Previous Chapter Return to Table of Contents Next Chapter

PART VI: Graph Algorithms

Introduction

Graphs are a pervasive data structure in computer science, and algorithms for working with them are fundamental to the field. There are hundreds of interesting computational problems defined in terms of graphs. In this part, we touch on a few of the more significant ones.

Chapter 23 shows how we can represent a graph on a computer and then discusses algorithms based on searching a graph using either breadth-first search or depth-first search. Two applications of depth-first search are given: topologically sorting a directed acyclic graph and decomposing a directed graph into its strongly connected components.

Chapter 24 describes how to compute a minimum-weight spanning tree of a graph. Such a tree is defined as the least-weight way of connecting all of the vertices together when each edge has an associated weight. The algorithms for computing minimum spanning trees are good examples of greedy algorithms (see Chapter 17).

Chapters 25 and 26 consider the problem of computing shortest paths between vertices when each edge has an associated length or "weight." Chapter 25 considers the computation of shortest paths from a given source vertex to all other vertices, and Chapter 26 considers the computation of shortest paths between every pair of vertices.

Finally, Chapter 27 shows how to compute a maximum flow of material in a network (directed graph) having a specified source of material, a specified sink, and specified capacities for the amount of material that can traverse each directed edge. This general problem arises in many forms, and a good algorithm for computing maximum flows can be used to solve a variety of related problems efficiently.

In describing the running time of a graph algorithm on a given graph $G = (V, E)$, we usually measure the size of the input in terms of the number of vertices $|V|$ and the number of edges $|E|$ of the graph. That is, there are two relevant parameters describing the size of the input, not just one. We adopt a common notational convention for these parameters. Inside asymptotic notation (such as O-notation or Θ-notation), and *only* inside such notation, the symbol V denotes $|V|$ and the symbol E denotes $|E|$. For example, we might say, "the algorithm runs in time $O(V E)$," meaning that the algorithm runs in time $O(|V| |E|)$. This convention makes the running-time formulas easier to read, without risk of ambiguity.

Another convention we adopt appears in pseudocode. We denote the vertex set of a graph G by $V[G]$ and its edge set by $E[G]$. That is, the pseudocode views vertex and edge sets as attributes of a graph.

Go to Chapter 23 Back to Table of Contents