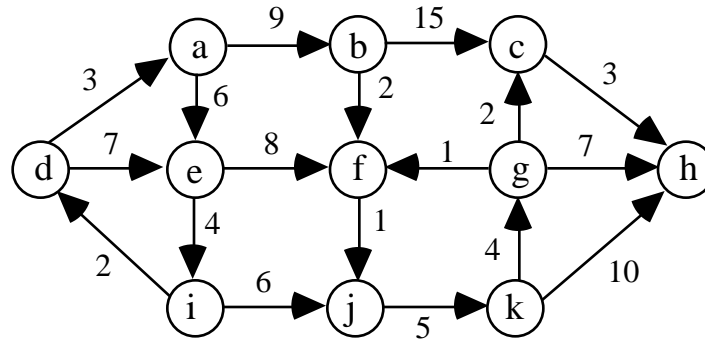


**PROBLEM SET 9**  
**Due: Monday, May 7**

**Reading:** Handouts 28 (Depth-First Search), 30 (P&NP), 31 (NP-Completeness); CLR Sections 23.3--23.5 (Depth-First Search, Topological Sort, Strongly Connected Components); CLR Chapter 36;

**Suggested Problems:** ; 23.3-9; 23.4-2, 23.4-5; 23.5-1, 23.5-3, 23.5-4, 23.5-4; 23-1, 23-2; 36.1-2, 36.1-7; 36.2-2, 36.2-4;

**Problem 1 [30]** Consider the following weighted graph G (this is the same graph from PS8 Problem 1):



In the following problems, you should assume that G is represented as a collection of adjacency lists, and that vertices are ordered alphabetically within each adjacency list.

**a [15]** Draw the tree the is induced by performing a depth-first-search starting at node a. Label each vertex by its discovery and finish times (as on CLR p. 479). Using dotted lines, draw the edges of G that are not in the depth-first tree and label them as forward edges (F), back edges (B), or cross edges (C).

**b [5]** Give a topological sort of the vertices in G.

**c [10]** Use the strongly connected component algorithm from CLR 23.5 to determine the strongly connected components of G.

**Problem 2 [15]**

**a [10]** CLR 23.4-3 (p. 488).

**b [5]** Does your algorithm from part a work for directed graphs? Explain.

**Problem 3 [15]** CLR 23.4-5 (p. 488)

**Problem 4 [10]:** CLR 36.1-1 (p. 923). Note that the graph is an *unweighted* graph, so the length of a path is just the number of edges in it. To prove the “if and only if” condition, you must show two things:

- 1) If there is an algorithm that computes LONGEST-PATH-LENGTH in polynomial time, then LONGEST-PATH  $\in$  P.
- 2) If LONGEST-PATH  $\in$  P, then there is an algorithm that computes LONGEST-PATH-LENGTH in polynomial time.

**Problem 5 [15]:** CLR 36.2-8 (p. 929).

**Problem 6 [15]:** CLR 36.3-2 (p. 938).

**Extra Credit Problems:**

**Problem EC1 [15]:** CLR 36.2-3 (p. 928). The problem could be rephrased as follows: Suppose you are given a black box predicate HAS-HAM-CYCLE?(G) that can determine in polynomial time whether G has a hamiltonian cycle. How could you use such a predicate to compute in polynomial time an ordered list of vertices for a hamiltonian path in a graph H for which HAS-HAM-CYCLE?(H) returns true? You do *not* need to give pseudocode, but need to give enough details so that it is clear how your algorithm works.

**Problem EC2 [10]:** CLR 36.2-7 (p. 929). *Hint:* Topologically sort the vertices. What must be true of the result?

*Problem Set Header Page*  
*Please make this the first page of your hardcopy submission.*

**CS231 Problem Set 9**  
**Due Monday, May 7, 2001**

Name:

Date & Time Submitted (*only if late*):

Collaborators (*anyone you collaborated with in the process of doing the problem set*):

*In the **Time** column, please estimate the time you spent on the parts of this problem set. Please try to be as accurate as possible; this information will help me to design future problem sets. I will fill out the **Score** column when grading your problem set.*

| <b>Part</b>         | <b>Time</b> | <b>Score</b> |
|---------------------|-------------|--------------|
| General Reading     |             |              |
| Problem 1 [30]      |             |              |
| Problem 2 [15]      |             |              |
| Problem 3 [15]      |             |              |
| Problem 4 [10]      |             |              |
| Problem 5 [15]      |             |              |
| Problem 6 [15]      |             |              |
| Extra Credit 1 [15] |             |              |
| Extra Credit 1 [10] |             |              |
| <b>Total</b>        |             |              |