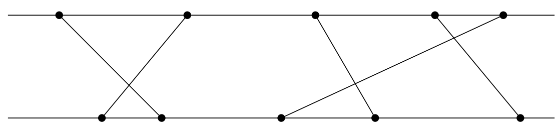
CS/ECE 374 A (Spring 2022)
Homework 10 (due April 21 Thursday at 10am)

Instructions: As in previous homeworks.

Problem 10.1: Consider the following geometric matching problem: Given a set A of n points and a set B of n points in 2D, find a set of n pairs $S = \{(a_1, b_1), \ldots, (a_n, b_n)\}$, with $\{a_1, \ldots, a_n\} = A$ and $\{b_1, \ldots, b_n\} = B$, minimizing $f(S) = \sum_{i=1}^{n} d(a_i, b_i)$. Here, $d(a_i, b_i)$ denotes the Euclidean distance between a_i and b_i (which you may assume can be computed in $O(1)$ time).

Assume that all points in A have y-coordinate equal to 0 and all points in B have y-coordinate equal to 1. (Thus, all points lie on two horizontal lines.) The points are not sorted. See the example below, which shows a solution that is definitely not optimal.



Assignment Project Exam Help
https://tutorcs.com
WeChat: cstutorcs

(a) (20 pts) Consider the following greedy strategy: pick a pair $(a, b) \in A \times B$ minimizing $d(a, b)$; then remove a from A and b from B, and repeat. Give a counterexample showing that this algorithm does not always give an optimal solution.

(b) (40 pts) Let a be the point in A with the smallest x-coordinate. Let b be the point in B with the smallest x-coordinate. Consider a solution S in which a is paired with some point b' with $b' \neq b$, and b is paired with some point a' with $a' \neq a$. Prove that the solution S can be modified to obtain a new solution S' with $f(S') < f(S)$.
(Hint: the triangle inequality[1] might be useful.)

(c) (40 pts) Now give a correct greedy algorithm to solve the problem. (The correctness should follow from (b).) Analyze the running time.

[1] $d(p, q) \leq d(p, z) + d(z, q)$ for any points p, q, z.

Problem 10.2: We are given an unweighted undirected connected graph $G = (V, E)$ with n vertices and m edges (with $m \geq n - 1$), We are also given two vertices $s, t \in V$ and an ordering of the edges $e_1, \ldots, e_m \in E$. Suppose the edges $e_1, \ldots, e_m$ are deleted one by one in that order. We want to determine the first time when s and t become disconnected. In other words, we want to find the smallest index j such that s and t are not connected in the graph $G_j = (V, E - \{e_1, \ldots, e_j\})$.

A naive approach to solve this problem is to run BFS/DFS on G_j for each $j = 1, \ldots, m$, but this would require $O(mn)$ time.[2] You will investigate a more efficient algorithm:

(a) (80 pts) Define a weighted graph G' with the same vertices and edges as G, where edge e_i is given weight $-i$. Let T be the minimum spanning tree of G'. Let π be the path from s to t in T. Let j^* be the smallest index such that e_{j^*} is in π. Prove that the answer to the above problem is exactly j^*.

(b) (20 pts) Following the approach in (a), analyze the running time needed to compute j^*.

Problem 10.3: Consider the following search problem:

> MAX-DISJOINT-TRIPLES:
>
> *Input:* a set S of n positive integers and an integer L.
>
> *Output:* pairwise disjoint triples $\{a_1, b_1, c_1\}, \ldots, \{a_{k^*}, b_{k^*}, c_{k^*}\} \subseteq S$, maximizing the number of triples k^*, such that $a_i + b_i + c_i \leq L$ for each i.

Assignment Project Exam Help

For example, if $S = \{3, 10, 29, 30, 35, 55, 70, 83, 90\}$ and $L = 100$, an optimal solution is $\{3, 10, 83\}, \{29, 30, 35\}$, with two triples (there is no solution with three triples).

https://tutorcs.com

Consider the following decision problem:

WeChat: cstutorcs

> DISJOINT-TRIPLES-DECISION:
>
> *Input:* a set S of n positive integers, an integer L, and an integer k.
>
> *Output:* True iff there exist k pairwise disjoint triples $\{a_1, b_1, c_1\}, \ldots, \{a_k, b_k, c_k\} \subseteq S$, such that $a_i + b_i + c_i \leq L$ for each i.

Prove that MAX-DISJOINT-TRIPLES has a polynomial-time algorithm iff DISJOINT-TRIPLES-DECISION has a polynomial-time algorithm.

(Note: One direction should be easy. For the other direction, see lab 12b for examples of this type of question. In MAX-DISJOINT-TRIPLES, the output is not the optimal value k^* but an optimal set of triples, although it may be helpful to give a subroutine to compute the optimal value k^* as a first step, as in the lab examples.)

[2] Oops, I meant $O(m^2)$.