S.Y.B.Sc.

(Computer Science)

SEMESTER - III (CBCS)

COMBINATORICS AND GRAPH THEORY

SUBJECT CODE : USCS305

© UNIVERSITY OF MUMBAI

Prof. Suhas Pednekar
Vice-Chancellor,
University of Mumbai,

Prof. Ravindra D. Kulkarni
Pro Vice-Chancellor,
University of Mumbai,

Prof. Prakash Mahanwar
Director,
IDOL, University of Mumbai,

Programme Co-ordinator : **Shri Mandar Bhanushe**
Head, Faculty of Science and Technology,
IDOL, University of Mumbai, Mumbai

Course Co-ordinator : **Mr. Sumedh Shejole**
Asst. Professor,
IDOL, University of Mumbai, Mumbai

Editor : **Mr.Vijay Kothawade**
Smt. Janakibai Rama Salvi College of Arts
Commerce and science, kalwa

Writers : **Ms. Prachi Abhijeet Surve**
Ramniranjan Jhunjhunwala College,
Ghatkopar West, Mumbai 400086

: **Prof. Prachita Sawant**
Smt. Janakibai Rama Salvi College of Arts
Commerce and science, kalwa

: **Dr. Priyanka**
J. K. College of Science and Commerce,
Navi Mumbai

July 2022, Print - I

Published by : Director
Institute of Distance and Open Learning ,
University of Mumbai,
Vidyanagari, Mumbai - 400 098.

DTP Composed and Printed by : Mumbai University Press
Vidyanagari, Santacruz (E), Mumbai - 400098

CONTENTS

Course: USCS305	TOPICS (Credits : 02 Lectures/Week: 03) Combinatorics and Graph Theory	

Objectives:

To give the learner a broad exposure of combinatorial Mathematics through applications especially the Computer Science applications.

Expected Learning Outcomes:

1. Appreciate beauty of combinatorics and how combinatorial problems naturally arise in many settings.
2. Understand the combinatorial features in real world situations and Computer Science applications.
3. Apply combinatorial and graph theoretical concepts to understand Computer Science concepts and apply them to solve problems

Unit I	**Introduction to Combinatorics:** Enumeration, Combinatorics and Graph Theory/ Number Theory/Geometry and Optimization, Sudoku Puzzles. Strings, Sets, and Binomial Coefficients: Strings- A First Look, Combinations, Combinatorial, The Ubiquitous Nature of Binomial Coefficients, The Binomial, Multinomial Coefficients. Induction: Introduction, The Positive Integers are Well Ordered, The Meaning of Statements, Binomial Coefficients Revisited, Solving Combinatorial Problems Recursively, Mathematical Induction, and Inductive Definitions Proofs by Induction. Strong Induction	**15L**
Unit II	**Graph Theory:** Basic Notation and Terminology, Multigraphs: Loops and Multiple Edges, Eulerian and Hamiltonian Graphs, Graph Coloring, Planar Counting, Labeled Trees, A Digression into Complexity Theory. Applying Probability to Combinatorics, Small Ramsey Numbers, Estimating Ramsey Numbers, Applying Probability to Ramsey Theory, Ramsey's Theorem The Probabilistic Method	**15L**
Unit III	**Network Flows:** Basic Notation and Terminology, Flows and Cuts, Augmenting Paths, The Ford-Fulkerson Labeling Algorithm,	**15L**

	A Concrete Example, Integer Solutions of Linear Programming Problems. Combinatorial Applications of Network Flows: Introduction, Matching in Bipartite Graphs, Chain partitioning, Pólya's Enumeration Theorem: Coloring the Vertices of a Square.	

Textbook(s):

1) Applied Combinatorics, Mitchel T. Keller and William T. Trotter, 2016, http://www.rellek.net/appcomb.

Additional Reference(s):

1) Applied Combinatorics, sixth.edition, Alan Tucker, Wiley; (2016)
2) Graph Theory and Combinatorics, Ralph P. Grimaldi, Pearson Education; Fifth edition (2012)
3) Combinatorics and Graph Theory, John Harris, Jeffry L. Hirst, Springer(2010).
4) Graph Theory: Modeling, Applications and Algorithms, Agnarsson, Pearson Education India (2008).

1

INTRODUCTION TO COMBINATORICS

Unit Structure:

1.0 OBJECTIVE

- Combinatorics, also referred to as Combinatorial Mathematics, is the field of mathematics concerned with problems of selection, arrangement, and operation within a finite or discrete system.
- It characterizes Mathematical relations and their properties.
- Mathematicians uses the term "Combinatorics" as it refers to the larger subset of Discrete Mathematics.
- It is frequently used in computer Science to derive the formulas and it is **used for the estimation of the analysis of the algorithms**.
- The important features of the combinatorics are as follows:
  - Counting the structures of the provided kind and size.
  - To decide when particular criteria can be fulfilled and analyzing elements of the criteria, such as combinatorial designs.
  - To identify "greatest", "smallest" or "optimal" elements, known as external combinatorics.
- Combinatorial structures that rise in an algebraic concept, or applying algebraic techniques to combinatorial problems, known as algebraic combinatorics.
- **In this chapter we are going to learn about how to :**
  - carry over an apply knowledge from Combinatorics and previous proof-based course

- we give a few examples of how we can use graphs to help with a variety of problems. These problems and examples will continue to come up throughout the course as we learn new graph or combinatorial ideas.
- read and understand assigned sections of the textbook.
- Independently study a new combinatorial topic and present this topic to their peers.
- use graphs to model real life situations.
- recognize graph theoretic properties of graphs and use these properties in problem-solving.
- use algorithms to study properties of graphs.

1.1 ENUMERATION

- Enumeration is just another word for counting.
- Enumeration (accurately determining how many) depends on several skills or methods, including careful counting one by one and subitizing.
- Enumeration also requires understanding key ideas, for example that the number of objects in a set stays the same even when objects are covered or moved.
- So let me show you an example of enumeration.
- We have five fingers. We can count through them. It turns out there's more ways we can enumerate fingers.
- Many basic problems in combinatorics involve counting the number of distributions of objects into cells.
- In this we may or may not be able to distinguish between the objects and the same for the cells. Also, the cells may be arranged in patterns.
- In short, the basic problem of enumerative combinatorics is that of counting the number of elements of a finite set.
- Many areas of discrete mathematics involve problems of counting or enumerating.
- Combinatorics is **all about number of ways of choosing some objects out of a collection and/or number of ways of their arrangement**.
- For example, suppose there are five members in a club, let's say there names are A, B, C, D, and E, and one of them is to be chosen as the coordinator.
- This section starts with two elementary but fundamental counting principles:

- **The Addition Principle**

If there are r1 different objects in the first set, r_2 different objects in the second set, ..., and r_m different objects in the mth set, and if the different sets are disjoint, then the number of ways to select an object from one of the m sets is $r_1 + r_2 + \cdots + r_m$.

- **The Multiplication Principle**

Suppose a procedure can be broken into m successive (ordered) stages, with r_1 different outcomes in the first stage, r_2 different outcomes in the second stage, ... , andrm different outcomes in the mth stage. If the number of outcomes at each stage is independent of the choices in previous stages and if the composite outcomes are all distinct, then the total procedure has $r_1 \times r_2 \times \cdots \times r_m$ different composite outcomes.

- Remember that the addition principle requires disjoint sets of objects and the multiplication principle requires that the procedure break into ordered stages and that the composite outcomes be distinct.

Example 1 :

In how many ways can you type one character on a keyboard if the keyboard has 26 letter keys, 10 digit keys and no others?

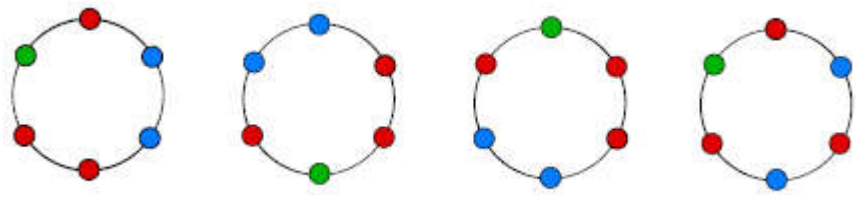
If one type of object can be selected in rways and another type can be selected in sways, then the number of ways of selecting any object is:

r+s.

In our example it is important that we know that there are no letters which are also digits.

Problem Scenario:

A circular necklace with a total of six beads will be assembled using beads of three different colors. In Figure we show four such necklaces—however, note that the first three are actually the same necklace. Each has three red beads, two blues and one green. On the other hand, the fourth necklace has the same number of beads of each color but it is a different necklace.



1. How many different necklaces of six beads can be formed using three reds, two blues and one green?

2. How many different necklaces of six beads can be formed using red, blue and green beads (not all colors have to be used)?

3. How many different necklaces of six beads can be formed using red, blue and green beads if all three colors have to be used?

4. How would we possibly answer these questions for necklaces of six thousand beads made with beads from three thousand different colors?

What special software would be required to find the exact answer and how long would the computation take?

1.2 COMBINATORICS AND GRAPH THEORY

- A graph G consists of a vertex set V and a collection E of 2-element subsets of V.
- Elements of E are called edges.
- In our course, we will (almost always) use the convention that V {1, 2, 3, . . ., n} for some positive integer n.
- With this convention, graphs can be described precisely with a text file:
  - The first line of the file contains a single integer n, the number of vertices in the graph.
  - Each of the remaining lines of the file contains a pair of distinct integers and specifies an edge of the graph.

We illustrate this convention in Figure 1.2 with a text file and the diagram for the graph G it defines.

```
graph1.txt
9
6 2
1 5
1 7
6 8
9 1
4 3
5 7
1 3
5 9
7 9
```

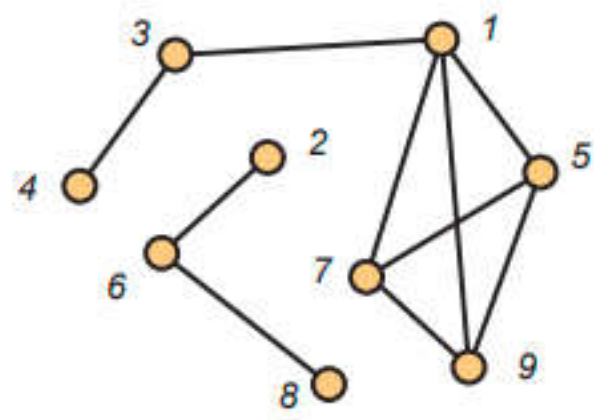


FIGURE 1.2: A GRAPH DEFINED BY DATA

The above graph G can be defined or explained as following:

➔ G has 9 vertices and 10 edges.

➔ {2, 6} is an edge.

➔ Vertices 5 and 9 are adjacent.

➔ {5, 4} is not an edge.

➔ Vertices 3 and 7 are not adjacent.

➔ P (4, 3, 1, 7, 9, 5) is a path of length 5 from vertex 4 to vertex 5.

➔ C (5, 9, 7, 1) is cycle of length 4.

➔ G is disconnected and has two components. One of the components has vertex set {2, 6, 8}.

➔ {1, 5, 7} is a triangle.

➔ {1, 7, 5, 9} is a clique of size 4.

➔ {4, 2, 8, 5} is an independent set of size 4.

1.3 NUMBER THEORY

- Number theory generally concerns itself with the properties of the positive integers.
- G.H. Hardy was a brilliant British mathematician who lived through both World Wars and conducted a large deal of number-theoretic research.
- He wrote in his 1940 essay A Mathematician's Apology "[n]o one has yet discovered any warlike purpose to be served by the theory of numbers or relativity, and it seems very unlikely that anyone will do so for many years." [1]
- Little did he know, the purest mathematical ideas of number theory would soon become indispensable for the cryptographic techniques that kept communications secure.
- Our subject here is not number theory, but we will see a few times where combinatorial techniques are of use in number theory.

Example:

Form a sequence of positive integers using the following rules.

Start with a positive integer $n > 1$.

If n is odd, then the next number is $3n + 1$.

If n is even, then the next number is $n/2$.

Halt if you ever reach 1.

➔ For example, if we start with 28,

➔ the sequence is 28, 14, 7, 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1.

➔ Now suppose you start with 19.

➔ Then the first few terms are 19, 58, 29, 88, 44, 22.

➔ But now we note that the integer 22 appears in the first sequence, so the two sequences will agree from this point on.

➔ **Sequences formed by this rule are called Collatz sequences.**

➔ Pick a number somewhere between 100 and 200 and write down the sequence you get. Regardless of your choice, you will eventually halt with a 1.

However, is there some positive integer n (possibly quite large) so that if you start from n, you will never reach 1?

Questions arising in number theory can also have an enumerative flair, as the following example shows.

Example In we show the integer partitions of 8.

8 distinct parts	4+1+1+1+1
7+1 distinct parts, odd parts	3+3+2
6+2 distinct parts	3+3+1+1 odd parts
6+1+1	3+2+2+1
5+3 distinct parts, odd parts	3+2+1+1+1
5+2+1 distinct parts	3+1+1+1+1+1 odd parts
5+1+1+1 odd parts	2+2+2+2
4+4	2+2+2+1+1
4+3+1 distinct parts	2+2+1+1+1+1
4+2+2	2+1+1+1+1+1+1
4+2+1+1	1+1+1+1+1+1+1+1 odd parts

FIGURE : THE PARTITIONS OF 8, NOTING THOSE INTO DISTINCT PARTS AND THOSE INTO ODD PARTS.

There are 22 partitions altogether, and as noted, exactly 6 of them are partitions of 8 into odd parts.

Also, exactly 6 of them are partitions of 8 into distinct parts.

Example:

How many positive factors does the number $N = 2^3 5^4 7^3 11^5$ have?

Solution:

➔ From the Unique Factorization Theorem for integers,

➔ A divides $N = p_1^{k_1} \dots p_s^{k_s}$ (p_i 's are distinct primes)

➔ iff $A = p_1^{l_1} \dots p_s^{l_s}$, with $0 \le l_i \le k_i$, i = 1, . . . , s.

➔ By the product rule, there are $(k_1 + 1)(k_2 + 1) \dots (k_s + 1)$ choices for A, since l_i 's can be chosen independently in $k_i + 1$ ways each.

➔ In our case, the number of positive factors is $4 \cdot 5 \cdot 4 \cdot 6 = 48$

1.4 COMBINATORICS AND GEOMETRY:

- There are many problems in geometry that are innately combinatorial or for which combinatorial techniques shed light on the problem.
- Combinatorial geometry is a blending of principles from the areas of combinatorics and geometry.
- It deals with combinations and arrangements of geometric objects and with discrete properties of these objects.

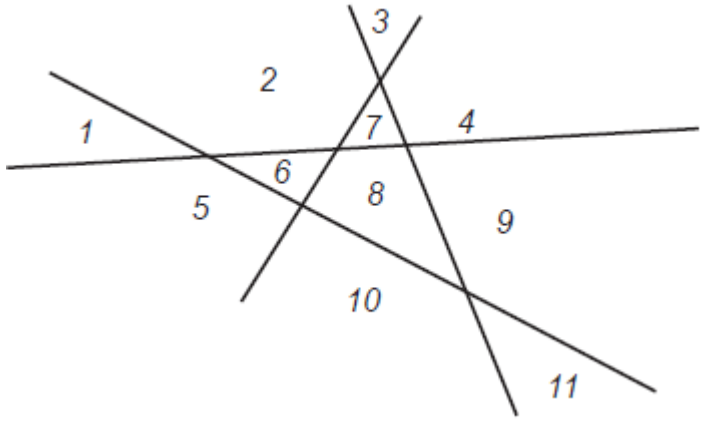
- Combinatorics, also called combinatorial mathematics, the field of mathematics concerned with problems of selection, arrangement, and operation within a finite or discrete system.
- Included is the closely related area of combinatorial geometry.
- Combinatorial geometry is a blending of principles from the areas of combinatorics and geometry.
- It deals with combinations and arrangements of geometric objects and with discrete properties of these objects.
- It is concerned with such topics as:
  - packing,
  - covering,
  - coloring,
  - folding,
  - symmetry,
  - tiling,
  - partitioning,
  - decomposition, and
  - illumination problems.
- Combinatorial geometry includes aspects of topology, graph theory, number theory, and other disciplines.
- Example:

In the given Figure, we show a family of 4 lines in the plane.

Each pair of lines intersects and no point in the plane belongs to more than two lines. These lines determine 11 regions.

Under these same restrictions,

1. how many regions would a family of 8947 lines determine?
2. Can different arrangements of lines determine different numbers of regions?



1.5 COMBINATORICS AND OPTIMIZATION

- Combinatorial optimization is a subset of mathematical optimization that is related to operations research, algorithm theory, and computational complexity theory.
- Combinatorics looks at permutations and combinations.
- Optimization explores ways to make any operation work more efficiently within given constraints.
- Together, they provide powerful methods for modelling and solving large management problems, from optimizing flight schedules to making a factory's layout as efficient as possible.
- Combinatorial optimization is the process of searching for maxima (or minima) of an objective function F whose domain is a discrete but large configuration space (as opposed to an N-dimensional continuous space).
- Some simple examples of typical combinatorial optimization problems are:
  - **The Traveling Salesman Problem:** given the (x, y) positions of N different cities, find the shortest possible path that visits each city exactly once.
  - **Bin-Packing:** given a set of N objects each with a specified size s_i, fit them into as few bins (each of size B) as possible.
  - **Integer Linear Programming:** maximize a specified linear combination of a set of integers X_1 ... X_N subject to a set of linear constraints each of the form
  - $a_1X_1 + ... + a_NX_N <= c$.
  - **Job-shop Scheduling:** given a set of jobs that must be performed, and a limited set of tools with which these jobs can be performed, find a schedule for what jobs should be done when and with what tools that minimizes the total amount of time until all jobs have been completed.

- **Boolean Satisfiability:** assign values to a set of boolean variables in order to satisfy a given boolean expression. (A suitable objective function might be the number of satisfied clauses if the expression is a CNF formula.)
- **The space of possible solutions** is typically too large to search exhaustively using pure brute force. In some cases, problems can be solved exactly using Branch and Bound techniques.
- However, in other cases no exact algorithms are feasible, and randomized search algorithms must be employed, such as:
  - Random-restart hill-climbing
  - Simulated annealing
  - Genetic algorithms
  - Tabu search
- A large part of the field of Operations Research involves algorithms for solving combinatorial optimization problems.
- However, these problems are inherently continuous.
- In theory, you can cross the river at any point you want, even if it were irrational. (OK, so not exactly irrational, but a good decimal approximation.)
- In this course, we will examine a few optimization problems that are not continuous, as only integer values for the variables will make sense.
- It turns out that many of these problems are very hard to solve in general.

1.6 SUDOKU PUZZLES

- The class of Sudoku puzzles consists of a partially completed row-column grid of cells partitioned into N regions each of size N cells, to be filled in ("solved") using a prescribed set of N distinct symbols (typically the numbers {1, ..., N}), so that each row, column and region contains exactly one of each element of the set.
- A Sudoku puzzle is a 9×9 array of cells that when completed have the integers 1, 2, . . . , 9 appearing exactly once in each row and each column.
- Also, the numbers 1, 2, 3, . . . , 9 appear once in each of the nine 3×3 subsquares identified by the darkened borders.
- To be considered a legitimate Sudoku puzzle, there should be a unique solution.

- In the following Figure, we show two Sudoku puzzles.

		7				8	2	
	9				1			
	4		9	7				
					5	4		6
		3				7		
5		6	7					
				8	4		5	
			6				1	
	2	4				6		

[A]

	8	1	3		2	6		
6		9	5		1		2	
2	3							
5		2		3		7	8	9
4	6	3		8		2		1
							6	2
	2		7		9	5		3
		6	8		3	9	4	

[B]

- Figure [A] is fairly easy, and figure [B] is far more challenging.
- There are many sources of Sudoku puzzles, and software that generates Sudoku puzzles and then allows you to play them with an attractive GUI is available for all operating systems.

1.7 EXERCISE

Solve the following:

1. Consider the graph G shown in Figure, and answer the following:

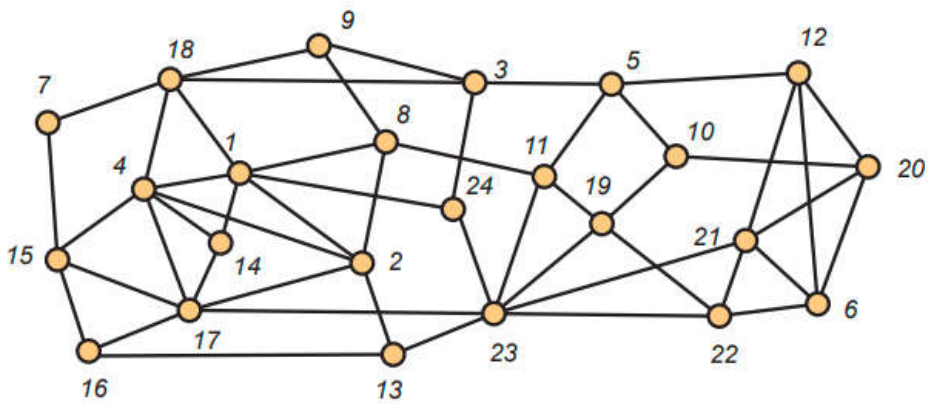


FIGURE : A CONNECTED GRAPH

1. What is the largest k for which G has a path of length k?
2. What is the largest k for which G has a cycle of length k?
3. What is the largest k for which G has a clique of size k?
4. What is the largest k for which G has an independent set of size k?
5. What is the shortest path from vertex 7 to vertex 6?

2. In the given Figure we use letters for the labels on the vertices to help distinguish visually from the integer weights on the edges.

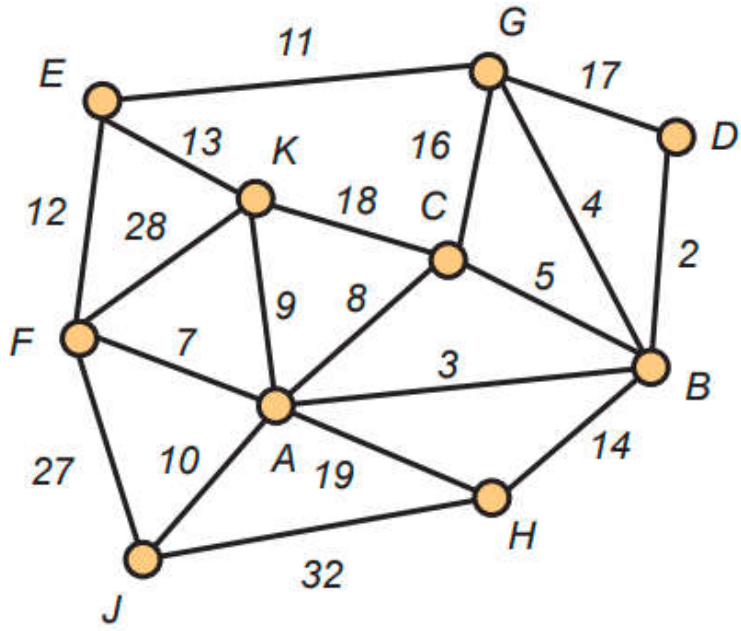


Figure: A labeled graph with weighted edges

Suppose the vertices are cities, the edges are highways and the weights on the edges represent distance.

1. What is the shortest path from vertex E to vertex B?

2. Suppose Ariel is a salesperson whose home base is city A. In what order should Ariel visit the other cities so that she goes through each of them at least once and returns home at the end—while keeping the total distance traveled to a minimum? Can Ariel accomplish such a tour visiting each city exactly once?

3. Sanjay is a highway inspection engineer and must traverse every highway each month. Sanjay's homebase is City E. In what order should Sanjay traverse the highways to minimize the total distance traveled? Can Sanjay make such a tour traveling along each highway exactly once?

3. n lines on a plane cut the plane into parts. Assume that every two lines intersect and there is no triple intersections. Find the number of parts. How many of these parts are bounded?

4. Consider the graph G with 4 vertices v1,v2,v3 and v4 and degree of vertices are 3,5,2 and 1 respectively. Is it possible to construct such graph? Explain.

5. (a) How many ways are there to pick a sequence of two different letters of the alphabet that appear in the word CRAB? In STATISTICS?

(b) How many ways are there to pick first a vowel and then a consonant from CRAB? From STATISTICS?

6. (a) How many integers are there between 0 and 50 (inclusive)?

(b) How many of these integers are divisible by 2?

(c) How many (unordered) pairs of these integers are there whose difference is 5?

7. A store carries eight styles of pants. For each style, there are 12 different possible waist sizes, five pants lengths, and four color choices. How many different types of pants could the store have?

8. How many different sequences of heads and tails are possible if a coin is flipped 100 times? Using the fact that $210 = 1024 \approx 1000 = 103$, give your answer in terms of an (approximate) power of 10.

9. How many six-letter "words" (sequence of any six letters with repetition) are there? How many with no repeated letters?

10. How many ways are there to pick a man and a woman who are not husband and wife from a group of n married couples?

11. Given 10 different English books, six different French books, and four different German books, (a) How many ways are there to select one book? (b) How many ways are there to select three books, one of each language?

Reference :

1. Applied Combinatorics2017 Edition, Mitchel T. Keller William T. Trotter

❖❖❖❖

2

STRINGS, SETS, AND BINOMIAL COEFFICIENTS

Unit Structure

2.0 OBJECTIVE

- Much of combinatorial mathematics can be reduced to the study of strings, as they form the basis of all written human communications.
- Also, strings are the way humans communicate with computers, as well as the way one computer communicates with another.
- As we shall see, sets and binomial coefficients are topics that fall under the string umbrella.So it is important to begin our in-depth study of combinatorics with strings.

2.1 STRINGS- A FIRST LOOK

- Let n be a positive integer.
- Throughout this chapter, we will use the shorthand notation [n] to denote the n-element set {1, 2, . . ., n}.
- Now let X be a set.
- Then a function s: [n] → X is also called an X-string of length n.
- In discussions of X-strings, it is customary to refer to the elements of X as characters, while the element s(i) is the i^{th} character of s.

- Whenever practical, we prefer to denote a string s by writing s "$x_1x_2x_3 \ldots x_n$", rather than the more cumbersome notation $s(1) = x_1$, $s(2) = x^2$, …, $s(n) = x_n$.
- There are a number of alternatives for the notation and terminology associated with strings.
- First, the characters in a string s are frequently written using subscripts as s1, s2, . . . , sn, so the i^{th}-term of s can be denoted si rather than s(i).
- Strings are also called sequences, especially when X is a set of numbers and the function s is defined by an algebraic rule.
  - For example, the sequence of odd integers is defined by $s_i = 2i-1$.
- Alternatively, strings are called words, the set X is called the alphabet and the elements of X are called letters.
  - For example, aababbccabcbb is a 13-letter word on the 3-letter alphabet {a, b, c}.
- In many computing languages, strings are called arrays.
- Also, when the character s(i) is constrained to belong to a subset $X_i \subseteq X$, a string can be considered as an element of the cartesian product $X_1 \times X_2 \times \cdots \times X_n$, which is normally viewed as n-tuples of the form (x1, x2, . . . , xn) such that $xi \in Xi$ for all $i \in [n]$.

Example

In the state of India, license plates consist of:

a) four digits

b) followed by a space

c) followed by three capital letters.

d) the first digit cannot be a 0.

How many license plates are possible?

Solution.

Let X consist of the digits {0, 1, 2, . . ., 9},

let Y be the singleton set whose only element is a space, and

let Z denote the set of capital letters.

A valid license plate is just a string from

$(X - \{0\}) \times X \times X \times X \times Y \times Z \times Z \times Z$

so the number of different license plates is

$9 \times 10^3 \times 1 \times 26^3 = 158\ 184\ 000$,

since the size of a product of sets is the product of the sets' sizes.

Explanation:

- ➔ We can get a feel for why this is the case by focusing just on the digit part of the string here.
- ➔ We can think about the digits portion as being four blanks that need to be filled.
- ➔ The first blank has 9 options (the digits 1 through 9).
- ➔ If we focus on just the digit strings beginning with 1, one perspective is that they range from 1000 to 1999, so there are 1000 of them.
- ➔ However, we could also think about there being 10 options for the second spot, 10 options for the third spot, and 10 options for the fourth.
- ➔ Multiplying 10×10×10 gives 1000.
- ➔ Since our analysis of filling the remaining digit blanks didn't depend on our choice of a 1 for the first position, we see that each of the 9 choices of initial digit gives 1 000 strings, for a total of 9 000 9×10^3 .
- ➔ In the case that X {0, 1}, an X-string is called a 0–1 string (also a binary string or bit string.).
- ➔ When X {0, 1, 2}, an X-string is also called a ternary string.

2.2 PERMUTATIONS :

Fundamental Principles Of Counting:

- ***Fundamental principle of multiplication –***

If there are three different events such that one event occurs in m different ways, second event happens in n different ways and the third event occurs in p different ways, then all three events simultaneously will happen in **m×n×p way**s.

- ***Fundamental principle of addition –***

If there are two jobs such that the first one can be performed independently in m number of ways and the second work independently can be done in n number of ways, then either of the two jobs can be performed in **(m+n) ways.**

- A permutation is a mathematical technique that determines the number of possible arrangements in a set when the order of the arrangements matters.
- Common mathematical problems involve choosing only several items from a set of items with a certain order.
- Permutation refers to a particular arrangement of a set of objects in a defined order or a process of arranging numbers or letters in a sequence.
- We can form many different permutations from a given set of objects taking all of the digits from the set at a time or a particular number of objects at a time.
- **Permutations are frequently confused with another mathematical technique called combinations.** However, in combinations, the order of the chosen items does not influence the selection.
- **Permutations –** It is the linear arrangements of distinct objects taken some or all at a time. The number of arrangements possible is called the permutations. If we have two positive integers r and n such that **l ≤ r ≤ n,** then the total number of arrangements or permutations possible for n distinct items taken r at a time is mathematically given by,
  - **Formula for Calculating Permutations**

The general permutation formula is expressed in the following way:

$$P(n, k) = \frac{n!}{(n - k)!}$$

Where:

- n – the total number of elements in a set
- k – the number of selected elements arranged in a specific order
- ! – factorial

- Factorial (noted as "!") is the product of all positive integers less than or equal to the number preceding the factorial sign. **For example, 3! = 1 x 2 x 3 = 6**.
- The formula above is used in situations when we want to select only several elements from a set of elements and arrange the selected elements in a special order.
- **Permutations Under Certain Conditions:**

The total number of arrangements or permutations taken r at a time from a set of n different objects;

- When we always have to include a particular object in every arrangement is $^{n-1}C_{r-1} \times r!$.
- When we don't have to include a particular object in any arrangement it is $^{n-1}C_r \times r!$.

Example:

Let's say we have 8 people:

1: Alice

2: Bob

3: Charlie

4: David

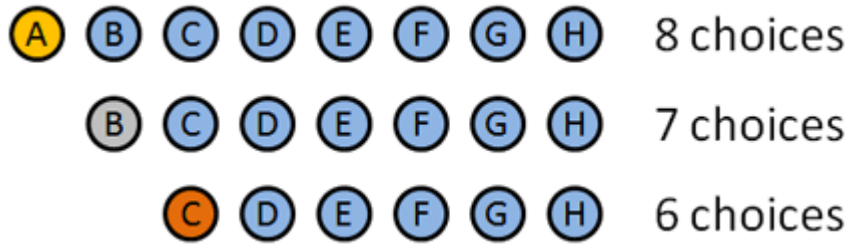
5: Eve

6: Frank

7: George

8: Horatio

How many ways can we award a 1st, 2nd and 3rd place prize among eight contestants? (Gold / Silver / Bronze)



Solution:

We're going to use permutations since the order we hand out these medals matters. Here's how it breaks down:

- **Gold medal:**
  - 8 choices:
  - A B C D E F G H
  - Let's say A wins the Gold.
- **Silver medal:**
  - 7 choices:
  - B C D E F G H.
  - Let's say B wins the silver.

- **Bronze medal:**
  - 6 choices:
  - C D E F G H.
  - Let's say C wins the bronze.
- We picked certain people to win,
  - we had 8 choices at first,
  - then 7,
  - then 6.

- **The total number of options was**

8*7*6=336

Explanation:

➔ Let's look at the details. We had to order 3 people out of 8.

➔ To do this, we started with all options (8) then took them away one at a time (7, then 6) until we ran out of medals.

➔ We know the factorial is:

$$8! = 8 \cdot 7 \cdot 6 \cdot 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1$$

➔ Unfortunately, that does too much! We only want 8*7*6

➔ **How can we "stop" the factorial at 5?**

➔ This is where permutations get cool: notice how we want to get rid of 5*4*3*2*1

➔ **What's another name for this? 5 factorial!**

So, if we do 8!/5! we get:

$$\frac{8!}{5!} = \frac{8 \cdot 7 \cdot 6 \cdot 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1}{5 \cdot 4 \cdot 3 \cdot 2 \cdot 1} = 8 \cdot 7 \cdot 6$$

➔ And why did we use the number 5? Because it was left over after we picked 3 medals from 8. So, a better way to write this would be:

$$\frac{8!}{(8-3)!}$$

➔ If we have n items total and want to pick k in a certain order, we get:

$$\frac{n!}{(n-k)!}$$

➔ **And this is the permutation formula:**

You have n items and want to find the number of ways k items can be ordered:

$$P(n,k) = \frac{n!}{(n-k)!}$$

2.3 COMBINATIONS

- Combinations are studied in combinatorics but are also used in different disciplines, including mathematics and finance.
- A combination is a mathematical technique that determines the number of possible arrangements in a collection of items where the order of the selection does not matter.
- In combinations, you can select the items in any order.
- Combinations can be confused with permutations.
- However, in permutations, the order of the selected items is essential.
- For example, the arrangements ab and ba are equal in combinations (considered as one arrangement), while in permutations, the arrangements are different.
- **Combinations –** If we have to select combinations of items from a given set of items such that the order or arrangement doesn't matter, then we use combinations. Such that to find the number of ways of selecting **r** objects from a set of **n** objects, then mathematically it is given by,

Formula for Combination

$$C(n,k) = \binom{n}{k} = \frac{n!}{k!\,(n-k)!}$$

Note that the formula above can be used only when the objects from a set are selected without repetition.

- Where:
  - n – the total number of elements in a set
  - k – the number of selected objects (the order of the objects is not important)
  - ! – factorial
- Factorial (noted as "!") is a product of all positive integers less or equal to the number preceding the factorial sign. For example, 3! = 1 x 2 x 3 = 6.

Example:

Let's say we have 8 people:

1: Alice

2: Bob

3: Charlie

4: David

5: Eve

6: Frank

7: George

8: Horatio

But here instead of giving separate Gold, Silver and Bronze medals,we are offering empty tin cans.

How many ways can I give 3 tin cans to 8 people?

Solution:

- In this case, the order we pick people doesn't matter.
- If I give a can to Alice, Bob and then Charlie, it's the same as giving to Charlie, Alice and then Bob.
- For a moment, let's just figure out how many ways we can rearrange 3 people.
- Well, we have
  - 3 choices for the first person,
  - 2 for the second, and
  - only 1 for the last.
- So we have

  $3*2*1$ ways to re-arrange 3 people.

- **If we want to figure out how many combinations we have, we just create all the permutations and divide by all the redundancies.**
- In our case, we get 336 permutations (from above), and
- we divide by the 6 redundancies for each permutation and get

  336/6 = 56.

Explanation:

➔ The general formula is

$$C(n,k) = \frac{P(n,k)}{k!}$$

➔ **which means "Find all the ways to pick k people from n, and divide by the k! variants".**

➔ Writing this out, we get our **combination formula**, or the number of ways to combine k items from a set of n:

$$C(n,k) = \frac{n!}{(n-k)!k!}$$

➔ Sometimes C(n,k) is written as:

$$\binom{n}{k}$$

➔ **which is the binomial coefficient.**

- **Difference Between Permutation and Combination**

To help students better understand this topic, we have formulated a table that contains all the points related to the difference between combination and permutation.

That table is mentioned below.

Permutation	**Combination**
It refers to the task of arranging digits, people, alphabets, colours, numbers, and letters	It is the selection of food, menu, clothes, teams, subjects, and other objects
Example: ● Permutation is to pick a team captain, picture, or shortstop from a group ● Deciding on your two favourite colours in a particular order from a colour brochure ● Picking winners for the first, second, and third place	Example: ● Combinations includes picking any three team members from a group ● Selecting any two colours from a colour brochure ● Picking any three winners for an award

A few examples:

Here's a few examples of **combinations (order doesn't matter) from permutations (order matters).**

Example 1

Combination:

Picking a team of 3 people from a group of 10.

C(10,3)=10!/(7!*3!)

=10*9*8/(3*2*1)

=120

Example 2

Permutation:

Picking a President, VP and Waterboy from a group of 10.

P(10,3)=10!/7!

=10*9*8

=720

2.4 COMBINATORIAL

- **Combinatorics** is a stream of mathematics that concerns the study of finite discrete structures.
- It deals with the study of permutations and combinations, enumerations of the sets of elements.
- Mathematicians use the term "Combinatorics" as it refers to the larger subset of Discrete Mathematics.
- It is frequently used in computer Science to derive the formulas and it is used for the estimation of the analysis of the algorithms.
- In this section, let us discuss combinatorics, its features, formulas, applications and examples in detail.
- Combinatorics, also called combinatorial mathematics, the **field of mathematics concerned with problems of selection, arrangement, and operation within a finite or discrete system**.
- Included is the closely related area of combinatorial geometry.
- **Features of combinatorics**

Some of the important features of the combinatorics are as follows:

- Counting the structures of the provided kind and size.
- To decide when particular criteria can be fulfilled and analyzing elements of the criteria, such as combinatorial designs.
- To identify "greatest", "smallest" or "optimal" elements, known as external combinatorics.
- Combinatorial structures that rise in an algebraic concept, or applying algebraic techniques to combinatorial problems, known as algebraic combinatorics.

- **Applications of combinatorics**

Combinatorics is applied in most of the areas such as:

- Communication networks, cryptography and network security
- Computational molecular biology
- Computer architecture
- Scientific discovery
- Languages
- Pattern analysis
- Simulation
- Databases and data mining
- Homeland security
- Operations research

- **Example:**

Let n be a positive integer. Use following figure explain why

$$1+2+3+\cdots+n=\frac{n(n+1)}{2}.$$

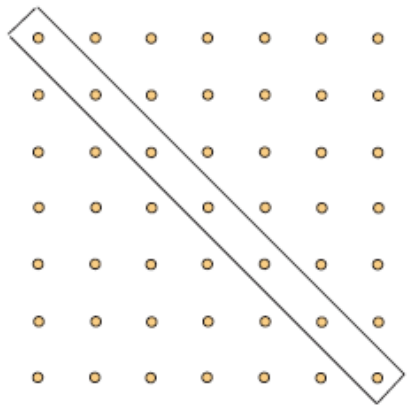


Figure: The sum of the first n integers

Solution:

- Consider an (n+1) (n+1) array of dots as depicted in Figure.
- There are$(n+1)^2$ dots altogether, with exactly n+1 on the main diagonal.
- The off-diagonal entries split naturally into two equal size parts, those above and those below the diagonal.
- Furthermore, each of those two parts has $S(n) = 1 + 2 + 3 + \cdots + n$ dots.
- It follows that:

$$S(n) = \frac{(n+1)^2 - (n+1)}{2}$$

2.5 THE UBIQUITOUS NATURE OF BINOMIAL COEFFICIENTS

- In this section, we present several combinatorial problems that can be solved by appealing to binomial coefficients, even though at first glance, they do not appear to have anything to do with sets.

Example

The office assistant is distributing supplies. In how many ways can he distribute 18 identical folders among four office employees: Audrey, Bart, Cecilia and Darren, with the additional restriction that each will receive at least one folder?

Solution:

➔ Imagine the folders placed in a row.

➔ Then there are 17 gaps between them.

➔ Of these gaps, choose three and place a divider in each.

➔ Then this choice divides the folders into four non-empty sets.

➔ The first goes to Audrey, the second to Bart, etc.

➔ Thus the answer isC(17,3).

➔ We can illustrate this scheme with Audrey receiving 6 folders, Bart getting 1 , Cecilia 4 and Darren 7 as follows:

Figure. Distributing Identical Objects into Distinct Cells

Example:

Suppose we redo the preceding problem but drop the restriction that each of the four employees gets at least one folder. Now how many ways can the distribution be made?

Solution.

➔ The solution involves a "trick" of sorts.

➔ First, we convert the problem to one that we already know how to solve.

➔ This is accomplished by *artificially* inflating everyone's allocation by one.

➔ In other words, if Bart will get 7 folders, we say that he will get .8.

➔ Also, artificially inflate the number of folders by ,4, one for each of the four persons. So now imagine a row of 22=18+4 folders.

➔ Again, choose 3 gaps.

➔ This determines a non-zero allocation for each person.

➔ The actual allocation is one less—and may be zero. So the answer is .C(21,3).

2.6 THE BINOMIAL

- Binomial theorem, statement that **for any positive integer n, the nth power of the sum of two numbers a and b may be expressed as the sum of n + 1 terms of the form**.
- The binomial theorem is used heavily in **Statistical and Probability Analyses**.
- It is so useful as our economy depends on Statistical and Probability Analysis.
- In higher mathematics and calculation, the Binomial Theorem is used in finding roots of equations in higher powers.
- In Algebra, a binomial expression contains two terms joined by either addition or subtraction sign.
- For instance, (x + y) and (2 – x) are examples of binomial expressions.

- Sometimes, we may need to expand binomial expressions as shown below.

$(a + b)^0 = 1$

$(a + b)^1 = a + b$

$(a + b)^2 = a^2 + 2ab + b^2$

$(a + b)^3 = a^3 + 3a^2b + 3ab^2 + b^3$

$(a + b)^4 = a^4 + 4a^3b + 6a^2b^2 + 4ab^3 + b^4$

$(a + b)^5 = a^5 + 5a^4b + 10a^3b^2 + 10a^2b^3 + 5ab^4 + b^5$

- In this section, we will learn how to use the Binomial theorem to expand binomial expression without having to multiply everything out the long way.

The Binomial Theorem?

- The traces of the binomial theorem were known to human beings since the 4th century BC.
- The binomial for cubes were used in the 6th century AD.
- An Indian mathematician, Halayudha, explains this method using Pascal's triangle in the 10th century AD.
- The clear statement of this theorem was stated in the 12th century.
- The mathematicians took these findings to the next stages till Sir Isaac Newton generalized the binomial theorem for all exponents in 1665.

The Binomial Theorem states the algebraic expansion of exponents of a binomial, which means it is possible to expand a polynomial $(a + b)^n$ into the multiple terms.

- *Mathematically, this theorem is stated as:*

$$(a + b)^n = a^n + \binom{n}{1} a^{n-1}b^1 + \binom{n}{2} a^{n-2}b^2 + \binom{n}{3} a^{n-3}b^3 + \ldots\ldots + b^n$$

where $\binom{n}{1}$, $\binom{n}{2}$, ... are the binomial coefficients.

- we can express the Binomial formula as:

$(a + b)^n = {}^nC_0\, a^n + {}^nC_1\, a^{n-1}b + {}^nC_2\, a^{n-2}b^2 + {}^nC_3\, a^{n-3}b^3 + \ldots\ldots\ldots + {}^nC_n\, b^n$

Where $\binom{n}{r} = {}^nC_r = n! / \{r!\,(n - r)!\}$ and (C) and (!) are the combinations and factorial respectively.

For example:

$^{10}C_6 = 10! / (10 - 6)!\, 6!$

$= 10! / 4!\, 6!$

$= (1 \times 2 \times 3 \times 4 \times 5 \times 6 \times 7 \times 8 \times 9 \times 10) / 1 \times 2 \times 3 \times 4 \times 1 \times 2 \times 3 \times 4 \times 5 \times 6$

$= 7 \times 8 \times 9 \times 10 / 1 \times 2 \times 3 \times 4$

$= 7 \times 3 \times 10 = 210$

Steps to use the Binomial Theorem:

- There are a few things which you need to remember while applying the Binomial Theorem.*These are:*
  - The exponents of the first term (a) decreases from n to zero
  - The exponents of the second term (b) increases from zero to n
  - The sum of the exponents of a and b is equal to n.
  - The coefficients of the first and last term are both 1.
- Let's use Binomial Theorem on certain expressions to practically understand the theorem.

Example 1

Expand $(a + b)^5$

Solution

$(a + b)^5 = a^n + (^5_1)\, a^{5-1}b^1 + (^5_2)\, a^{5-2}b^2 + (^5_3)\, a^{5-3}b^3 + (^5_4)\, a^{5-4}b^4 + b^5$

$= a^5 + 5a^4b + 10a^3b^2 + 10a^2b^3 + 5ab^4 + b^5$

Example 2

Expand $(x + 2)^6$ using the Binomial Theorem.

Solution

Given a = x;

b = 2 and n = 6

Substitute the values in binomial formula

$(a + b)^n = a^n + na^{n-1}b^1 + [n(n-1)/2!]\, a^{n-2}b^2 + [n(n-1)(n-2)/3!]a^{n-3}b^3 + \ldots\ldots\ldots + b^n$

We get,

$(x + 2)^6 = x^6 + 6x^5(2)^1 + [(6)(5)/2!]\,(x^4)\,(2^2) + [(6)(5)(4)/3!]\,(x^3)\,(2^3) + [(6)(5)(4)(3)/4!]\,(x^2)\,(2^4) + [(6)(5)(4)(3)(2)/5!]\,(x)\,(2^5) + (2)^6$

$= x^6 + 12x^5 + 60x^4 + 160x^3 + 240x^2 + 192x + 64$

2.7 MULTINOMIAL COEFFICIENTS

Multinomial coefficients are generalizations of binomial coefficients, with a similar combinatorial interpretation. They are **the coefficients of terms in the expansion of a power of a multinomial**, in the multinomial theorem.

The multinomial coefficient is nearly always introduced by way of die tossing

$$\binom{N}{n_1\ n_2\ n_3\ n_4\ n_5\ n_6} = \frac{N!}{n_1!\,n_2!\,n_3!\,n_4!\,n_5!\,n_6!}$$

$$= \frac{N!}{\prod n_i!}\ .$$

Note the use of the product operator $\prod$ in the last expression; it is similar to the summation $\sum$ operator .

It is handy in many instances in statistics.

A **practical example of when you might use the multinomial coefficient** is given by Harris et. al in *Combinatorics and Graph Theory* .

Let's say you wanted to count the number of ways to order *n* objects.

If the objects are all different, then there are n! ways to order them.

But if some of the objects are a multiset and some of the objects are the same, n! will produce too many permutations.

For example, let's say you're trying to find the number of different permutations for the letters in the word **Mississippi**. Note there are actually only four unique letters: MISP. The number of times the letters appear:

- M = 1
- I = 4
- S = 4
- P = 2

Inserting those values into the multinomial coefficient formula, where *n* is the total number of letters in the word MISSISSIPPI, and kn is the individual letter count (from the above list):

n! / k1! * k2! * k3! * k4! = 11! / (1! * 4! * 4! * 2!) = 11! / (1 * 24 * 24 * 2) = 34,650.

Note that n! gives 11! = 39916800, which is way larger than the actual number of distinguishable permutations.

Example:

How many different rearrangements of the string:

MITCHELTKELLERANDWILLIAMTTROTTERAREGENIUSES!!

are possible if all letters and characters must be used?

Solution:

Note that there are a total of characters distributed as follows:

3 A's,

1 C,

1 D,

7 E's,

1 G,

1 H,

4 I's,

1 K,

5 L's,

2 M's,

2 N's,

1 O,

4 R's,

2 S's,

6 T's,

1 U,

1 W, and

2 !'s.

So the number of rearrangements is

$$\frac{45!}{3!1!1!7!1!1!4!1!5!2!2!1!4!2!6!1!1!2!}.$$

Example What is the coefficient of $x^{99}y^{60}z^{14}$ in $(2x^3 + y - z^2)^{100}$? What about $x^{99}y^{61}z^{13}$?

Solution:

By the Multinomial Theorem, the expansion of $(2x^3 + y - z^2)^{100}$ has terms of the form

$$\binom{100}{k_1, k_2, k_3}(2x^3)^{k_1}y^{k_2}(-z^2)^{k_3} = \binom{100}{k_1, k_2, k_3}2^{k_1}x^{3k_1}y^{k_2}(-1)^{k_3}z^{2k_3}.$$

The $x^{99}y^{60}z^{14}$ arises when $k_1 = 33$, $k_2 = 60$, and $k_3 = 7$, so it must have coefficient

$$-\binom{100}{33, 60, 7}2^{33}.$$

For $x^{99}y^{61}z^{13}$, the exponent on z is odd, which cannot arise in the expansion of $(2x^3 + y - z^2)^{100}$, so the coefficient is 0.

2.8 EXERCISE

Solve the following

1. A machine instruction in a 32-bit operating system is just a bit string of length 32. Thus, there are 2 options for each of 32 positions to fill, making the number of such strings $2^{32} = 4\,294\,967\,296$. In general, the number of bit strings of length n is 2^n.

2. Find the total number of permutations and combinations if the value of n is 12 and the value of r is 2.

3. You know that in a dictionary all permutations of the letters that are used in the word AGAIN are arranged in a particular order. Find out the 49th word by using this information.

4. In the class, there are 27 boys and 14 girls. The teacher wants to select 1 boy and 1 girl to represent the class for a function. In how many ways can the teacher make this selection?

5. Let n be a positive integer. Explain why
   $1 + 3 + 5 + \cdots + 2n - 1 = n^2$.

6. Calculate the number of ways a cricket eleven can be selected out of a batch of 15 players if;

a. no restriction on the selection.

b. A specific player is always selected.

c. A specific player is never chosen.

Reference:

Applied Combinatorics2017 Edition, Mitchel T. Keller William T. Trotter

3

INDUCTION

Unit Structure

3.0 OBJECTIVE

Mathematical Induction is a technique of proving a statement, theorem or formula which is thought to be true, for each and every natural number n.

By generalizing this in the form of a principle which we would use to prove any mathematical statement is 'Principle of Mathematical Induction'.

On completing this chapter, you should be able to:

- state the principle of mathematical induction
- prove formulas that are valid for all $n \in \mathbb{N}$ by using the principle of mathematical induction
- state the principle of inclusion and exclusion
- solve counting problems using the principle of inclusion and exclusion
- state and prove the result on the number of functions from a finite set onto another finite set
- state the pigeon-hole principle

- solve simple counting problems using the pigeon-hole principle
- to find the greatest common divisor in detail.

3.1 INTRODUCTION

- A professor decides to liven up the next combinatorics class by giving a door prize.
- As students enter class they draw a ticket from a box.
- On each ticket, a positive integer has been printed.
- No information about the range of ticket numbers is given, although they are guaranteed to be distinct.
- The box of tickets was shaken robustly before the drawing, so the contents are thoroughly mixed, and the selection is done without looking inside the box.
- After each student has selected a ticket, the professor announces that a cash prize of one dollar will be awarded to the student holding the lowest numbered ticket—from among those drawn.

3.2 THE POSITIVE INTEGERS ARE WELL ORDERED

Number theory studies the properties of integers. Some basic results in number theory rely on the existence of a certain number. The next theorem can be used to show that such a number exists.

Definition:

A set T of real numbers is said to be well-ordered if every nonempty subset of T has a smallest element.

Principle: Well Ordered Property of the Positive Integers.

Every non-empty set of positive integers has a least element.

Example:

Consider the sets

$$\begin{aligned} A &= \{n \in \mathbb{N} \mid n \text{ is a multiple of } 3\}, \\ B &= \{n \in \mathbb{N} \mid n = -11 + 7m \text{ for some } m \in \mathbb{Z}\}, \\ C &= \{n \in \mathbb{N} \mid n = x^2 - 8x + 12 \text{ for some } x \in \mathbb{Z}\}. \end{aligned}$$

It is easy to check that all three sets are nonempty, and since they contain only positive integers, the principle of well-ordering guarantees that each of them has a smallest element.

These smallest elements may not be easy to find.

It is obvious that the smallest element in A is 3.

To find the smallest element in B, we need $-11+7m>0$, which means $m>11/7\approx1.57$.

Since m has to be an integer, we need $m\geq2$.

Since $-11+7m$ is an increasing function in m, its smallest value occurs when $m=2$.

The smallest element in B is $-11+7\cdot2=3$.

To determine the smallest element in C, we need to solve the inequality $x2-8x+12>0$.

Factorization leads to $x2-8x+12=(x-2)(x-6)>0$, so we need $x<2$ or $x>6$.

Because $x\in Z$, we determine that the minimum value of $x2-8x+12$ occurs at $x=1$ or $x=7$.

Since,

$12-8\cdot1+12=72-8\cdot7+1=5$

The smallest element in C is 5.

An equivalent statement to the well-ordering principle is as follows:

The set of positive integers does not contain any infinite strictly decreasing sequences.

- A set of real numbers is said to be well-ordered if every nonempty subset in it has a smallest element.
- A well-ordered set must be nonempty and have a smallest element.
- Having a smallest element does not guarantee that a set of real numbers is well-ordered.
- A well-ordered set can be finite or infinite, but a finite set is always well-ordered.

3.3 THE MEANING OF STATEMENTS

Communication in mathematics requires more precision than many other subjects, and thus we should take a few pages here to consider the basic building blocks: **mathematical statements.**

A **statement** is any declarative sentence which is either true or false.

The first few terms of a sequence.

Example:

Find the answer for statement: 1+2+3+···+6 .

Solution:

First, let $f : \mathbb{N} \longrightarrow \mathbb{N}$ be a function.

Set $\sum_{i=1}^{1} f(i) = f(1)$

and if , n>1, define

$$\sum_{i=1}^{n} f(i) = f(n) + \sum_{i=1}^{n-1} f(i)$$

To see that these two statements imply that the expression $\sum_{i=1}^{n} f(i)$

is defined for all positive integers, apply the Well Ordered Property to the set of all positive integers for which the expression is not defined and use the recursive definition to define it for the least element.

So if we want to talk about the sum of the first six positive integers, then we should write: $\sum_{i=1}^{6} i$

Hence the answer of 1+2+3+···+6 is 21.

Example 2:

Solve to find the answer for n!

Solution:

We need to find answer for statement:

$$n! = n \times (n-1) \times (n-2) \times \cdots \times 3 \times 2 \times 1$$

Multiplication, like addition, is a binary operation.

Such examples are called **recursive definitions**.

Here's a way to do the job more precisely.

Like,

Define n!

If n= 1 , then n!=1

If n >1, then n!= n (n - 1)!

3.4 BINOMIAL COEFFICIENTS REVISITED

➔ The binomial coefficient $\binom{n}{k}$ was originally defined in terms of the factorial notation, and with our recursive definitions of the factorial notation, we also have a complete and legally-correct definition of binomial coefficients.

➔ The following recursive formula provides an efficient computational scheme.

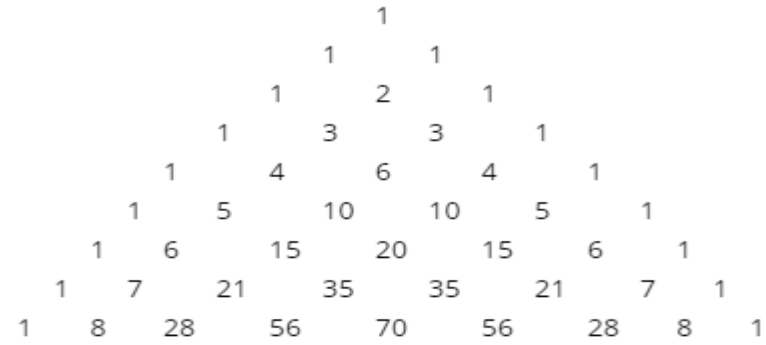
➔ Let n and k be integers with
$0 \leq k \leq n$. If $k = 0$ or $k = n$, set $\binom{n}{k} = 1$. If $0 < k < n$,

➔ Set, $\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}.$

➔ This recursion has a natural combinatorial interpretation.

➔ Both sides count the number of k-element subsets of ,{1,2,…,n}, with the right-hand side first grouping them into those which contain the element n and then those which don't.

➔ The traditional form of displaying this recursion is shown in following figure:



Pascal's Triangle

- ➔ This pattern is called “Pascal's triangle.”
- ➔ Other than the 1 s at the ends of each row, an entry of the triangle is determined by adding the entry to the left and the entry to the right in the row above.
- ➔ Xing was intrigued by the fact that he now had two fundamentally different ways to calculate binomial coefficients.
- ➔ One way is to write $\binom{n}{m} = P(n, m)/(n-m)!$ and just carry out the specified arithmetic.
- ➔ The second way is to use the recursion of Pascal's triangle, so that you are just performing additions.
- ➔ So he experimented by writing a computer program to calculate binomial coefficients, using a library that treats big integers as strings.

3.5 SOLVING COMBINATORIAL PROBLEMS RECURSIVELY

In this section, we present examples of combinatorial problems for which solutions can be computed recursively.

Example:

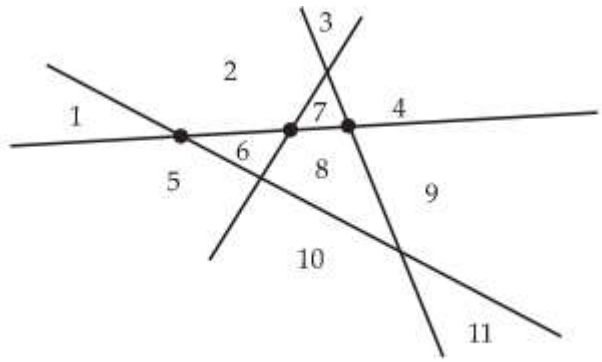
A family of n lines is drawn in the plane with condition as:

(1) each pair of lines crossing and

(2) no three lines crossing in the same point.

Solution:

- ➔ Let r(n) denote the number of regions into which the plane is partitioned by these lines.
- ➔ Evidently, r(1) = 2, r(2) = 4, r(3) = 7 and r(4) = 11.
- ➔ To determine r(n) for all positive integers, it is enough to note that:
- ➔ r(1) = 1, and
- ➔ when n > 1,
- ➔ r(n) = n + r(n - 1).
- ➔ This formula follows from the observation that if we label the lines as L1, L2,..., Ln, then the n - 1 points on line Ln where it crosses the other lines in the family divide Ln into n segments, two of which are infinite.
- ➔ Each of these segments is associated with a region determined by the first n - 1 lines that has now been subdivided into two, giving us n more regions than were determined by n-1 lines.
- ➔ This situation is illustrated in the following Figure, where the line containing the three dots is L4.



Lines and regions in the plane

➔ The other lines divide it into four segments, which then divide larger regions to create regions 1 and 5, 2 and 6, 7 and 8, and 4 and 9.

➔ With the recursive formula, we thus have

➔ r(5) = 5 + 11

= 16,

➔ r(6) = 6 + 16

= 22

and

➔ r(7) = 7 + 22

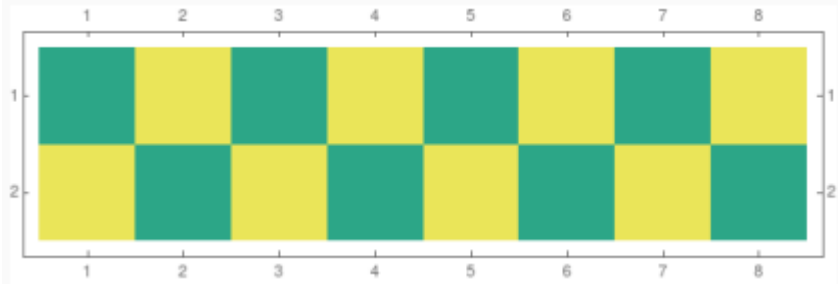
= 29.

Example

A 2 X n checkerboard will be tiled with rectangles of size 2 X 1 and 1 X 2.

Find a recursive formula for the number tⁱnº of tilings.

Solution:



➔ Clearly, t(1) = 1 and t(2) = 2.

➔ When n > 2, consider the rectangle that covers the square in the upper right corner.

➔ If it is vertical, then preceding it, we have a tiling of the first n - 1 columns.

➔ If it is horizontal, then so is the rectangle immediately underneath it, and proceeding them is a tiling of the first n - 2 columns.

➔ This shows that:

$t(n) = t(n - 1) + t(n - 2)$.

➔ In particular,

➔ $t(3) = 1 + 2$

$= 3$

➔ $t(4) = 2 + 3$

$=5$ and

➔ $t(5) = 3 + 5$

$=8$.

3.5.1 Finding Greatest Common Divisors:

- The greatest common divisor (GCD) of two or more numbers is the **greatest common factor number that divides them**, exactly.
- It is also called the highest common factor (HCF).
- For example, the greatest common factor of 15 and 10 is 5, since both the numbers can be divided by 5.
- What is Greatest Common Divisor?
  - For a set of positive integers (a, b), the greatest common divisor is defined as the greatest positive number which is a common factor of both the positive integers (a, b).
  - GCD of any two numbers is never negative or 0 as the least positive integer common to any two numbers is always 1.
  - There are two ways to determine the greatest common divisor of two numbers:
    - By finding the common divisors
    - By Euclid's algorithm
- How to Find the Greatest Common Divisor?
  - For a set of two positive integers (a, b) we use the below-given steps to find the greatest common divisor:
    - **Step 1:** Write the divisors of positive integer "a".
    - **Step 2:** Write the divisors of positive integer "b".
    - **Step 3:** Enlist the common divisors of "a" and "b".
    - **Step 4:** Now find the divisor which is the highest of both "a" and "b".

Example: Find the greatest common divisor of 13 and 48.

Solution:

➔We will use the below steps to determine the greatest common divisor of (13, 48).

➔Divisors of 13 are 1, and 13.

➔Divisors of 48 are 1, 2, 3, 4, 6, 8, 12, 16, 24 and 48.

➔The common divisor of 13 and 48 is 1.

➔The greatest common divisor of 13 and 48 is 1.

➔Thus, GCD(13, 48) = 1.

Division Theorem:

Let m and n be positive integers. Then there exist unique integers q and r so that:

m = q . n + r and 0 <= r < n:

We call q the quotient and r the remainder.

Proof:

➔ Let t be the least positive integer for which there are integers

m and n with m+n = t, but there do not exist integers q and r with m = qn+r

and 0<= r < n.

➔ First, we note that $n \neq 1$, for if n = 1, then we could take q = m and r = 0.

➔ Also, we cannot have m = 1, for if m = 1, then we can take q = 0 and r = 1.

➔ Now the statement holds for the pair m - 1, n so there are integers q and r so that

m - 1 = q . n + r and 0 <= r < n:

➔ Since r < n, we know that r + 1 n.

➔ If r + 1 < n, then

m = q . n + (r + 1) and 0 <= r + 1 < n:

➔ On the other hand, if r + 1 n, then

m = q . n + (r + 1)

= nq + n

= (q + 1)n

= (q + 1)n + 0:

➔ The contradiction completes the proof.

Theorem Euclidean Algorithm:

Let be positive integers with m > n and let q and r be the unique integers for which

$$m = q \cdot n + r \quad \text{and} \quad 0 \leq r < n.$$

and

If r > 0, then gcd(m,n) = gcd(n,r).

If r = 0 , then n divides m , and gcd(m,n)=n.

Proof:

➔Here is a code snippet that computes the greatest common divisor of m and n when m and n are positive integers with m>=n.

➔We use the familiar notation m%n to denote the remainder r in the expression

m=q . n + r, with 0 <= r < n.

➔

```
1  def gcd(m,n):
2      if m % n == 0:
3          return n
4      else:
5          return gcd(n,m%n)
6  gcd(12,5)
```

3.5.2 Sorting:

One of the most common and most basic computing problems is sorting:

Given a sequence $a_1,a_2,\ldots,a_n$ of n distinct integers, rearrange them so that they are in increasing order.

We describe here an easy recursive strategy for accomplishing this task.

This strategy is known as **Merge Sort**

It is one of several optimal algorithms for sorting.

To present merge sort, must first develop a strategy for solving a special case of the sorting problem. Suppose we have s+t distinct integers

$$\{u_0, u_1, \ldots, u_{s-1}, v_0, v_1, \ldots, v_{t-1}\}$$

arranged as two lists with $u_0 < u_1 < \cdots < u_{s-1}$ and $v_0 < v_1 < \cdots < v_{t-1}$.

```
u = [1,2,7,9,11,15]
v = [3,5,8,100,130,275]
a = []
p = 0
q = 0
for i in range(len(u) + len(v)):
    if (p < len(u) and q < len(v)):
        a.append(min(u[p], v[q]))
        if (min(u[p], v[q])==u[p]):
```

3.6 MATHEMATICAL INDUCTION

- An essential property of the set N = {1, 2, 3, ...} of positive integers follows:
- Principle of Mathematical Induction I:
- Let P be a proposition defined on the positive integers N; that is, P(n) is either true or false for each n $\in$ N.
- Suppose P has the following two properties:

  (i) P(1) is true.

  (ii) P (k + 1) is true whenever P (k) is true.
- Then P is true for every positive integer n $\in$ N.
- We shall not prove this principle. In fact, this principle is usually given as one of the axioms when N is developed axiomatically.

3.7 INDUCTIVE DEFINITIONS PROOFS BY INDUCTION

Principle of Mathematical Induction Solution and Proof:

Consider a statement P(n), where n is a natural number. Then to determine the validity of P(n) for every n, use the following principle:

Step 1: Check whether the given statement is true for n = 1.

Step 2: Assume that given statement P(n) is also true for n = k, where k is any positive integer.

Step 3: Prove that the result is true for P(k+1) for any positive integer k.

If the above-mentioned conditions are satisfied, then it can be concluded that P(n) is true for all n natural numbers.

Proof:

➔ The first step of the principle is a *factual statement* and the second step is a *conditional one*.

➔ According to this if the given statement is true for some positive integer k only then it can be concluded that the statement P(n) is valid for n = k + 1.

➔ This is also known as the *inductive step* and the assumption that P(n) is true for n=k is known as the *inductive hypothesis.*

Example:

Show that $1 + 3 + 5 + \ldots + (2n-1) = n^2$

Solution:

Step 1:

Result is true for n = 1

That is $1 = (1)^2$ (True)

Step 2:

Assume that result is true for n = k

$1 + 3 + 5 + \ldots + (2k-1) = k^2$

Step 3:

for n = k + 1

i.e. $1 + 3 + 5 + \ldots + (2(k+1)-1) = (k+1)^2$

We can write the above equation as,

$1 + 3 + 5 + \ldots + (2k-1) + (2(k+1)-1) = (k+1)^2$

Using step 2 result, we get

$k^2 + (2(k+1)-1) = (k+1)^2$

$k^2 + 2k + 2 - 1 = (k+1)^2$

$k^2 + 2k + 1 = (k+1)^2$

$(k+1)^2 = (k+1)^2$

L.H.S. and R.H.S. are same.

So the result is true for n = k+1

By mathematical induction, the statement is true.

We see that the given statement is also true for n=k+1.

Hence we can say that by the principle of mathematical induction this statement is valid for all natural numbers n.

3.8 STRONG INDUCTION

The principle of strong induction states that

if for some property P(n), we have that P(0) is true and

For any $n \in \mathbb{N}$ with $n \neq 0$, if P(0), P(1), …, and P(n – 1) are true, then P(n) is true then

For any $n \in \mathbb{N}$, P(n) is true.

- **Weak induction (regular induction)** is good for showing that some property holds by incrementally adding in one new piece.
- **Strong induction** is good for showing that some property holds by breaking a large structure down into multiple small pieces.
- Any proof done by weak induction can be done by strong induction.
- It is never wrong to use strong induction.
- However, if you only need the immediately previous result, weak induction can be a lot cleaner.

Proof by Strong Induction

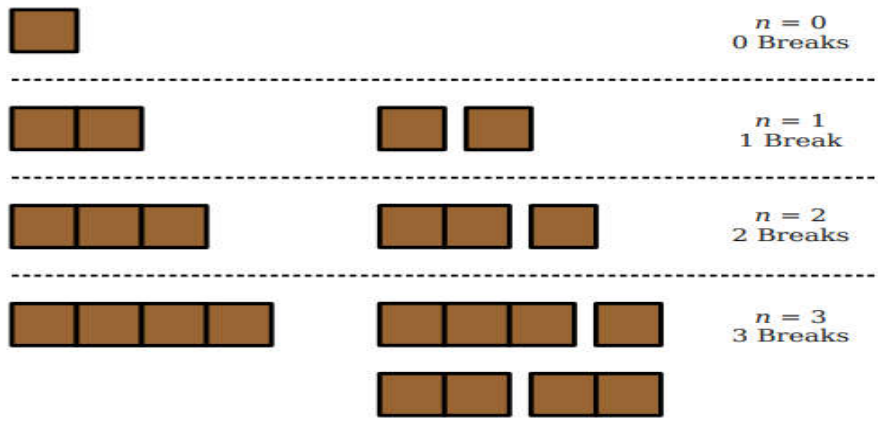
- State that you are attempting to prove something by strong induction.
- State what your choice of P(n) is.
- Prove the base case:
- State what P(0) is, then prove it.
- Prove the inductive step:
- State that you assume for all $0 \leq n' < n$, that P(n') is true.
- State what P(n) is. (this is what you're trying to prove)
- Go prove P(n).

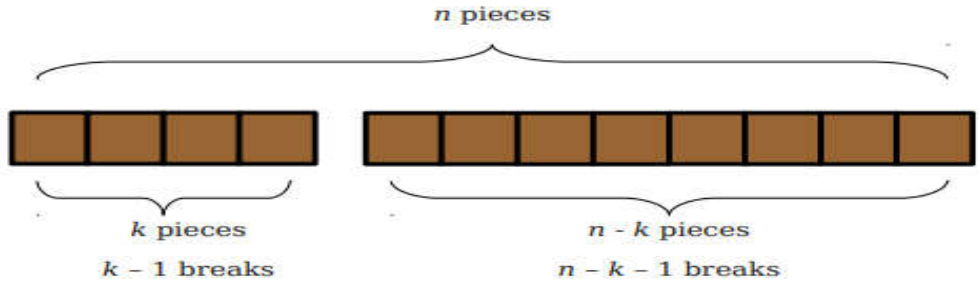
Example:

The Candy Bar Problem.

You are given a candy bar with $n \geq 1$ pieces.

How many breaks do you have to make to separate it into n individual pieces?





Theorem:

Breaking a chocolate bar with n ≥ 1 pieces into individual pieces requires n – 1 breaks.

Proof:

➔ By strong induction. Let P(n) be "breaking a chocolate bar with n ≥ 1 pieces into individual pieces requires n – 1 breaks."

➔ We prove P(n) holds for all n ∈ ℕ with n ≥ 1.

➔ For our base case, we prove P(1), that breaking a candy bar with one piece into individual pieces takes zero breaks.

➔ Since the candy bar is already in individual pieces, no breaks are required, so P(1) holds.

➔ For the inductive step, assume that for some n ∈ ℕ with n ≥ 1, that for any n' ∈ ℕ with 1 ≤ n' < n, that P(n') is true and any candy bar with n' pieces requires n' – 1 breaks.

➔ We will prove P(n), that breaking any candy bar with n pieces requires n – 1 breaks.

➔ To see this, consider any possible break made in the candy bar.

➔ This will split the candy bar into two pieces, one of size k and one of size n – k, where 1 ≤ k < n.

➔ Note that 1 ≤ n – k < n as well, since the second piece is not empty and is not the entire candy bar.

➔ Since 1 ≤ k < n, by our inductive hypothesis, breaking the k-piece candy bar requires k – 1 breaks.

➔ Since 1 ≤ n – k < n, by our inductive hypothesis, breaking the (n–k)-piece candy bar takes n – k – 1 breaks.

➔ Counting all breaks, we need k – 1 + n – k – 1 + 1 = n – 1 breaks for the n-piece candy bar. Thus P(n) holds, completing the induction.

3.9 EXERCISE

Solve the following:

1. Find the smallest element in each of these subsets of N.
   a) $\{n \in N \mid n=m2-10m+28$ for some integer $m\}$.
   b) $\{n \in N \mid n=5q+3$ for some integer $q\}$.
   c) $\{n \in N \mid n=-150-17d$ for some integer $d\}$.
   d) $\{n \in N \mid n=4s+9t$ for some integers s and $t\}$.
2. Solve the following statement:

$$2+3+6+11+18+27+38+51+\cdots+(n^2-2n+3)$$

Reference :

Applied Combinatorics2017 Edition, Mitchel T. Keller William T. Trotter

4

GRAPH THEORY

Unit Structure

4.1 Basic Notation and Terminology for Graphs

4.2 Multigraphs- Loops and Multiple Edges

4.3 Eulerian and Hamiltonian Graphs

4.4 Graph Coloring

4.5 Planar Graphs

4.6 Counting Labeled Trees

4.1 BASIC NOTATION AND TERMINOLOGY FOR GRAPHS

A ***graph*** GG is a pair (V,E)(V,E) where VV is a set (almost always finite) and EE is a set of 2-element subsets of VV. Elements of VV are called ***vertices*** and elements of EE are called ***edges***. We call VV the ***vertex set*** of GG and EE is the ***edge set***. For convenience, it is customary to abbreviate the edge {x,y}{x,y} as just xyxy. Remember though that xy∈Exy∈E means exactly the same as yx∈Eyx∈E. If xx and yy are distinct vertices from V,xV,x and yy are ***adjacent*** when xy∈Exy∈E; otherwise, we say they are ***non-adjacent***. We say the edge xyxy is ***incident to*** the vertices xx and yy.

For example, we could define a graph G=(V,E)G=(V,E) with vertex set V={a,b,c,d,e}V={a,b,c,d,e} and edge set E={{a,b},{c,d}, {a,d}} E={{a,b},{c,d},{a,d}}. Notice that no edge is incident to ,e, which is perfectly permissible based on our definition. It is quite common to identify a graph with a visualization in which we draw a point for each vertex and a line connecting two vertices if they are adjacent. The graph GG we've just defined is shown in **Figure 2.1**. It's important to remember that while a drawing of a graph is a helpful tool, it is not the same as the graph. We could draw GG in any of several different ways without changing what it is as a graph.

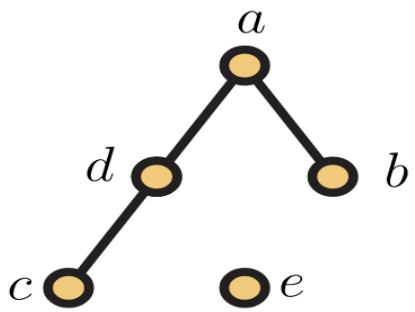


Figure **4.1.** A graph on 5 vertices

As is often the case in science and mathematics, different authors use slightly different notation and terminology for graphs. As an example, some use ***nodes*** and ***arcs*** rather than vertices and edges. Others refer to vertices as ***points*** and in this case, they often refer to ***lines*** rather than edges. We will try to stick to vertices and edges but confess that we may occasionally lapse into referring to vertices as points. Also, following the patterns of many others, we will also say that adjacent vertices are ***neighbors***. And we will use the more or less standard terminology that the ***neighborhood*** of a vertex xx is the set of vertices adjacent to xx. Thus, using the graph GG we have depicted in **Figure 4.1**, vertices dd and aa are neighbors, and the neighborhood of dd is {a,c}{a,c} while the neighborhood of ee is the empty set. Also, the ***degree*** of a vertex vv in a graph GG, denoted degG(v)degGf_0(v), is then the number of vertices in its neighborhood, or equivalently, the number of edges incident to it. For example, we have degG(d)=degG(a)=2,, degG(c)=degG(b)=1 degGf_0(d)=degG f_0(a)=2,,degGf_0(c)=degGf_0(b)=1, and deg G(e)=0degGf_0(e)=0. If the graph being discussed is clear from context, it is not uncommon to omit the subscript and simply write deg(v)degf_0(v) for the degree of vv.

When **G** =(V,E)=(V,E) and **H** =(W,F)=(W,F) are graphs, we say **H** is a ***subgraph*** of **G** when W⊆VW⊆V and F⊆EF⊆E.We say **H** is an ***induced subgraph*** when W⊆VW⊆V and F={xy∈E:x,y∈W} F={xy∈E:x,y∈W}. In other words, an induced subgraph is defined completely by its vertex set and the original graph **G**. We say **H** is a *spanning subgraph* when W=VW=V. In **Figure 4.2**, we show a graph, a subgraph and an induced subgraph. Neither of these subgraphs is a spanning subgraph.

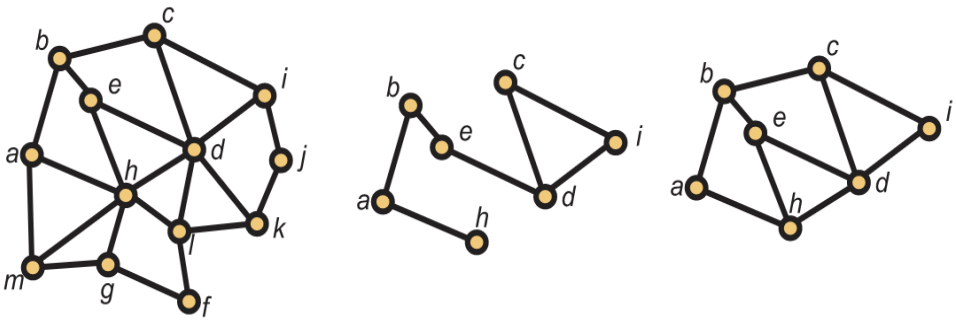


Figure **4.2.** A Graph, a Subgraph and an Induced Subgraph

A graph **G** =(V,E)=(V,E) is called a ***complete graph*** when xyxy is an edge in **G** for every distinct pair x,y∈Vx,y∈V. Conversely, **G** is an ***independent graph*** if xy∈Exy∈E, for every distinct pair x,y∈Vx,y∈V. It is customary to denote a complete graph on nn vertices by KnKn and an independent graph on nn vertices by InIn. In **Figure 4.3**, we show the complete graphs with at most 5 vertices.

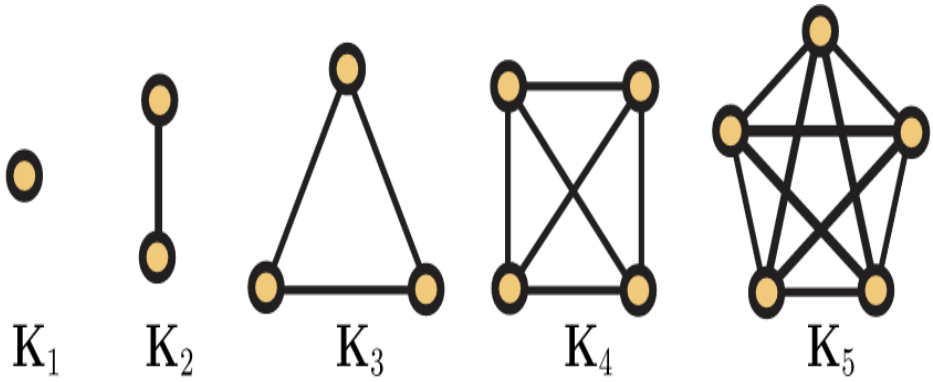


Figure **4.3.** Small complete graphs

A sequence (x1,x2,…,xn)(x1,x2,…,xn) of vertices in a graph **G**=(V,E) (V,E) is called a ***walk*** when xixi+1xixi+1 is an edge for each i=1,2,…, n−1i=1,2,…,n−1. Note that the vertices in a walk need not be distinct. On the other hand, if the vertices are distinct, then the sequence is called a ***path***, and often to emphasize where a path starts and ends, we will say that a sequence (x1,x2,…,xn)(x1,x2,…,xn) of distinct vertices is a path from x1x1 to xnxn in **G**. Similarly, when n≥3n≥3, a path (x1,x2,…,xn) (x1,x2,…,xn) of nn distinct vertices is called a ***cycle*** when x1xnx1xn is also an edge in **G**. It is customary to denote a path on n vertices by PnPn, while CnCn denotes a cycle on nn vertices. The ***length*** of a path or a cycle is the number of edges it contains. Therefore, the length of PnPn is n−1n−1 and the length of CnCn is nn. In **Figure 4.4**, we show the paths of length at most 4, and in **Figure 4.5**, we show the cycles of length at most 2.

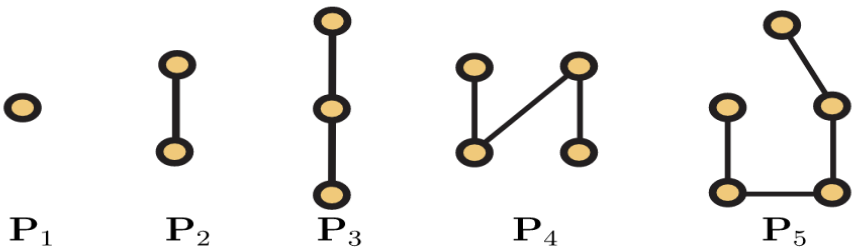


Figure **2.4.** Short paths

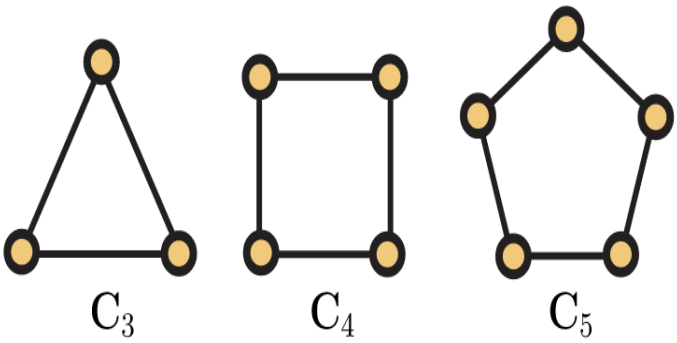


Figure **4.2.** Small cycles

If G=(V,E)G=(V,E) and H=(W,F)H=(W,F) are graphs, we say GG is ***isomorphic*** to HH and write G≅HG≅H when there exists a bijection f: V−→−onto1−1Wf:V→onto1−1W so that xx is adjacent to yy in GG if and only if f(x) f(x) is adjacent to f(y)f(y) in HH. Often writers will say that GG "contains" HH when there is a subgraph of GG which is isomorphic to HH. In particular, it is customary to say that GG contains the cycle CnCn (same for PnPn and KnKn) when GG contains a subgraph isomorphic to CnCn. The graphs in **Figure 4.6** are isomorphic. An isomorphism between these graphs is given by

f(a)=5,f(b)=3,f(c)=1,f(d)=6,f(e)=2,f(h)=4f(a)=5,f(b)=3,f(c)=1,f(d)=6,f(e)=2,f(h)=4.

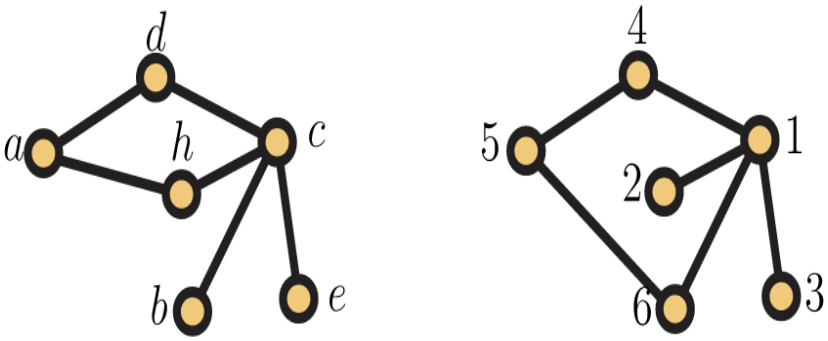


Figure **4.6.** A pair of isomorphic graphs

On the other hand, the graphs shown in **Figure 4.7** are *not* isomorphic, even though they have the same number of vertices and the same number of edges. Can you tell why?

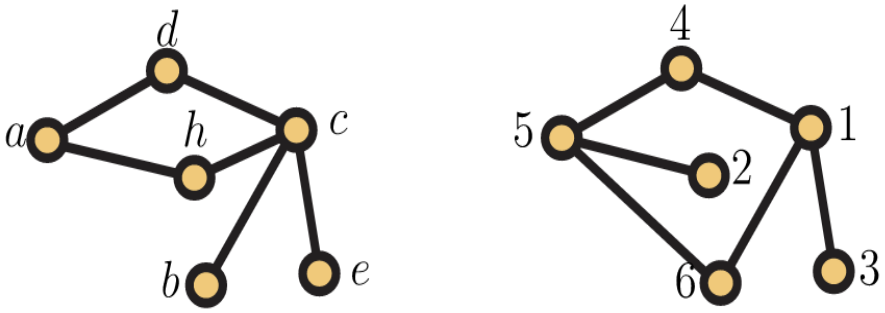


Figure **4.7.** A pair of nonisomorphic graphs

A graph GG is ***connected*** when there is a path from xx to yy in GG, for every x,y∈Vx,y∈V; otherwise, we say GG is ***disconnected***. The graph of **Figure 4.1** is disconnected (a sufficient justification for this is that there is no path from ee to cc), while those in **Figure 4.6** are connected. If GG is disconnected, we call a maximal connected subgraph of GG a ***component***. By this we mean that a subgraph HH of GG is a component of GG provided that there does not exist a connected subgraph H'H′ of GG such that HH is a subgraph of H′H′.

A graph is ***acyclic*** when it does not contain any cycle on three or more vertices. Acyclic graphs are also called ***forests***. A connected acyclic graph

is called a ***tree***. When G=(V,E) G=(V,E) is a connected graph, a subgraph H=(W,F)H=(W,F) of GG is called a ***spanning tree*** if HH is both a spanning subgraph of GG and a tree. In **Figure 4.8**, we show a graph and one of its spanning trees. We will return to the subject of spanning trees in Chapter 12.

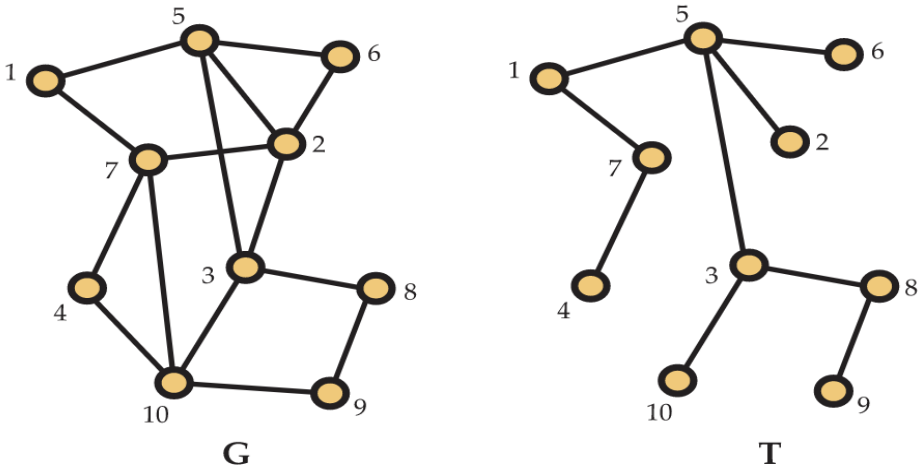


Figure **4.8.** A Graph and a Spanning Tree

The following theorem is very elementary, and some authors refer to it as the "first theorem of graph theory". However, this basic result can be surprisingly useful.

Theorm 4.9

Let degG(v)degG(v) *denote the degree of vertex* vv *in graph* G=(V,E) G=(V,E). *Then*

∑v∈VdegG(v)=2|E|∑v∈VdegG(v)=2|E|.

Proof

We consider how many times an edge e=vw∈ Ee=vw∈ E contributes to each side of **(4.1.1)**. The degG (x)deg G(x) and deg G(y)degG(y) terms on the left hand side each count ee once, so ee is counted twice on that side. On the right hand side, ee is clearly counted twice. Therefore, we have the equality claimed.

Corollary 4.10.

For any graph, the number of vertices of odd degree is even.

We will return to the topic of trees later, but before moving on, let us prove one elementary proposition about trees. First, a ***leaf*** in a tree **T** is a vertex vv with degT(v)=1degT(v)=1.

Our proof is by induction on nn. For n=2n=2, there is precisely one tree, which is isomorphic to K2K2. Both vertices in this graph are leaves, so the proposition holds for n=2n=2. Now suppose that for some integer m≥2m≥2, every tree on at most mm vertices has at least two leaves and let T=(V,E)T=(V,E) be a tree on m+1m+1 vertices. Pick an edge e∈Ee∈E

and form a new graph $T'=(V',E')T'=(V',E')$ by deleting ee from TT. That is, $V'=VV'=V$ and $E'=E-\{e\}E'=E-\{e\}$. Now since $T'T'$ does not contain a path from one endpoint of ee to its other endpoint, $T'T'$ is not connected. However, deleting an edge cannot create a cycle, so $T'T'$ is a forest. Furthermore, it has precisely two components, each of which is a tree with at most m vertices. If each component has at least two vertices, then by induction, each has at least two leaves. In the worst case scenario, two of these leaves are the endpoints of ee, so at least two of the vertices are leaves in TT, too. If each component of $T'T'$ has only one vertex, then $T\cong K2T\cong K2$, which has two leaves. If exactly one of the components has only one vertex, then it must be a leaf in TT. Thus, applying the inductive hypothesis to the other component ensures that there is a second leaf in TT.

4.2: MULTIGRAPHS- LOOPS AND MULTIPLE EDGES

Consider a graph in which the vertices represent cities and the edges represent highways. Certain pairs of cities are joined by an edge while other pairs are not. The graph may or may not be connected (although a disconnected graph is likely to result in disgruntled commuters). However, certain aspects of real highway networks are not captured by this model. First, between two nearby cities, there can actually be several interconnecting highways, and traveling on one of them is fundamentally different from traveling on another. This leads to the concept of ***multiple edges***, i.e., allowing for more than one edge between two adjacent vertices. Also, we could have a highway which leaves a city, goes through the nearby countryside and the returns to the same city where it originated. This leads to the concept of a ***loop***, i.e., an edge with both end points being the same vertex. Also, we can allow for more than one loop with the same end point.

Accordingly, authors frequently lead off a discussion on a graph theory topic with a sentence or two like:

1. In this paper, all graphs will be ***simple***, i.e., we will not allow loops or multiple edges.

2. In this paper, graphs can have loops and multiple edges.

The terminology is far from standard, but in this text, a graph will always be a ***simple*** graph, i.e., no loops or multiple edges. When we want to allow for loops and multiple edges, we will use the term ***multigraph***. This suggests the question of what we would call a graph if it is allowed to have loops but not multiple edges, or if multiple edges are allowed but not loops. If we *really* needed to talk about such graphs, then the English language comes to our rescue, and we just state the restriction explicitly!

4.3: EULERIAN AND HAMILTONIAN GRAPHS

Graph theory is an area of mathematics that has found many applications in a variety of disciplines. Throughout this text, we will encounter a

number of them. However, graph theory traces its origins to a problem in Königsberg, Prussia (now Kaliningrad, Russia) nearly three centuries ago. The river Pregel passes through the city, and there are two large islands in the middle of the channel. These islands were connected to the mainland by seven bridges as indicated in **Figure 4.12**. It is said that the citizens of Königsberg often wondered if it was possible for one to leave his home, walk through the city in such a way that he crossed each bridge precisely one time, and end up at home again. Leonhard Euler settled this problem in 1736 by using graph theory in the form of **Theorem 4.13**.

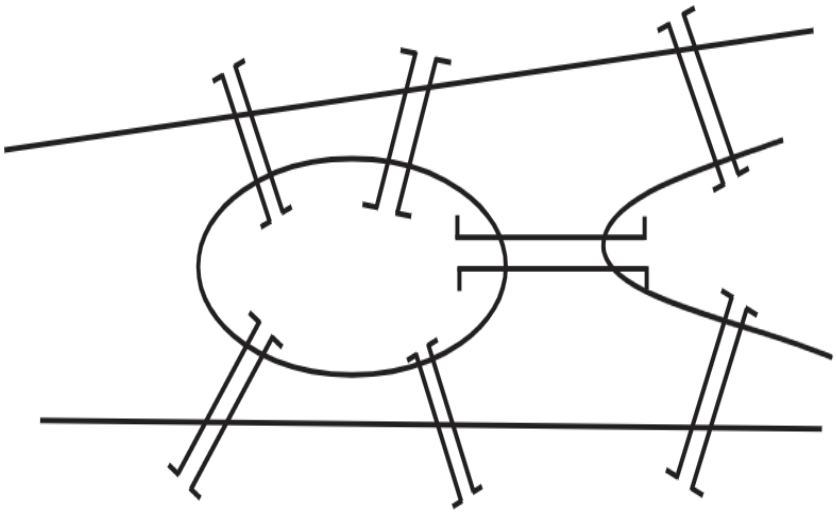


Figure **4.12.** The bridges of Königsberg

Let GG be a graph without isolated vertices. We say that GG is ***eulerian*** provided that there is a sequence (x0,x1,x2,…,xt) (x0,x1,x2,…,xt) of vertices from GG, with repetition allowed, so that

1. x0=xtx0=xt;
2. for every i=0,1,...,t−1i=0,1,...,t−1, xixi+1xixi+1 is an edge of GG;
3. for every edge e∈Ee∈E, there is a unique integer ii with 0≤i<t0≤i<t for which e=xixi+1e=xixi+1.

When GG is eulerian, a sequence satisfying these three conditions is called an ***eulerian circuit***. A sequence of vertices (x0,x1,…,xt)(x0,x1,…,xt) is called a ***circuit*** when it satisfies only the first two of these conditions. Note that a sequence consisting of a single vertex is a circuit. Before proceeding to Euler's elegant characterization of eulerian graphs, let's use SageMath to generate some graphs that are and are not eulerian.

Run the code below. It will execute until it finds a graph GG that is eulerian. The output that will be produced is a list of the degrees of the vertices of the graph GG followed by a drawing of GG.

// code 1

We encourage you to evaluate the run the code above multiple times, even changing the number of vertices and edges. If it seems to be running a log time, it may be that you have made the number of edges too small, so try increasing it a bit. Do you notice anything about the degrees of the vertices in the graphs produced?

Now let's try to find a graph HH that is *not* eulerian. Again, the output is the list of degrees of HH followed by a drawing of HH.

// code 2

One thing you probably noticed in running this second block of code is that it tended to come back much faster than the first. That would suggest that the non-eulerian graphs outnumber the eulerian graphs. Did you notice anything different about the degrees of the vertices in these graphs compared to the ones that were eulerian?

The following elementary theorem completely characterizes eulerian graphs. Its proof gives an algorithm that is easily implemented.

Theorem 4.13

A graph GG *is eulerian if and only if it is connected and every vertex has even degree.*

Proof

Clearly, an eulerian graph must be connected. Also, if (x0,x1,…,xt) (x0,x1,…,xt) is an eulerian circuit in GG, then for each i=0,1,…, t−1i=0,1,…,t−1, we can view the edge xixi+1xixi+1 as exiting xixi and entering xi+1xi+1. The degree of every vertex must be even, since for each vertex xx, the number of edges exiting xx equals the number of edges entering xx. Furthermore, each edge incident with xx either exits from xx or enters xx.

We now describe a deterministic process that will either (a) find an eulerian circuit, (b) show that the graph is disconnected, or (c) find a vertex of odd degree. The description is simplified by assuming that the vertices in GG have been labelled with the positive integers 1,2,…, n1,2,…,n, where nn is the number of vertices in GG. Furthermore, we take x0=1x0=1.

We launch our algorithm with a trivial circuit CC consisting of the vertex x0=(1)x0=(1). Thereafter suppose that we have a partial circuit CC defined by (x0,x1,…,xt)(x0,x1,…,xt) with x0=xt=1x0=xt=1. The edges of the form xixi+1xixi+1 have been ***traversed***, while the remaining edges in GG (if any) have not. If the third condition for an euler circuit is satisfied, we are done, so we assume it does not hold.

We then choose the least integer ii for which there is an edge incident with xixi that has not already been traversed. If there is no such integer,

since there are edges that have not yet been traversed, then we have discovered that the graph is disconnected. So we may assume that the integer ii exists. Set u0=xiu0=xi. We define a sequence (u0,u1,…,us) (u0,u1,…,us) recursively. If j≥0j≥0, set

Nj={y:ujyNj={y:ujy is an edge in **G** and has not yet been traversed.}}

If Nj≠\0Nj≠\0, we take uj+1uj+1 as the least positive integer in NjNj. If Nj=0Nj=0, then j≥1j≥1 and we take s=js=j and halt this subroutine.

When the subroutine halts, we consider two cases. If u0 nequs, then (u0u0 nequs,then\(u0 and usus are vertices of odd degree in GG. So we are left to consider the case where u0=us=xiu0=us=xi. In this case, we simply expand our original sequence (x0,x1,...,xt)(x0,x1,...,xt) by replacing the integer xixi by the sequence (u0,u1,...,us)(u0,u1,...,us).

As an example, consider the graph GG shown in **Figure 4.14**. Evidently, this graph is connected and all vertices have even degree. Here is the sequence of circuits starting with the trivial circuit CC consisting only of the vertex 1.

begin{alligned}

C &= (1)

&= (1,2,4,3,1) text{start next from 2}

&=(1,2,5,8,2,4,3,1) text{start next from 4}

&=(1,2,5,8,2,4,6,7,4,9,6,10,4,3,1) text{start next from 7}

&=(1,2,5,8,2,4,6,7,9,11,7,4,9,6,10,4,3,1) text{Done!!}

end{alligned}

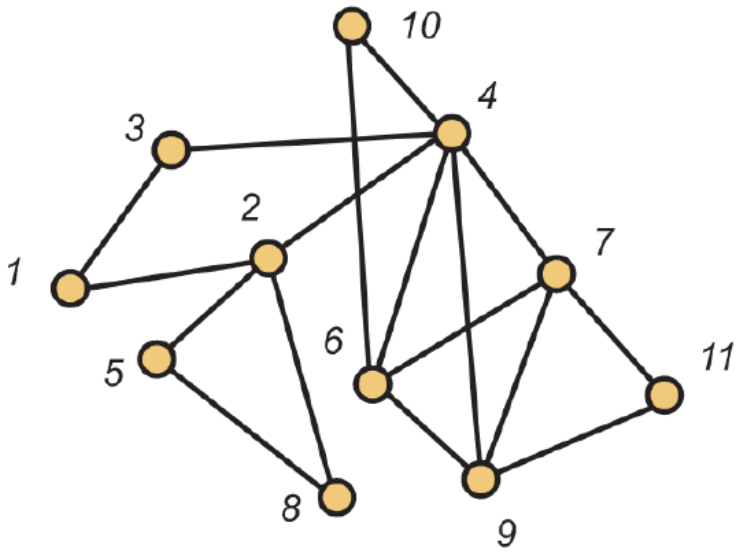


Figure 4.14. An Eulerian Graph

You should note that **Theorem 4.13** holds for loopless graphs in which multiple edges are allowed. Euler used his theorem to show that the multigraph of Königsberg shown in **Figure 4.15**, in which each land mass is a vertex and each bridge is an edge, is *not* eulerian, and thus the citizens could not find the route they desired. (Note that in **Figure 4.15** there are multiple edges between the same pair of vertices.)

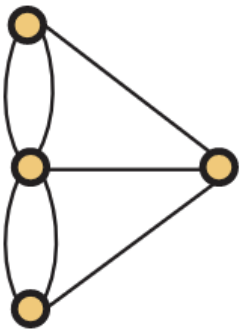


Figure 4.15 The multigraph of Königsberg's bridges

A graph G=(V,E)G=(V,E) is said to be ***hamiltonian*** if there exists a sequence (x1,x2,…,xn)(x1,x2,…,xn) so that

1. every vertex of GG appears exactly once in the sequence
2. x1xnx1xn is an edge of GG
3. for each i=1,2,...,n−1,xixi+1i=1,2,...,n−1,xixi+1 is an edge in GG.

Such a sequence of vertices is called a ***hamiltonian cycle***.

The first graph shown in **Figure 4.16** both eulerian and hamiltonian. The second is hamiltonian but not eulerian.

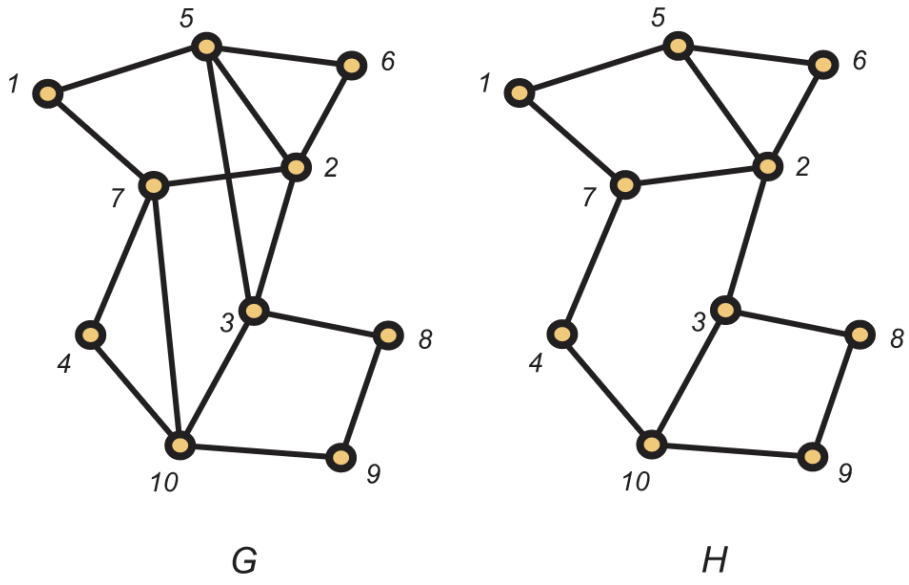


Figure 4.16. Eulerian and Hamiltonian Graphs

In **Figure 4.17**, we show a famous graph known as the Petersen graph. It is not hamiltonian.

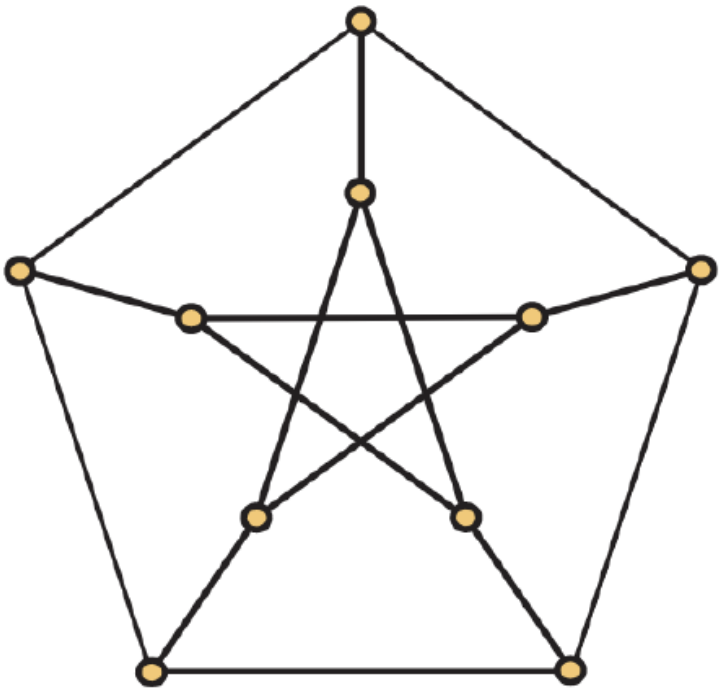


Figure **4.17.** The Petersen Graph

Unlike the situation with eulerian circuits, there is no known method for quickly determining whether a graph is hamiltonian. However, there are a number of interesting conditions which are sufficient. Here is one quite well known example, due to Dirac.

Theorem 4.18

If GG *is a graph on* nn *vertices and each vertex in* GG *has at least* ⌈n2⌉⌈n2⌉ *neighbors, then* GG *is hamiltonian.*

Proof

Suppose the theorem fails and let nn be the least positive integer for which there exists a graph GG on nn vertices so that each vertex in GG has at least ⌈n/2⌉⌈n/2⌉ neighbors, yet there is no hamiltonian cycle in GG. Clearly, n≥4n≥4.

Now let tt be the largest integer for which GG has a path P=(x1,x2,…,xt) P=(x1,x2,…,xt) on tt vertices. Clearly all neighbors of both x1x1 and xtxt appear on this path. By the pigeon hole principle, there is some integer ii with 1≤i<t1≤i<t so that x1xi+1x1xi+1 and xixtxixt are edges in GG. However, this implies that

C=(x1,x2,x3,...,xi,xt,xt−1,xt−2,...,xi+1)C=(x1,x2,x3,...,xi,xt,xt−1,xt−2,...,xi+1)

is a cycle of length tt in GG. In turn, this requires $\lceil n/2 \rceil < t < n \lceil n/2 \rceil < t < n$. But if yy is any vertex not on the cycle, then yy must have a neighbor on CC, which implies that GG has a path on $t+1t+1$ vertices. The contradiction completes the proof.

4.4: GRAPH COLORING

Let's return now to the subject of **Example 1.5**, assigning frequencies to radio stations so that they don't interfere. The first thing that we will need to do is to turn the map of radio stations into a suitable graph, which should be pretty natural at this juncture. We define a graph $G=(V,E)$ $G=(V,E)$ in which VV is the set of radio stations and $xy \in E xy \in E$ if and only if radio station xx and radio station yy are within 200 miles of each other. With this as our model, then we need to assign different frequencies to two stations if their corresponding vertices are joined by an edge. This leads us to our next topic, coloring graphs.

When $G=(V,E)G=(V,E)$ is a graph and CC is a set of elements called ***colors***, a ***proper coloring*** of GG is a function $\phi: V \to C \phi: V \to C$ such that if $\phi(x) \neq \phi(y) \phi(x) \neq \phi(y)$ whenever xyxy is an edge in GG. The least tt for which GG has a proper coloring using a set CC of tt colors is called the ***chromatic number*** of GG and is denoted $\chi(G) \chi(G)$. In **Figure 4.19**, we show a proper coloring of a graph using 5 colors. Now we can see that our radio frequency assignment problem is the much-studied question of finding the chromatic number of an appropriate graph.

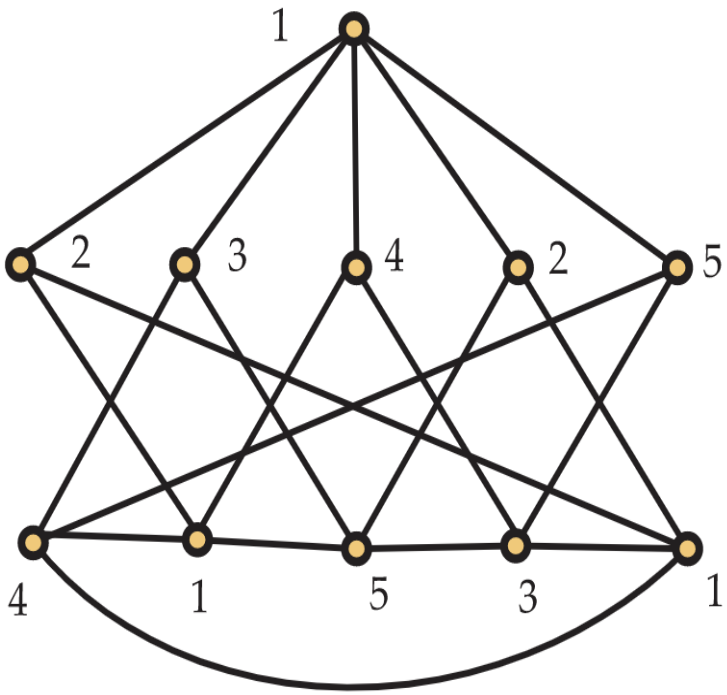


Figure **4.19.** A proper coloring using 5 colors

Discussion 4.20.

Everyone agrees that the graph GG in **Figure 4.19** has chromatic number at most 2. However, there's a bit of debate going on about if $\chi(G)=5\chi(G)=2$. Bob figures the authors would not have used five colors if they didn't need to. Carlos says he's glad they're having the discussion, since all having a proper coloring does is provide them with an upper bound on $\chi(G)\chi(G)$. Bob sees that the graph has a vertex of degree 5 and claims that must mean $\chi(G)=5\chi(G)=2$. Alice groans and draws a graph with 101 vertices, one of which has degree 100, but with chromatic number 2. Bob is shocked, but agrees with her. Xing wonders if the fact that the graph does not contain a K3K3 has any bearing on the chromatic number. Dave's in a hurry to get to the gym, but on his way out the door he says they can get a proper 4-coloring pretty easily, so $\chi(G)\leq4\chi(G)\leq4$. The rest decide it's time to keep reading.

- What graph did Alice draw that shocked Bob?
- What changed did Dave make to the coloring in **Figure 4.19** to get a proper coloring using four colors?

4.4.1 Bipartite Graphs

A graph $G=(V,E)G=(V,E)$ with $\chi(G)\leq2\chi(G)\leq2$ is called a ***2-colorable graph***. A couple of minutes of reflection should convince you that for $n\geq2n\geq2$, the cycle $C2nC2n$ with $2n2n$ vertices is 2-colorable. On the other hand, $C3\cong K3C3\cong K3$ is clearly not 2-colorable. Furthermore, no odd cycle $C2n+1C2n+1$ for $n\geq1n\geq1$ is 2-colorable. It turns out that the property of containing an odd cycle is the only impediment to being 2-colorable, which means that recognizing 2-colorable graphs is easy, as the following theorem shows.

Theorem 4.21

A graph is 2-colorable if and only if it does not contain an odd cycle.

Proof

Let $G=(V,E)G=(V,E)$ be a 2-colorable graph whose coloring function partitions VV as $A\cup BA\cup B$. Since there are no edges between vertices on the same side of the partition, any cycle in GG must alternate vertices between AA and BB. In order to complete the cycle, therefore, the number of vertices in the cycle from AA must be the same as the number from BB, implying that the cycle has even length.

Now suppose that GG does not contain an odd cycle. Note that we may assume that GG is connected, as each component may be colored individually. The ***distance*** $d(u,v)d(u,v)$ between vertices $u,v\in Vu,v\in V$ is the length of a shortest path from uu to vv, and of course $d(u,u)=0d(u,u)=0$. Fix a vertex $v0\in Vv0\in V$ and define

$A=\{v\in V:d(u0,v)A=\{v\in V:d(u0,v)$ is even$\}$
and $B=\{v\in V:d(v0,v)B=\{v\in V:d(v0,v)$ is odd$\}$.

We claim that coloring the vertices of AA with color 1 and the vertices of BB with color 2 is a proper coloring. suppose not. Then without loss of generality, there are vertices x,y∈Ax,y∈A such that xy∈Exy∈E. Since x,y∈Ax,y∈A, d(v0,x)d(v0,x) and d(v0,y)d(v0,y) are both even. Let

v0,x1,x2,...,xn=xv0,x1,x2,...,xn=x

and

v0,y1,y2,...,ym=yv0,y1,y2,...,ym=y

be shortest paths from v0v0 to xx and yy, respectively. If x1≠yjx1≠yj for all 1≤i≤n1≤i≤n and 1≤j≤m1≤j≤m, then since mm and nn are both even,

v0,x1,x2,...,xn=x,y=ym,ym−1,...,y2,y1,v0v0,x1,x2,...,xn=x,y=ym,ym−1,..., y2,y1,v0

is an odd cycle in GG, which is a contradiction. Thus, there must be i,ji,j such that xi=yjxi=yj, and we may take i,ji,j as large as possible. (That is, after xi=yjxi=yj, the two paths do not intersect again.) Thus,

xi,xi+1,...,xn=x,y=ym,ym−1,...,yj=xixi,xi+1,...,xn=x,y=ym,ym−1,...,yj=xi

is a cycle in GG. How many vertices are there in this cycle? A quick count shows that it has

n−(i−1)+m−(j−1)−1=n+m−(i+j)+1n−(i−1)+m−(j−1)−1=n+m−(i+j)+1

vertices. We know that n and m are even, and notice that ii and jj are either both even or both odd, since xi=yjxi=yj and the odd-subscripted vertices of our path belong to BB while those with even subscripts belong to AA. Thus, i+ji+j is even, so n+m−(i+j)+1n+m−(i+j)+1 is odd, giving a contradiction.

A graph GG is called a ***bipartite graph*** when there is a partition of the vertex VV into two sets AA and BB so that the subgraphs induced by AA and BB are independent graphs, i.e., no edge of GG has both of its endpoints in AA or in BB. Evidently, bipartite graphs are 2-colorable. On the other hand, when a 2-colorable graph is disconnected, there is more than one way to define a suitable partition of the vertex set into two independent sets.

Bipartite graphs are commonly used as models when there are two distinct types of objects being modeled and connections are only allowed between two objects of different types. For example, on one side, list candidates who attend a career fair and on the other side list the available positions. The edges might naturally correspond to candidate/position pairs which link a person to a responsibility they are capable of handling.

As a second example, a bipartite graph could be used to visualize the languages spoken by a group of students. The vertices on one side would be the students with the languages listed on the other side. We would then have an edge xyxy when student xx spoke language yy. A concrete example of this graph for our favorite group of students is shown

in **Figure 4.22**, although Alice isn't so certain there should be an edge connecting Dave and English.

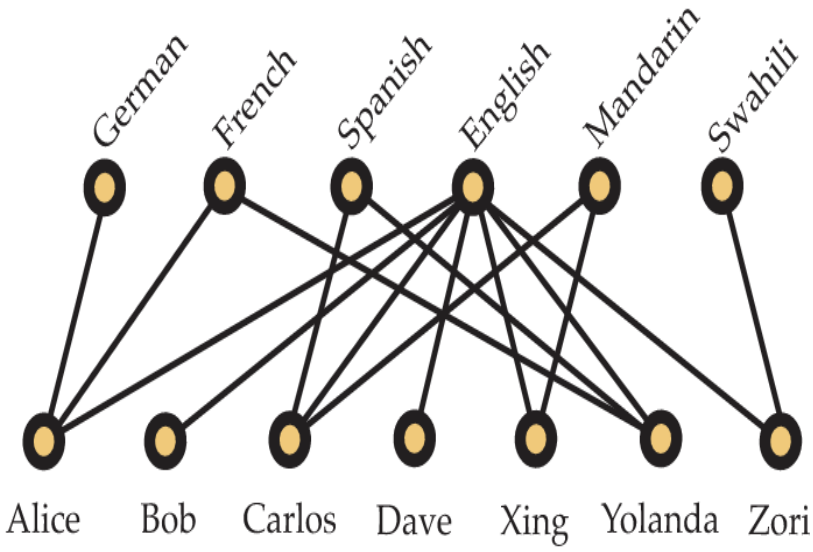


Figure **4.22.** A bipartite graph

One special class of bipartite graphs that bears mention is the class of ***complete bipartite graphs***. The complete bipartite graph Km,nKm,n has vertex set V=V1∪V2V=V1∪V2 with |V1|=m|V1|=m and |V2|=n|V2|=n. It has an edge xyxy if and only if x∈V1x∈V1 and y∈V2y∈V2. The complete bipartite graph K3,3K3,3 is shown in **Figure 4.23**.

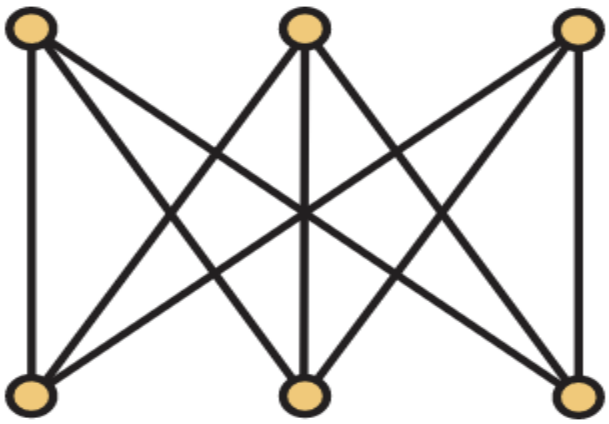


Figure **4.23.** The complete bipartite graph K3,3K3,3

4.4.2 Cliques and Chromatic Number

A ***clique*** in a graph G=(V,E)G=(V,E) is a set K⊂VK⊂V such that the subgraph induced by KK is isomorphic to the complete graph K|K|K|K|. Equivalently, we can say that every pair of vertices in KK are adjacent. The ***maximum clique size*** or ***clique number*** of a graph GG, denoted ω(G)ω(G), is the largest tt for which there exists a clique KK with |K|=t|K|=t. For example, the graph in **Figure 4.14** has clique number 4 while the graph in **Figure 2.19** has maximum clique size 2.

For every graph GG, it is obvious that $\chi(G) \geq \omega(G)\chi(G) \geq \omega(G)$. On the other hand, the inequality may be far from tight. Before showing how bad it can be, we need to introduce a more general version of the **Pigeon Hole Principle**. Consider a function $f:X \rightarrow Yf:X \rightarrow Y$ with $|X|=2|Y|+1|X|=2|Y|+1$. Since $|X|>|Y||X|>|Y|$, the Pigeon Hole Principle as stated in **Proposition 4.1** only tells us that there are distinct $x,x' \in Xx,x' \in X$ with $f(x)=f(x')f(x)=f(x')$. However, we can say more here. Suppose that each element of YY has at most two elements of XX mapped to it. Then adding up the number of elements of XX based on how many are mapped to each element of YY would only allow XX to have (at most) $2|Y|2|Y|$ elements. Thus, there must be $y \in Yy \in Y$ so that there are three distinct elements $x,x',x'' \in Xx,x',x'' \in X$ with $f(x)=f(x')=f(x'')=yf(x)=f(x')=f(x'')=y$.
This argument generalizes to give the following version of the Pigeon Hole Principle:

Proposition 4.24. Generalized Pigeon Hole Principle

If $f:X \rightarrow Yf:X \rightarrow Y$ *is a function and* $|(|X| \geq (m-1)|Y|+1\backslash)$, *then there exists an element* $y \in Yy \in Y$ *and distinct elements* $x1,\ldots,xm \in Xx1,\ldots,xm \in X$ *so that* $f(xi)=yf(xi)=y$ *for* $i=1,...,mi=1,...,m$.

We are now prepared to present the following proposition showing that clique number and chromatic number need not be close at all. We give two proofs. The first is the work of J. Kelly and L. Kelly, while the second is due to J. Mycielski.

Proposition 4.22.

For every $t \geq 3t \geq 3$, *there exists a graph* $GtGt$ *so that* $\chi(Gt)=t\chi(Gt)=t$ *and* $\omega(Gt)=2\omega(Gt)=2$.

Proof

We proceed by induction on tt. For $t=3t=3$, we take $G3G3$ to be the cycle $C5C5$ on five vertices. Now assume that for some $t \geq 3t \geq 3$, we have determined the graph $GtGt$. Suppose that $GtGt$ has $ntnt$ vertices. Label the vertices of $GtGt$ as $x1,x2,\ldots,xntx1,x2,\ldots,xnt$. Construct $Gt+1Gt+1$ as follows. Begin with an independent set II of cardinality $t(nt-1)+1t(nt-1)+1$. For every subset SS of II with $|S|=nt|S|=nt$, label the elements of SS as $y1,y2,\ldots,ynty1,y2,\ldots,ynt$. For this particular $ntnt$-element subset attach a copy of $GtGt$ with $yiyi$ adjacent to $xixi$ for $i=1,2,\ldots,nti=1,2,\ldots,nt$. Vertices in copies of $GtGt$ for distinct $ntnt$-element subsets of II are nonadjacent, and a vertex in II has at most one neighbor in a particular copy of $GtGt$.

To see that $\omega(Gt+1)=2\omega(Gt+1)=2$, it will suffice to argue that $Gt+1Gt+1$ contains no triangle ($K3K3$). Since $GtGt$ is triangle-free, any triangle in $Gt+1Gt+1$ must contain a vertex of II. Since none of the vertices of II are adjacent, any triangle in $Gt+1Gt+1$ contains only one point of II. Since each vertex of II is adjacent to at most one vertex of any fixed copy of $GtGt$, if $y \in Iy \in I$ is part of a triangle, the other two vertices must come from distinct copies of $GtGt$. However, vertices in different

copies of GtGt are not adjacent, so $\omega(Gt+1)=2\omega(Gt+1)=2$. Notice that $\chi(Gt+1)\geq t\chi(Gt+1)\geq t$ since Gt+1Gt+1 contains GtGt. On the other hand, $\chi(Gt+1)\leq t+1\chi(Gt+1)\leq t+1$ since we may use tt colors on the copies of GtGt and a new color on the independent set II. To see that $\chi(Gt+1)=t+1\chi(Gt+1)=t+1$, observe that if we use only tt colors, then by the generalized Pigeon Hole Principle, there is an ntnt-element subset of II in which all vertices have the same color. Then this color cannot be used in the copy of GtGt which is attached to that ntnt-element subset.

Proof

We again start with G3G3 as the cycle C5C2. As before we assume that we have constructed for some $t\geq 3t\geq 3$ a graph GtGt with $\omega(Gt)=2\omega$ $(Gt)=2$ and $\chi(Gt)=t\chi(Gt)=t$. Again, label the vertices of GtGt as x1,x2,…, xntx1,x2,…,xnt. To construct Gt+1Gt+1, we now start with an independent set II, but now II has only ntnt points, which we label as y1,y2,…,ynty1,y2,…,ynt. We then add a copy of GtGt with yiyi adjacent to xjxj if and only if xixi is adjacent to xjxj. Finally, attach a new vertex zz adjacent to all vertices in II.

Clearly, $\omega(Gt+1)=2\omega(Gt+1)=2$. Also, $\chi(Gt+1)\geq t\chi(Gt+1)\geq t$, since it contains GtGt as a subgraph. Furthermore, $\chi(Gt+1)\leq t+1\chi(Gt+1)\leq t+1$, since we can color GtGt with colors from $\{1,2,\ldots,t\}\{1,2,\ldots,t\}$, use color $t+1t+1$ on the independent set II, and then assign color 1 to the new vertex zz. We claim that in fact $\chi(Gt+1)=t+1\chi(Gt+1)=t+1$. Suppose not. Then we must have $\chi(Gt+1)=t\chi(Gt+1)=t$. Let $\phi\phi$ be a proper coloring of Gt+1Gt+1. Without loss of generality, $\phi\phi$ uses the colors in $\{1,2,\ldots,t\}$ $\{1,2,\ldots,t\}$ and $\phi\phi$ assigns color tt to zz. Then consider the nonempty set SS of vertices in the copy of GtGt to which $\phi\phi$ assigns color tt. For each xixi in SS, change the color on xixi so that it matches the color assigned to yiyi by $\phi\phi$, which cannot be tt, as zz is colored tt. What results is a proper coloring of the copy of GtGt with only $t-1t-1$ colors since xixi and yiyi are adjacent to the same vertices of the copy of GtGt. The contradiction shows that $\chi(Gt+1)=t+1\chi(Gt+1)=t+1$, as claimed.

Since a 3-clique looks like a triangle, **Proposition 4.25** is often stated as "There exist triangle-free graphs with large chromatic number." As an illustration of the construction in the proof of Mycielski, we again refer to **Figure 4.19**. The graph shown is G4G4. We will return to the topic of graphs with large chromatic number in Section 11.6 where we show that are there graphs with large chromatic number which lack not only cliques of more than two vertices but also *cycles* of fewer than gg vertices for *any* value of gg. In other words, there is a graph GG with χ $(G)=106\chi(G)=106$ but no cycle with fewer than 10101010 vertices!

4.4.3 Can we Determine Chromatic Number?

Suppose you are given a graph GG. It's starting to look like it is not easy to find an algorithm that answers the question "Is $\chi(G)\leq t\chi(G)\leq t$?" It's easy to verify a certificate (a proper coloring using at most t colors), but how could you even find a proper coloring, not to mention one with the fewest number of colors? Similarly for the question "Is $?\omega(G)\geq k\omega(G)\geq k$?", it is

easy to verify a certificate. However, finding a maximum clique appears to be a very hard problem. Of course, since the gap between $\chi(G)\chi$ (G) and $\omega(G)\omega(G)$ can be arbitrarily large, being able to find one value would not (generally) help in finding the value of the other. No polynomial-time algorithm is known for either of these problems, and many believe that no such algorithm exists. In this subsection, we look at one approach to finding chromatic number and see a case where it does work efficiently.

A very naïve algorithmic way to approach graph coloring is the First Fit, or "greedy", algorithm. For this algorithm, fix an ordering of the vertex set $V=\{v1,v2,\ldots vn\}V=\{v1,v2,\ldots vn\}$. We define the coloring function $\phi\phi$ one vertex at a time in increasing order of subscript. We begin with $\phi(v1)=1\phi(v1)=1$ and then we define $\phi(vi+1)$ $\phi(vi+1)$ (assuming vertices $v1,v2,\ldots,viv1,v2,\ldots,vi$ have been colored) to be the least positive integer color that has not already been used on any of its neighbors in the set $\{v1,...vi\}\{v1,...vi\}$.

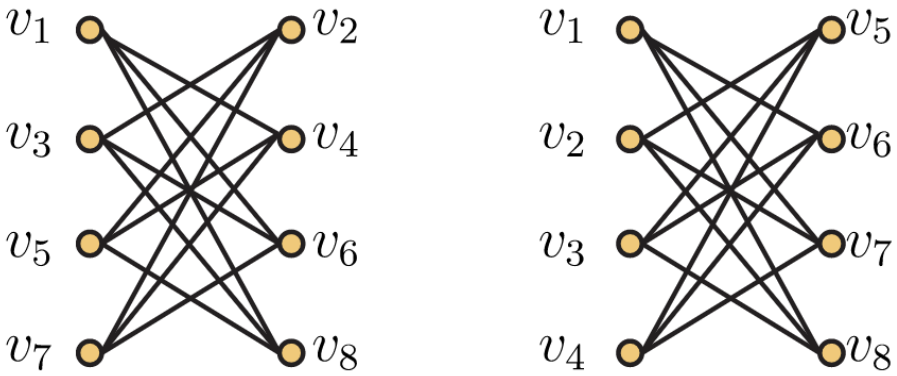


Figure **4.26.** Two orderings of the vertices of a bipartite graph.

Figure 2.26 shows two different orderings of the same graph. **Exercise 4.9.24** demonstrates that the ordering of VV is vital to the ability of the First Fit algorithm to color GG using $\chi(G)\chi(G)$ colors. In general, finding an optimal ordering is just as difficult as coloring GG. Thus, this very simple algorithm does not work well in general. However, for some classes of graphs, there is a "natural" ordering that leads to optimal performance of First Fit. Here is one such example—one that we will study again in the next chapter in a different context.

Given an indexed family of sets $F=\{S\alpha:\alpha\in V\}F=\{S\alpha:\alpha\in V\}$, we associate with FF a graph GG defined as follows. The vertex set of GG is the set VV and vertices xx and yy in VV are adjacent in GG if and only if $Sx\cap Sy\neq\emptyset Sx\cap Sy\neq\emptyset$. We call GG an ***intersection graph***. It is easy to see that every graph is an intersection graph (*Why?*), so it makes sense to restrict the sets which belong to FF. For example, we call GG an ***interval graph*** if it is the intersection graph of a family of closed intervals of the real line RR. For example, in **Figure 4.27**, we show a collection of six intervals of the real line on the left. On the right, we show the

corresponding interval graph having an edge between vertices xx and yy if and only if intervals xx and yy overlap.

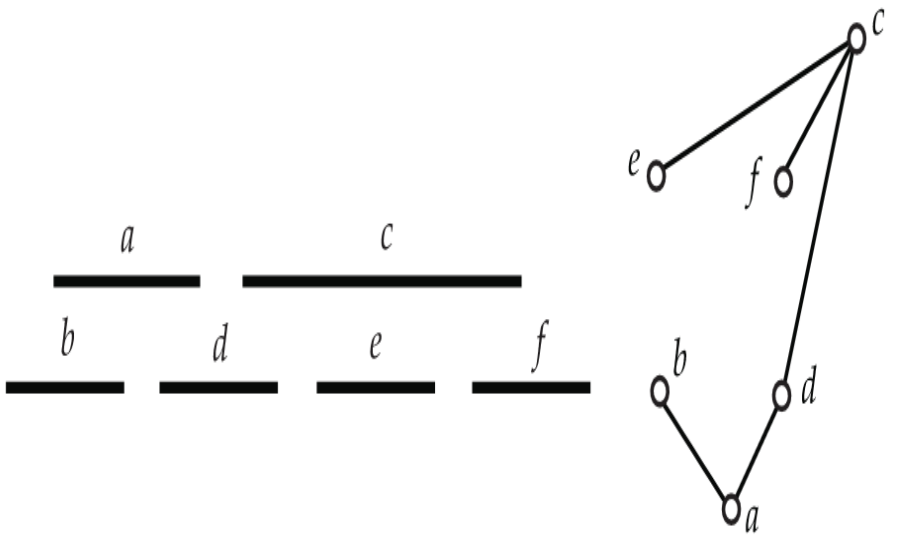


Figure **4.27.** A collection of intervals and its interval graph

Theorem 4.27

If G=(V,E)G=(V,E) *is an interval graph, then* χ(G)=ω(G)χ(G)=ω(G).

Proof

For each v∈Vv∈V, let I(v)=[av,bv]I(v)=[av,bv] be a closed interval of the real line so that uvuv is an edge in GG if and only if I(u)∩I(v)≠∅I(u)∩I(v)≠∅. Order the vertex set VV as {v1,v2,…vn}{v1,v2,…vn} such that a1≤a2≤⋯≤ana1≤a2≤⋯≤an. (Ties may be broken arbitrarily.) Apply the First Fit coloring algorithm to GG with this ordering on VV. Now when the First Fit coloring algorithm colors vivi, all of its neighbors have left end point at most aiai. Since they are neighbors of vivi, however, we know that their right endpoints are all at least aiai. Thus, vivi and its previously-colored neighbors form a clique. Hence, vivi is adjacent to at most ω(G)−1ω(G)−1 other vertices that have already been colored, so when the algorithm colors vivi, there will be a color from {1,2,…,ω(G)}{1,2,…,ω(G)} not already in use on its neighbors. The algorithm will assign vivi the smallest such color. Thus, we never need to use more than ω(G)ω(G) colors, so χ(G)=ω (G)χ (G)=ω(G).

A graph GG is said to be ***perfect*** if χ(H)=ω(H)χ(H)=ω(H) for every induced subgraph HH. Since an induced subgraph of an interval graph is an interval graph, **Theorem 4.28** shows interval graphs are perfect. The study of perfect graphs originated in connection with the theory of communications networks and has proved to be a major area of research in graph theory for many years now.

4.5: PLANAR GRAPHS

Let's return to the problem of providing lines for water, electricity, and natural gas to three homes which we discussed in the introduction to this chapter. How can we model this problem using a graph? The best way is to have a vertex for each utility and a vertex for each of the three homes. Then what we're asking is if we can draw the graph that has an edge from each utility to each home so that none of the edges cross. This graph is shown in **Figure 4.29**. You should recognize it as the complete bipartite graph K3,3K3,3 we introduced earlier in the chapter.

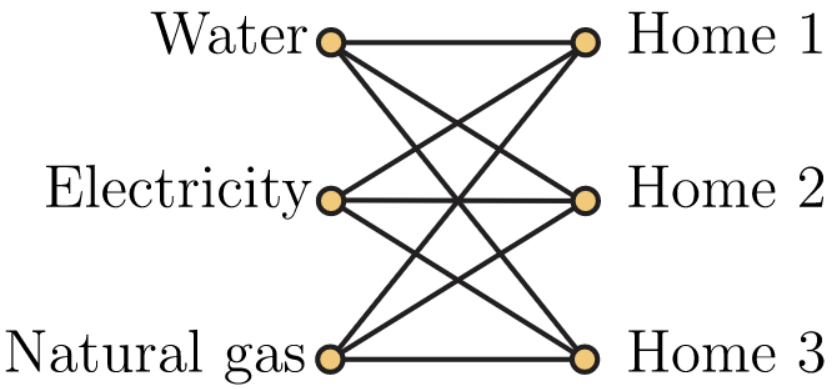


Figure **4.29.** A graph of connecting homes to utilities

While this example of utility lines might seem a bit contrived, since there's really no good reason that the providers can't bury their lines at different depths, the question of whether a graph can be drawn in the plane such that edges intersect only at vertices is a long-studied question in mathematics that does have useful applications. One area where it arises is in the design of microchips and circuit boards. In those contexts, the material is so thin that the option of placing connections at different depths either does not exist or is severely restricted. There is much deep mathematics that underlies this area, and this section is intended to introduce a few of the key concepts.

By a ***drawing*** of a graph, we mean a way of associating its vertices with points in the Cartesian plane R2R2 and its edges with simple polygonal arcs whose endpoints are the points associated to the vertices that are the endpoints of the edge. You can think of a polygonal arc as just a finite sequence of line segments such that the endpoint of one line segment is the starting point of the next line segment, and a simple polygonal arc is one that does not cross itself. (Our choice of polygonal arcs rather than arbitrary curves actually doesn't cause an impediment, since by taking very, very, very short line segments we can approximate any curve.) A ***planar drawing*** of a graph is one in which the polygonal arcs corresponding to two edges intersect only at a point corresponding to a vertex to which they are both incident. A graph is ***planar*** if it has a planar drawing. A ***face*** of a planar drawing of a graph is a region bounded by edges and vertices and not containing any other vertices or edges.

Figure 4.30 shows a planar drawing of a graph with 6 vertices and 9 edges. Notice how one of the edges is drawn as a true polygonal arc rather than a straight line segment. This drawing determines 5 regions, since we also count the unbounded region that surrounds the drawing.

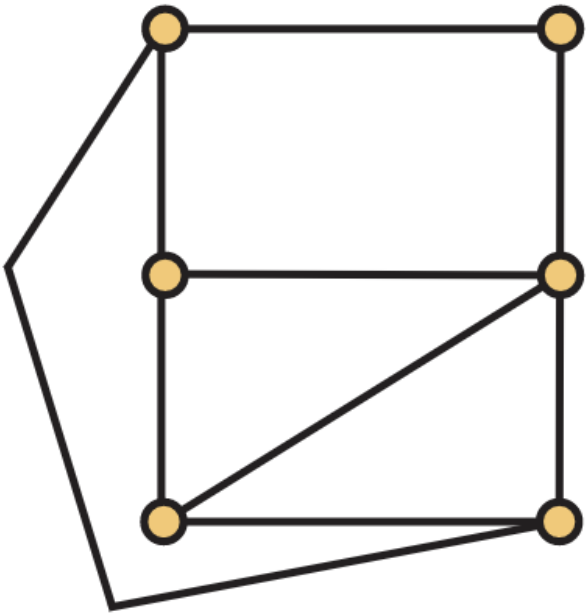


Figure **4.30.** A planar drawing of a graph

Figure 4.31 shows a planar drawing of the complete graph K4K4. There are 4 vertices, 6 edges, and 4 faces in the drawing.

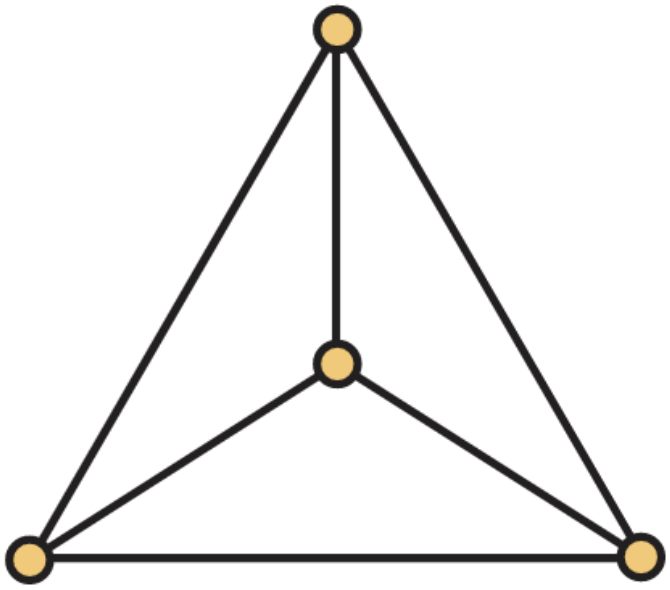


Figure **4.31.** A planar drawing of K4K4

What happens if we compute the number of vertices minus the number of edges plus the number of faces for these drawings? We have

6−9+5=26−9+5=2

4−6+4=24−6+4=2

While it might seem like a coincidence that this computation results in 2 for these planar drawings, there's a more general principle at work here, and in fact it holds for *any* planar drawing of *any* planar graph.

In fact, the number 2 here actually results from a fundamental property of the plane, and there are a corresponding theorems for other surfaces. However, we only need the result as stated above.

Theorem 4.32. Euler's Formula

Let GG *be a connected planar graph with* nn *vertices and* mm *edges. Every planar drawing of* GG *has* ff *faces, where* ff *satisfies*

n−m+f=2n−m+f=2

Proof

Our proof is by induction on the number mm of edges. If m=0m=0, then since GG is connected, our graph has a single vertex, and so there is one face. Thus n−m+f=1−0+1=2n−m+f=1−0+1=2 as needed. Now suppose that we have proven Euler's formula for all graphs with less than m edges and let GG have mm edges. Pick an edge ee of GG. What happens if we form a new graph G'G′ by deleting e from GG? If G'G′ is connected, our inductive hypothesis applies. Say that G'G′ has n'n′ vertices, m'm′ edges, and f'f′ faces. Then by induction, these numbers satisfy

n′−m′+f′=2n′−m′+f′=2.

Since we only deleted one edge, n'=nn′=n and m'=m−1m′=m−1. What did the removal of ee do to the number of faces? In G'G′ there's a new face that was formerly two faces divided by ee in GG. Thus, f'=f−1f′=f−1. Substituting these into n'−m'+f'=2n′−m′+f′=2, we have

n−(m−1)+(f−1)=2⟺n−m+f=2n−(m−1)+(f−1)=2⟺n−m+f=2.

Thus, if G'G′ is connected, we are done. If G'G′ is disconnected, however, we cannot apply the inductive assumption to G'G′ directly. Fortunately, since we removed only one edge, G'G′ has two components, which we can view as two connected graphs G1'G1′ and G2'G2′. Each of these has fewer than mm edges, so we may apply the inductive hypothesis to them. For i=1,2i=1,2, let ni'ni′ be the number of vertices of Gi',mi'Gi′,mi′ the number of edges of Gi',andfi'Gi′,andfi′ the number of faces of Gi'Gi′. Then by induction we have

n′1−m′1+f′1=2n1′−m1′+f1′=2 and n′2−m′2+f′2=2n2′−m2′+f2′=2.

Adding these together, we have

(n′1+n′2)−(m′1+m′2)+(f′1+f′2)=4(n1′+n2′)−(m1′+m2′)+(f1′+f2′)=4.

But now $n=n'1+n'2n=n1'+n2'$ and $m'1+m'2=m-1m1'+m2'=m-1$, so the equality becomes

$$n-(m-1)+(f'1+f'2)=4\Leftrightarrow n-m+(f'1+f'2)=3n-(m-1)+(f1'+f2')=4\Leftrightarrow n-m+(f1'+f2')=3.$$

The only thing we have yet to figure out is how $f1'+f2'f1'+f2'$ relates to ff, and we have to hope that it will allow us to knock the 3 down to a 2. Every face of $G1'G1'$ and $G2'G2'$ is a face of GG, since the fact that removing ee disconnects GG means that ee must be part of the boundary of the unbounded face. Further, the unbounded face is counted twice in the sum $f1'+f2'f1'+f2'$, so $f=f1'+f2'-1f=f1'+f2'-1$. This gives exactly what we need to complete the proof.

Taken by itself, Euler's formula doesn't seem that useful, since it requires counting the number of faces in a planar embedding. However, we can use this formula to get a quick way to determine that a graph is not planar. Consider a drawing without edge crossings of a graph on nn vertices and m edges, with $n\geq 3n\geq 3$. We consider pairs $(e,F)(e,F)$ where ee is an edge of GG and FF is a face that has ee as part of its boundary. How many such pairs are there? Let's call the number of pairs pp. Each edge can bound either one or two faces, so we have that $p\leq 2mp\leq 2m$. We can also bound pp by counting the number of pairs in which a face FF appears. Each face is bounded by at least 3 edges, so it appears in at least 3 pairs, and so $p\geq 3fp\geq 3f$. Thus $3f\leq 2m3f\leq 2m$ or $f\leq 2m/3f\leq 2m/3$. Now, utilizing Euler's formula, we have

$$m=n+f-2\leq n+2m3-2\Leftrightarrow m3\leq n-2m=n+f-2\leq n+2m3-2\Leftrightarrow m3\leq n-2.$$

Thus, we've proven the following theorem.

Theorem 4.33

A planar graph on nn *vertices has at most* $3n-63n-6$ *edges when* $n\geq 3n\geq 3$.

The contrapositive of this theorem, namely that an nn-vertex graph with more than $3n-63n-6$ edges is not planar, is usually the most useful formulation of this result. For instance, we've seen (**Figure 4.31**) that $K4K4$ is planar. What about $K5K5$? It has 5 vertices and $C(5,2)=10>9=3\cdot 5-6C(5,2)=10>9=3\cdot 5-6$ edges, so it is not planar, and thus for $n\geq 5n\geq 5$, $KnKn$ is not planar, since it contains $K5K2$. It's important to note that **Theorem 4.33** is not the be-all, end-all of determining if a graph is planar. To see this, let's return to the subject of drawing $K3,3K3,3$ in the plane. This graph has 6 vertices and 9 edges, so it passes the test of **Theorem 4.33**. However, if you spend a couple minutes trying to find a way to draw $K3,3K3,3$ in the plane without any crossing edges, you'll pretty quickly begin to believe that it can't be done—and you'd be right!

To see why $K3,3K3,3$ is not planar, we'll have to return to Euler's formula, and we again work with edge-face pairs. For $K3,3K3,3$, we see that every edge would have to be part of the boundary of two faces, and faces are

bounded by cycles. Also, since the graph is bipartite, there are no odd cycles. Thus, counting edge-face pairs from the edge perspective, we see that there are 2m=182m=18 pairs. If we let fkfk be the number of faces bounded by a cycle of length kk, then f=f4+f6f=f4+f6. Thus, counting edge-face pairs from the face perspective, there are 4f4+6f64f4+6f6 pairs. From Euler's formula, we see that the number of faces ff must be 5, so then 4f4+6f6≥204f4+6f6≥20. But from our count of edge-face pairs, we have 2m=4f4+6f62m=4f4+6f6, giving 18≥2018≥20, which is clearly absurd. Thus, K3,3K3,3 is not planar.

At this point, you're probably asking yourself “So what?” We've invested a fair amount of effort to establish that K5K5 and K3,3K3,3 are nonplanar. Clearly any graph that contains them is also nonplanar, but there are a lot of graphs, so you might think that we could be at this forever. Fortunately, we won't be, since at its core, planarity really comes down to just these two graphs, as we shall soon see.

If G=(V,E)G=(V,E) is a graph and uv∈Euv∈E, then we may form a new graph G'G′ called an ***elementary subdivision*** of GG by adding a new vertex v'v′ and replacing the edge uvuv by edges uv'uv′ and v'vv′v. In other words, G'G′ has vertex set V'=V∪{v'}V′=V∪{v′} and edge set E'=(E−{uv})∪{uv',v'v}E′=(E−{uv})∪{uv′,v′v}. Two graphs G1G1 and G2G2 are ***homeomorphic*** if they can be obtained from the same graph by a (potentially trivial) sequence of elementary subdivisions.

The purpose of discussing homeomorphic graphs is that two homeomorphic graphs have the same properties when it comes to being drawn in the plane. To see this, think about what happens to K5K5 if we form an elementary subdivision of it via any one of its edges. Clearly it remains nonplanar. In fact, if you take any nonplanar graph and form the elementary subdivision using any one of its edges, the resulting graph is nonplanar. The following very deep theorem was proved by the Polish mathematician Kazimierz Kuratowski in 1930. Its proof is beyond the scope of this text.

Theorem 4.34. Kuratowski's Theorem

A graph is planar if and only if it does not contain a subgraph homeomorphic to either K5K5 or K3,3K3,3.

Kuratowski's Theorem gives a useful way for checking if a graph is planar. Although it's not always easy to find a subgraph homeomorphic to K5K5 or K3,3K3,3 by hand, there are efficient algorithms for planarity testing that make use of this characterization. To see this theorem at work, let's consider the Petersen graph shown in **Figure 4.17**. The Petersen graph has 10 vertices and 15 edges, so it passes the test of **Theorem 4.33**, and our argument using Euler's formula to prove that K3,3K3,3 is nonplanar was complex enough, we probably don't want to try it for the Petersen graph. To use Kuratowski's Theorem here, we need to decide if we would rather find a subgraph homeomorphic to K5K5 or to K3,3K3,3. Although the Petersen graph looks very similar to K5K5, it's actually simultaneously *too* similar and too different for us to be able to find a

subgraph homeomorphic to K5K5, since each vertex has degree 3. Thus, we set out to find a subgraph of the Petersen graph homeomorphic to K3,3K3,3. To do so, note that K3,3K3,3 contains a cycle of length 6 and three edges that are in place between vertices opposite each other on the cycle. We identify a six-cycle in the Petersen graph and draw it as a hexagon and place the remaining four vertices inside the cycle. Such a drawing is shown in **Figure 4.32.** The subgraph homeomorphic to K3,3K3,3 is found by deleting the black vertex, as then the white vertices have degree two, and we can replace each of them and their two incident edges (shown in bold) by a single edge.

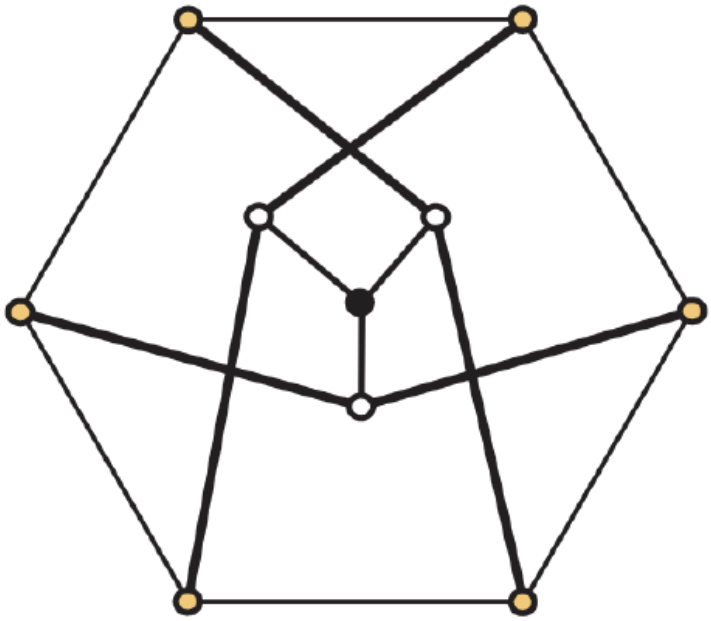


Figure **4.32.** A more illustrative drawing of the Petersen graph

We close this section with a problem that brings the current section together with the topic of graph coloring. In 1852 Francis Guthrie, an Englishman who was at the time studying to be lawyer but subsequently became a professor of mathematics in South Africa, was trying to color a map of the counties of England so that any two counties that shared a boundary segment (meaning they touched in more than a single point) were colored with different colors. He noticed that he only needed four colors to do this, and was unable to draw any sort of map that would require five colors. (He was able to find a map that required four colors, an example of which is shown in **Figure 4.36**.)

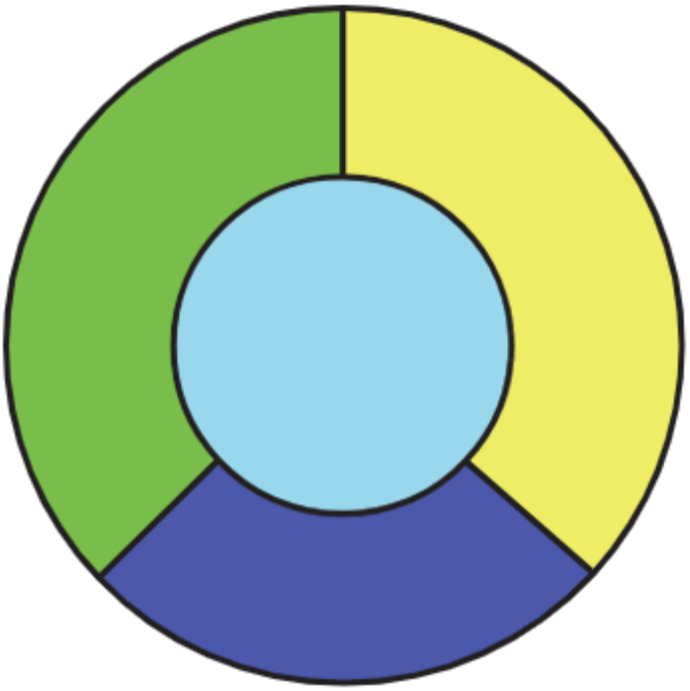


Figure **4.36.** A map that requires four colors

Could it possibly be true that *every* map could be colored with only four colors? He asked his brother Frederick Guthrie, who was a mathematics student at University College, London, about the problem, and Frederick eventually communicated the problem to Augustus de Morgan (of de Morgan's laws fame), one of his teachers. It was in this way that one of the most famous (or infamous) problems, known for a century as the Four Color Problem and now the Four Color Theorem, in graph theory was born. De Morgan was very interested in the Four Color Problem, and communicated it to Sir William Rowan Hamilton, a prominent Irish mathematician and the one for whom hamiltonian cycles are named, but Hamilton did not find the problem interesting. Hamilton is one of the few people who considered the Four Color Problem but did not become captivated by it.

We'll continue our discussion of the history of the Four Color Theorem in a moment, but first, we must consider how we can turn the problem of coloring a map into a graph theory question. Well, it seems natural that each region should be assigned a corresponding vertex. We want to force regions that share a boundary to have different colors, so this suggests that we should place an edge between two vertices if and only if their corresponding regions have a common boundary. (As an example, the map in **Figure 4.36** corresponds to the graph K4K4.) It is not difficult to see that this produces a planar graph, since we may draw the edges through the common boundary segment. Furthermore, with a little bit of thought, you should see that given a planar drawing of a graph, you can create a map in which each vertex leads to a region and edges lead to common

boundary segments. Thus, the Four Color Problem could be stated as "Does every planar graph have chromatic number at most four?"

Interest in the Four Color Problem languished until 1877, when the British mathematician Arthur Cayley wrote a letter to the Royal Society asking if the problem had been resolved. This brought the problem to the attention of many more people, and the first "proof" of the Four Color Theorem, due to Alfred Bray Kempe, was completed in 1878 and published a year later. It took 11 years before Percy John Heawood found a flaw in the proof but was able to salvage enough of it to show that every planar graph has chromatic number at most five. In 1880, Peter Guthrie Tait, a British physicist best known for his book Treatise on Natural Philosophy with Sir William Thomson (Lord Kelvin), made an announcement that suggested he had a proof of the Four Color Theorem utilizing hamiltonian cycles in certain planar graphs. However, consistent with the way Tait approached some conjectures in the mathematical theory of knots, it appears that he subsequently realized around 1883 that he could not prove that the hamiltonian cycles he was using actually existed and so Tait likely only believed he had a proof of the Four Color Theorem for a short time, if at all. However, it would take until 1946 to find a counterexample to the conjecture Tait had used in his attempt to prove the Four Color Theorem.

In the first half of the twentieth century, some incremental progress toward resolving the Four Color Problem was made, but few prominent mathematicians took a serious interest in it. The final push to prove the Four Color Theorem came with about at the same time that the first electronic computers were coming into widespread use in industry and research. In 1976, two mathematicians at the University of Illinois announced their computer-assisted proof of the Four Color Theorem. The proof by Kenneth Appel and Wolfgang Haken led the University of Illinois to add the phrase "FOUR COLORS SUFFICE" to its postage meter's imprint.

Theorem 4.37. Four Color Theorem

Every planar graph has chromatic number at most four.

Appel and Haken's proof of the Four Color Theorem was at a minimum unsatisfactory for many mathematicians, and to some it simply wasn't a proof. These mathematicians felt that the using a computer to check various cases was simply too uncertain; how could you be certain that the code that checked the 1,482 "unavoidable configurations" didn't contain any logic errors? In fact, there were several mistakes found in the cases analyzed, but none were found to be fatal flaws. In 1989, Appel and Haken published a 741-page tome entitled Every Planar Map is Four Colorable which provided corrections to all known flaws in their original argument. This still didn't satisfy many, and in the early 1990's a team consisting of Neil Robertson from The Ohio State University; Daniel P. Sanders, a graduate student at the Georgia Institute of Technology; Paul Seymour of Bellcore; and Robin Thomas from Georgia Tech announced a new proof of the Four Color Theorem. However, it still required the use of

computers. The proof did gain more widespread acceptance than that of Appel and Haken, in part because the new proof used fewer than half (633) of the number of configurations the Appel-Haken proof used and the computer code was provided online for anyone to verify. While still unsatisfactory to many, the proof by Robertson, et al. was generally accepted, and today the issue of the Four Color Theorem has largely been put to rest. However, many still wonder if anyone will ever find a proof of this simple statement that does not require the assistance of a computer.

4.6 COUNTING LABELED TREES

How many trees are there with vertex set $[n]=\{1,2,\ldots,n\}[n]=\{1,2,\ldots,n\}$? Let $TnTn$ be this number. For $n=1n=1$, there is clearly only one tree. Also, for $n=2n=2$, there is only one tree, which is isomorphic to $K2K2$. In determining,$T3T3$, we finally have some work to do; however, there's not much, since all trees on 3 vertices are isomorphic to $P3P3$. Thus, there are $T3=3T3=3$ ***labeled trees*** on 3 vertices, corresponding to which vertex is the one of degree 2. When $n=4n=4$, we can begin by counting the number of nonisomorphic trees and consider two cases depending on whether the tree has a vertex of degree 3. If there is a vertex of degree 3, the tree is isomorphic to $K1,3K1,3$ or it does not have a vertex of degree three, in which case it is isomorphic to $P4P4$, since there must be precisely two vertices of degree 2 in such a graph. There are four labelings by $[4][4]$ for $K1,3K1,3$ (choose the vertex of degree three). How many labelings by $[4][4]$ are there for $P4P4$? There are $C(4,2)C(4,2)$ ways to choose the labels i,ji,j given to the vertices of degree 2 and two ways to select one of the remaining labels to be made adjacent to ii. Thus, there are 12 ways to label $P4P4$ by $[4][4]$ and so $T4=16T4=16$.

To this point, it looks like maybe there's a pattern forming. Perhaps it is the case that for all $n\geq1,Tn=nn-2n\geq1,Tn=nn-2$. This is in fact the case, but let's see how it works out for $n=5n=5$ before proving the result in general. What are the nonisomorphic trees on five vertices? Well, there's $K1,4K1,4$ and $P5P5$ for sure, and there's also the third tree shown in **Figure 4.38**. After thinking for a minute or two, you should be able to convince yourself that this is all of the possibilities. How many labelings by $[5][5]$ does each of these have? There are 5 for $K1,4K1,4$ since there are 5 ways to choose the vertex of degree 4. For $P5P5$, there are 5 ways to choose the middle vertex of the path, $C(4,2)=6C(4,2)=6$ ways to label the two remaining vertices of degree 2 once the middle vertex is labeled, and then 2 ways to label the vertices of degree 1. This gives 60 labelings. For the last tree, there are 5 ways to label the vertex of degree 3, $C(4,2)=6C(4,2)=6$ ways to label the two leaves adjacent to the vertex of degree 3, and 2 ways to label the remaining two vertices, giving 60 labelings. Therefore, $T5=125=53=55-2T5=125=53=55-2$.

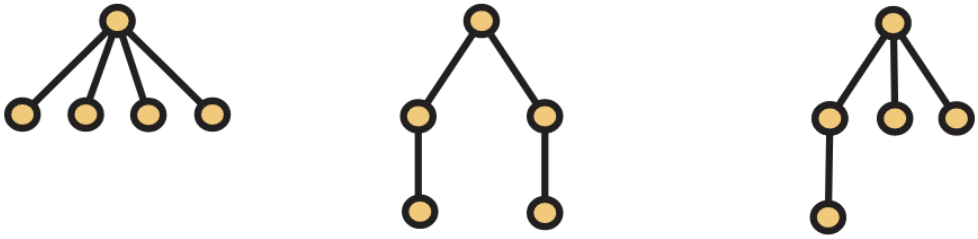


Figure **4.38.** The nonisomorphic trees on n=5n=5 vertices

It turns out that we are in fact on the right track, and we will now set out to prove the following:

Theorem 4.39. Cayley's Formula

The number TnTn *of labeled trees on* nn *vertices is* nn−2nn−2.

This result is usually referred to as Cayley's Formula, although equivalent results were proven earlier by James J. Sylvester (1857) and Carl W. Borchardt (1860). The reason that Cayley's name is most often affixed to this result is that he was the first to state and prove it in graph theoretic terminology (in 1889). (Although one could argue that Cayley really only proved it for n=6n=6 and then claimed that it could easily be extended for all other values of nn, and whether such an extension can actually happen is open to some debate.) Cayley's Formula has many different proofs, most of which are quite elegant. If you're interested in presentations of several proofs, we encourage you to read the chapter on Cayley's Formula in Proofs from THE BOOK by Aigner, Ziegler, and Hofmann, which contains four different proofs, all using different proof techniques. Here we give a fifth proof, due to Prüfer and published in 1918. Interestingly, even though Prüfer's proof came after much of the terminology of graph theory was established, he seemed unaware of it and worked in the context of permutations and his own terminology, even though his approach clearly includes the ideas of graph theory. We will use a recursive technique in order to find a bijection between the set of labeled trees on nn vertices and a natural set of size nn−2nn−2, the set of strings of length n−2n−2 where the symbols in the string come from [n][n].

We define a recursive algorithm that takes a tree TT on k≥2k≥2 vertices labeled by elements of a set SS of positive integers of size kk and returns a string of length k−2k−2 whose symbols are elements of SS. (The set SS will usually be [k][k], but in order to define a recursive procedure, we need to allow that it be an arbitrary set of kk positive integers.) This string is called the ***Prüfer code*** of the tree TT. Let prüferprüfer(TT) denote the Prüfer code of the tree TT, and if vv is a leaf of TT, let T−vT−v denote the tree obtained from TT by removing vv (i.e., the subgraph induced by all the other vertices). We can then define prüferprüfer(TT) recursively by the following procedure.

1. If T≅K2T≅K2, return the empty string.

2. Else, let vv be the leaf of TT with the smallest label and let uu be its unique neighbor. Let ii be the label of uu. Return (ii, prüfer(T−vT−v)).

Example 4.40

Before using Prüfer codes to prove Cayley's Formula, let's take a moment to make sure we understand how they are computed given a tree. Consider the 9-vertex tree TT in **Figure 4.41**.

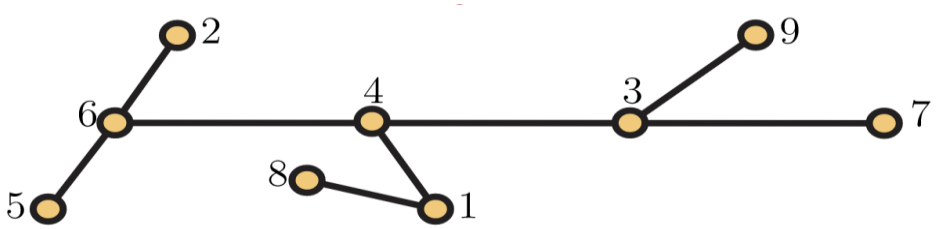
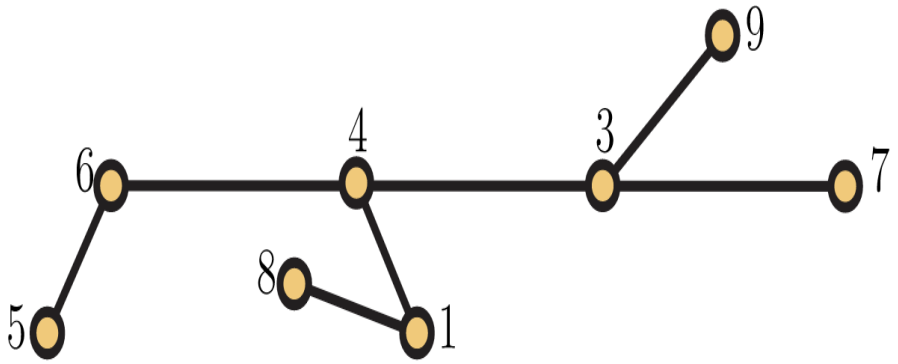


Figure **4.41.** A labeled 9-vertex tree

How do we compute prüfer(TT)? Since TT has more than two vertices, we use the second step and find that vv is the vertex with label 2 and uu is the vertex with label 6, so prüfer(TT)=(6,prüfer(T−vT−v)). The graph T−vT−v is shown in **Figure 4.42**.



Fi

gure **4.42.** The tree T−vT−v

The recursive call prüfer(T−vT−v) returns (6,prüfer(T−v−v'T−v−v′)), where v'v′ is the vertex labeled 2. Continuing recursively, the next vertex deleted is 6, which appends a 4 to the string. Then 7 is deleted, appending 3. Next 8 is deleted, appending 1. This is followed by the deletion of 1, appending 4. Finally 4 is deleted, appending 3, and the final recursive call has the subtree isomorphic to K2K2 with vertices labeled 3 and 9, and an empty string is returned. Thus, prüfer(TT) = 6643143.

We're now prepared to give a proof of Cayley's Formula.

Proof

It is clear that prüfer(T) takes an nn-vertex labeled tree with labels from [n][n] and returns a string of length n−2n−2 whose symbols are

elements of [n][n]. What we have yet to do is determine a way to take such a string and construct an nn-vertex labeled tree from it. If we can find such a construction, we will have a bijection between the set TnTn of labeled trees on nn vertices and the set of strings of length n−2n−2 whose symbols come from [n][n], which will imply that Tn=nn−2Tn=nn−2.

First, let's look at how prüfer(T) behaves. What numbers actually appear in the Prüfer code? The numbers that appear in the Prüfer code are the labels of the *nonleaf* vertices of TT. The label of a leaf simply cannot appear, since we always record the label of the *neighbor* of the leaf we are deleting, and the only way we would delete the neighbor of a leaf is if that neighbor were also a leaf, which can only happen T≅K2T≅K2, in which case prüfer(T) simply returns the empty string. Thus if I⊂[n]I⊂[n] is the set of symbols that appear in prüfer(T), the labels of the leaves of TT are precisely the elements of [n]−I[n]−I.

With the knowledge of which labels belong to the leaves of TT in hand, we are ready to use induction to complete the proof. Our goal is to show that if given a string s=s1s2⋯sn−2s=s1s2⋯sn−2 whose symbols come from a set SS of nn elements, there is a unique tree TT with prüfer(T)=ss. If n=2n=2, the only such string is the empty string, so 1 and 2 both label leaves and we can construct only K2K2. Now suppose we have the result for some m≥2m≥2, and we try to prove it for m+1m+1. We have a string s=s1s2⋯sm−1s=s1s2⋯sm−1 with symbols from [m+1][m+1]. Let II be the set of symbols appearing in ss and let kk be the least element of [m+1]−I[m+1]−I. By the previous paragraph, we know that kk is the label of a leaf of TT and that its unique neighbor is the vertex labeled s1s1. The string s'=s2s3⋯sm−1s′=s2s3⋯sm−1 has length m−2m−2 and since kk does not appear in ss, its symbols come from S=[m+1]−{k}S=[m+1]−{k}, which has size mm. Thus, by induction, there is a unique tree T'T′ whose Prüfer code is s's′. We form TT from T'T′ by attaching a leaf with label kk to the vertex of T'T′ with label s1s1 and have a tree of the desired type.

Example 4.43

We close this section with an example of how to take a Prüfer code and use it to construct a labeled tree. Consider the string s=75531s=75531 as a Prüfer code. Then the tree TT corresponding to ss has 7 vertices, and its leaves are labeled 2, 4, and 6. The inductive step in our proof attaches the vertex labeled 2 to the vertex labeled 7 in the tree T'T′ with Prüfer code 5531 and vertex labels {1,3,4,5,6,7}{1,3,4,5,6,7}, since 2 is used to label the last vertex added. What are the leaves of T'T′? The symbols in {4,6,7}{4,6,7} do not appear in 5531, so they must be the labels of leaves, and the construction says that we would attach the vertex labeled 4 to the vertex labeled 5 in the tree we get by induction. In **Figure 4.44**, we show how this recursive process continues.

Prüfer code	Label set	Edge added
75531	$\{1, 2, 3, 4, 5, 6, 7\}$	2–7
5531	$\{1, 3, 4, 5, 6, 7\}$	4–5
531	$\{1, 3, 5, 6, 7\}$	6–5
31	$\{1, 3, 5, 7\}$	5–3
1	$\{1, 3, 7\}$	3–1
(empty string)	$\{1, 7\}$	1–7

Figure **4.44.** Turning the Prüfer code 75531 into a labeled tree

We form each row from the row above it by removing the first label used on the edge added from the label set and removing the first symbol from the Prüfer code. Once the Prüfer code becomes the empty string, we know that the two remaining labels must be the labels we place on the ends of K2K2 to start building TT. We then work back up the edge added column, adding a new vertex and the edge indicated. The tree we construct in this manner is shown in **Figure 4.42.**

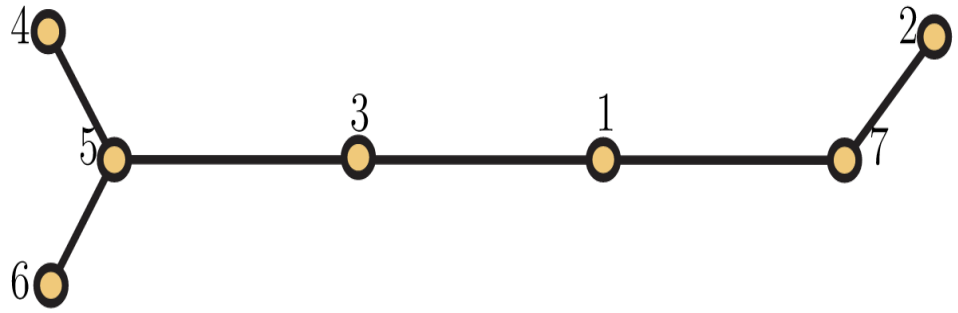


Figure **4.42.** The labeled tree with Prüfer code 75531

4.7: A DIGRESSION INTO COMPLEXITY THEORY

We have already introduced in Chapter 4 a few notions about efficient algorithms. We also discussed the difficulty of determining a graph's chromatic number and clique number earlier in this chapter. We conclude with a brief discussion of some issues involving computational complexity for other problems discussed in this chapter.

Let's begin with some problems for which there are polynomial-time algorithms. Suppose you are given a graph on nn vertices and asked whether or not the graph is connected. Here a positive answer can be justified by providing a spanning tree. On the other hand, a negative answer can be justified by providing a partition of the vertex sets V=V1∪V2V=V1∪V2 with V1V1 and V2V2 non-empty subsets and having no edges with one end-point in V1V1 and the other in V2V2. In Chapter 12 we will discuss two efficient algorithms that find spanning

trees in connected graphs. They can easily be modified to produce a partition showing the graph is disconnected.

If you are asked whether a connected graph is eulerian, then a positive answer can be justified by producing the appropriate sequence. We gave an algorithm to do this earlier in the chapter. A negative answer can be justified by producing a vertex of odd degree, and our algorithm will identify such a vertex if it exists. (Depending on the data structures used to represent the graph, it may be most efficient to simply look for vertices of odd degree without using the algorithm to find an eulerian circuit.)

On the surface, the problem of determining if a graph is hamiltonian looks similar to that of determining if the graph is eulerian. Both call for a sequence of vertices in which each pair of consecutive vertices is joined by an edge. Of course, each problem has an additional requirement on yes certificates. However, justifying a negative answer to the question of whether a graph is hamiltonian is not straightforward. **Theorem 4.18** only gives a way to confirm that a graph *is* amiltonian; there are many nonhamiltonian graphs that do not satisfy its hypothesis. At this time, no one knows how to efficiently justify a negative answer—at least not in the general case.

5

PROBABILITY TO COMBINATORICS AND RAMSEY THEORY

Unit Structure

5.0 OBJECTIVES:

This chapter would make you understand the following concepts:

- Permutations and Combinations
- Simple concepts related to probability and combinatorics
- The meaning of R(m, n)
- Estimating Ramsey Numbers

5.1 INTRODUCTION

Combinatorics, a branch of Mathematics consists methods of counting and arranging those quantities that are too large to be counted in regular way. It is used for the study of Discrete probability and also counting possible outcomes in a uniform probability experiment.

5.2 BASIC CONCEPTS: PERMUTATIONS AND COMBINATIONS

Definition 2.1:

Permutations (Arrangements): Different arrangements of objects that can be made out of a given number of things by taking some or all of them at a time is called a permutation. The permutations of n different things taken r at a time is denoted by n_{P_r} or P(n, r) and is equal to

$n_{P_r} = \frac{n!}{(n-r)!}$, $r \leq n$.

If in the collection of n things, p are of same type, q are of other same type, r are of other same type and so on.., then the number of permutations $= \frac{n!}{p! \cdot q! \cdot r! \ldots}$

Definition 2.2:

Combinations (Selections): Selections that can be made by taking some or all of things without reference to the order of things is called a combination. To choose r things out of n things is denoted by n_{C_r} or C(n, r) and is equal to $n_{C_r} = \frac{n!}{r! \cdot (n-r)!}$, $r \leq n$.

5.2.1 Combinatorics and its principles

There are are two basic principles of counting that are used in enumerative combinatorics:

Rule of product: (Count arrangements using probability)- If there are m ways to arrange something and then n ways to arrange another things after that, then there m*n ways to perform both of these actions.

Rule of Sum: If there are m ways for one action, and n ways for another action and the two actions cannot be done simultaneously, then there are (m + n) ways to choose one of these actions.

Counting Integers in a Range: In a closed interval [m, n], the number of integers is n-m+1.

5.2.2 Probability: probability refers to the chance of happening or not happening of an event. The probability of happening an event E is denoted by P(E) and is equal to

$P(E) = \frac{\text{Number of favourable outcomes}}{\text{Number of all possible outcomes}}$

5.2.3 Examples on applying probability to combinatorics:

Example 1. To unlock a mobile phone, a user must enter 6-digit correct password. How many passwords are possible? And if a user gets only 10 attempts, then find the probability that he will unlock the phone.

Sol. Since there are 10-10 choices (Choices can be 0, 1, 2, 3, 4, 5, 6, 7, 8, 9) for choosing each digit of password, hence by rule of product the total number of possible passwords = 10*10*10*10*10*10 = 1000000.

From 1000000 possible passwords, only 1 password can unlock the mobile phone. If a user gets only 10 attempts, then the probability P(correct password) is = $\frac{10}{1000000}$ = 0.00001

Example 2. How many ways are there to select first 3 finishers among 10 candidates in a race? Find the probability of selecting the first 3 finishers of the race in correct order.

Sol. Since anyone among 10 candidates can come first, so the choices for first position are 10. For the second and third position, there are 9 and 8 choices respectively. By the rule of product, the total number of ways to select first 3 finishers = 10*9*8 = 720.

From 720 possible ways, only 1 is in the correct order. So the probability of selecting the first 3 finishers of the race in correct order = $\frac{1}{720}$

Example 3. How many integers are there from 200 to 400? Find the probability that a number picked randomly from 200 to 400 and is divisible by 3.

Sol. The number of integers from 200 to 400 = 400 – 200 + 1 = 201

Let the number of integers from 200 to 400 that are divisible by 3 is n. Then n can be found by using general formula for arithmetic series whose first term a is 201 and last term t_n is 399. Then,

399 = 201 + (n – 1)*3

⇒ n = 67

The probability that a number picked randomly from 200 to 400 and is divisible by 3 = $\frac{67}{201}$ = 0.33

Example 4. A committee of 6 members is to be formed out of a group of 7 men and 4 women. Calculate the probability that the committee will consist of exactly 2 women.

Sol. Total no. of all possible outcomes = 11_{C_6} = 462

Number of all favourable outcomes = $7_{C_4} * 4_{C_2}$ = 210

The probability that the committee will consist of exactly 2 women = $\frac{210}{462}$ = 0.45

Example 5. Five children called A, B, C, D and E sit randomly on five chairs. What is the probability that A sits on the first chair?

Sol. Five children can sit on 5 chairs in 5! Ways. A can sit on first chair in 4! Ways.

P(A sits on the first chair) $= \frac{\text{number of outcomes where A sits on the first chair}}{\text{total number of possible outcomes}}$

$= \frac{1}{5}$

Example 6. If we draw two cards from a standard pack, what is the probability that they are of the same suit?

Sol. The total number of ways choosing 2 cards from a standard pack = 52_{C_2} = 1326

Now choosing cards of same suit, first choose the suit, and then choose two cards out of that suit. So, the number of ways to choose cards of same suit = 4 × 13_{C_2} = 312.

The probability that both cards are of same suit = $\frac{312}{1326}$ = 0.2353

5.3 RAMSEY NUMBER:

Introduction: Ramsey theory was introduced by Frank Plumpton Ramsey to study about those complete graphs whose subgraphs can have some regular properties. Generally we look for monochromatic complete graph.

The Ramsey number R(m, n) gives the solution to the party problem, which asks the minimum number of guests R(m, n) that must be invited so that at least m will know each other or at least n will not know each other.

Definition 2.3:

Ramsey Number: Let K_n be a complete graph with n vertices. Then the Ramsey number R(m, n) is the least number of vertices that a graph must have so that in any red-blue coloring, there exists either a red K_m or a blue K_n.

5.4 SMALL RAMSEY NUMBER:

Ramsey numbers R(m, n) are called trivial for which either m = 2 or n = 2.

Theorem 1. For all m, n ∈ N, the relationship R(m, n) = R(n, m) holds.

Proof. The result is based on the symmetry of graphs. From the standpoint of edge colorings, consider that a 2-colored complete graph G will have an inversely 2-colored complete graph G', where any red edge in G will be colored blue in G' and vice versa.

We know that R(m, n) requires that any edge coloration of $K_{R(m, n)}$ will have a red monochromatic subgraph K_r or a blue monochromatic subgraph K_b , that also means that the inversely 2- colored graph $K'_{R(m, n)}$ will have a blue monochromatic subgraph K_r or a red monochromatic subgraph K_b.

Thus, since the inverses of all edge colorings are just all edge colorings, we have the equivalent conditions for R(n, m).

Theorem 2. R(1, n) = 1

Proof. A monochromatic K_1 is simply a single vertex, which requires no edges and thus either a "red" or "blue" monochromatic K_1 will simply require one vertex to satisfy the conditions of R(1, n) or R(n, 1).

Thus, all Ramsey numbers with m = 1 or n = 1 will only need a single vertex to guarantee the existance of one of their two required subgraphs.

Hence R(1, n) = R(n, 1) = 1

Theorem 3. R(n, 2) = n.

Proof. Let us consider a complete (n−1)-gon in which every edge is colored blue. In this case, there is neither a red edge, nor a complete blue n-gon, so R(n, 2) > n − 1.

Now we consider any graph with n vertices. If any edge is colored red, then we have found the red pair of vertices. Otherwise, all edges are blue, so we have found the blue n-gon. This means that in any graph of n vertices there is either a blue K_n or a red K2, so R(n, 2) ≤ n.

Combining the above two results, we get that R(n, 2) = n. By symmetry of R(s, t) and R(t, s), we also get that R(2, n) = n.

Theorem 4. R(3, 3) = 6.

Proof. First, we show that R(3, 3) > 5. Let us consider the pentagon shown in Figure 1. There is no monochromatic triangle, hence our claim is true. Next, we claim that R(3, 3) ≤ 6. Consider an arbitrary coloring of the edges of a complete graph with 6 vertices. There are 5 edges incident to each vertex of this complete graph. Since there exist just two colors, at least 3 of those edges will be colored by the same color.

Figure 1

Theorem 5. If m > 2 and n > 2, then R(m, n) ≤ R(m − 1, n) + R(m, n − 1).

Proof. Let us assume contrary statement that R(m, n) > R(m−1, n) + R(m, n −1) for some values of m and n. Let k = R(m−1, n) + R(m, n −1) and consider a complete graph of k vertices and a reb-blue coloring such that there is no red K_m or blue K_n . Pick a random vertex v. Let N_R be the set of vertices which are connected to v with a red edge and N_B be the set of vertices which are connected to v with a blue edge. It holds that | N_R | + |

N_B | = k − 1. By assumptions for the graph, there should be no blue K_n in N_R. Also, if there exists a red K_{m-1} in N_R, then the set $N_R \cup \{v\}$ has a red K_m, contradiction. Thus $|N_R| \leq R(m-1, n) - 1$. Using the same argument, we can get, $|N_B| \leq R(m, n-1) - 1$.

So $k - 1 = |N_R| + |N_B| \leq R(m-1, n) + R(m, n-1) - 2 = k - 2$, contradiction, and we showed that $R(m, n) \leq R(m-1, n) + R(m, n-1)$.

Theorem 6. $R(m, n) \leq \binom{m+n-2}{m-1}$

Proof. We will prove the theorem by Mathematical Induction.

Basis Step: Let us take m = n = 2, we get,

$R(2, 2) = 2 \leq \binom{2+2-2}{2-1}$

Induction step: Now assume that the relation holds for all m = x − 1, n = y and m = x, n = y − 1. We demonstrate that the m = x, n = y case holds using Theorem 1.

By theorem 4, we have,

$$R(m, n) \leq R(m-1, n) + R(m, n-1)$$

$$\leq \binom{(m-1)+n-2}{(m-1)-1} + \binom{m+(n-1)-2}{m-1}$$

$$\leq \binom{m+n-2}{m-1}$$

$$\Rightarrow R(m, n) \leq \binom{m+n-2}{m-1}$$

Corollary 1. $R(k, k) \leq \binom{k+k-2}{k-1}$

$$\Rightarrow R(k, k) \leq \binom{2k-2}{k-1}$$

Theorem 7. $R(m, n) \leq R(m-1, n) + R(m, n-1) - 1$

Proof. Suppose $R(m-1, n) = 2p$ and $R(m, n-1) = 2q$. Let us assume a graph of 2p + 2q − 1 vertices and choose a vertex A among them. There are 2p + 2q − 2 edges ending at A. Then, consider the following cases:

1. 2p or more edges end at A

2. 2q or more edges end at A

3. 2p − 1 red edges end at A and 2q − 1 blue edges end at A

For first case, consider the set T_1 of the vertices at the farther ends of the 2p or more segments. Since the numbers of vertices in T_1 is greater than or equal to R(m − 1, n), there is either a red K_{m-1} or a blue K_n . However, if there is a red K_{m-1}, then the set $T_1 \cup \{A\}$ is a red K_m. Thus, the theorem holds in this case.

The same argument shows that the theorem holds for second case as well.

The third case cannot hold for every vertex A of the graph. Indeed, if it did, there would be $(2p + 2q - 1)(2p - 1)$ red endpoints, which is an odd number. However, every edge has two endpoints, so this number should be even. This means that there exists at least one vertex for which either case 1 or case 2 holds. Since theorem was shown for these two cases, it holds for the third case, too.

5.5 ESTIMATING RAMSEY NUMBERS:

Theorem 8. $R(3, 4) = 9$ and $R(3, 5) = 14$.

Proof. From Theorem 4, it follows that $R(3, 4) \leq R(2, 4) + R(3, 3) - 1 = 4 + 6 - 1 = 9$.

Then, we claim that $R(3, 5) > 13$. Indeed, we consider a K_{13} in which we number vertices with numbers 0-12 and color the edges such that an edge is red if and only if the difference of the numbers of the two adjacent vertices is 1, 5, 8 or 12 (modulo 13). Then, the graph contains no red triangle and no blue K_5. It is easy to see that there is no red triangle. We can also show that there is no blue K_5. Assume on the contrary that a blue K_5 exists. By symmetry, assume that a vertex of the k_5 is the 0. Then, the other vertices must be in the "clusters" 2, 3, 4, or 6, 7, or 9, 10, 11. By pigeon-hole principle, at least two are in the same cluster. Since the edge between them is not blue, they are in a cluster of three total numbers. Without loss of generality assume they are 2 and 4. Then the others can only be 6 and 11. But these two differ by 5, contradiction. Thus $R(3, 5) > 13 \Rightarrow R(3, 5) \geq 14$. However, $R(3, 5) \leq R(2, 5) + R(3, 4) \leq 5 + 9 = 14$. This means that we must have $R(3, 4) = 9$ and $R(3, 5) = 14$.

Theorem 9. $R(4, 4) = 18$.

Proof. By Theorem 3, we have $R(4, 4) \leq 2\ R(3, 4) = 18$. It is enough to show that $R(4, 4) > 17$. We consider a K_{17} in which we number vertices with numbers 0-16 and color the edges such that an edge is if and only if the difference of the numbers of the two adjacent vertices is 1, 2, 4, 8, 9, 13, 15, 16 (modulo 17). By symmetry, it is enough to show that vertex 0 cannot be in a red K_4 or a blue K_4. Vertex 0 is connected by red edges with the vertices 1, 2, 4, 8, 9, 13, 15 and 16. Assume there is a red K_4. If 1 is in that, the remaining vertices must be in the set {2, 9, 16}, but no two of them are connected with red vertices. Similarly, for 2, the set of remaining vertices should be in {1, 4, 15}, for 4, the set of remaining vertices should be in {2, 8, 13}, and for 8, the set of remaining vertices should be in {4, 9, 16}. No red edges are contained in these sets. The rest are symmetric. Thus there can be no red K_4 that contains 0. Vertex 0 is connected by red edges with the vertices 3, 5, 6, 7, 10, 11, 12 and 14. Assume there is a blue K_4. If 3 is in that, the remaining vertices must be in the set {6, 10, 14}, but no two of them are connected with blue vertices. Similarly, for 5, the set of remaining vertices should be in {10, 11, 12}, for 6, the set of remaining vertices should be in {3, 11, 12}, and for 7, the set of remaining values should be in {10, 12, 14}. No blue edges are contained in these sets. The rest are symmetric. Thus there can be no blue K4 that contains 0.

Hence R(4, 4) > 17 ⇒R(4, 4) = 18

There are five more numbers which are known:

R(3, 6) = 18,

R(3, 7) = 23,

R(3, 8) = 28,

R(3, 9) = 36, and

R(4, 5) = 25.

5.6 APPLYING PROBABILITY TO RAMSEY THEORY

Theorem 10. If n is a positive integer, then $R(n, n) \geq \frac{n}{e\sqrt{2}} 2^{\frac{1}{2}n}$.

Proof: Let us consider probability space (S, P) where the outcomes are graphs with vertex set {1,2,…,t}. For each i and j with, $1 \leq I < j \leq t$, edge ij is present in the graph with probability 1/2. Furthermore, the events for distinct pairs are independent.

Let X_1 denote the random variable which counts the number of n-element subsets of {1, 2, …, t} for which all $\binom{n}{2}$ pairs are edges in the graph. Similarly, X_2 is the random variable which counts the number of n-element independent subsets of {1,2,…,t}.

Then set $X = X_1 + X_2$.

By linearity of expectation, $E(X) = E(X_1) + E(X_2)$ while

$E(X_1) = E(X_2) = \binom{t}{n} \frac{1}{2^{C(n,2)}}$

If E(X) < 1, then there must exist a graph with vertex set {1,2,…,t} without a K_n or I_n .

5.7 RAMSEY'S THEOREM:

Theorem 11. (Ramsey's Theorem for graph):

If m and n are positive integers, then there exists a least positive integer R(m, n) so that if G is a graph and G has at least R(m, n) vertices, then either G contains a complete subgraph on m vertices, or G contains an independent set of size n.

Proof: First of all, we have to show that such a r(m, n) exists and the maximum value it can attain = $\binom{m+n-2}{m-1}$.

If m ≤ 2 or n ≤ 2, the claim is trivial.

For m, n ≥ 3, we show the result by using mathematical induction on t = m + n assuming the result holds when t ≤ 5.

Now let x be any vertex in .G. Then there are at least $\binom{m+n-2}{m-1} - 1$ other vertices, which we partition as $S_1 \cup S_2$, where S_1 are those vertices adjacent to x in G and S_2 are those vertices which are not adjacent to x.

We recall that the binomial coefficients satisfy

$$\binom{m+n-2}{m-1} = \binom{m+n-3}{m-2} + \binom{m+n-3}{m-1} = \binom{m+n-3}{n-2} + \binom{m+n-2}{n-1}.$$

So either $|S_1| \geq \binom{m+n-3}{m-2}$ or $|S_1| \geq \binom{m+n-3}{m-1}$. If the first option holds, and S_1 does not have an independent set of size n, then it contains a complete subgraph of size m−1. It follows that we may add x to this set to obtain a complete subgraph of size m in G.

Similarly, if the second option holds, and S_2 does not contain a complete subgraph of size m, then S_2 contains an independent set of size n−1, and we may add x to this set to obtain an independent set of size n in G.

Theorem 12. (General Ramsey's Theorem):

For positive integers k, $l_1, l_2, \ldots, l_r$ with each $l_i \geq k$, there exists a least positive integer $N = R_k(l_1, l_2, \ldots, l_r)$ such that, for every r–colouring of all k–subsets of [N], there exists a monochromatic set of size l_i for some $i \in [r]$.

5.8 A LOWER BOUND OF RAMSEY NUMBERS USING PROBABILISTIC METHOD:

Theorem 13. Let $k, p \in N$ be such that $\binom{p}{k} 2^{1-\frac{k}{2}} < 1$. Then $R(k, k) > p$.

Proof. It is sufficient to show that there exists a colouring of the edges of K_p that contains no monochromatic K_k. Consider an edge colouring of K_p in which colours are assigned rabdomly. Let each edge be coloured independently, and such that for all edges e,

P(edge e is red) = P(edge e is blue) = $\frac{1}{2}$.

There are $\binom{p}{k}$ K_k in K_p. Let A_i be the event that the i^{th} K_k is monochromatic.

Then, $P(A_i) = 2 \cdot \left(\frac{1}{2}\right)^{\binom{k}{2}} = 2^{1-\binom{k}{2}}$ where the leading 2 is because there are two colours from which to choose.

Then, P(∃ a monochromatic K_k) = $P(\cup_i A_i) = \binom{p}{k} 2^{1-\binom{k}{2}}$.

However, $\binom{p}{k} 2^{1-\binom{k}{2}} < 1$ by the assumption of the theorem, so

P(∃ a colouring with no monochromatic K_k) > 0.

Hence, there exists a colouring with no monochromatic K_k.

5.9 SUMMARY:

In Computer Science, there are numerous problems that need to count things and measure the likelihood of events. The concepts that surround attempts to measure the likelihood of events can be studied in probability theory. A gentle introduction and application to Ramsey theory for

students interested in becoming familiar with this dynamic segment of combinatorics has been given. The chapter provides lower and upper bound of Ramsey numbers.

5.10 REFERENCE FOR FURTHER READING

1. J. M. Harris, J. L. Hirst, and M. J. Mossinghoff, Combinatorics and Graph Theory. Springer, 2000.

2. R. Balakrishnan and K. Ranganathan, A Textbook of Graph Theory. Springer, 2000.

5.11 MODEL QUESTIONS:

1. A postman has to deliver four letters to four different houses in a street. Find the probability that every house gets the right letter if address has been erased from each letter.

2. In a lottery you have to guess 5 out of 49 numbers. What is the probability that you get all of them right?

3. A password consists 4 characters allowing any upper case, lower case or digits can be used. Find the chance one can get it correct.

4. A lottery consists of 8 numbers from the numbers 1 to 30. What is the probability of getting exactly 4 winning numbers correct with one ticket?

5. If we draw one card from a standard pack, what is the probability that it is black and 6? Also find the probability that it is black or 6?

6. What is the value of R(2, 8)?

7. What is the value of R(1, 13)?

8. Find the maximum value of R(5, 8).

Answers:

1. 0.0417

2. 0.00000052

3. 0.000000068

4. 0.0875

5. 0.0384, 0.5384

6. 8

7. 1

8. 330

6

NETWORK FLOWS

Unit Structure

This chapter continues our look at the topics of algorithms and optimization. On an intuitive level, networks and network flows are fairly simple. We want to move something (merchandise, water, data) from an initial point to a destination. We have a set of intermediate points (freight terminals, valves, routers) and connections between them (roads, pipes, cables) with each connection able to carry a limited amount. The natural goal is to move as much as possible from the initial point to the destination while respecting each connection's limit. Rather than just guessing at how to perform this maximization, we will develop an algorithm that does it. We'll also see how to easily justify the optimality of our solution though the classic Max Flow-Min Cut Theorem.

6.1.1 BASIC NOTATION AND TERMINOLOGY

A directed graph in which for each pair of vertices x,y at most one of the directed edges (x,y) and (y,x) between them is present is called an ***oriented graph***. The basic setup for a network flow problem begins with an oriented graph ,G, called a ***network***, in which we have two special vertices called the ***source*** and the ***sink***. We use the letter S to denote the source, while the letter T is used to denote the sink (terminus). All edges incident with the source are oriented away from the source, while all edges incident with the sink are oriented with the sink. Furthermore, on each edge, we have a non-negative ***capacity***, which functions as a constraint on how much can be transmitted via the edge. The capacity of the edge e=(x,y) is denoted c(e) or by .c(x,y). In a computer program, the nodes of a network may be identified with integer keys, but in this text, we will typically use letters in labeling the nodes of a network. This helps to distinguish nodes from capacities in diagrams of networks. We illustrate a network in Figure 6.1.1. The numbers associated with the edges are their capacities, so, for instance, c(E,B)=24 and .c(A,T)=56.

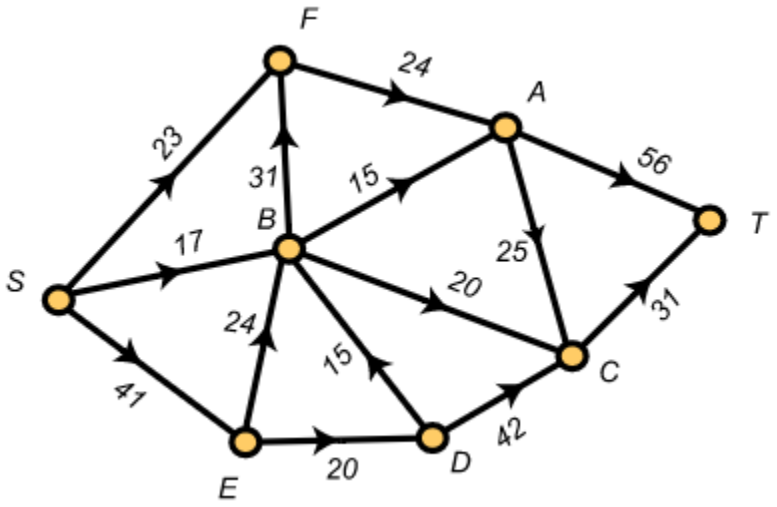


Figure 6.1.1. A Network

A ***flow*** ϕ in a network is a function which assigns to each directed edge e=(x,y) a non-negative value $\phi(e)=\phi(x,y)\leq c(x,y)$ so that the following ***conservation laws*** hold:

1. $,\sum x\phi(S,x)=\sum x\phi(x,T)$, i.e., the amount leaving the source is equal to the amount arriving at the sink. This quantity is called the ***value*** of the flow $.\phi$.

2. For every vertex y which is neither the source nor the sink the amount leaving y is equal to the amount entering .y. That is, $.\sum x\phi(x,y)= \sum x\phi(y,x)$.

We illustrate a flow in a network in Figure 6.1.2.

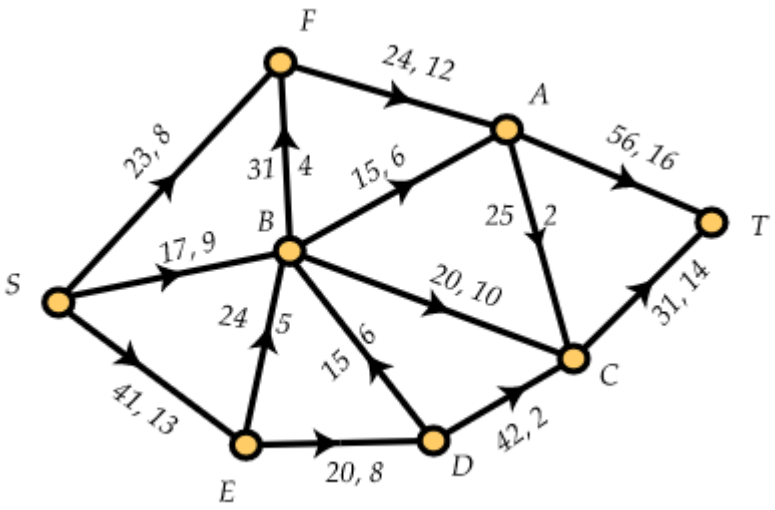


Figure 6.1.2. A Network Flow

In this figure, the numbers associated with each edge are its capacity and the amount of flow that ϕ places on that edge. For example, the edge (E,D) has capacity 20 and currently carries a flow of .8. (Since , $\phi(x,y) \leq c(x,y)$, it is always easy to determine which number is the capacity and which is the flow.) The value of this flow is .30=ϕ(S,F)+ϕ (S,B)+ϕ(S,E)=ϕ(A,T)+ϕ(C,T). To see that the second conservation law holds at, for example, vertex ,B, note that the flow into B is ϕ(S,B)+ϕ (E,B)+ϕ(D,B)=20 and the flow out of B is .ϕ(B,F)+ϕ(B,A)+ϕ(B,C)=20.

Given a network, it is very easy to find a flow. We simply assign $\phi(e)=0$ for every edge .e. It is very easy to *underestimate* the importance of this observation, actually. Network flow problems are a special case of a more general class of optimization problems known as ***linear programs***, and in general, it may be very difficult to find a feasible solution to a linear programming problem. In fact, conceptually, finding a feasible solution—*any* solution—is just as hard as finding an *optimal* solution.

6.1.2 FLOWS AND CUTS

Considering the applications suggested at the beginning of the chapter, it is natural to ask for the maximum value of a flow in a given network. Put another way, we want to find the largest number v0 so that there exists a flow ϕ of value v0 in the network. Of course, we not only want to find the maximum value ,v0, but we also want to find a flow ϕ having this value. Although it may seem a bit surprising, we will develop an efficient algorithm which both finds a flow of maximum value *and* finds a certificate verifying the claim of optimality. This certificate makes use of the following important concept.

A partition $V=L \cup U$ of the vertex set V of a network with $S \in L$ and $T \in U$ is called a ***cut***.[1] The ***capacity*** of a cut ,$V=L \cup U$, denoted , c(L,U), is defined by

$$c(L,U)=\sum_{x \in L, y \in U} c(x,y).$$

Put another way, the capacity of the cut $V=L \cup U$ is the total capacity of all edges *from* L *to* .U. Note that in computing the capacity of the cut ,$V=L \cup U$, we only add the capacities of the edges from L to .U. We do *not* include the edges from U to L in this sum.

Example 6.1.1

Let's again take a look at the network in Figure 6.1.2. Let's first consider the cut $V=L1 \cup U1$ with

$L1=\{S,F,B,E,D\}$ and $U1=\{A,C,T\}$.

Here we see that the capacity of the cut is

$$c(L1,U1)=c(F,A)+c(B,A)+c(B,C)+c(D,C)=24+15+20+42=101.$$

We must be a bit more careful, however, when we look at the cut V=L2∪U2 with

andL2={S,F,B,E}andU2={A,D,C,T}.

Here the capacity of the cut is

c(L2,U2)=c(F,A)+c(B,A)+c(B,C)+c(E,D)=24+15+20+20=79.

Notice that we do not include c(D,B) in the calculation as the directed edge (D,B) is from U2 to .L2.

The relationship between flows and cuts rests on the following fundamentally important theorem.

6.1.3 AUGMENTING PATHS

In this section, we develop the classic labeling algorithm of Ford and Fulkerson which starts with any flow in a network and proceeds to modify the flow—always increasing the value of the flow—until reaching a step where no further improvements are possible. The algorithm will also help resolve the debate Alice, Bob, Carlos, and Yolanda were having in the previous section.

Our presentation of the labeling algorithm makes use of some natural and quite descriptive terminology. Suppose we have a network G=(V,E) with a flow ϕ of value .v. We call ϕ the ***current flow*** and look for ways to ***augment*** ϕ by making a relatively small number of changes. An edge (x,y) with ϕ(x,y)>0 is said to be ***used***, and when ,ϕ(x,y)=c(x,y)>0, we say the edge is ***full***. When ,ϕ(x,y)<c(x,y), we say the edge (x,y) has ***spare capacity***, and when ,0=ϕ(x,y)<c(x,y), we say the edge (x,y) is ***empty***. Note that we simply ignore edges with zero capacity.

The key tool in modifying a network flow is a special type of path, and these paths are not necessarily directed paths. An ***augmenting path*** is a sequence P=(x0,x1,…,xm) of distinct vertices in the network such that ,x0=S, ,xm=T, and for each ,i=1,2,…,m, either

a. (xi−1,xi) has spare capacity or

When condition (Item a) holds, it is customary to refer to the edge (xi−1,xi) as a ***forward*** edge of the augmenting path .P. Similarly, if condition (Item b) holds, then the (nondirected) edge (xi−1,xi) is called a ***backward*** edge since the path moves from xi−1 to ,xi, which is opposite the direction of the edge.

Example 6.1.2.

Let's look again at the network and flow in Figure 6.1.2. The sequence of vertices (S,F,A,T) meets the criteria to be an augmenting path, and each edge in it is a forward edge. Notice that increasing the flow on each of ,(S,F), ,(F,A), and (A,T) by any positive amount δ≤12 results in increasing the value of the flow and preserves the conservation laws.

If our first example jumped out at you as an augmenting path, it's probably less clear at a quick glance that (S,E,D,C,B,A,T) is also an augmenting path. All of the edges are forward edges except for ,(C,B), since it's actually (B,C) that is a directed edge in the network. Don't worry if it's not clear how this path can be used to increase the value of the flow in the network, as that's our next topic.

Ignoring, for the moment, the issue of finding augmenting paths, let's see how they can be used to modify the current flow in a way that increases its value by some .$\delta>0$. Here's how for an augmenting path .$P=(x0,x1,\ldots,xm)$. First, let $\delta1$ be the positive number defined by:

a forward edge of $\delta1=\min\{c(xi-1,xi)-\phi(xi-1,xi):(xi-1,xi)$ a forward edge of P.$\}$

The quantity $c(xi-1,xi)-\phi(xi-1,xi)$ is nothing but the spare capacity on the edge ,$(xi-1,xi)$, and thus $\delta1$ is the largest amount by which *all* of the forward edges of .P. Note that the edges $(x0,x1)$ and $(xm-1,xm)$ are always forward edges, so the *positive* quantity $\delta1$ is defined for every augmenting path.

When the augmenting path P has no backward edges, we set .$\delta=\delta1$. But when P has one or more backward edges, we pause to set a backward edge of $\delta2=\min\{\phi(xi,xi-1):(xi-1,xi)$ a backward edge of P$\}$.

Since every backward edge is used, $\delta2>0$ whenever we need to define it. We then set .$\delta=\min\{\delta1,\delta2\}$.

In either case, we now have a positive number δ and we make the following elementary observation.

Proposition 6.1.3.

Suppose we have an augmenting path $P=(x0,x1,\ldots,xm)$ with $\delta>0$ calculated as above. Modify the flow ϕ by changing the values along the edges of the path P by an amount which is either $+\delta$ or $-\delta$ according to the following rules:

1. *Increase the flow along the edges of P which are forwards.*
2. *Decrease the flow along the edges of P which are backwards.*

Then the resulting function ϕ^ is a flow and it has value .$v+\delta$.

Example 6.1.4.

The network flow shown in Figure 6.1.2 has many augmenting paths. We already saw two of them in Example 6.1.6, which we call P1 and P3 below. In the list below, be sure you understand why each path is an augmenting path and how the value of δ is determined for each path.

1. P1=(S,F,A,T) with .$\delta=12$. All edges are forward.
2. P2=(S,B,A,T) with .$\delta=8$. All edges are forward.

3. P3=(S,E,D,C,B,A,T) with .δ=9. All edges are forward, except (C,B) which is backward.
4. P4=(S,B,E,D,C,A,T) with .δ=2. All edges are forward, except (B,E) and (C,A) which are backward.

6.1.3.1 Caution on Augmenting Paths

Bob's gotten really good at using augmenting paths to increase the value of a network flow. He's not sure how to find them quite yet, but he knows a good thing when he sees it. He's inclined to think that any augmenting path will be a good deal in his quest for a maximum-valued flow. Carlos is pleased about Bob's enthusiasm for network flows but is beginning to think that he should warn Bob about the dangers in using just any old augmenting path to update a network flow. They agree that the best situation is when the number of updates that need to be made is small in terms of the number of vertices in the network and that the size of the capacities on the edges and the value of a maximum flow should not have a role in the number of updates.

Bob says he can't see any way that the edge capacities could create a situation where a network with only a few vertices requires many updates, Carlos is thinking that an example is in order. He asks Bob to pick his favorite very large integer and to call it .M. He then draws the network on four vertices shown in Figure 3..1.9. Bob quickly recognizes that the maximum value of a flow in this network is .2M. He does this using the flow
with ,ϕ(S,A)=M, ,ϕ(A,T)=M, ,ϕ(S,B)=M, ϕ(B,T)=M and .ϕ(A,B)=0. Carlo s is pleased with Bob's work.

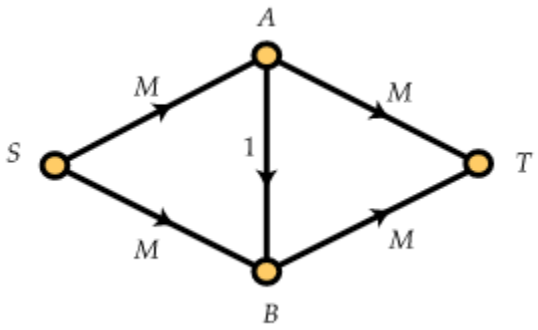


Figure 6.1.3 A Small Network

Since this network is really small, it was easy for Bob to find the maximum flow. However, Bob and Carlos agree that “eyeballing” is not an approach that scales well to larger networks, so they need to have an approach to finding that flow using augmenting paths. Bob tells Carlos to give him an augmenting path, and he'll do the updating. Carlos suggests the augmenting path ,(S,A,B,T), and Bob determines that δ=1 for this augmenting path. He updates the network (starting from the zero flow, i.e., with ϕ(e)=0 for every edge e) and it now has value .1. Bob asks Carlos for another augmenting path, so Carlos gives him .(S,B,A,T). Now (B,A) is

backward, but that doesn't phase Bob. He performs the update, obtaining a flow of value 2 with (A,B) empty again.

Despite Carlos' hope that Bob could already see where this was heading, Bob eagerly asks for another augmenting path. Carlos promptly gives him ,(S,A,B,T), which again has .δ=1. Bob's update gives them a flow of value .3. Before Carlos can suggest another augmenting path, Bob realizes what the problem is. He points out that Carlos can just give him (S,B,A,T) again, which will still have δ=1 and result in the flow value increasing to .4. He says that they could keep alternating between those two augmenting paths, increasing the flow value by 1 each time, until they'd made 2M updates to finally have a flow of value .2M. Since the network only has four vertices and M is very large, he realizes that using any old augmenting path is definitely not a good idea.

Carlos leaves Bob to try to figure out a better approach. He realizes that starting from the zero flow, he'd only need the augmenting paths (S,A,T) and ,(S,B,T), each with δ=M to quickly get the maximum flow. However, he's not sure why an algorithm should find those augmenting paths to be preferable. About this time, Dave wanders by and mumbles something about the better augmenting paths using only two edges, while Carlos' two evil augmenting paths each used three. Bob thinks that maybe Dave's onto something, so he decides to go back to reading his textbook.

6.1.4 THE FORD-FULKERSON LABELING ALGORITHM

In this section, we outline the classic Ford-Fulkerson labeling algorithm for finding a maximum flow in a network. The algorithm begins with a linear order on the vertex set which establishes a notion of **precedence**. Typically, the first vertex in this linear order is the source while the second is the sink. After that, the vertices can be listed in any order. In this book, we will use the following convention: the vertices will be labeled with capital letters of the English alphabet and the linear order will be ,(S,T,A,B,C,D,E,F,G,…), which we will refer to as the **pseudo-alphabetic order**. Of course, this convention only makes sense for networks with at most 26 vertices, but this limitation will not cramp our style. For real world problems, we take comfort in the fact that computers can deal quite easily with integer keys of just about any size.

Before providing a precise description of the algorithm, let's take a minute to consider a general overview. In carrying out the labeling algorithm, vertices will be classified as either **labeled** or **unlabeled**. At first, we will start with only the source being labeled while all other vertices will be unlabeled. By criteria yet to be spelled out, we will systematically consider unlabeled vertices and determine which should be labeled. If we ever label the sink, then we will have discovered an augmenting path, and the flow will be suitably updated. After updating the flow, we start over again with just the source being labeled.

This process will be repeated until (and we will see that this always occurs) we reach a point where the labeling halts with some vertices labeled (one of these is the source) and some vertices unlabeled (one of these is the sink). We will then note that the partition V=L∪U into labeled and unlabeled vertices (hence our choice of L and U as names) is a cut whose capacity is exactly equal to the value of the current flow. This resolves the debate from earlier in the chapter and says that the maximum flow/minimum cut question is more like antichains and partitioning into chains than clique number and chromatic number. In particular, the labeling algorithm will provide a proof of the following theorem:

Theorem The Max Flow–Min Cut Theorem.

Let G=(V,E) be a network. If v0 is the maximum value of a flow and c0 is the minimum capacity c0 of a cut, then .v0=c0.

We're now ready to describe the ***Ford-Fulkerson labeling algorithm*** in detail.

Algorithm **Ford-Fulkerson Labeling Algorithm.**

Labeling the Vertices

Vertices will be labeled with ordered triples of symbols. Each time we start the labeling process, we begin by labeling the source with the triple .(∗,+,∞). The rules by which we label vertices will be explicit.

Potential on a Labeled Vertex

Let u be a labeled vertex. The third coordinate of the label given to u will be positive real number—although it may be infinite. We call this quantity the ***potential*** on u and denote it by .p(u). (The potential will serve as the amount that the flow can be updated by.) Note that the potential on the source is infinite.

First Labeled, First Scanned

The labeling algorithm involves a scan from a ***labeled*** vertex .u. As the vertices are labeled, they determine another linear order. The source will always be the first vertex in this order. After that, the order in which vertices are labeled will change with time. But the important rule is that we scan vertices in the order that they are labeled—until we label the sink. If for example, the initial scan—always done from the source—results in labels being applied to vertices ,D, G and ,M, then we next scan from vertex .D. If that scan results in vertices ,B, ,F, G and Q being labeled, then we next scan from ,G, as it was labeled before ,B, even though B precedes G in the pseudo-alphabetic order. This aspect of the algorithm results in a ***breadth-first*** search of the vertices looking for ways to label previously unlabeled vertices.

Never Relabel a Vertex

Once a vertex is labeled, we do not change its label. We are content to label previously unlabeled vertices—up until the time where we label the sink. Then, after updating the flow and increasing the value, all labels, except of course the special label on the source, are discarded and we start all over again.

Labeling Vertices Using Forward Edges

Suppose we are scanning from a labeled vertex u with potential .p(u)>0. From ,u, we consider the unlabeled neighbors of u in pseudo-alphabetic order. Now suppose that we are looking at a neighbor v of u with the edge (u,v) belonging to the network. This means that the edge is directed from u to .v. If e=(u,v) is not full, then we label the vertex v with the triple (u,+,p(v)) where . $p(v)=\min\{p(u), c(e)-\phi(e)\}$. We use this definition since the flow cannot be increased by more than the prior potential or the spare capacity on .e. Note that the potential p(v) is positive since a is the minimum of two positive numbers.

Labeling Vertices Using Backward Edges

Now suppose that we are looking at a neighbor v of u with the edge (v,u) belonging to the network. This means that the edge is directed from v to .u. If e=(v,u) is used, then we label the vertex v with the triple $(u,-,p(v))$ where $.p(v)=\min\{p(u),\phi(e)\}$. Here p(v) is defined this way since the flow on e cannot be decreased by more than $\phi(e)$ or .p(u). Again, note that the potential p(v) is positive since a is the minimum of two positive numbers.

What Happens When the Sink is Labeled?

The labeling algorithm halts if the sink is ever labeled. Note that we are always trying our best to label the sink, since in each scan the sink is the very first vertex to be considered. Now suppose that the sink is labeled with the triple .(u,+,a). Note that the second coordinate on the label must be + since all edges incident with the sink are oriented towards the sink.

We claim that we can find an augmenting path P which results in an increased flow with ,$\delta=a$, the potential on the sink. To see this, we merely back-track. The sink T got its label from ,$u=u1$, u1 got its label from ,u2, and so forth. Eventually, we discover a vertex um which got its label from the source. The augmenting path is then.

The value of δ for this path is the potential p(T) on the sink since we've carefully ensured that $.p(um)\geq p(um-1)\geq\cdots\geq p(u1)\geq p(T)$.

And if the Sink is Not Labeled?

On the other hand, suppose we have scanned from every labeled vertex and there are still unlabeled vertices remaining, one of which is the sink. Now we claim victory. To see that we have won, we simply observe that if L is the set of labeled vertices, and U is the set of unlabeled vertices,

then every edge e=(x,y) with x∈L and y∈U is full, i.e., .ϕ(e)=c(e). If this were not the case, then y would qualify for a label with x as the first coordinate. Also, note that ϕ(y,x)=0 for every edge e with x∈L and . y∈U. Regardless, we see that the capacity of the cut V=L∪U is exactly equal to the value of the current flow, so we have both a maximum flow and minimum cut providing a certificate of optimality.

6.1.5 A CONCRETE EXAMPLE

Let's apply the Labeling Algorithm to the network flow shown in Figure 6.1.2. Then we start with the source:

S:(∗,+,∞)

Since the source S is the first vertex labeled, it is also the first one scanned. So we look at the neighbors of S using the pseudo-alphabetic order on the vertices. Thus, the first one to be considered is vertex B and since the edge (S,B) is not full, we label B as

B:(S,+,8).

We then consider vertex E and label it as

E:(S,+,28).

Next is vertex ,F, which is labeled as

F:(S,+,15).

At this point, the scan from S is complete.

The first vertex after S to be labeled was ,B, so we now scan from .B. The (unlabeled) neighbors of B to be considered, in order, are ,A, ,C, and . D. This results in the following labels:

A:(B,+,8)C:(B,+,8)D:(B,−,6)

The next vertex to be scanned is ,E, but E has no unlabeled neighbors, so we then move on to ,F, which again has no unlabeled neighbors. Finally, we scan from ,A, and using the pseudo-alphabetic order, we first consider the sink T (which in this case is the only remaining unlabeled vertex). This results in the following label for .T.

T:(A,+,8)

Now that the sink is labeled, we know there is an augmenting path. We discover this path by backtracking. The sink T got its label from ,A, A got its label from ,B, and B got its label from .S. Therefore, the augmenting path is P=(S,B,A,T) with .δ=8. All edges on this path are forward. The flow is then updated by increasing the flow on the edges of P by .8. This results in the flow shown in fig. The value of this flow is .38.

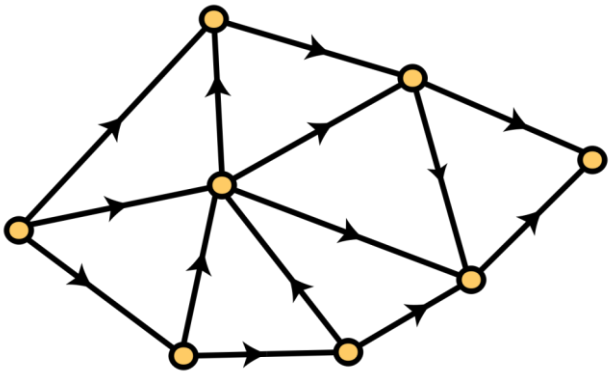


Figure 6.1.4 An Updated Network Flow

Here is the sequence (reading down the columns) of labels that will be found when the labeling algorithm is applied to this updated flow. (Note that in the scan from ,S, the vertex B will not be labeled, since now the edge (S,B) is full.)

S:(∗,+,∞)D:(E,+,12)E:(S,+,28)A:(F,+,12)F:(S,+,15)C:(B,+,10)B:(E,+,19)T :(A,+,12)

This labeling results in the augmenting path P=(S,F,A,T) with .δ=12.

After this update, the value of the flow has been increased and is now .50=38+12. We start the labeling process over again and repeat until we reach a stage where some vertices (including the source) are labeled and some vertices (including the sink) are unlabeled.

6.5.1 How the Labeling Algorithm Halts

Consider the network flow in Figure 6.1.5.

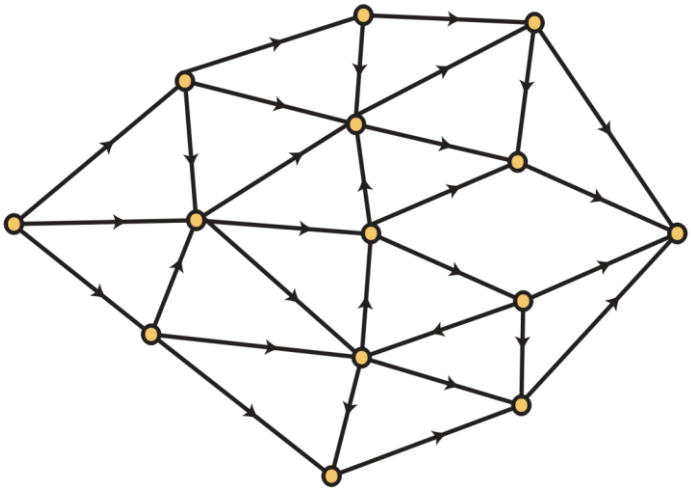


Figure 6.1.5 Another Network Flow

The value of the current flow is .172. Applying the labeling algorithm using the pseudo-alphabetic order results in the following labels (reading down the columns):

S:(∗,+,∞)E:(I,−,3)C:(S,+,8)G:(E,−,3)F:(S,+,23)L:(E,+,3)H:(C,+,7)B:(G,+,3)I:(H,+,7)T:(L,+,3)

These labels result in the augmenting path P=(S,C,H,I,E,L,T) with . δ=3. After updating the flow and increasing its value to ,175, the labeling algorithm halts with the following labels:

S:(∗,+,∞)H:(C,+,4)C:(S,+,5)I:(H,+,4)F:(S,+,23)

Now we observe that the labeled and unlabeled vertices are L={S,C,F,H,I} and .U={T,A,B,D,E,G,J,K}. Furthermore, the capacity of the cut V=L∪U is

41+8+23+8+13+29+28+25=175.

This shows that we have found a cut whose capacity is exactly equal to the value of the current flow. In turn, this shows that the flow is optimal.

6.1.6 INTEGER SOLUTIONS OF LINEAR PROGRAMMING PROBLEMS

A linear programming problem is an optimization problem that can be stated in the following form: Find the maximum value of a linear function

c1x1+c2x2+c3x3+⋯+cnxn

subject to m constraints ,C1, ,C2,…,Cm, where each constraint Ci is a linear equation of the form:

Ci:ai1x1+ai2x2+ai3x3+⋯+ainxn=bi

where all coefficients and constants are real numbers.

While the general subject of linear programming is far too broad for this course, we would be remiss if we didn't point out that:

Linear programming problems are a very important class of optimization problems and they have many applications in engineering, science, and industrial settings.

There are relatively efficient algorithms for finding solutions to linear programming problems.

A linear programming problem posed with rational coefficients and constants has an optimal solution with rational values—if it has an optimal solution at all.

A linear programming problem posed with integer coefficients and constants need not have an optimal solution with integer values—even when it has an optimal solution with rational values.

A very important theme in operations research is to determine when a linear programming problem posed in integers has an optimal solution with integer values. This is a subtle and often very difficult problem.

The problem of finding a maximum flow in a network is a special case of a linear programming problem.

A network flow problem in which all capacities are integers has a maximum flow in which the flow on every edge is an integer. The Ford-Fulkerson labeling algorithm guarantees this!

In general, linear programming algorithms are not used on networks. Instead, special purpose algorithms, such as Ford-Fulkerson, have proven to be more efficient in practice.

7

COMBINATORIAL APPLICATIONS OF NETWORK FLOWS

Unit Structure

Clearly finding the maximum flow in a network can have many direct applications to problems in business, engineering, and computer science. However, you may be surprised to learn that finding network flows can also provide reasonably efficient algorithms for solving combinatorial problems. In this chapter, we consider a restricted version of network flows in which each edge has capacity .1. Our goal is to establish algorithms for two combinatorial problems: finding maximum matchings in bipartite graphs and finding the width of a poset as well as a minimal chain partition.

7.1 INTRODUCTION

Before delving into the particular combinatorial problems we wish to consider in this chapter, we will state a key theorem. When working with network flow problems, our examples thus far have always had integer capacities and we always found a maximum flow in which every edge carried an integer amount of flow. It is not, however, immediately obvious that this can always be done. Why, for example, could it not be the case that the maximum flow in a particularly pathological network with integer capacities is ?23/3? Or how about something even worse, such as ?21π? We can rule out the latter because network flow problems fall into a larger class of problems known as linear programming problems, and a major theorem tells us that if a linear program is posed with all integer constraints (capacities in our case), the solution must be a rational number. However, in the case of network flows, something even stronger is true.

Theorem 7.1.

In a network flow problem in which every edge has integer capacity, there is a maximum flow in which every edge carries an integer amount of flow.

Notice that the above theorem does not guarantee that every maximum flow has integer flow on every edge, just that we are able to find one. With this theorem in hand, we now see that if we consider network flow problems in which the capacities are all 1 we can find a maximum flow in

which every edge carries a flow of either 0 or .1. This can give us a combinatorial interpretation of the flow, in a sense using the full edges as edges that we “take” in some useful sense.

7.2 MATCHINGS IN BIPARTITE GRAPHS

Recall that a bipartite graph G=(V,E) is one in which the vertices can be properly colored using only two colors. It is clear that such a coloring then partitions V into two independent sets V1 and ,V2, and so all the edges are between V1 and .V2. Bipartite graphs have many useful applications, particularly when we have two distinct types of objects and a relationship that makes sense only between objects of distinct types. For example, suppose that you have a set of workers and a set of jobs for the workers to do. We can consider the workers as the set V1 and the jobs as V2 and add an edge from worker w∈V1 to job j∈V2 if and only if w is qualified to do .j.

For example, the graph in Figure 7.2 is a bipartite graph in which we've drawn V1 on the bottom and V2 on the top.

Figure 7**.2.** A bipartite graph

If G=(V,E) is a graph, a set M⊆E is a **matching** in G if no two edges of M share an endpoint. If v is a vertex that is the endpoint of an edge in ,M, we say that M saturates v or v is saturated by .M. When G is bipartite with ,V=V1∪V2, a matching is then a way to pair vertices in V1 with vertices in V2 so that no vertex is paired with more than one other vertex. We're usually interested in finding a **maximum matching**, which is a matching that contains the largest number of edges possible, and in bipartite graphs we usually fix the sets V1 and V2 and seek a maximum matching from V1 to .V2. In our workers and jobs example, the matching problem thus becomes trying to find an assignment of workers to jobs such that each worker is assigned to a job for which he is qualified (meaning there's an edge), each worker is assigned to at most one job, andeach job is assigned at most one worker.

As an example, in Figure 7.3, the thick edges form a matching from V1 to .V2. Suppose that you're the manager of these workers (on the bottom) and must assign them to the jobs (on the top). Are you really making the best use of your resources by only putting four of six workers to work? There are no trivial ways to improve the number of busy workers, as the two without responsibilities right now cannot do any of the jobs that are unassigned. Perhaps there's a more efficient assignment that can be made by redoing some of the assignments, however. If there is, how should you go about finding it? If there is not, how would you justify to your boss that there's no better assignment of workers to jobs?

Figure 7**.3.** A matching in a bipartite graph

At the end of the section, we'll briefly look at a theorem on matchings in bipartite graphs that tells us precisely when an assignment of workers to jobs exists that ensures each worker has a job. First, however, we want to see how network flows can be used to find maximum matchings in bipartite graphs. The algorithm we give, while decent, is not the most efficient algorithm known for this problem. Therefore, it is not likely to be the one used in practice. However, it is a nice example of how network flows can be used to solve a combinatorial problem. The network that we use is formed from a bipartite graph G by placing an edge from the source S to each vertex of V1 and an edge from each vertex of V2 to the sink .T. The edges between V1 and V2 are oriented from V1 to , V2, and every edge is given capacity .1. Figure 7.4 contains the network corresponding to our graph from Figure 7.2. Edges in this network are all oriented from bottom to top and all edges have capacity .1. The vertices in V1 are x1,…,x6 in order from left to right, while the vertices in V2 are y1,…,y7 from left to right.

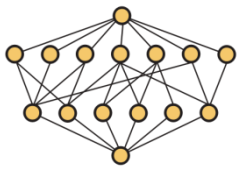


Figure 7**.4.** The network corresponding to a bipartite graph

Now that we have translated a bipartite graph into a network, we need to address the correspondence between matchings and network flows. To turn a matching M into a network flow, we start by placing one unit of flow on the edges of the matching. To have a valid flow, we must also place one unit of flow on the edges from S to the vertices of V1 saturated by .M. Since each of these vertices is incident with a single edge of ,M, the flow out of each of them is ,1, matching the flow in. Similarly, routing one unit of flow to T from each of the vertices of V2 saturated by M takes care of the conservation laws for the remaining vertices. To go the other direction, simply note that the full edges from V1 to V2 in an integer-valued flow is a matching. Thus, we can find a maximum matching from V1 to V2 by simply running the labeling algorithm on the associated network in order to find a maximum flow.

In Figure 7.5, we show thick edges to show the edges with flow 1 in the flow corresponding to our guess at a matching from Figure 7.3.

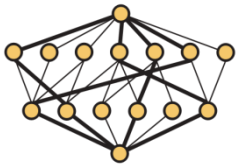


Figure 7**.5.** The flow corresponding to a matching

With priority sequence S,T,x1,x2,…,x6,y1,y2,…,y7 replacing our usual pseudo-alphabetic order, the labeling algorithm produces the labels shown below.

S:(∗,+,∞)y6:(x6,+,1)x3:(S,+,1)x1:(y6,−,1)x5:(S,+,1)y1:(x1,+,1)y4:(x3,+,1) y2:(x1,+,1)y5:(x3,+,1)y3:(x1,+,1)x6:(y4,−,1)x2:(y1,−,1)x4:(y5,−,1)T:(y2, +,1)

This leads us to the augmenting path ,S,x3,y4,x6,y6,x1,y2,T, which gives us the flow shown in Figure 7.6.

Figure 7**.6.** The augmented flow

Is this a maximum flow? Another run of the labeling algorithm produces

S:(∗,+,∞)x4:(y5,−,1)x5:(S,+,1)y4:(x4,+,1)y5:(x5,+,1)x3:(y4,−,1)

and then halts. Thus, the flow in Figure 7.6 is a maximum flow.

Now that we know we have a maximum flow, we'd like to be able to argue that the matching we've found is also maximum. After all, the boss isn't going to be happy if he later finds out that this fancy algorithm you claimed gave an optimal assignment of jobs to workers left the fifth worker (x5) without a job when all six of them could have been put to work. Let's take a look at which vertices were labeled by the Ford-Fulkerson labeling algorithm on the last run. There were three vertices (,x3, ,x4, and x5) from V1 labeled, while there were only two vertices (y4 and y5) from V2 labeled. Notice that y4 and y5 are the only vertices that are neighbors of ,x3, ,x4, or x5 in .G. Thus, no matter how we choose the matching edges from ,{x3,x4,x5}, one of these vertices will be left unsaturated. Therefore, one of the workers must go without a job assignment. (In our example, it's the fifth, but it's possible to choose different edges for the matching so another one of them is left without a task.)

The phenomenon we've just observed is not unique to our example. In fact, in every bipartite graph G=(V,E) with V=V1∪V2 in which we cannot

find a matching that saturates all the vertices of ,V, we will find a similar configuration. This is a famous theorem of Hall, which we state below.

Theorem 7.7. Hall's Theorem.

Let $G=(V,E)$ be a bipartite graph with $V=V1 \cup V2$. There is a matching which saturates all vertices of $V1$ if and only if for every subset $A \subseteq V1$, the set $N \subseteq V$ of neighbors of the vertices in A satisfies .

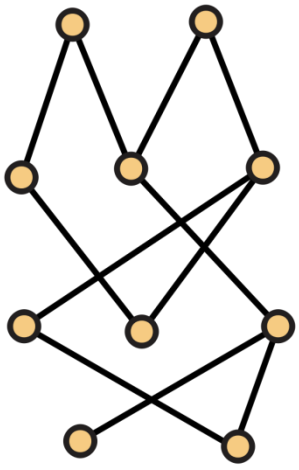
7.3 CHAIN PARTITIONING

Dilworth's Theorem, which told us that for any poset P of width ,w, there is a partition of P into ,w, but no fewer, chains. However, we were only able to devise an algorithm to find this chain partition (and a maximum antichain) in the special case where P was an interval order. Now, through the magic of network flows, we will be able to devise an efficient algorithm that works in general for all posets. However, to do so, we will require a slightly more complicated network than we devised in the previous section.

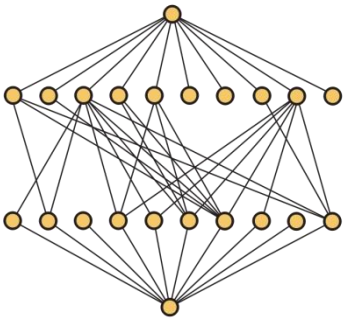
Suppose that the points of our poset P are $\{x1,x2,\ldots,xn\}$. We construct a network from P consisting of the source ,S, sink ,T, and two points xi' and xi'' for each point xi of .P. All edges in our network will have capacity .1. We add edges from S to xi' for $1 \leq i \leq n$ and from xi'' to T for $.1 \leq i \leq n$. Of course, this network wouldn't be too useful, as it has no edges from the single-prime nodes to the double-prime nodes. To resolve this, we add an edge directed from xi' to xj'' if and only if $xi<xj$ in .P.

Our running example in this section will be the poset in Figure 7

.8.(a). We'll discuss the points of the poset as xi where i is the number printed next to the point in the diagram.



(a) A small poset



(b) The associated network

Figure 7**.8.** A partially ordered set (a) and the associated network (b).

The first step is to create the network, which we show in Figure 7.8.(b). In this network, all capacities are ,1, edges are directed from bottom to top, the first row of ten vertices is the xi′ arranged consecutively with x1′ at the left and x10′ at the right, and the second row of ten vertices is the xi″ in increasing order of index. To see how this network is constructed, notice that x1<x3 in the poset, so we have the directed edge .(x1′,x3″). Similarly, x4 is less than ,x3, ,x5, and x9 in the poset, leading to three directed edges leaving x4′ in the network. As a third example, since x9 is maximal in the poset, there are no directed edges leaving .x9′.

We have not yet seen how we might turn a maximum flow (or minimum cut) in the network we've just constructed into a minimum chain partition or a maximum antichain. It will be easier to see how this works once we have a confirmed maximum flow. Rather than running the labeling algorithm starting from the zero flow, we eyeball a flow, such as the one shown in Figure 7.9. (Again, we use the convention that thick edges are full, while thin edges are empty.)

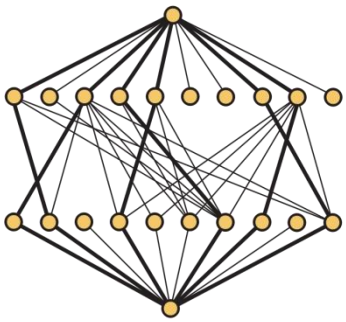


Figure 7**.9.** An initial flow

When we run the labeling algorithm (using priority S, T,x1′,…,x10′,x1″,…,x10″), we obtain the following list of labels:

S:(∗,+,∞)x9″:(x5′,+,1)x3′:(S,+,1)x3′:(S,+,1)x4″:(x6′,+,1)x1″:(x7′,+,1)x5′:(S,+,1)x5″:(x6′,+,1)x2″:(x7′,+,1)x6′:(S,+,1)x1′:(x3″,−,1)x2′:(x7′,+,1)x9′:(S,+,1)x8′:(x9″,−,1)T:(x2″,+,1)x3″:(x5′,+,1)x7′:(x4″,−,1)

Thus, we find the augmenting path ,(S,x6′,x4″,x7′,x2″,T), and the updated flow can be seen in Figure 7.10.

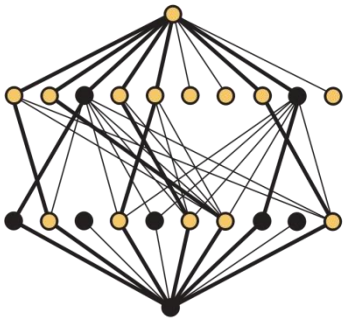


Figure 7**.10.** A better flow

If we run the labeling algorithm again, the algorithm assigns the labels below, leaving the sink unlabeled.

S:(∗,+,∞)x5′:(S,+,1)x3″:(x5′,+,1)x1′:(x3″,−,1)x3′:(S,+,1)x9′:(S,+,1)x9″:(x5′,+,1)x8′:(x9″,−,1)

In Figure 7.10, the black vertices are those the labeled in the final run, while the gold vertices are the unlabeled vertices.

Now that we've gone over the part you already knew how to do, we need to discuss how to translate this network flow and cut into a chain partition and an antichain. If there is a unit of flow on an edge ,(xi′,xj″), then a good first instinct is to place xi and xj in the same chain of a chain partition. To be able to do this successfully, of course, we need to ensure that this won't result in two incomparable points being placed in a chain. A way to see that everything works as desired is to think of starting with (xi′,xj″) and then looking for flow leaving .xj′. If there is, it goes to a vertex ,xk″, so we may add xk to the chain since .xi<xj<xk. Continue in this manner until reaching a vertex in the network that does not have any flow leaving it. Then see if xi″ has flow coming into it. If it does, it's from a vertex xm′ that can be added since .xm<xi<xj.

Let's see how following this process for the flow in Figure 7.10 leads to a chain partition. If we start with ,x1′, we see that (x1′,x3″) is full, so we place x1 and x3 in chain .C1. Since x3′ has no flow leaving it, there are no greater elements to add to the chain. However, x1″ has flow in from ,x2′, so we add x2 to .C1. We now see that x2″ has flow in from ,x7′, so now .C1={x1,x2,x3,x7}. Vertex x7″ has no flow into it, so the building of the first chain stops. The first vertex we haven't placed into a chain is ,x4, so we note that (x4′,x5″) is full, placing x4 and x5 in chain .C2. We then look from x5′ and see no flow leaving. However, there is flow into x4″ from ,x6′, so x6 is added to .C2. There is no flow out of ,x6″, so .C2={x4,x5,x6}. Now the first point not in a chain is ,x8, so we use the flow from x8′ to x9″ to place x8 and x9 in chain .C3. Again, no flow out of ,x9′, so we look to ,x8″, which is receiving flow

from .x10″. Adding x10 to C3 gives ,C3={x8,x9,x10}, and since every point is now in a chain, we may stop.

Even once we see that the above process does in fact generate a chain partition, it is not immediately clear that it's a minimum chain partition. For this, we need to find an antichain of as many points as there are chains in our partition. (In the example we've been using, we need to find a three-element antichain.) This is where tracking the labeled vertices comes in handy. Suppose we have determined a chain C={x1<x2<⋯<xk} using the network flow. Since x1 is the minimal element of this chain, there is no flow into x1″ and hence no flow out of .x1″. Since T is unlabeled, this must mean that x1″ is unlabeled. Similarly, xk is the maximal element of ,C, so there is no flow out of .xk′. Thus, xk′ is labeled. Now considering the sequence of vertices

xk′,xk″,xk−1′,xk−1″,…,x2′,x2″,x1′,x1″,

there must be a place where the vertices switch from being labeled to unlabeled. This must happen with xi′ labeled and xi″ unlabeled. To see why, suppose that xi′ and xi″ are both unlabeled while xi+1′ and xi+1″ are both labeled. Because xi and xi+1 are consecutive in ,C, there is flow on .(xi′,xi+1″). Therefore, when scanning from ,xi+1″, the vertex xi′ would be labeled. For each chain of the chain partition, we then take the first element y for which y′ is labeled and y″ is unlabeled to form an antichain .A={y1,…,yw}. To see that A is an antichain, notice that if ,yi<yj, then (yi′,yj″) is an edge in the network. Therefore, the scan from yi′ would label .yj″. Using this process, we find that a maximum antichain in our example is .

7.4 PÓLYA'S ENUMERATION THEOREM

In this chapter, we introduce a powerful enumeration technique generally referred to as Pólya's enumeration theorem . Pólya's approach to counting allows us to use symmetries (such as those of geometric objects like polygons) to form generating functions. These generating functions can then be used to answer combinatorial questions such as

How many different necklaces of six beads can be formed using red, blue and green beads? What about 500-bead necklaces?

How many musical scales consisting of 6 notes are there?

How many isomers of the compound xylenol, CHCHOH,C6H3 (CH3)2(OH), are there? What about CH?CnH2n+2? (In chemistry, **isomers** are chemical compounds with the same number of molecules of each element but with different arrangements of those molecules.)

How many nonisomorphic graphs are there on four vertices? How many of them have three edges? What about on 1000 vertices with 257,000 edges? How many r-regular graphs are there on 40 vertices? (A graph is ***r*-regular** if every vertex has degree .r.)

To use Pólya's techniques, we will require the idea of a permutation group. However, our treatment will be self-contained and driven by examples. We begin with a simplified version of the first question above.

Coloring the Vertices of a Square

Let's begin by coloring the vertices of a square using white and gold. If we fix the position of the square in the plane, there are 24=16 different colorings. These colorings are shown in Figure 3.1.

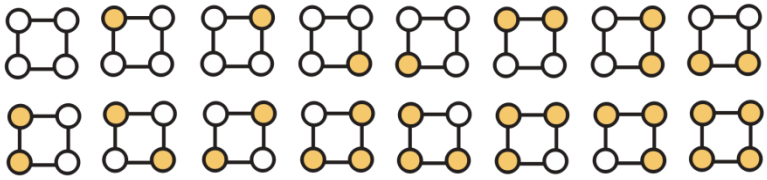


Figure 7.1. The 16 colorings of the vertices of a square.

However, if we think of the square as a metal frame with a white bead or a gold bead at each corner and allow the frame to be rotated and flipped over, we realize that many of these colorings are equivalent. For instance, if we flip coloring C7 over about the vertical line dividing the square in half, we obtain coloring .C9. If we rotate coloring C2 clockwise by ,90∘, we obtain coloring .C3. In many cases, we want to consider such equivalent colorings as a single coloring. (Recall our motivating example of necklaces made of colored beads. It makes little sense to differentiate between two necklaces if one can be rotated and flipped to become the other.)

To systematically determine how many of the colorings shown in Figure 3.1 are not equivalent, we must think about the transformations we can apply to the square and what each does to the colorings. Before examining the transformations' effects on the colorings, let's take a moment to see how they rearrange the vertices. To do this, we consider the upper-left vertex to be ,1, the upper-right vertex to be ,2, the lower-right vertex to be ,3, and the lower-left vertex to be .4. We denote the clockwise rotation by 90∘ by r1 and see that r1 sends the vertex in position 1 to position ,2, the vertex in position 2 to position ,3, the vertex in position 3 to position ,4, and the vertex in position 4 to position .1. For brevity, we will write ,r1(1)=2, ,r1(2)=3, etc. We can also rotate the square clockwise by 180∘ and denote that rotation by .r2. In this case, we find that ,r2(1)=3, ,r2(2)=4, ,r2(3)=1, and .r2(4)=2. Notice that we can achieve the transformation r2 by doing r1 twice in succession. Furthermore, the clockwise rotation by ,270∘, ,r3, can be achieved by doing r1 three times in succession. (Counterclockwise rotations can be avoided by noting that they have the same effect as a clockwise rotation, although by a different angle.)

When it comes to flipping the square, there are four axes about which we can flip it: vertical, horizontal, positive-slope diagonal, and negative-slope diagonal. We denote these flips by ,v, ,h, ,p, and ,n, respectively. Now notice that ,v(1)=2, ,v(2)=1, ,v(3)=4, and .v(4)=3. For the flip about the

horizontal axis, we have ,h(1)=4, ,h(2)=3, ,h(3)=2, and .h(4)=1. For ,p, we have ,p(1)=3, ,p(2)=2, ,p(3)=1, and .p(4)=4. Finally,for n we find ,n(1)=1, ,n (2)=4, ,n(3)=3, and .n(4)=2. There is one more transformation that we must mention; the transformation that does nothing to the square is called the **identity transformation**, denoted .ι. It has ,ι(1)=1, ,ι(2)=2, ,ι (3)=3, and .ι(4)=4.

Now that we've identified the eight transformations of the square, let's make a table showing which colorings from Figure 7.1 are left unchanged by the application of each transformation. Not surprisingly, the identity transformation leaves all of the colorings unchanged. Because r1 moves the vertices cyclically, we see that only C1 and C16 remain unchanged when it is applied. Any coloring with more than one color would have a vertex of one color moved to one of the other color. Let's consider which colorings are fixed by ,v, the flip about the vertical axis. For this to happen, the color at position 1 must be the same as the color at position ,2, and the color at position 3 must be the same as the color at position .4. Thus, we would expect to find 2·2=4 colorings unchanged by .v. Examining Figure 7.1, we see that these colorings are ,C1, ,C6, ,C8, and .C16. Performing a similar analysis for the remaining five transformations leads to Figure 7.2.

Transformation	Fixed colorings
ι	All 16
r1	,C1, C16
r2	,C1, ,C10, ,C11, C16
r3	,C1, C16
v	,C1, ,C6, ,C8, C16
h	,C1, ,C7, ,C9, C16
p	,C1, ,C3, ,C5, ,C10, ,C11, ,C13, ,C15, C16
n	,C1, ,C2, ,C4, ,C10, ,C11, ,C12, ,C14, C16

Figure 7**.2.** Colorings fixed by transformations of the square

At this point, it's natural to ask where this is going. After all, we're trying to count the number of nonequivalent colorings, and Figure 7.2 makes no effort to group colorings based on how a transformation changes one coloring to another. It turns out that there is a useful connection between counting the nonequivalent colorings and determining the number of colorings fixed by each transformation. To develop this connection, we first need to discuss the equivalence relation created by the action of the

transformations of the square on the set C of all 2-colorings of the square. (Refer to Section B.13 for a refresher on the definition of equivalence relation.) To do this, notice that applying a transformation to a square with colored vertices results in another square with colored vertices. For instance, applying the transformation r1 to a square colored as in C12 results in a square colored as in .C13. We say that the transformations of the square **act** on the set C of colorings. We denote this action by adding a star to the transformation name. For instance, r1∗(C12)=C13 and .v∗(C10)=C11.

If τ is a transformation of the square with ,τ∗(Ci)=Cj, then we say colorings Ci and Cj are **equivalent** and write .Ci∼Cj. Since ι∗(C)=C for all ,C∈C, ∼ is reflexive. If τ1∗(Ci)=Cj and ,τ2∗(Cj)=Ck, then ,τ2∗(τ1∗(Ci))=Ck, so ∼ is transitive. To complete our verification that ∼ is an equivalence relation, we must establish that it is symmetric. For this, we require the notion of the **inverse** of a transformation ,τ, which is simply the transformation τ−1 that undoes whatever τ did. For instance, the inverse of r1 is the counterclockwise rotation by ,90∘, which has the same effect on the location of the vertices as .r3. If ,τ∗(Ci)=Cj, then ,τ−1∗(Cj)=Ci, so ∼ is symmetric.

Before proceeding to establish the connection between the number of nonequivalent colorings (equivalence classes under ∼) and the number of colorings fixed by a transformation in full generality, let's see how it looks for our example. In looking at Figure 7.1, you should notice that ∼ partitions C into six equivalence classes. Two contain one coloring each (the all white and all gold colorings). One contains two colorings (C10 and C11). Finally, three contain four colorings each (one gold vertex, one white vertex, and the remaining four with two vertices of each color). Now look again at Figure 7.2 and add up the number of colorings fixed by each transformation. In doing this, we obtain ,48, and when 48 is divided by the number of transformations (8), we get 6 (the number of equivalence classes)! It turns out that this is far from a fluke, as we will soon see. First, however, we introduce the concept of a permutation group to generalize our set of transformations of the square.

❖❖❖❖