George Washington University
Department of Computer Science

CS6212 - Final Exam

2 hours (closed book)
5/8/2012

1. (10 pts each) Consider the following graph G.

(a) Find a minimum spanning tree of G obtained by applying the Kruskal's algorithm. Show all your work to illustrate the details of $O(m \log n)$ time implementation of Kruskal's algorithm, where $n = |V(G)|$ and $m = |E(G)|$.

(b) Apply Dijkstra's shortest path algorithm to find a shortest path from node 7 to each of the remaining node. show all your work.

(c) Apply the $O(n^3)$ dynamic programming algorithm to find all-to-all shortest paths.

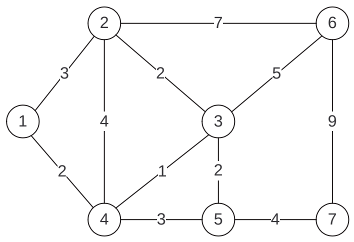

Figure 1: G

2. Consider a set of keys $K_1 < K_2 < \cdots < K_n$. Let p_i for $1 \le i \le n$ denote the probability of searching for key K_i and q_i for $0 \le i \le n$ denote the probability of searching for a key E_i where $a_i < E_i < a_{i+1}$, $a_0 = -\infty$, and $a_{n+1} = +\infty$. Let $T(i,j)$ denote an optimal binary search tree containing $E_i, a_{i+1}, E_{i+1}, \cdots, a_j, E_j$. Let $C(i,j)$ and $R(i,j)$ denote the cost (i.e., the average number of comparisons to search for a key) and the root of $T(i,j)$. Define $W(i,j) = q_i + p_{i+1} + q_{i+1} + \cdots p_j + q_j$.

(a) (10 pts) Describe a dynamic programming algorithm to find an optimal binary search tree.

(b) (10 pts) Consider the following example: $a_1 < a_2$ with $q_0 = 2/10$, $p_1 = 1/10$, $q_1 = 3/10$, $p_2 = 3/10$, and $q_2 = 1/10$. Apply your algorithm in (a) to compute $C(i,j)$ and $R(i,j)$, and construct an OBST for this example.

3. Consider the following graph G_2 where the number next to each vertex shows a depth-first-number obtained by applying the depth-first-search. A sequence of low points (as revised following the algorithm discussed in class) to each vertex is given as:

a:	1
b:	2, 1
c:	3, 1
d:	10, 3, 1
e:	9, 1

f:	4, 1
g:	5, 4
h:	6
i:	7
j:	8, 4

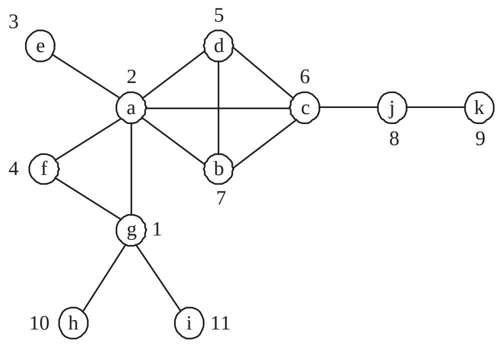

Figure 2: G_2

(a) (10 pts) Show the resulting depth-first-search tree of G_2.

(b) (10 pts) Show all articulation points and all biconnected components of G_2. Note that you do not have to show details of how your algorithm works; you just have to list all articulation points and all biconnected components.

(c) (10 pts) List a sequence of 13 edges **in the order that they are examined** so that low point at each node is revised as given above.

4. (10 pts) Do one problem from the following. (If you give answers to more than one problem, you will receive the lowest score.)

(a) Give a dynamic programming algorithm to find the longest monotonically increasing subsequence of a sequence of n elements. Apply your algorithm to a sequence $A = \langle 1, 3, 2, 4, 6, 13, 14, \ 15, 5, 6, 8, 12, 13 \rangle$. Note that in this example, sequences $\langle 1, 3, 4, 6, 13, 14, 15 \rangle$ and $\langle 1, 2, 4, 5, 6, 8, 12, 13 \rangle$ both are monotonically increasing subsequence of A.

(b) The input to this problem is a sequence of n points $p_1, \cdots, p_n$ in the Euclidean plane. You are to find the shortest routes for two taxis to service these requests in order. The two taxis start at the origin p_1. If a taxi visits a point p_i before p_j then it must be the case that $i < j$. Each point must be visited by one of the two taxis. The cost of a routing is the total distance traveled by the first taxi plus the total distance traveled by the second taxi. Design an efficient dynamic programming algorithm to find the minimum cost routing such that each point is covered by one of the two taxis.

(c) Consider a 2-D map with a horizontal river passing through its center. There are n cities on the southern bank with x-coordinates $a(1), \cdots, a(n)$ and n cities on the northern bank with x-coordinates $b(1), \cdots, b(n)$. You want to connect as many north-south pairs of cities as possible with bridges such that no two bridges cross. When connecting cities, you can only connect city i on the northern bank to city i on the southern bank. Give a dynamic programming algorithm to solve this problem and analyze the time complexity of your algorithm. Note that those x-coordinate values are not sorted, i.e., $a(i)$'s and $b(i)$'s are in an arbitrary order.

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.