14

Applications of Subgraph Enumeration

Author: Fred J. Rispoli, Department of Mathematics, Dowling College.

Prerequisites: The prerequisites for this chapter are counting, probability, graphs, and trees. See Sections 5.1, 5.3, and 6.1, and Chapters 9 and 10 of *Discrete Mathematics and Its Applications*.

Introduction

Many applications of graph theory involve enumerating subgraphs to determine the number of subgraphs satisfying various properties, or to find a subgraph satisfying various properties. Some interesting examples are:

Example 1 How many distinct paths are there joining locations v_1 to v_3 in the transportation network represented by the graph in Figure 1? Given the length, l, and cost, c, of each edge, as displayed in Figure 1, does there exist a path joining v_1 to v_3 with total length 15 or less, and total cost \$40 or less? □

Example 2 How many different isomers are there of the saturated hydrocarbons C_5H_{12}? □

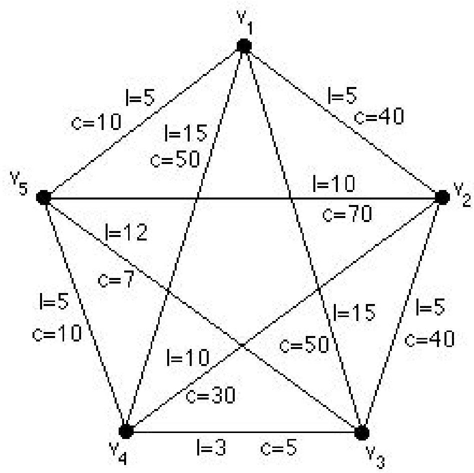


Figure 1. A transportation network.

Example 3 How many ways are there to construct an electrical network which connects all of the nodes in the network and uses the smallest number of wires possible? □

Example 4 A salesman wishes to visit a number of cities and return to the starting point in such a way that each city is visited exactly once. In how many ways can this be done? If a route is selected at random, what is the probability that two given cities are visited in succession? Given the distances between cities, what route should be chosen so that the total distance covered is as short as possible? □

In this chapter we will discuss how to solve these problems, and other similar problems. The approach is to define each problem in terms of subgraphs of K_n, the complete graph on n vertices, and then derive a method to generate and count the set of all subgraphs of K_n satisfying the required conditions. In particular, we will count the number of simple paths joining any pair of vertices in K_n, the number of spanning trees in K_n, the number of Hamilton circuits in K_n, and the number of perfect matches in K_n. These counts will then be used to determine the algorithmic complexity of exhaustive search procedures, to compute various probabilities, and to solve some counting problems.

Counting Paths

We begin by discussing paths and enumeration problems involving paths. Given any graph $G = (V, E)$ and a positive integer n, a *path of length* n from vertex u

to vertex v is a sequence of edges e_1, e_2,... , e_n of E such that $e_1 = \{x_0, x_1\}$, $e_2 = \{x_1, x_2\}$,... , $e_n = \{x_{n-1}, x_n\}$ where $x_0 = u$ and $x_n = v$. A path is *simple* if it does not contain the same edge more than once.

Since any path of a graph $G = (V, E)$ consists of a subset of vertices of V and a subset of edges of E, a path is a subgraph of G. We will only consider simple paths in this chapter, and will omit the term “simple”.

Theorem 1 and its proof allow us to solve Example 1 of the introduction. We use the notation

$$P(n, r) = n(n-1)(n-2)\dots(n-r+1) \qquad \text{if } r > 0,$$

and $P(n, 0) = 1$.

Theorem 1 Given any two vertices in K_n, the complete graph with n vertices, the number of paths joining them is

$$\sum_{k=1}^{n-1} P(n-2, k-1) = O(n^{n-2}).$$

Proof: Let K_n have vertex set $V = \{v_1, v_2, \dots, v_n\}$ and let v_i and v_j be any pair of vertices in K_n. We count the number of paths joining v_i to v_j of length k, by establishing a one-to-one correspondence from the set of paths joining v_i to v_j of length k to the set of $(k+1)$-permutations of the set $\{1, 2, \dots, n\}$ which begin with i and end with j.

Given any path P of length k joining v_i to v_j, to obtain a $(k+1)$-permutation simply list the subscripts of the vertices that P visits as it is traversed from v_i to v_j. Conversely, let $i_1 i_2 \dots i_{k+1}$ be a $(k+1)$-permutation of $\{1, 2, \dots, n\}$ such that $i_1 = i$ and $i_{k+1} = j$. The corresponding path P is made up of edges $\{v_{i_s}, v_{i_{s+1}}\}$, for $s = 1, 2, \dots, k$. Since every path joining v_i to v_j of length k corresponds to a unique $(k+1)$-permutation of $\{1, 2, \dots, n\}$ beginning with i and ending with j, and every $(k+1)$-permutation of $\{1, 2, \dots, n\}$ which begins with i and ends with j of length k corresponds to a unique path joining v_i to v_j, the correspondence between these two sets is one-to-one. The number of $(k+1)$-permutations of $\{1, 2, \dots, n\}$ which begin with i and end with j is $P(n-2, k-1)$. Thus the total number of paths joining v_i to v_j is obtained by summing $P(n-2, k-1)$ as k varies from 1 to $n-1$.

To obtain the big-O estimate, note that

$$P(n-2, k-1) = (n-2)(n-3)\dots(n-k) \le nn\dots n = n^{k-1}.$$

Hence,

$$\sum_{k=1}^{n-1} P(n-2, k-1) \le n^0 + n^1 + \cdots + n^{n-2} = O(n^{n-2}). \qquad \blacksquare$$

The proof of Theorem 1 indicates how to enumerate all paths joining any pair of vertices in K_n by generating permutations. (A method for generating permutations is given in Section 4.7 of the text.) This allows us to solve Example 1 using an exhaustive search.

Solution to Example 1. By Theorem 1, there are

$$\sum_{k=1}^{4} P(3, k-1) = 1 + 3 + 6 + 6 = 16$$

paths joining v_1 to v_3. To determine if there is a path from v_1 to v_3 with total length 15 or less and total cost \$40 or less, we list each $(k+1)$-permutation of $\{1, 2, 3, 4, 5\}$ beginning with 1 and ending with 3 corresponding to a path, along with its total length and total cost for $k = 1, 2, 3, 4$.

Paths with 1 or 2 edges

13	length: 15,	cost: 50
123	length: 10,	cost: 80
143	length: 18,	cost: 55
153	length: 17,	cost: 17

Paths with 3 edges

1243	length: 18,	cost: 75
1423	length: 30,	cost: 120
1253	length: 27,	cost: 117
1523	length: 20,	cost: 120
1453	length: 32,	cost: 67
1543	length: 13,	cost: 25

Paths with 4 edges

12453	length: 32,	cost: 87
12543	length: 23,	cost: 125
14253	length: 47,	cost: 157
15243	length: 28,	cost: 115
14523	length: 35,	cost: 170
15423	length: 25,	cost: 90

This shows that there is one path joining v_1 to v_3 which has total length 15 or less, and total cost \$40 or less, namely the path corresponding to 1543. □

Example 1 is an example of a *shortest weight-constrained path problem*, which we now define.

Shortest Weight-Constrained Path Problem: Given positive integers W and L, and a weighted graph $G = (V, E)$ with weights $w(e)$ and lengths $l(e)$, which are both positive integers, for all $e \in E$. Is there a path between two given vertices with weight $\leq W$ and length $\leq L$?

There is no known algorithm with polynomial complexity which solves the shortest weight-constrained problem. (See [2] in the suggested readings for an explanation why.) Thus, using an exhaustive search is a useful method for solving such a problem, as long as n is not too large. Theorem 1 tells us precisely just how large n can be. For example, suppose $n = 10$, and each path along with its weight and length can be computed in 10^{-4} seconds of computer time. Then, by Theorem 1, the are at most 10^8 paths to consider in K_{10}. So the problem can be solved in at most $10^8 \cdot 10^{-4} = 10^4$ seconds, or roughly 3 hours. Whereas if $n = 20$, the amount of computer time required is at most $20^{18} \cdot 10^{-4}$ seconds, or roughly $8 \cdot 10^{12}$ years.

A problem closely related to the above problem is the well-known shortest path problem, defined as follows.

> **Shortest Path Problem**: Given a weighted graph, find a path between two given vertices that has the smallest possible weight.

The shortest path problem may also be solved using an exhaustive search. However, Dijkstra's algorithm is a much better method. (See Section 7.6 of the text for a description of the algorithm). This is true because Dijkstra's algorithm requires $O(n^2)$ operations (additions and comparisons) to solve the problem. Whereas, if an exhaustive search is used, the number of additions used to compute the weight of each path is $O(n)$, and, by Theorem 1, there are $O(n^{n-2})$ such paths to examine. Thus, the number of additions required to compute the weight of all paths is $O(n^{n-1})$. This shows that Dijkstra's algorithm is much more efficient than an exhaustive search.

Counting Spanning Trees

In this section we shall study the enumeration of spanning trees. Recall that a *tree* is a connected graph with no circuits. If $G = (V, E)$ is a graph, a *spanning tree* of G is a subgraph of G that is a tree containing every vertex of V.

Spanning trees were first used by the German physicist Gustav Kirchoff who developed the theory of trees in 1847. Kirchoff used spanning trees to solve systems of simultaneous linear equations which give the current in each branch and around each circuit of an electrical network.

In 1857, the English mathematician Arthur Cayley independently discovered trees when he was trying to enumerate all isomers for certain hydrocarbons. Hydrocarbon molecules are composed of carbon and hydrogen atoms where each carbon atom can form up to four chemical bonds with other atoms, and each hydrogen atom can form one bond with another atom. A saturated hydrocarbon is one that contains the maximum number of hydrogen atoms for a given number of carbon atoms. Cayley showed that if a saturated hydrocarbon has n carbon atoms, then it must have $2n + 2$ hydrogen atoms, and hence has the chemical formula C_nH_{2n+2}. His approach was to represent the structure of a hydrocarbon molecule using a graph in which the vertices represent atoms of hydrogen (H) and carbon (C), and the edges represent the chemical bonds between the atoms (see Figure 2). He then showed that any graph representing a saturated hydrocarbon must be a tree. Thus, any graph representing the saturated hydrocarbon C_nH_{2n+2} must be a tree with n vertices of degree 4 and $2n + 2$ vertices of degree 1.

When two molecules have the same chemical formula but different chemical bonds they are called isomers. One can enumerate the isomers of C_nH_{2n+2} by enumerating the nonisomorphic trees with n vertices of degree 4 and $2n + 2$ vertices of degree 1. The problem may be simplified further by removing vertices representing hydrogen atoms, thereby obtaining a subgraph called the *carbon-graph*. The vertices of carbon-graphs all represent carbon atoms and the edges represent chemical bonds between the carbon atoms. Given any graph representing a saturated hydrocarbon C_nH_{2n+2}, removing all vertices of degree 1 leaves a tree, namely, the carbon-graph, containing n vertices which all have degree at most 4.

Conversely, given any tree T with n vertices such that every vertex has degree at most 4, edges may be added to T to obtain a tree, T', in which all of the original n vertices have degree 4. So T' represents a molecule with the chemical formula C_nH_{2n+2}. Since any tree with n vertices such that every vertex has degree at most 4 corresponds to a unique isomer with chemical formula C_nH_{2n+2}, and vice versa, there is a one-to-one correspondence between the isomers of C_nH_{2n+2} and the nonisomorphic trees with n vertices such that every vertex has degree at most 4. We shall exploit this fact to solve the problem posed in Example 2.

Solution to Example 2: Figure 2 gives all nonisomorphic trees with 5 vertices such that every vertex has degree 4 or less. The corresponding isomer is given below each tree along with its name. □

Cayley did not immediately succeed at obtaining a formula, in terms of n, for the number of isomers of C_nH_{2n+2}. So he altered the problem until he was able to obtain such a formula for trees satisfying various conditions. In 1889 he discovered Theorem 2, known as Cayley's Theorem, which states: the number of spanning trees of K_n is n^{n-2}. The proof we will give was discovered by H. Prüfer in 1918. Several other completely different proofs are also known. (See [5] in the suggested readings.) The idea behind the proof is to establish a one-to-one correspondence from the set of all spanning trees of K_n to the set of ordered $(n-2)$-tuples $(a_1, a_2, \ldots, a_{n-2})$, where each a_i is an integer satisfying $1 \leq a_i \leq n$. Given any spanning tree T of K_n, we obtain an $(n-2)$-tuple

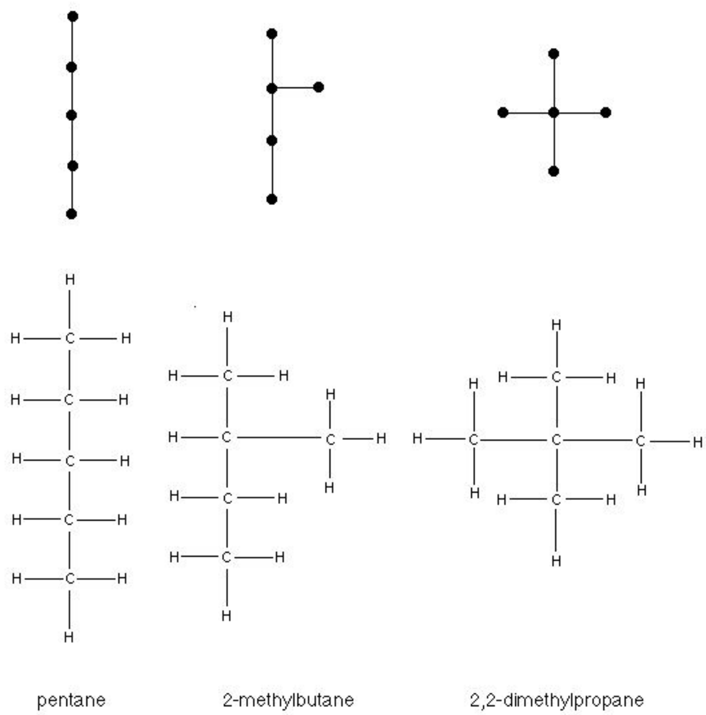


Figure 2. Trees and isomers.

as follows. Choose a vertex of degree 1 . (The existence of such a vertex is proved in Exercise 5.) Assume the vertices are labeled $v_1, \ldots, v_n$ and remove the vertex of degree 1 with with the smallest subscript, along with its incident edge. Let a_1 be the subscript of the unique vertex which was adjacent to the removed vertex. Repeat this procedure on the remaining tree with $n-1$ vertices to determine a_2. Iterate this procedure until there are only two vertices left, thereby obtaining the $(n-2)$-tuple, $(a_1, a_2, \ldots, a_{n-2})$.

Example 5 Find the 5-tuple which corresponds to the spanning tree given in Figure 3.

Solution: Figure 3 corresponds to the 5-tuple $(2, 3, 4, 3, 6)$. To see this, notice that v_1 is the vertex with the smallest subscript which has degree 1 and v_2 is adjacent to v_1, thus $a_1 = 2$. Now remove edge $\{v_1, v_2\}$. In the reduced graph, v_2 is the vertex with the smallest subscript which has degree 1. Vertex v_3 is adjacent to v_2; thus $a_2 = 3$. Now remove edge $\{v_2, v_3\}$. Iterating this procedure gives the result. □

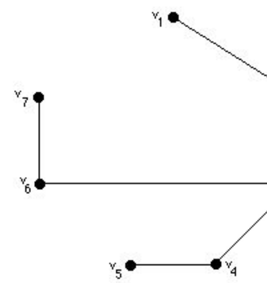


Figure 3. A spanning tree.

To obtain a spanning tree of K_n, given any $(n-2)$-tuple, begin with the list $\{1, 2, \ldots, n\}$. Find the smallest number, i, in the list but not in the $(n-2)$-tuple and take the first number in the $(n-2)$-tuple, a_1. Then add the edge joining the vertices v_i and v_{a_1}. Remove i from the list and a_1 from the $(n-2)$-tuple and repeat the procedure. Iterate until there are only two numbers left in the list, then join the vertices with these subscripts. The graph G thus obtained does not contain any circuits. For if C is a circuit in G, let $\{u, v\}$ be the last edge in C adjoined to G. Then both u and v were included in an edge previously adjoined to G. If the first time u was included in an edge adjoined to G, u was from the list, then u was not in the tuple, and was crossed off the list. So it may not be an endpoint of any edge subsequently adjoined to G. Thus u must have been from the tuple the first time it was an endpoint of an edge adjoined to G. Similarly, v must have been from the tuple the first time it was an endpoint of an edge adjoined to G. Now let $v_1, v_2, \ldots, v_k$ be the vertices visited by C, where $u = v_1$ and $v = v_k$, as C is traversed from u to v without passing through edge $\{u, v\}$. Since v_1 was in the tuple when edge $\{v_1, v_2\}$ was adjoined to G, v_2 must have been from the list. This implies v_2 must have been from the tuple when $\{v_2, v_3\}$ is adjoined to G, hence, v_3 is from the list when $\{v_2, v_3\}$ is adjoined to G. Similarly, v_4 must have been from the list when $\{v_3, v_4\}$ was adjoined to G, and so on. But this implies that $v_k = v$ was from the list when $\{v_{k-1}, v_k\}$ was adjoined to G, a contradiction. Thus G can not have any circuits. Exercise 6 shows that any graph with n vertices, $n-1$ edges, and no circuits must be a tree. Thus G is a spanning tree of K_n. We have shown that every spanning tree of K_n corresponds to a unique $(n-2)$-tuple and every $(n-2)$-tuple corresponds to a unique spanning tree of K_n. Therefore, there is a one-to-one correspondence between these two sets.

Example 6 Find the spanning tree of K_7 which corresponds to the 5-tuple $(7, 2, 1, 2, 1)$.

Solution: The spanning tree is given in Figure 4. To see why, start with the list $\{1, 2, 3, 4, 5, 6, 7\}$. The number 3 is the smallest number in the list but not

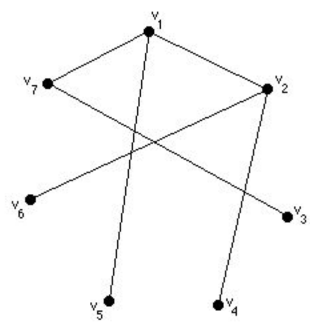


Figure 4. The spanning tree corresponding to (7,2,1,2,1).

in $(7, 2, 1, 2, 1)$, and 7 is the first number in the 5-tuple. So we adjoin edge (v_3, v_7). Now remove 3 from the list to obtain the new list $\{1, 2, 4, 5, 6, 7\}$, and remove 7 from the 5-tuple to obtain the 4-tuple $(2, 1, 2, 1)$. The number 4 is the smallest number in the list but not in $(2, 1, 2, 1)$, and 2 is the first number in the 4-tuple. So we adjoin edge (v_4, v_2). Iterate this procedure until there are only two numbers left, namely 1 and 7. Now adjoin edge (v_1, v_7) to obtain the tree. □

Theorem 2 Cayley's Theorem The number of spanning trees of K_n is n^{n-2}.

Proof: Construct the one-to-one correspondence outlined above from the set of all spanning trees of K_n with vertices $\{v_1, v_2, \ldots, v_n\}$, to the set of all $(n-2)$-tuples $(a_1, a_2, \ldots, a_{n-2})$, where each a_i is an integer satisfying $1 \leq a_i \leq n$. The count is obtained by observing that there are n^{n-2} such $(n-2)$-tuples, since there are n ways to select each a_i. ■

Examples 7 and 8 involve a direct application of Cayley's Theorem.

Example 7 How many ways are there to construct an electrical network with 12 nodes which connects all of the nodes using the fewest possible number of wires?

Solution: Any electrical network consisting of 12 nodes and wires connecting the nodes can be represented by a subgraph of K_{12}, where each node is represented by a vertex, and each wire is represented by an edge. The graph representing any electrical network which connects all 12 nodes and uses the fewest number of wires, must be a connected graph with no circuits. Hence, it must be a spanning tree of K_{12}. By Cayley's Theorem, there are 12^{10} spanning trees of K_{12}. Thus, there are 12^{10} ways to construct the electrical network. □

Example 8 Determine the probability that a spanning tree selected at random from K_n does not contain a given edge e.

Solution: Exercise 22 shows that the number of spanning trees of the graph obtained by deleting the edge e from K_n is $(n-2)n^{n-3}$. By Cayley's Theorem, the number of spanning trees of K_n is n^{n-2}. So the probability is

$$\frac{(n-2)n^{n-3}}{n^{n-2}} = \frac{n-2}{n} = 1 - \frac{2}{n}.$$ □

The proof of Theorem 2 describes how the set of all spanning trees of K_n may be generated by generating $(n-2)$-tuples. We now describe an algorithm which generates n-tuples $(a_1, a_2, \ldots, a_n)$, where each a_i is an integer satisfying $r \le a_i \le s$, where r and s are any integers satisfying $r < s$. The algorithm is based on the *lexicographic ordering* of n-tuples. In this ordering, the n-tuple $(a_1, a_2, \ldots, a_n)$ precedes the n-tuple $(b_1, b_2, \ldots, b_n)$ if, for some k with $1 \le k \le n$, $a_1 = b_1, a_2 = b_2, \ldots, a_{k-1} = b_{k-1}$, and $a_k < b_k$. In words, an n-tuple precedes a second n-tuple if the number in this n-tuple in the first position where the two n-tuples disagree is smaller than the number in that position in the second n-tuple. For example, the 5-tuple $a = (2,3,1,5,7)$ precedes the 5-tuple $b = (2,3,1,6,2)$, since $a_1 = b_1$, $a_2 = b_2$, $a_3 = b_3$, but $a_4 < b_4$.

Example 9 What is the next largest 5-tuple in lexicographic order after $(3,2,4,7,7)$ in the set of all 5-tuples $(a_1, a_2, a_3, a_4, a_5)$, with $1 \le a_i \le 7$?

Solution: To find the next largest 5-tuple, find the largest subscript i such that $a_i < 7$, which is $i = 3$. Then add one to a_3. This gives the 5-tuple $(3,2,5,7,7)$. Any other 5-tuple $(a_1, a_2, a_3, a_4, a_5)$ that is larger than $(3,2,5,7,7)$ satisfies either $a_1 > 3$, or $a_1 = 3$ and $a_2 > 2$, or $a_1 = 3$, $a_2 = 2$, and $a_3 > 5$. In every case $(a_1, a_2, a_3, a_4, a_5)$ is larger than $(3,2,4,7,7)$. Therefore, $(3,2,5,7,7)$ is the next largest 5-tuple. □

Algorithm 1 displays the pseudocode description for finding the next largest n-tuple after an n-tuple that is not $(s, s, \ldots, s)$, which is the largest n-tuple.

Next we look at a problem for which there is no known algorithm. Given any weighted graph G and any spanning tree T of G, define the **range** of T to be the weight of the edge in T with the largest weight minus the weight of the edge in T with the smallest weight.

Example 10 Use an exhaustive search to find a spanning tree with the smallest possible range for the graph in Figure 5.

ALGORITHM 1. Generating the next largest n-tuple in lexicographic order.

procedure *next n-tuple* $((a_1, a_2, \ldots, a_n)$: n-tuple of integers between r and s, $r < s$, not equal to $(s, s, \ldots, s))$
$j := 1$
for $i := 1$ **to** n
  if $a_i < s$ **then** $j := i$
$\{j$ is the largest subscript with $a_j < s\}$
$a_j := a_j + 1$
$\{(a_1, a_2, \ldots, a_n)$ is now the next largest n-tuple$\}$

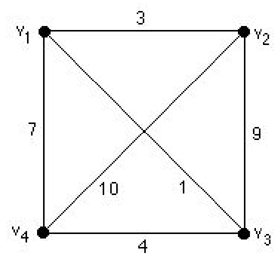


Figure 5. A weighted graph.

Solution: By Cayley's Theorem there are $4^2 = 16$ spanning trees of K_4. We list each 2-tuple corresponding to a spanning tree of K_4, along with the corresponding range of each spanning tree:

(1,1) range: 6	(2,1) range: 6	(3,1) range: 8	(4,1) range: 9
(1,2) range: 9	(2,2) range: 7	(3,2) range: 9	(4,2) range: 3
(1,3) range: 3	(2,3) range: 6	(3,3) range: 8	(4,3) range: 5
(1,4) range: 4	(2,4) range: 7	(3,4) range: 9	(4,4) range: 6

This shows that the spanning trees corresponding to $(1, 3)$ or $(4, 2)$, having range 3, are spanning trees with the smallest possible range. □

Another important problem involving spanning trees is the well known *minimal spanning tree problem.*

Minimal Spanning Tree Problem: Given a weighted graph, find a spanning tree with the smallest possible weight.

This problem may also be solved using an exhaustive search. However this approach should be avoided in this case since there are several well known algorithms which are much more efficient. For example, Prim's algorithm is known to find a minimal spanning tree for a graph with n vertices using $O(n^2)$ comparisons and no additions. (See Section 10.5 of *Discrete Mathematics and Its Applications* for a description of Prim's algorithm.) Whereas if an exhaustive search were used, the number of additions required to compute the weights of each spanning tree is $n - 2 = O(n)$. By Cayley's Theorem, this must be performed at most $n^{n-2} = O(n^{n-2})$ times. Thus, the number of additions required to compute the weights of the spanning trees of K_n is $O(n^{n-1})$. This shows that Prim's algorithm is much more efficient than an exhaustive search.

Counting Hamilton Circuits

Next we discuss Hamilton circuits and some related problems. Given any graph $G = (V, E)$, a path is called a *circuit* if it begins and ends at the same vertex. A circuit $x_0, x_1, \ldots, x_n$, where $x_0 = x_n$, is called a *Hamilton circuit* if $V = \{x_0, x_1, \ldots, x_n\}$ and $x_i \neq x_j$, for $0 \leq i < j \leq n$.

The terminology is due to the Irish mathematician Sir William Rowan Hamilton, who was a child prodigy, and is famous for his contributions in algebra. Perhaps his most famous discovery was the existence of algebraic systems in which the commutative law for multiplication ($ab = ba$) does not hold. His *algebra of quaternions* , as it is now known, can be expressed in terms of Hamilton circuits on the regular dodecahedron (a regular solid with 20 vertices and 12 regular pentagons as faces). Hamilton's discovery lead to a puzzle in which the vertices of the dodecahedron are labeled with different cities of the world. The player is challenged to start at any city, travel "around the world", and return to the starting point, visiting each of the other 19 cities exactly once. In the puzzle that was marketed in 1859, the player must find a Hamilton circuit starting with five given initial cities.

An important problem involving Hamilton circuits is the *traveling salesman problem*. In such problems, a salesman wishes to visit a number of cities and return to the starting point, in such a way that each city is visited exactly once, and the total distance covered is as small as possible. The problem may also be stated using graph terminology.

> **Traveling Salesman Problem**: Given a weighted graph, find a Hamilton circuit that has the smallest possible weight.

The origin of the traveling salesman problem is somewhat obscure. George Dantzig, Ray Fulkerson, and Selmer Johnson were among the first mathematicians who studied the problem in 1954. They showed that a certain Hamilton circuit of a graph representing 49 cities, one in each of the 48 contiguous states and Washington D.C., has the shortest distance. (See [1] in the suggested readings.) Since then, many researchers have worked on the problem. However, there is no known algorithm having polynomial complexity which solves the traveling salesman problem. On the other hand, there has been a lot of progress towards finding good algorithms which either solve the problem, or find approximate solutions to the problem. (This problem is also studied in another chapter of this book. In addition, see [4] in the suggested readings for a comprehensive discussion.)

Theorem 3 and its proof allow us to solve the traveling salesman problem using an exhaustive search, as well as determine probabilities concerning Hamilton circuits selected at random. To enumerate the Hamilton circuits in K_n, we establish a one-to-one correspondence that characterizes the set of all Hamilton circuits of K_n in terms of permutations. The idea behind the correspondence is to label the vertices of K_n, using $v_1, v_2, \ldots, v_n$, and then associate a permutation of $1, 2, \ldots, n$ to every Hamilton circuit using the subscripts of the vertices v_i.

For example, consider the circuit C given in Figure 6. We can associate the permutation 13425 with C. However, the permutations 34251, 42513, 25134, 51342, and the permutations 15243, 52431, 24315, 43152, 31524 all give rise to the same circuit, C. To obtain a one-to-one correspondence, we will pick an arbitrary starting point v_1, and associate the permutation beginning with 1, in which the second number is smaller than the last. According to this rule, 13425 is the only permutation associated to C.

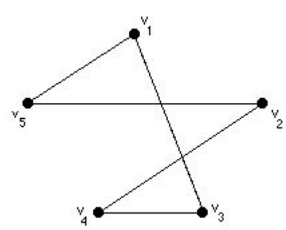


Figure 6. A circuit C.

Theorem 3 The number of Hamilton circuits in K_n is $\frac{1}{2}(n-1)!$.

Proof: We show that the set of all Hamilton circuits in K_n is in one-to-one correspondence with the set of all permutations σ of $\{1, 2, \ldots, n\}$ beginning with 1, such that the second number of σ is smaller than the last. Let C be any Hamilton circuit in K_n. Take the vertex v_1 and let v_j and v_k be the two vertices which are joined to v_1 by edges in C. Clearly $j \neq k$, so assume $j < k$. To obtain the permutation σ corresponding to C let the ith element of σ be the subscript of the ith vertex visited by C as C is traversed by beginning at v_1 and proceeding in the direction such that v_1 is followed by v_j.

Conversely, given any permutation σ of $\{1, 2, \ldots, n\}$ beginning with 1, such that the second number of σ is smaller than the last, the Hamilton circuit corresponding to σ is obtained by starting at v_1, then visiting the vertices $\{v_2, v_3, \ldots, v_n\}$ in the order prescribed by σ.

The number of permutations of $\{1, 2, 3, \ldots, n\}$ beginning with 1, such that the second number is smaller than the last number, is equal to the number of ways to choose the second and last numbers times the number of ways to choose the remaining $n - 3$ numbers. Note that there is only one way to choose the first number since it must be 1. Moreover, when we choose two numbers, say a and b, one is larger than the other, so there is only one way to place them as second and last elements in the permutation. Therefore, the count is $C(n-1, 2)(n-3)! = \frac{1}{2}(n-1)!$. ■

We are now ready to answer the questions posed in Example 4 of the introduction.

Example 11 A salesman wishes to visit all the locations listed in Figure 7 and return to the starting point in such a way that each city is visited exactly once. If such a route is selected at random, what is the probability that the route visits v_1 and v_2 in succession?

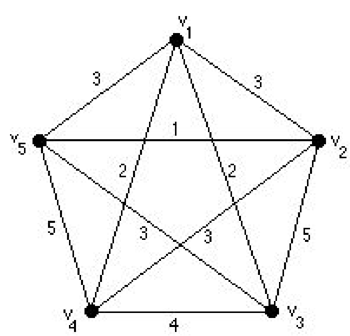


Figure 7. Salesman's network.

Solution: The number of Hamilton circuits which visit v_1 and v_2 in succession is obtained by observing that if a Hamilton circuit passes through v_1 followed by v_2, then there are three ways to visit the next vertex, two ways to visit the next, and one way to visit the last. Thus, there are a total of $3 \cdot 2 \cdot 1 = 6$ Hamilton circuits. By Theorem 3, there are $\frac{1}{2}4! = 12$ Hamilton circuits in K_5. So the probability is $6/12 = 1/2$. □

Example 12 Use an exhaustive search to find a Hamilton circuit of smallest weight in the graph of Figure 7 by generating the permutations of $\{1,2,3,4,5\}$ which begin with 1, such that the second number is smaller than the last.

Solution: By Theorem 3, there are $\frac{1}{2}4! = 12$ Hamilton circuits in K_5. We give each permutation along with its corresponding weight.

12345	weight: 20	12354	weight: 18	12435	weight: 16
12453	weight: 16	12534	weight: 13	12543	weight: 15
13245	weight: 18	13254	weight: 15	13425	weight: 13
13524	weight: 11	14235	weight: 16	14325	weight: 15

This shows that the Hamilton circuit $v_1, v_3, v_5, v_2, v_4, v_1$, having weight 11, is a Hamilton circuit with the smallest possible weight. □

Theorem 3 tells us that for a the graph K_{10} there would be $\frac{1}{2}9! = 181,440$ different Hamilton circuits. If each circuit could be found and its weight computed in 10^{-4} seconds, it would require approximately 3 minutes of computer time to solve a traveling salesman problem with 10 vertices. So an exhaustive search is a reasonable way to solve the problem. However, under the same assumption, a problem with 25 vertices would require $(3 \cdot 10^{23}) \cdot 10^{-4} = 3 \cdot 10^{19}$ seconds, or roughly $9.5 \cdot 10^{11}$ years.

Counting Perfect Matches

A class of ten students must be paired off to form five study groups. How many ways can the study groups be formed? After a preliminary examination the instructor assigns a rating from 1 to 10 to each pair such that the lower the rating, the more productive the pair, in the opinion of the instructor. How can the students be paired so that the sum of the ratings of the five pairs is minimal, thus maximizing the productivity of the class? These questions concern a certain type of matching, called a *perfect matching*, which we now define.

Definition 1 A *matching* in a graph $G = (V, E)$ is a subset of edges, M, contained in E such that no two edges in M have a common endpoint. A matching M is called *perfect* if every vertex of G is an endpoint of an edge of M. □

For example, the set of all perfect matches of the graph given in Figure 8 are the matches

$$M_1 = \{\{1,2\},\{3,4\}\} \qquad M_2 = \{\{1,3\},\{2,4\}\} \qquad M_3 = \{\{1,4\},\{2,3\}\}.$$

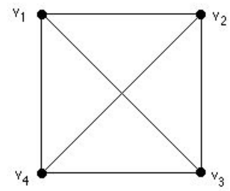


Figure 8. Finding perfect matchings.

Theorems 1, 2, and 3 were all proved by establishing a one-to-one correspondence. The following theorem uses mathematical induction to count the perfect matches in K_n.

Theorem 4 The number of perfect matches in K_n is 0 if n is odd and $(n-1)(n-3)\ldots 5\cdot 3\cdot 1 = O(n^{n/2})$ if n is even.

Proof: We will prove this theorem using mathematical induction. If $n = 1$, there is no perfect matching and if $n = 2$, then there is only 1 perfect matching.

For the induction step, assume the theorem holds for all complete graphs with k vertices, where $k < n$. It is clear that K_n has no perfect matching if n is odd, so we assume n is even. We count the number of perfect matches in K_n by considering a vertex v_1, which can be matched to any of the other $n-1$ vertices. Suppose v_1 is matched to v_2. Then remove v_1, v_2, and all the edges incident to v_1 and v_2, to obtain the graph K_{n-2} with vertices $\{v_3, v_4, \ldots, v_n\}$. By the inductive assumption, since $n-2$ is even, the number of perfect matches in K_{n-2} is $(n-3)(n-5)\cdots 5\cdot 3\cdot 1$. Since there are $n-1$ ways to match v_1, and for each of these there are $(n-3)(n-5)\cdots 5\cdot 3\cdot 1$ ways to match the remaining $n-2$ vertices, the total number of perfect matches is

$$(n-1)(n-3)\ldots 5\cdot 3\cdot 1 \le nn\ldots n = O(n^{n/2}).$$ ■

Theorem 4 can be used to answer the question posed at the beginning of this section.

Example 13 How many ways can a class of 10 students be paired off to form 5 study groups?

Solution: The number of study groups is equal to the number of perfect matches in K_{10}. By Theorem 4, this number is $9\cdot 7\cdot 5\cdot 3\cdot 1 = 945$. □

Example 14 Use an exhaustive search to find a perfect matching of minimal weight for the graph given in Figure 9 by listing all perfect matches along with their weights.

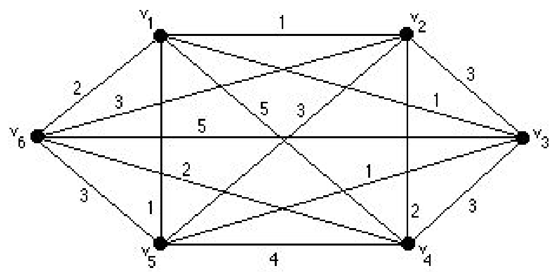


Figure 9. A weighted graph.

Solution: By Theorem 4, there are $5 \cdot 3 \cdot 1 = 15$ perfect matches in K_6. We shall list the edges in each perfect matching of K_6 along with the weight of the matching.

$\{\{1,2\},\{3,4\},\{5,6\}\}$	weight: 7	$\{\{1,4\},\{3,5\},\{2,6\}\}$	weight: 9
$\{\{1,2\},\{3,5\},\{4,6\}\}$	weight: 4	$\{\{1,5\},\{2,3\},\{4,6\}\}$	weight: 6
$\{\{1,2\},\{4,5\},\{3,6\}\}$	weight: 10	$\{\{1,5\},\{2,4\},\{3,6\}\}$	weight: 8
$\{\{1,3\},\{4,5\},\{2,6\}\}$	weight: 8	$\{\{1,5\},\{3,4\},\{2,6\}\}$	weight: 7
$\{\{1,3\},\{2,4\},\{5,6\}\}$	weight: 6	$\{\{1,6\},\{3,4\},\{2,5\}\}$	weight: 8
$\{\{1,3\},\{2,5\},\{4,6\}\}$	weight: 6	$\{\{1,6\},\{2,3\},\{4,5\}\}$	weight: 9
$\{\{1,4\},\{2,3\},\{5,6\}\}$	weight: 11	$\{\{1,6\},\{2,4\},\{3,5\}\}$	weight: 5
$\{\{1,4\},\{2,5\},\{3,6\}\}$	weight: 13		

This shows that the perfect matching $\{\{1,2\},\{3,5\},\{4,6\}\}$, with weight 4 is a perfect matching of the smallest possible weight. □

Example 14 is an example of a *perfect matching problem*, defined as follows.

Perfect Matching Problem: Given a weighted graph, find a perfect matching that has the smallest possible weight.

The proof of Theorem 4 indicates how to recursively generate the set of perfect matches of K_n. Specifically, first generate all perfect matches of K_2, use these to generate all those of K_4, use the perfect matches of K_4 to generate those of K_6, and so on.

Example 15 Use the perfect matching $\{\{1,2\},\{3,4\}\}$ in K_4 to generate 5 perfect matches of K_6.

Solution: First, replace 1 by 5 and match 1 to 6 to obtain the perfect matching

$$\{\{5,2\},\{3,4\},\{1,6\}\}$$

in K_6. Next, replace 2 by 5 and match 2 to 6 to obtain the perfect matching

$$\{\{1,5\},\{3,4\},\{2,6\}\}$$

in K_6. Iterate this procedure two more times to get the perfect matches

$$\{\{1,2\},\{5,4\},\{3,6\}\}$$

$$\{\{1,2\},\{3,5\},\{4,6\}\}.$$

The fifth perfect matching is obtained by matching 5 to 6, giving

$$\{\{1,2\},\{3,4\},\{5,6\}\}.$$ □

The procedure used in the solution of Example 16 is generalized in Algorithm 2, which displays the pseudocode description for finding the $n-1$ perfect matches of K_n, given a perfect matching of K_{n-2}, where n is an even integer, $n \geq 4$.

ALGORITHM 2. Generating $n-1$ perfect matches of K_n, given a perfect matching of K_{n-2}.

procedure *perfect matches* ($\{\{a_1,a_2\},\{a_3,a_4\},\ldots,\{a_{n-3},a_{n-2}\}\}$: a perfect matching of K_{n-2}, n an even integer, $n \geq 4$)
for $i := 1$ **to** $n-1$
begin
  for $j := 1$ **to** $n-2$
    if $j = i$ **then** $b_j := n-1$ **and** $b_{n-1} := a_i$
      else $b_j := a_j$
  if $i = n-1$ **then** $b_{n-1} := n-1$
  $M_i := \{\{b_1,b_2\},\{b_3,b_4\},\ldots,\{b_{n-1},n\}\}$
end $\{M_i$ is a perfect matching of $K_n\}$

Using Algorithm 2 one can solve a perfect matching problem using an exhaustive search. How efficient is this? The number of additions required to compute the weight of each perfect matching is $n/2-1 = O(n)$. By Theorem 4, the weights of $(n-1)(n-3)\cdots 5 \cdot 3 \cdot 1 = O(n^{n/2})$ perfect matches must be computed. So the number of additions required to compute the weights of all the perfect matches in K_n is $O(n^{\frac{n}{2}+1})$. There are more efficient ways to solve a perfect matching problem. For example, [3] in the suggested readings describes an algorithm that solves the perfect matching problem which requires $O(n^3)$ operations (additions and comparisons).

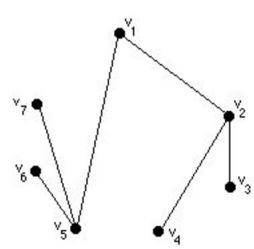
Suggested Readings

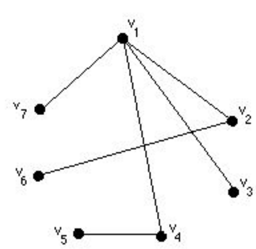
1. G. Dantzig, D. Fulkerson, and S. Johnson, "Solution of a Large-Scale Traveling Salesman Problem", *Operations Research*, volume 2 (1954), 393–410.
2. M. Garey and D. Johnson, *Computers and Intractability. A Guide to the Theory of NP-Completeness*, W. H. Freeman, New York, 1979.
3. E. Lawler, *Combinatorial Optimization: Networks and Matroids*, Dover Publications, Mineola, N.Y., 2000.
4. E. Lawler, A. Lenstra, A. Rinnooy Kan, and D. Shmoys, *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*, John Wiley & Sons, Hoboken, N.J., 1991.
5. J. Moon, "Various Proofs of Cayley's Formula for Counting Trees", *A Seminar on Graph Theory*, (ed. F. Harary), Holt, Rinehart and Winston, New York, 1967, 70–78.

Exercises

1. For the graph K_8, determine the number of
   a) paths joining any pair of vertices. b) spanning trees.
   c) Hamilton circuits. d) perfect matches.

2. For each of the following trees, determine the 5-tuple described in the proof of Cayley's Theorem.
   a) b)





3. For each of the following 5-tuples, construct the corresponding spanning tree of K_7 as described in the proof of Cayley's Theorem.

a) (7, 2, 4, 4, 1)

b) (2, 2, 2, 4, 6).

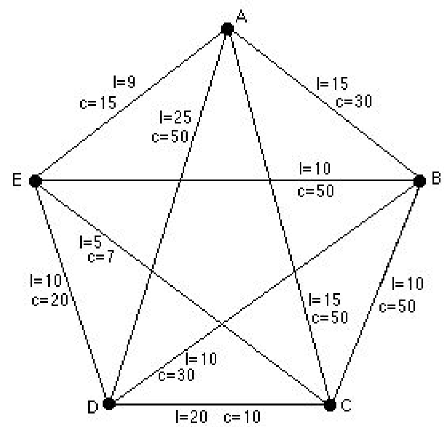
4. List all the perfect matches of K_6 by first listing all the perfect matches of K_4 and then using these to obtain the perfect matches of K_6. (See Example 15.)

5. Show that any tree with at least two vertices has at least two vertices of degree 1.

6. Show that any graph with n vertices, n-1 edges, and no cycles is a tree.

7. How many different isomers do the saturated hydrocarbon C_6H_{14} have?

8. For the following graph determine if there is a path from A to C which has total length 40 or less and total cost \$45 or less.

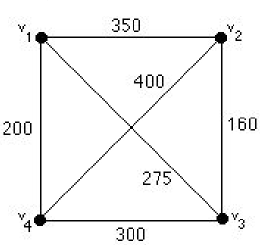


9. For the following graph

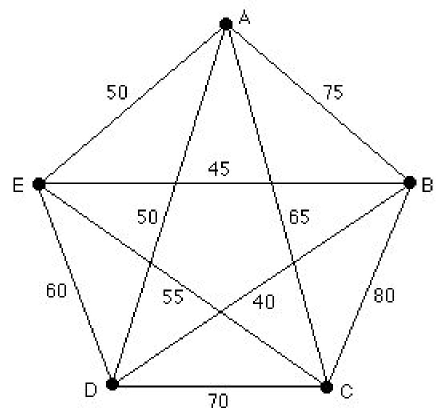
a) find a spanning tree with the smallest possible range.

b) find a Hamilton circuit with the smallest possible weight.

c) find a perfect matching with the smallest possible weight.



10. A doctor, who lives in village A, wishes to visit his patients who live in the four villages B, C, D, and E, as illustrated in the following graph. Find a route for him which involves the least possible total distance.



11. Let K_8 have the vertex set $V = \{v_1, v_2, \ldots, v_8\}$. Determine the probability that a path joining v_1 to v_5 selected at random from K_8 contains fewer than five edges.

12. Let K_n have vertex set $V = \{v_1, v_2, \ldots, v_n\}$. Determine the probability that a spanning tree selected at random from K_n contains a vertex having degree $n - 1$.

13. Let K_n have vertex set $V = \{v_1, v_2, \ldots, v_n\}$ where $n \geq 4$. Determine the probability that a Hamilton circuit selected at random from K_n visits v_1, v_2, and v_3 in succession.

14. Let K_n have vertex set $V = \{v_1, v_2, \ldots, v_n\}$ and assume n is even with $n \geq 6$. Determine the probability that a perfect matching selected at random from K_n contains the edges $\{v_1, v_2\}$ and $\{v_3, v_4\}$.

15. Determine the number of perfect matches in the complete bipartite graph $K_{n,n}$.

16. Explain how the perfect matches of the bipartite graph $K_{n,n}$ may be generated on a computer.

17. Given a perfect matching M of K_n, where n is even, determine how many spanning trees of K_n contain M.

18. a) Let $W = \{w_1, w_2, \ldots, w_n\}$ be a set of n real numbers and let r be an integer where $r < n$. Describe a procedure to find a subset of r numbers with the smallest possible sum, by checking all possible subsets of size r.

b) Give a formula in terms of n and r which indicates how many candidates must be checked to solve the problem.

19. a) Give an algorithm that is more efficient than the exhaustive approach for the problem described in Exercise 18.

b) Provide a big-O estimate for your algorithm to prove that the algorithm is more efficient than the algorithm of Exercise 18.

20. Determine the largest value of n for which all of the Hamilton circuits of K_n may be generated in less than 10 minutes of computer time, assuming the computer requires 10^{-4} seconds of computer time to generate one Hamilton circuit and compute its weight.

21. How many spanning trees does the complete bipartite graph $K_{2,n}$ have?

22. Let $K_n - e$ be the graph obtained by deleting the edge e from K_n. Show that the number of spanning trees of $K_n - e$, for any edge e, is $(n-2)n^{n-3}$.

⋆**23.** Let K_n have vertex set $V = \{v_1, v_2, \ldots, v_n\}$. Show that the number of spanning trees of K_n such that vertex v_i has degree d_i in the spanning tree is

$$\frac{(n-2)!}{(d_1-1)!(d_2-1)!\ldots(d_n-1)!.}$$

⋆**24.** Describe a method which generates the set of all spanning trees of K_n such that vertex v_i has degree d_i.

⋆⋆**25.** Let $K_{m,n}$ be the complete bipartite graph with vertices $V = V_1 \cup V_2$, where $V_1 = \{u_1, u_2, \ldots, u_m\}$ and $V_2 = \{v_1, v_2, \ldots, v_n\}$. Show that the number of spanning trees of $K_{m,n}$ such that vertex u_i has degree d_i and vertex v_j has degree f_j is

$$\frac{(m-1)!(n-1)!}{(d_1-1)!\ldots(d_m-1)!(f_1-1)!\ldots(f_n-1)!}.$$

Computer Projects

1. Let K_{10} have vertex set $V = \{v_1, v_2, \ldots, v_{10}\}$. Write a computer program that takes as input the weights of the edges of K_{10} and finds a path of length 3 of smallest possible weight that joins a given pair of vertices.

2. Write a program that generates all the Hamilton circuits of K_6.

3. Write a computer program that generates all the perfect matches of K_8.