Homework 4

Due date: 2/03/2021 right before class

Problem 1

In a neighborhood there are n radio stations $R_1 ... R_n$. Each radio station i has a broadcast B_i associated with it. B_i is a tuple that has (*start time, end time, power*). When more than one station broadcasts at the same point in time, only the broadcast with the highest power can be heard.

Write a greedy algorithm to output a sequence of times alternating with stations, namely between time t_i and t_{i+1}, station R_j will be heard. Explain why your algorithm is Greedy (Remember, you are asked to solve the problem by a Greedy Algorithm). Your algorithm should run in $O(n(log(n)))$ time.

0.1 Solution 1

Create a max heap and push all broadcasts B into it. Heap will sort based on the power of each broadcast. At each iteration, pop the highest power broadcast, and if not already reserved, reserve the amount of time (or only the non reserved part) from start time to end time in a result array. Keep popping the heap until no more broadcasts remain.

Complexity will be heap construction= O(n), and popping all elements= (O(nlogn)

By contradiciont we assume our heap popping results in the wrong time series. That means we popped an element with a lower power before an element with a higher power. This is impossible by the heap properties. Contradiction.

Problem 2

A cellular company is trying to locate one of their subscribers. The subscriber is in one of n cells $c_1 ... c_n$. Each cell is associated with a pair $c_i = (\pi_i, p_i)$, where π is the probability of the subscriber being found at that cell, and p_i is the amount of power needed to cover cell c_i. The search will continue until the customer is found. To find a customer in a cell, the tower in the cell has to be activated for a unit of time. If the subscriber is there it will respond. Otherwise, no response. The company can activate at most one tower in a unit of time.

You need to compute the order of cell activation that will minimize the expect power needed to find the subscriber. Write the expression that needs to be minimized over all orders of cells.

Write a Greedy algorithm to minimize that expression. For example, if all cells needed equal power, then you would simply conduct your search in decreasing order of probability. This will minimize the average time to find the subscriber (prove!).

0.2 Solution 2

Over all cells we wish to minimize $\sum_i^n \pi_i(\sum_j^i p_j)$.

To minimize our global term we sort cells by π_i/p_i and start exploring in decreasing order.

This works since we minimize the global term, by exploring cells based on their expected returns. We explore cells who have a higher expected value weighted over probability and power consumption.

Problem 3

Give an alternative Greedy Algorithm for the interval scheduling algorithm. Your algorithm should select the first interval to be in a solution based on a different criteria then earliest termination time.

0.3 Solution 3

The idea is to sort intervals by latest start time.

```
init array sol <- first item in the latest start time array

For all remaining items:
    If end time is earlier then start time of last item added
        sol <- item
return reverse(sol)
```

Listing 1: algo

This is exactly equivalent to the interval scheduling algorithm we learned in class, only going top down.

Problem 4

In the case of a MST

a. If we add the same constant value to each weight in the MST, would the MST change or remain the same? Argue why.

b. If we multiply each weight by the same constant factor, would the MST change or remain the same? Argue why.

Answer the above for the tree of shortest path graph with positive weights. A tree of shortest path is a subgraph which is tree, rooted at the node r from which we compute the shortest paths, and the path from r to node v in the tree, is a shortest path from r to v in the graph.

Now think of running Dijkstra on an undirected graph $G = (V, E)$. When running Dijkstra on the undirected graph, we think of each edge e_i as two directed anti parallel edges with equal weight as e_i.

Think about increasing the weights of the shortest path tree by an arbitrary variable, what happens when we gradually increase that variable? at some point the tree will converge, when it does, what is the property of the tree we have?

0.4 Solution 4

a If we add the same constant value to each weight, MST remains the same.
Reason: When generating a MST from a graph G, we focus on the ordering of the weights. Adding a constant value to each edge only results in the change in total weights of a MST. It does not affect the ordering of the weights.

b f we multiply each weight by the same constant factor, MST remains the same.
Reason: Multiplying a factor (more specifically, a positive factor) will not affect the ordering of the weights in an MST, since all weights are factored by the same factor.

If we raise the weights of a graph that contained a shortest path consistently higher and higher by a constant factor, we can see that we might need to rerun the shortest path algorithm to find a new shorter path. Eventually, when we increased the weights enough, the shortest path we will find will be an MST as well.