Jonathan Buss, Mark Petrick, Masoomeh Rudafshani

Copyright©2022.

Due Nov. 21, by 11:59pm (midnight), to Crowdmark.
All submitted work must be the student's own.

Question 1 (8 marks).

Consider communication networks consisting of a set of one-way links, over which one node (the source of the link) can send messages to another node (the target of the link). Since each link is one-way, a direct message in reply may not be possible – unless the reverse link also exists.

In such a network, a sequence of nodes $p_1, p_2, \ldots, p_k$ is a communication route from p_1 to p_k (of length $k-1$) iff for each i with $1 \leq i < k$, node p_i has a link to p_{i+1}. A *conversation* route between two nodes p and q consists of a communication route from p to q and a communication route from q to p. The length of a conversation route is the sum of the lengths of the two communication routes.

Give an efficient algorithm that computes, for each pair p and q ($p \neq q$), the length of a shortest conversation route between the two or determines that no such route exists. The input to the algorithm is a list, for each node, of the nodes to which it has a link.

Justify that the algorithm works, and determine its asymptotic run time.

Question 2 (12 marks).

Let G be a weighted, undirected graph with minimum spanning tree (MST) T. This question considers the effect of changing the weight of a single edge e of G.

(a) Suppose that some edge e has its weight changed from w to w'. Show that, in either of the following cases, the tree T is still a minimum spanning tree in the new graph.

- e is an edge in T, and $w' < w$, or
- e is not an edge in T, and $w < w'$.

(b) Suppose that some edge e of T has its weight increased from w to $w' > w$. Show that there is an edge e' such that $(T \setminus \{e\}) \cup \{e'\}$ is a MST in the re-weighted graph. (In some cases, e' is the same as e – that is, the MST is unchanged.)

Hint: you may wish to consider the operation of Kruskal's algorithm in the graph. (Or Prim's, if you prefer.)

(c) Suppose that some edge e not in T has its weight decreased from w to $w' < w$. Show that either T is still an MST, or there is an edge e' of T such that $(T \setminus \{e'\}) \cup \{e\}$ is a MST.

Question 3 (12 marks).

A *tour* in a graph is a simple cycle that meets each node exactly once. The cost of a tour in a weighted graph is the sum of the costs of its edges. For this question, we shall consider only graphs which have a directed edge from each node to each other node; thus a tour is simply a permutation of the nodes.

You are given a cost matrix C for a graph on $\{1, 2, \ldots, n\}$; the cost of the edge from i to j is $C[i][j]$. Each cost is positive, but they need not be symmetric (that is, it may be that $C[i][j] \neq C[j][i]$) and the triangle inequality need not hold (that is, the sum $C[i][j] + C[j][k]$ may be less than $C[i][k]$).

In this question, you will describe an algorithm that finds the tour of minimum total cost, using the technique of dynamic programming. For two nodes $a \neq b$ and a set $S \subseteq \{1, \ldots, n\} \setminus \{a, b\}$, define an "$a$-$S$-$b$" path to be a path starting at a and ending at b, with each vertex of S exactly once in between. Define $P(a, b, S)$ to be the minimum cost of an a-S-b path; these values form the matrix for the dynamic programming.

(a) Explain how to determine the costs for the cases where the set S is the empty set.

(b) Give a formula or algorithm, to compute the value $P(a, b, S)$ from some of the the values $P(a', b', S')$ with $|S'| < |S|$. For each entry $P(a, b, S)$, use at most $O(n)$ other entries.

(c) Explain how to compute the cost of the optimal tour, once all values in P are computed.

(d) Give an asymptotic expression for the total run-time of the algorithm. Assume that to add two distances, or to compare two distances, uses constant time. (Note that the run-time will not be polynomial; the problem is *NP*-complete.)

(e) Compare the cost of your algorithm to the cost of a brute-force algorithm that simply tries each permutation of the nodes. Let $P(n)$ be the worst-case time of the algorithm you constructed, on a graph of size n, and let $B(n)$ be the time used to try all permutations and take the minimum. Estimate a constant c such that $B(n) \in O(P(n)^c)$, or show that no such constant exists.

(Note: it is not necessary to determine either function precisely.)

Question 4 (0 marks).

Optional question. Will not be marked. Submissions will not be accepted in Crowdmark; discussion on Piazza is fine.

Define a function T (for n a power of 2) by the recurrence

$$T(1) = d$$
$$T(n) = n \cdot T(n/2) + cn \quad \text{if } n > 1 \ .$$

You may do the two parts below separately; alternatively, you may first find an approximate (or exact!) formula for $T(n)$, and then use that formula in your solutions.

(a) Show that T is not polynomially bounded; that is, for each constant $c > 0$, $T(n) \in \Omega(n^c)$.

(b) Show that T is not fully exponential; that is, for each rational constant $c > 0$, $T(n) \in O(2^{cn})$.