CSCI 570 - HW 3

Solutions

1. You are given a set $X = \{x_1, x_2, \ldots, x_n\}$ of points on the real line. Your task is to design a greedy algorithm that finds a smallest set of intervals, each of length-2 that contains all the given points. Linked list is a data structure consisting of a group of nodes which together represent a sequence. Under the simplest form, each node is composed of data and a reference (in other words, a link) to the next node in the sequence.
Example: Suppose that $X = \{1.5, 2.0, 2.1, 5.7, 8.8, 9.1, 10.2\}$. Then the three intervals [1.5, 3.5], [4, 6], and [8.7, 10.7] are length-2 intervals such that every $x \in X$ is contained in one of the intervals. Note that 3 is the minimum possible number of intervals because points 1.5, 5.7, and 8.8 are far enough from each other that they have to be covered by 3 distinct intervals. Also, note that the above solution is not unique.

a) Describe the steps of your greedy algorithm in plain English. What is its runtime complexity?
b) Argue that your algorithm correctly finds the smallest set of intervals.

Solution.

a) Sort X = $\{x_1, x_2, \ldots, x_n\}$. Let it be S.

Initialize T = {}
while S not empty:
    select the smallest x from S
    add $[x, x+2]$ into T
    remove all elements within $[x, x+2]$ from S
return T

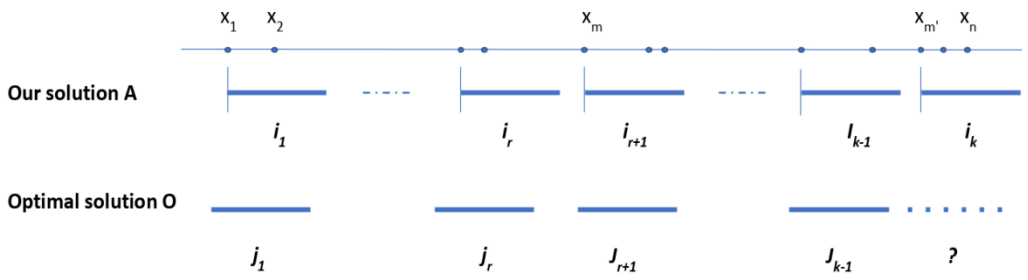
The time complexity of this algorithm is dominated by sorting the input, which can be done in O(n logn) time.

b)Assume there is an optimal solution O. We will prove that the size of our solution A is the same as the size of the optimal solution O.

We will first prove that intervals in our solution (i1 ,i2 …. , in) are never to the left of the corresponding intervals in the optimal solution (j1 ,j2 …. , ip). We will use mathematical induction:

Base Case: i1 is not to the left of j1 since if it were then point x1 will not be covered by j1

Inductive step: Assume ir is not to the left of jr for r ≥ 1, we will prove that ir+1 is not to the left of jr+1



Proof: In above diagram, xm is the leftmost point covered by interval ir+1. Our solution A uses interval ir+1 to cover this point since interval ir was not covering it. And since ir is not to the left of jr then jr will not be able to cover xm either. So, if jr+1 is any further to right of ir+1 then xm will not be covered in O.

Now assume our solution needs k intervals and that the optimal solution O needs fewer intervals. We will prove that this is not possible:

Proof: Our solution needed interval ik because xm' (the leftmost point covered by ik) was not covered by ik-1 and since in step 1 we proved that our intervals are never to the left of the corresponding intervals in the optimal solution, then xm' will not be covered by jk-1 either. So the optimal solution will also need an interval jk to cover xm'. Therefore, the size of our solution is the same as the size of optimal solution.

2. You are given a minimum spanning tree T in a graph $G = (V, E)$. Suppose we remove an edge from G creating a new graph G_1. Assuming that G_1 is still connected, devise a linear time algorithm to find a MST in G_1.

Solution. Once we delete an edge from T, the tree becomes disconnected. We need to find the minimum weight edge that connects two components, sat T_1 and T_2. Pick any component say T_2 and find all edges going to T_1. Among them choose the one which has the smallest cost. Runtime O(E).

3. Given a graph $G = (V, E)$ with nonnegative edge weights and the shortest path distances d(s, u) from a source vertex s to all other vertices in G. However, you are not given the shortest path tree. Devise a linear time algorithm, to find a shortest path from s to a given vertex t.

Solution. Reverse edges in the original graph. This could be done in linear time (for example, using matrix transform). For all vertices that are connected to a given vertex t, find ones such that d(s, t) = d(s, u) + d(u, t). Next, set u as t and repeat this to find new u until s equals to u. Nodes saved in the sequence are the shortest path from s to a given vertex t.

4. Suppose you were to drive from USC to Santa Monica along I-10. Your gas tank, when full, holds enough gas to go p miles, and you have a map that contains the information on the distances between gas stations along the route. Let $d_1 < d_2 < \ldots < d_n$ be the locations of all the gas stations along the route where d_i is the distance from USC to the gas station. We assume that the distance between neighboring gas stations is at most p miles. Your goal is to make as few gas stops as possible along the way. Design a greedy algorithm to determine at which gas stations you should stop and prove that your strategy yields an optimal solution.

Solution. The greedy strategy is to go as far as possible before stopping for gas. That is when you are at the i-th gas station, if you have enough gas to go the (i+1)-st gas station, then skip the i-th gas station. Otherwise stop at the i-th station and fill up the tank.
Let $\{g_1,g_2,\ldots,g_m\}$ be the set of gas stations at which our algorithm made us refuel.
Let $\{h_1,h_2,\ldots,h_k\}$ be an optimal solution.
We next prove that our choice is optimal.
Consider the first refuel. Note that h_1 cannot be less than g_1, because in this case the optimal solution will need more refuels (the range in the next drive segment would be greater). Also h_1 cannot be greater than g_1, because in this case the optimal solution will run out of gas before reaching the next gas station. So $g_1 = h_1$. Using induction we prove it for all other refuels.