ADA Homework Assignment 3

Deadline : April 6 (Tuesday) 11.59 pm.

The theory assignment has to be done in a team of at most two members, as already selected by you. The solutions are to be typed either as a word document or latex-ed and uploaded as pdf on GC. We shall strictly not accept solutions written in any other form. Remember that both team members need to upload the HW solution on GC.

Collaboration across teams or seeking help from any sources other than the lectures, notes and texts mentioned on the homepage will be considered an act of plagiarism.

Some questions below are from the text by Jeff Erickson. You can find an online copy here.

Problem 1 (10 points) Recall the following problem from the coding assignment.

Problem 1 *You are lord Indra and you want to cause rain in a village. There n festivals when the villagers worship you, say festival i occurs on date t_i. You want to cause rain on exactly k of these festivals. Each time it rains, the villagers become complacent and take the rain for granted. You don't want this to happen ; so you want the minimum duration between two consecutive rains to be as long as possible (assume that you have to cause rain on the very first festival).*

Hence, you want to pick k out of the n festival days on which you wish it to rain, such that the minimum duration between any pair of rains is as long as possible. Give a polynomial time algorithm to achieve this.

For the theory part, prove the correctness of your algorithm.

Solution.

Here will assume that $k \geq 2$, else the answer is undefined (or it can be thought of as infinity).

Let V denote any vector $t_1, t_2, \ldots, t_n$ of festival dates. Then suppose we can efficiently solve the following decision version of our problem

Problem 2 *Suppose we are given festivals at V and k rains (including a rain at the first festival). Additionally, we are given a number d. Then, is the optimum value for problem 1 at least d ?*

Then, we can simply binary search on d to solve our original optimization problem. More precisely, given an algorithm `DECIDE`(V, k, d) for problem 2, the algorithm `MAXDURATION`(V, k) solves problem 1.

The correctness of algorithm 1 follows from the invariant that at the start of each iteration of the while loop, the optimum duration lies in the interval $[a, b]$.

Algorithm 1 Optimization via decision version

```
1:  procedure MAXDURATION($V, k$)                    ▷ $V$ is a vector $t_1, t_2, \dots, t_n$ of dates
2:      $D \leftarrow t_n - t_1$.                               ▷ Maximum possible duration.
3:      Initialize $a \leftarrow 1, b \leftarrow D$.
4:      while $b > a$ do          ▷ Invariant : the optimum delay lies in the interval $[a, b]$
5:          $d \leftarrow \lceil (a + b)/2 \rceil$
6:          if DECIDE($V, k, d$) then
7:              $a \leftarrow d$
8:          else
9:              $b \leftarrow d - 1$
10:
11:             return $a$                                           ▷ $a = b$
12:
```

Time Complexity of Algorithm 1 makes $O(\log_2(t_n - t_1))$ calls to DECIDE because $b - a$ halves after each call to DECIDE and initially $b - a = D = t_n - t_1$ (and $a = b$ at the end).

Now to solve problem 2, it is enough to solve its following optimization version.

Problem 3 *Given festivals at dates V and a number d, find the maximum number of rains possible so that minimum duration between any pair of rains is at least d.*

Given an algorithm MAXRAIN(V, d) to solve problem 3, DECIDE(V, k, d) simply returns whether MAXRAIN(V, d) is at least k, see algorithm 2.

Algorithm 2 Decision version

```
procedure DECIDE($V, k, d$)                          ▷ $V$ is a vector $t_1, t_2, \dots, t_n$ of dates
    return (MAXRAIN($V, d$) $\geq k$)
end procedure=0
```

Hence, we now focus on solving problem 3.

Minimizing the number of towers needed to achieve a given maximum delay

We give a natural greedy algorithm for problem 3 : Make rain at festival 1. Consider the first festival i such that $t_i - t_1 \geq d$ (assume $d \geq 1$). Recurse on the festivals $i, i + 1, \dots, n$.

Algorithm 3 recursively solves this problem for festivals $j, j + 1, \dots, n$. Run this algorithm for $j = 1$.

Proof of correctness:

Proof:

Let us fix some feasible subset S of rains, and let G be the subset returned by the greedy algorithm. Clearly G is feasible by construction. We want to show that $|G| \geq |S|$. The following claim is the key to everything

Claim 4 *Suppose the second rain in G (respectively S) is at festival g (resp. s). Then, $s \geq g$ or equivalently, $t_s \geq t_g$.*

Algorithm 3 Maximizing rains to achieve a given minimum duration for festivals $j, j+1, \ldots n$

```
1: procedure MAXRAIN($V, d, j$)                    ▷ $V$ is a vector $t_1, t_2, \ldots, t_n$ of dates
2:     for all $i = j+1, \ldots$ do
3:         if $t_i - t_j \geq d$ then                  ▷ Next rain will be on $i$
4:             return 1 + MAXRAIN($V, d, i$)
5:         end if
6:     end for
7:     return 1                                     ▷ only one rain possible
8: end procedure
```

Proof: Since S is feasible, $t_s - t_1 \geq d$. Since g is the first festival i satisfying $t_i - t_1 \geq d$, it must be that $s \geq g$. Equivalently, $t_s \geq t_g$. □

Claim 4 shows that after the choice of the second rain, the greedy solution G is 'ahead' of any other solution S in the following sense :

Claim 5 *The subset of rains $S' = (S - \{1, s\}) \cup g$ is feasible for the festivals $g, g+1, \ldots$.*

Proof: The first rain in S' is at festival g. The second rain in S' occurs at some date t satisfying $t - t_s \geq d$ (since S is feasible for the set of all festivals and there is a rain at festival s). Hence, $t - t_g \geq d$ (since $t_g \leq t_s$ by claim 4). Hence, the duration between the first and the second rain in S' is at least d. Again, by the feasibility of S, the duration between any consecutive pair of rains in S' after the first pair is at least d. Therefore, S' is feasible for the festivals $g, g+1, \ldots$. □

We can inductively assume that greedy is optimal for any smaller subset of festivals than V, hence $G - 1$ is an optimal solution for the festivals $g, g+1, \ldots$. Since, S' is feasible for the festivals $g, g+1, \ldots$, we have that $|S'| = |S| - 1 \leq |G - 1|$ and hence $|S| \leq |G|$.

□

Time complexity of the greedy algorithm: In algorithm 3, we look at any festival at most once in the iterative loop. Hence, the time complexity is $O(n)$.

Final time complexity There are $O(\log(t_n - t_1))$ calls to `DECIDE` which makes a single call to `MAXRAIN`. `MAXRAIN` takes $O(n)$ time, hence the overall running time is $O(n \log O(t_n - t_1))$.

Problem 2

a. (5 points) Solve part $4(a)$ from the above text.(you might as well just state a reduction to a known problem and justify why a known algorithm for that problem also works in this case)

   Solution. There are several ways to solve this. The easiest is to reduce it to solving the standard MST problem. Suppose we define a new graph G' which is the same graph G but with the edge weights negated - that is $w'_e = -w_e, \forall e \in E$. Now clearly, the minimum spanning tree in G' will be the maximum spanning tree in G.

   Now consider any algorithm to compute MST - say Prim's. The crucial property that we use to prove that Prim's produces an MST is the cut property. The cut property continues to be satisfied even with negative edge weights. Hence, we are done.

b. (10 points) Solve part $9(a)$ from the above text, assuming that all edge weights are distinct. (*Hint:* Understand the proof of Prim's Algorithm till the last detail)

Solution. Suppose for the purpose of contradiction that there exists a pair of vertices u, v such that the uv path in the Max ST T is not the widest path between u, v as claimed. Suppose the minimum weight edge of this path is e. Clearly, there exists another path P' between u, v in G such that all edge weights are larger than e, by our assumption. Now, suppose we remove e from T. This forms a cut S, S' such that $u \in S, v \in S'$ and further both S and S' form connected components (this is very critical). Now, the path P' has to cross this cut - say using edge $e' \neq e$. Hence $T \setminus e \cup e'$ is another spanning tree of G which has strictly larger weight than T - this contradicts the optimality of T and hence we are done.