COL 702

Homework II

Due on Sept. 26, 2021

Justify your answers with proper reasonings/proofs.

1. You are given a directed graph G and two vertices s and t in G. Each edge of G is associated with a cost $c(e)$ that may be negative; however there is no negative cycle in G. Give an efficient algorithm to compute the total number of shortest s-t paths in G. Note that you are not supposed to output the set of such paths, but just a count of how many different shortest paths are there from s to t.

2. You are given a directed graph G where all edge lengths are positive except for one edge. Given a source vertex s, give $O(m \log n)$ time algorithm for finding a shortest path from a vertex s to a vertex t. Now assume there are a constant number of edges in G which have negative weights (rest have positive weights). Give an $O(m \log n)$ time algorithm to find a shortest path from s to t.

3. The second minimal spanning tree is one that is distinct from the minimal spanning tree (has to differ by at least one edge) and is an MST if the original tree is ignored (they may even have the same weight). Design an efficient algorithm to determine the second MST.

4. You are given two arrays A and B, each of length n, containing integers. Give an efficient algorithm to find a permutation σ of $\{1, 2, \ldots, n\}$ such that the following quantity is minimized:
$$\sum_{i=1}^{n} |A[i] - B[\sigma(i)]|.$$

5. You are given a positive integer N. You want to reach N by starting from 1, and performing one of the following two operations at each step: (i) increment the current number by 1, or (ii) double the current number. For example, if $N = 10$, you can start with 1, and then go in the sequence $1, 2, 4, 8, 9, 10$, or $1, 2, 4, 5, 10$. Give an efficient algorithm, which given the number N finds the minimum number of such operations to go from 1 to N.

6. You are given an array A of length N (numbered $A[1], \ldots, A[N]$). The entries in the array are integers (positive or negative). A subarray of the array A is of the form $(A[i], A[i+1], \ldots, A[j])$ for some $i \leq j$. Two subarrays are said to be disjoint if they do not have any common array element. A sub-array is said to be positive if the total sum of the values in it is positive. Now, given the array A, we would like to find the minimum number of mutually disjoint subarrays which cover all the **positive** elements in it. For example, if the array is $3, -5, 7, -4, 1, -8, 3, -7, 5, -9, 5, -2, 4$, one solution is to have three sub-arrays: $[3, -5, 7, -4, 1], [3, -7, 5], [5, -2, 4]$. Consider

the following greedy algorithm for solving this problem: let $A[i]$ be the first positive integer from left. Starting from $A[i]$ find the largest index j such that the sub-array $A[i], \ldots, A[j]$ is positive. Take this sub-array and then solve the remaining problems for $A[j+1], \ldots, A[n]$ using the same algorithm. Is this algorithm optimal ? Either prove or disprove it.