George Washington University
Department of Computer Science

CS212 - Final Exam

2 hours (closed book)
12/15/2010

1. (10 pts) Is there a comparison-based sorting algorithm that can sort 5 numbers in an arbitrary order using at most 6 comparisons? If so, give a sorting algorithm. Otherwise, explain why. Note that you cannot simply say no by arguing that a lower bound of sorting 5 numbers is $\Omega(5 \log 5)$ and $5 \log 5 > 6$ as Ω notation dose not specify the exact constant.

2. (5 pts each) A *dominating set* of a graph G is defined as a vertex subset $D \subseteq V(G)$ such that each vertex in $V(G)$ is either in D or is connected to a vertex in D.

(a) Describe a backtracking algorithm to determine whether a given graph G has a dominating set of size k or not. Apply this algorithm to an example of your own choice (with 4-6 vertices) and show the generated state space tree.

(b) Describe a branch-and-bound algorithm to find a minimum size dominating set of a given graph G. Apply this algorithm to the example you constructed in part (a) and show the generated state space tree.

4. Consider the following graph G where the number next to each vertex shows a depth-first-number obtained by applying the depth-first-search. A sequence of low points assigned and revised to each vertex is given as:

a:	1
b:	2, 1
c:	3, 1
d:	10, 1
e:	9, 3, 1
f:	4, 1
g:	5, 4
h:	6
i:	7
j:	8, 4

(a) (3 pts) Show the resulting depth-first-search tree of G.

(b) (4 pts) List a sequence of 13 edges in the order that they are examined. Note that the sequence is not necessarily unique.

(c) (3 pts) Show all articulation points.

(d) (10 pts) Apply the $O(|E|)$ time algorithm to compute all biconnected components of G. Show all your work including stack operations.

4. (10 pts) Consider two airline companies that serve connections among two groups of cities G_1 and G_2 where G_1 and G_2 are not necessarily disjoint. Now, two airlines are going to merge, and cities in $G_1 \cup G_2$ will be served by a single company. The central office has to

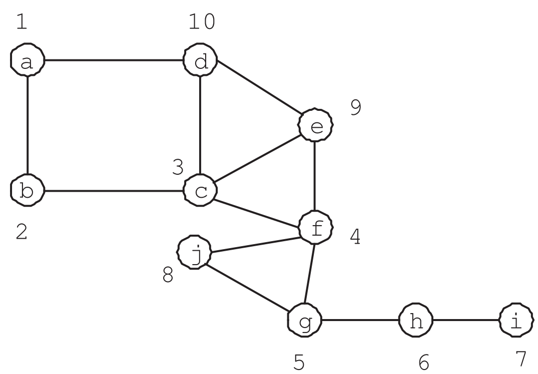

Figure 1: G

determine a shortest route between each pair of cities in $G_1 \cup G_2$. The available information is: the distance matrices (i.e., a shortest flying time between each pair of cities) for G_1 and G_2 before the merger, and the direct flying time for each new service between two cities (one in G_1 and the other in G_2) if the service will be provided after the merger. Design an *efficient* algorithm to determine the distance matrix for $G_1 \cup G_2$, and analyze the time complexity of your algorithm. Note that the topology information (i.e., direct flight information in G_1 or G_2) is not available.

5. (10 pts) A graph G is called *bipartite* if the vertex set of G can be partitioned into two parts X and Y such that for any edge $(u, v) \in E(G)$, $u \in X$ and $v \in Y$. Describe an $O(|E|)$ time algorithm to determine if a given graph is bipartite.

6. (10 pts) You are given an ordered sequence, $C_1, \cdots, C_n$, of n cities, and the distance $d(i, j)$ between every pair of cities. You must partition the cities into two subsequences (not necessarily contiguous) such that (i) person A visits all cities in the first subsequence (in order), (ii) person B visits all cities in the second subsequence (in order), and (iii) the sum of the total distances traveled by A and B is minimized. Assume that person A and person B start initially at the first city in their respective subsequences. Give a dynamic programming algorithm to solve this problem and analyze the time complexity of your algorithm.

7. (5 pts each)

(a) Given an arbitrary edge-weighted graph G and two arbitrary vertices v_0 and v_1 in G, give an algorithm that finds a shortest cycle that goes through v_0 but does not go through v_1.

(b) Given an edge-weighted graph G with n nodes, a pair s, t of source and destination nodes, and a vertex x ($x \in V(G) - \{s, t\}$), the problem is to find a shortest path from s to t going through x. For example, $s - u - v - x - z - t$ is a feasible path, but $s - u - v - t$ is not a feasible path.

Does this problem satisfy the principle of optimality ? If so, describe a dynamic programming algorithm to solve this problem. If not, show me an example to justify your answer.

8. (10 pts) *Arbitrage* is the use of discrepancies in currency exchange rates to make a profit. For example, there me be a small window of time during which 1 U.S. dollar buys a 0.75 British pounds, 1 British pound buys 2 Australian dollars, and 1 Australian dollar buys 0.70 U.S. dollars. At such a time, a smart trader can trade one U.S. dollar and end up with $0.75 \times 2 \times 0.7 = 1.05$ U.S. dollars, a profit of 5 %. Suppose that there are n currencies $c_1, \cdots, c_n$ and an $n \times n$ table R of exchange rates, such that one unit of currency c_i buys $R[i, j]$ units of currency c_j. Give a dynamic programming algorithm to determine the maximum value of

$$R[c_1, c_{i1}] \cdot R[c_{i1}, c_{i2}] \cdots R[c_{i(k-1)}, c_{ik}] \cdot R[c_{ik}, c_1],$$

i.e., trade 1 unit of currency c_1 and end with currency c_1. Analyze the time complexity of your algorithm.

9. Answer briefly but precisely to each of the following questions.

(a) (1 pts) Give a definition for class P.

(b) (1 pts) Give a definition for class NP.

(c) (2 pts) Give a definition for class NP-*complete*.

(d) (6 pts) Prove the following statement: $P = NP$ if there exists a problem $\phi \in P \cap NPC$.