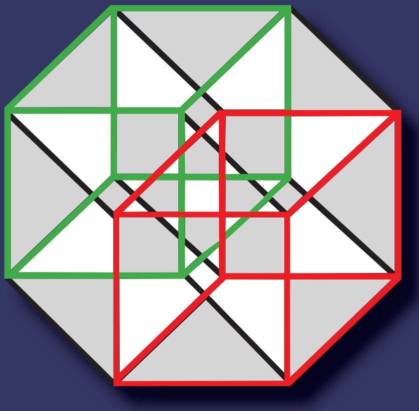
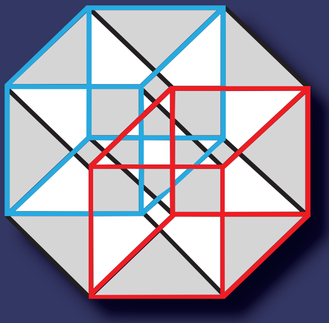
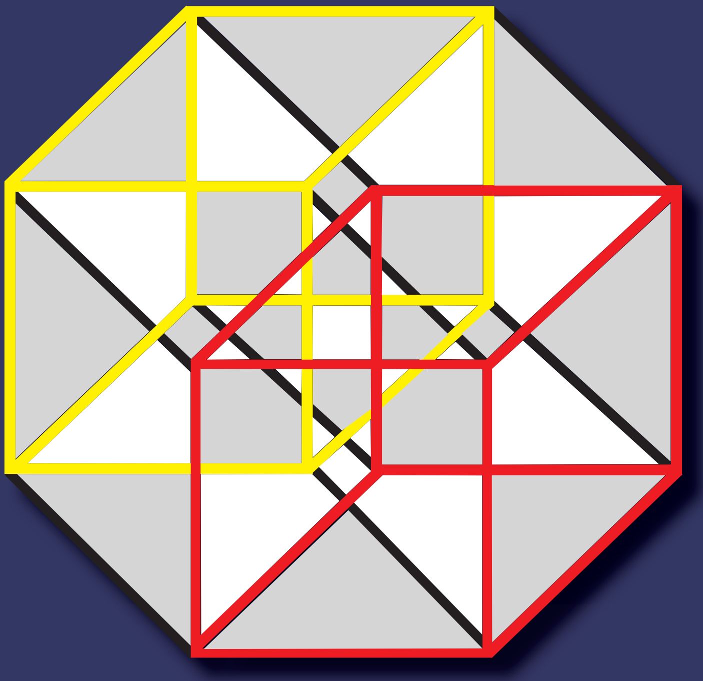



5th Edition

Introduction to Graph Theory

Robin J. Wilson



Introduction to Graph Theory

every vertex is **semi-Hamiltonian**. Little is known about Hamiltonian digraphs, and several theorems on Hamiltonian graphs do not generalize easily, if at all, to digraphs.

It is natural to ask whether there is a generalization to digraphs of Dirac's theorem (Corollary 2.17). One such generalization is due to M. A. Ghouila-Houri; its proof is more difficult than that of Dirac's theorem, and can be found in West [16].

THEOREM 2.18 *Let D be a strongly connected digraph with n vertices. If* outdeg(v) $\geq \frac{1}{2}n$ *and* indeg(v) $\geq \frac{1}{2}n$ *for each vertex v, then D is Hamiltonian.*

It seems that such results will not come easily, and so we consider instead which types of digraph are Hamiltonian. In this respect, the tournaments are particularly important, the results in this case taking a very simple form.

Because tournaments may have vertices with out-degree or in-degree 0, they are not in general Hamiltonian. However, the following theorem, due to L. Rédei and P. Camion, shows that every tournament is 'nearly Hamiltonian'.

THEOREM 2.19
(i) *Every non-Hamiltonian tournament is semi-Hamiltonian.*
(ii) *Every strongly connected tournament is Hamiltonian.*

Proof. (i) The statement is clearly true if the tournament has fewer than four vertices. We prove the result by induction on the number of vertices, and assume that every non-Hamiltonian tournament on n vertices is semi-Hamiltonian.

Let T be a non-Hamiltonian tournament on $n + 1$ vertices, and let T' be the tournament on n vertices obtained by removing from T a vertex v and its incident arcs. By the induction hypothesis, T' has a semi-Hamiltonian path $v_1 \to v_2 \to \cdots \to v_n$. There are now three cases to consider:

- if vv_1 is an arc in T, then the required path is

$$v \to v_1 \to v_2 \to \cdots \to v_n.$$

- if vv_1 is not an arc in T (which means that v_1v is), and if there exists an i such that vv_i is an arc in T, then choosing i to be the first such, we obtain the required path (see Fig. 2.29)

$$v_1 \to v_2 \to \cdots \to v_{i-1} \to v \to v_i \to \cdots \to v_n.$$

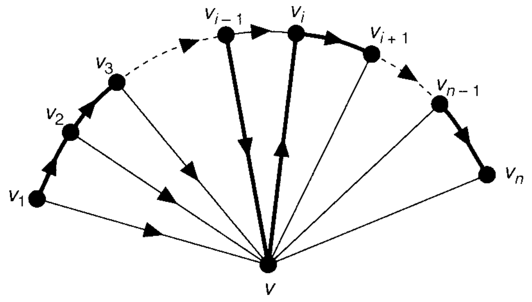


Figure 2.29

■ if there is no arc in T of the form vv_i, then the required path is

$$v_1 \to v_2 \to \dots \to v_n \to v.$$

(ii) We prove the stronger result that a strongly connected tournament T on n vertices contains cycles of length 3, 4, . . . , n.

To show that T contains a cycle of length 3, let v be any vertex of T, and let W be the set of all vertices w such that vw is an arc in T, and Z be the set of all vertices z such that zv is an arc. Since T is strongly connected, W and Z must both be non-empty, and there must be an arc in T of the form $w'z'$, where w' is in W and z' is in Z (see Fig. 2.30). The required cycle of length 3 is then

$$v \to w' \to z' \to v.$$

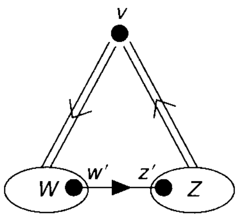


Figure 2.30

It remains only to show that, if there is a cycle of length k, where $k \le n$, then there is one of length $k + 1$. Let

$$v_1 \to v_2 \to \dots \to v_k \to v_1$$

be such a cycle. Suppose first that there exists a vertex v, not contained in this cycle, for which there exist arcs in T of the form vv_i and of the form v_jv. Then there must be a vertex v_i such that both $v_{i-1}v$ and vv_i are arcs in T. The required cycle is then (see Fig. 2.31)

$$v_1 \to v_2 \to \dots \to v_{i-1} \to v \to v_i \to \dots \to v_k \to v_1.$$

If no vertex exists with the above-mentioned property, then the set of vertices not contained in the cycle may be divided into two disjoint sets W and Z, where W is the set of vertices w such that v_iw is an arc for each i, and Z is the set of vertices z such

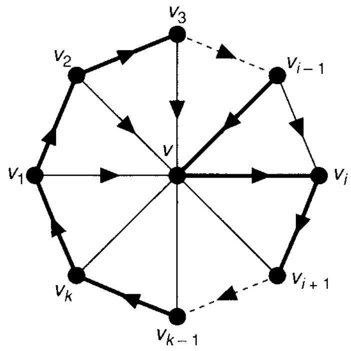


Figure 2.31

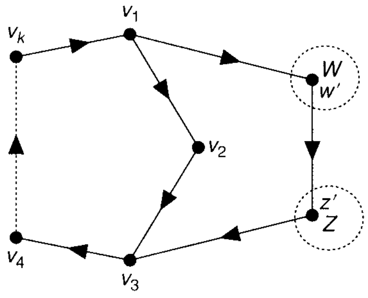


Figure 2.32

that zv_i is an arc for each i. Since T is strongly connected, W and Z must both be non-empty, and there must be an arc in T of the form $w'z'$, where w' is in W and z' is in Z. The required cycle is then (see Fig. 2.32)

$$v_1 \to w' \to z' \to v_3 \to \cdots \to v_k \to v_1. \qquad \blacksquare$$

Exercises

2.27ˢ Which of the following graphs are Hamiltonian or semi-Hamiltonian?

(i) the complete graph K_5;
(ii) the complete bipartite graph $K_{2,3}$;
(iii) the graph of the octahedron;
(iv) the wheel W_6;
(v) the 4-cube Q_4.

2.28ˢ In the table of Fig. 1.9, locate all the Hamiltonian and semi-Hamiltonian graphs.

2.29 (i) For which values of n is K_n Hamiltonian?
(ii) Which complete bipartite graphs are Hamiltonian?
(iii) Which Platonic graphs are Hamiltonian?
(iv) For which values of n is the wheel W_n Hamiltonian?
(v) For which values of k is the k-cube Q_k Hamiltonian?

2.30 Is the Grötzsch graph in Fig. 2.33 Hamiltonian?

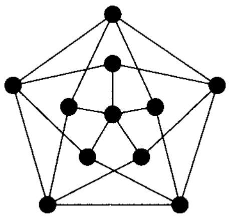


Figure 2.33

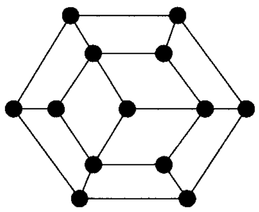


Figure 2.34

2.31 (i) Prove that, if G is a bipartite graph with an odd number of vertices, then G is non-Hamiltonian.
(ii) Deduce that the graph in Fig. 2.34 is non-Hamiltonian.

2.32 Use the result of Exercise 2.31(i) to show that a knight cannot visit all the squares of a 5×5 or 7×7 chessboard exactly once by knight's moves and return to its starting point. Can a knight visit all the squares of a 6×6 chessboard?

2.33ˢ Give an example to show that the condition '$\deg(v) \geq \frac{1}{2}n$', in the statement of Dirac's theorem, cannot be replaced by '$\deg(v) \geq \frac{1}{2}(n-1)$'.

2.34 (i) Let G be a graph with n vertices and $\{\frac{1}{2}(n-1)(n-2)\} + 2$ edges. Use Ore's theorem to prove that G is Hamiltonian.
(ii) Find a non-Hamiltonian graph with n vertices and $\{\frac{1}{2}(n-1)(n-2)\} + 1$ edges.

2.35ˢ Find a Hamiltonian cycle in the tournament of Fig. 2.23.

2.4 Applications

Many important advances in graph theory arose as a result of attempts to solve particular problems – Euler and the bridges of Königsberg (Section 2.2), Cayley and the enumeration of chemical molecules (Section 3.2), and Kirchhoff's work on electrical networks (Section 3.3), to name but three. Much present-day interest in the subject is due to the fact that, quite apart from being an elegant mathematical discipline in its own right, graph theory can be applied in a wide range of areas (see the Preface). In a book of this size we cannot discuss a large number of these applications, and you should consult Berge [9], Chartrand and Oellermann [20], Deo [21], Roberts [25] and Tucker [26] for a wide range of practical problems, often with algorithms for their solution.

In this section we outline four types of problem that involve paths and cycles: the *shortest path problem*, the *critical path problem*, the *Chinese postman problem* and the *travelling salesman problem*. The first two of these can be solved by efficient **algorithms** – finite step-by-step procedures that quickly yield the solutions. The third problem can also be solved by an efficient algorithm, but we consider only a special case here. For the fourth problem, no efficient algorithms are known; we must therefore choose between algorithms that take a long time to implement and heuristic algorithms that are quick to apply but give only an approximation to the solution.

The shortest path problem

Suppose that we have a diagram of the form shown in Fig. 2.35, in which the letters *A*–*L* refer to towns that are connected by roads. If the lengths of these roads are as marked, what is the length of a shortest path from *A* to *L*? (We write 'a shortest path' rather than 'the shortest path' since there may be more than one path with this shortest length.)

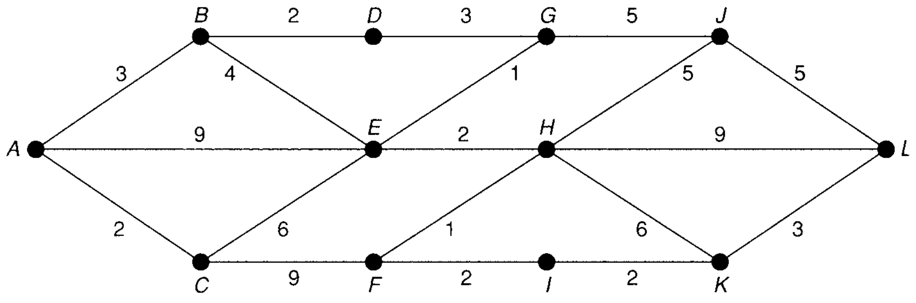


Figure 2.35

Note that the numbers in the diagram might also refer, not to the length of each road, but to the time taken to travel along it, or to the cost of doing so. Thus, if we have an algorithm for solving this problem in its original formulation, then this algorithm can also be used to find a quickest or a cheapest route.

Note also that we can easily obtain an upper bound for the answer by taking *any* path from *A* to *L* and calculating its length. For example, the path

$$A \to B \to D \to G \to J \to L$$

has total length 18, and so the length of a shortest path cannot exceed 18.

In solving such problems, we regard our diagram as a connected graph in which a non-negative number is assigned to each edge. Such a graph is called a **weighted graph**, and the number assigned to each edge e is the **weight** of e, denoted by $w(e)$. The problem is to find a path from A to L with minimum total weight. Note that, if we have a weighted graph in which each edge has weight 1, then the problem reduces to that of finding the number of edges in a shortest path from A to L.

There are several methods that we can use to solve this problem. One way is to make a model of the map by knotting together pieces of string whose lengths are proportional to the lengths of the roads. To find a shortest path, take hold of the knots corresponding to A and L – and pull tight!

However, there is a more mathematical way of solving it. The idea is to move across the map from left to right, labelling each vertex V with a number $l(V)$ that indicates the shortest distance from A to V. This means that, when we reach a vertex such as K in Fig. 2.35, then $l(K)$ is labelled either $l(H) + 6$ or $l(I) + 2$, whichever is the smaller. Our aim is to find $l(L)$.

To apply the algorithm, we first assign A the label 0 and give B, E and C the temporary labels $l(A) + 3$, $l(A) + 9$ and $l(A) + 2$ – that is, 3, 9 and 2. We take the *smallest* of these, and write $l(C) = 2$. *C is now permanently labelled 2.*

We next look at the vertices adjacent to this labelled vertex C. We assign F the temporary label $l(C) + 9 = 11$, and we can lower the temporary label at E to $l(C) + 6 = 8$. The smallest temporary label is now 3 (at B), so we write $l(B) = 3$. *B is now permanently labelled 3.*

Now we look at the vertices adjacent to B. We assign D the temporary label $l(B) + 2 = 5$, and we can lower the temporary label at E to $l(B) + 4 = 7$. The smallest temporary label is now 5 (at D), so we write $l(D) = 5$. *D is now permanently labelled 5.*

Now we look at the vertices adjacent to D. The only one is G, and we assign it the temporary label $l(D) + 3 = 8$. The smallest temporary label is now 7 (at E), so we write $l(E) = 7$. *E is now permanently labelled 7.*

Continuing in this way, we successively obtain the permanent labels

$$l(G) = 8,\ l(H) = 9,\ l(F) = 10,\ l(I) = 12,\ l(J) = 13,\ l(K) = 14,\ l(L) = 17;$$

these are shown on Fig. 2.36. It follows that the shortest path from A to L has length 17.

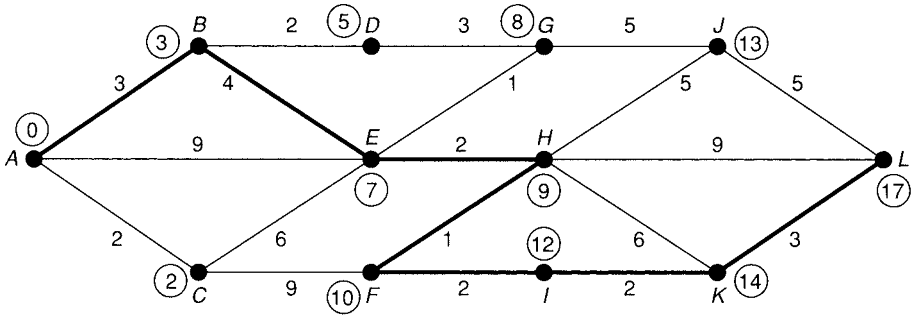


Figure 2.36

To find such a shortest path, we can restrict our attention to those edges whose length is the difference of the labels at its ends, such as KL and IK (since $l(L) - l(K) = 17 - 14 = 3$ and $l(K) - l(I) = 2$). Using the labels to help us, we then trace back from L, via K and I, obtaining

$$L \leftarrow K \leftarrow I \leftarrow F \leftarrow H \leftarrow E \leftarrow B \leftarrow A.$$

Thus the shortest path (which in this case is unique) from A to L is

$$A \rightarrow B \rightarrow E \rightarrow H \rightarrow F \rightarrow I \rightarrow K \rightarrow L.$$

The critical path problem

We now see how this algorithm can be adapted to yield the *longest* path in a digraph, and we illustrate its use in a 'critical path' problem relating to the scheduling of a series of operations.

Suppose that we have a job to perform, such as the building of a house, and that this job can be divided into a number of activities, such as laying the foundations, putting on the roof, doing the wiring, etc. Some of these activities can be performed simultaneously, whereas some may need to be completed before others can be started. Can we find an efficient method for determining how to schedule the activities so that the entire job is completed in minimum time?

In order to solve this problem, we construct a 'weighted digraph', or **activity network**, in which each arc represents the length of time taken for an activity. Such a network is given in Fig. 2.37. The vertex A represents the beginning of the job, and the vertex L represents its completion. Since the entire job cannot be completed until each path from A to L has been traversed, the problem reduces to that of finding the longest path from A to L.

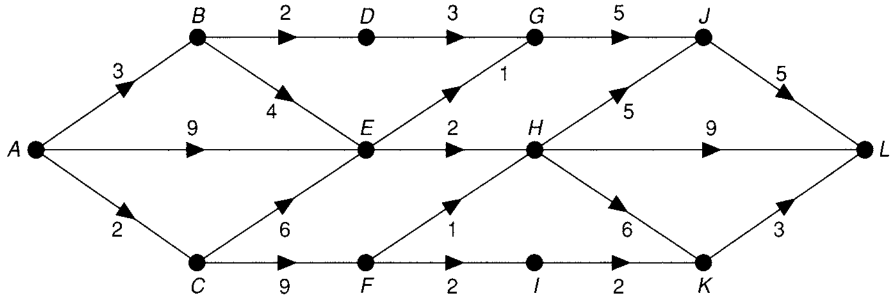


Figure 2.37

This is accomplished by using a technique known as programme evaluation and review technique (PERT), which is similar to the method we used to solve the shortest path problem above. This time, as we move across the digraph from left to right, we associate with each vertex V a label $l(V)$ indicating the length of the *longest* path from A to V. As in the shortest path problem, we keep track of these labels by writing them next to the vertices they represent. However, unlike the problem that we considered above, there is no 'zigzagging', since all arcs are directed from left to right, so we can assign the permanent labels as we proceed. For example, for the digraph of Fig. 2.37, we assign: