CSCI 570 - HW3 Isaac Delgado
4189182267

1.) You are given a set $X = \{x1, x2, \ldots, xn\}$ 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.

A. Describe the steps of your greedy algorithm in plain English. What is its runtime complexity?

If our set X is not sorted then we must first sort the set.

The lowest point will the beginning of our first interval of size 2. Thus our first interval will be from x1 to (x1+2).

We will traverse through our set and if the next item is within our interval we continue. If our next item is not in the interval then this item will be the start of our second interval. This will repeated until we have finished traversing through the set. Consequently every item in our set will be within an interval.

The runtime complexity will be O(n log n) for our sorting and O(n) as we go through our set creating our intervals. So our runtime complexity will just be O(n log n)

B. Argue that your algorithm correctly finds the smallest set of intervals.

Let our intervals chosen by our algorithm be the set {i1, i2, i3,…., ik} and let the optimal subset of intervals be the set {j1, j2,….,jk}.

Our base case will be the following:

i1 = j1

This is true because we always start our first interval with first point in our set. Any other starting position would reduce the amount of coverage from the interval and thus this must be identical to the optimal first interval.

Inductive Hypothesis:

ik-1 ≤ jk-1

Inductive Step:

Prove for ik ≤ jk

We know that our intervals cannot overlap because if they did then they would not be optimal. Since our optimal set is the optimal we know that jk-1 comes right before jk. Or more formally f(jk-1) ≤ s(jk) (where f() and s() denote finish time and start time respectively). Since our algorithm takes an interval ik-1 where f(ik-1) ≤ f(jk-1) then it follows that our algorithm would choose jk next as it would be the next available interval. This process would guarantee not overlapping intervals and since we always start or intervals at points existing in our set there will not be any gaps or points not covered by an interval.

Isaac Delgado
4189182267

2.) 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 $d_1 < d_2 < ... < d_n$ 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.

Our methodology should be to drive as far as we can go without stopping for gas. Since we know the distances to all the gas stations from USC then we can calculate p miles down the road and stop at the gas station closest and BEFORE to the point p miles.

Since in worst case the distance between gas stations is p miles, then this guarantees that we will always pass or get to a fast station on a full tank of gas. Additionally by stopping at the gas station immediately before or at p miles this guarantees that we stopped at the last possible gas station since we cannot traverse past p miles. Repeating this process will give us the least amount of stops as we always stop closest to p miles in every iteration.

3.) 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.

If the edge that was removed is not in our minimum spanning tree T then our MST is unchanged and we do not have to compute anything.

If the edge removed was in our original minimum spanning tree T then we simply travers our graph and add the smallest edged not already in our MST. This will take linear time as we will visit every vertex and edge in worst case. O(V+E).

Isaac Delgado
4189182267

4.) Given a directed 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.

The idea here should be to traverse the graph in a backtracking method. Since we know the distances from all the vertices we can simply backtrack by adding some constant to each vertex distance and test which one adds up to the d(s, t). Which ever distance adds up to d(s, t) will be the shortest edge and thus we repeat with the vertex that we found.

For example, say $d(s, g) + c = d(s, t)$, if this is true then we know this is our shortest edge. We will then repeat this process by finding another vertex with a constant c that will add up to d(s, g).

Time complexity is linear because in worst case we will be testing every vertex and every edge along the way, thus time complexity is $O(V + E)$.