Assignment 2

Mehul Jain - 112072812
CSE 548 - Analysis of Algorithm

October 2, 2018

Problem-1

If task J_i is scheduled before J_j , then following should be total execution time of the system with 2 process i and j: max($p_i + f_i, p_i + p_j + f_j$) ..... (1)
If task J_j is scheduled before J_i , then following should be total execution time of the system with 2 process i and j: max($p_j + f_j, p_j + p_i + f_i$) .....(2)
let us consider an optimal solution in which we have task J_i scheduled before task J_j and now lets try finding a condition when this should be true.
max($p_i + f_i, p_i + p_j + f_j$) < max($p_j + f_j, p_j + p_i + f_i$)
For the expression-2 to be more than expression-1, max($p_j + f_j, p_j + p_i + f_i$) should be equal to $p_j + p_i + f_i$ since expression-1 has a term $p_i + p_j + f_j$ which is greater than $p_j + f_j$ and will cause our condition to fail. So now we are left with 2 conditions.
$p_i + f_i < p_j + p_i + f_i$ ... (3) and
$p_i + p_j + f_j < p_j + p_i + f_i$ ..(4)
As we can see, condition 3 will always be true since p_j will always be greater than 0.
from condition 4 we get, $f_i > f_j$ if we want to execute optimally when Task J_i is executed before J_j. This condition should take be true between and 2 process in system having n Tasks.
Final Solution: Sort all Jobs in decreasing order of f time and then execute task with higher f time should be executed first. Complexity $O(nlogn)$ for sorting.

Problem-2

Let c be the total time elapsed after which we schedule two process J_i and J_j. If task J_i is scheduled before J_j , then following should be total weighted time of the of these 2 process i and j: (c+t_i)w_i + (c+t_i+t_j)w_2 ..... (1)
If task J_j is scheduled before J_i , then following should be total weighted time of the of these 2 process i and j: (c+t_j)w_j + (c+t_j+t_i)w_i .....(2)
let us consider an optimal solution in which we have task J_i scheduled before task J_j and now lets try finding a condition when this should be true.
(c+t_i)w_i + (c+t_i+t_j)w_2 < (c+t_j)w_j + (c+t_j+t_i)w_i
from this we get, $t_i w_j < t_j w_i$
$w_j/t_j < w_i/t_i$
We have proved this condition to be true to get an optimal solution for scheduling between two task. This condition can be extended between any two process in the system to decide the Job schedule.
Final Solution: Sort all Jobs in decreasing order of w/t and then execute task with higher w/j values before any other jobs. Complexity $O(nlogn)$ for sorting.

Problem-3

```
Convert all n Jobs time into 24 hour format.
Sort all jobs in increasing order of finish time.

Lets say for a particular Task Ti, Ti.start be the starting time and Ti.

MaxJobList = {}

for j=0 to n
 baseIndex =-1
 baseMax = 0
 currentMinStartTime=0
 CurrentJobList = {}
   for i=j to n
      if Ti.end > Ti.start and Ti.start > currentMinStartTime:
       currentMinStartTime = Ti.end
       CurrentJobList.add(Ti)
       if baseIndex == -1:
```

```
        baseIndex = i
        baseMax = T_i.start

for i=1 to j-1
    if T_i.start > currentMinStartTime & T_i.end < baseMax:
     CurrentJobList.add(T_i)
     currentMinStartTime = T_i.end
     AddJob(T_i)

if CurrentJobList.size > MaxJobList.size
 MaxJobList = CurrentJobList
```

Complexity = $O(n^2)$

Problem-4

Part-1 True
Let us consider a path P between two nodes u,v in MST with heaviest edge as E. Now let us assume we can also have a path P2 between u and v which has the heaviest edge less than E. Since both path P and P2 connect u and v, we can say it is a cycle. And according to the cycle property of MST, we cannot have edge E, which is not the case. So if P2 exist it will surely have path in which it will have edge E' which will be greater than E. Cycle Property: If there exist a cycle in a graph than the heaviest edge in that path will not be a part of MST, since if we exclude it we will have a path which will have the least value and it will still be able to connect all the nodes in that circle.
Part -2 True
Suppose we divide our Minimum altitude graph into any two connected components. The only way to connect them would be to connect them would be to use them minimum edge. This is what we do in MST algorithm. So any two connected component is connected using minimum weight edge, and hence the entire Minimum altitude graph will contain all edges of MST.

Problem-5

(In this question We will refer to component- as a tree build so far from all the leaves in that tree) Initially, let us create n components C1 to Cn, where P1 is a member of C1, P2 is a member of C2 and so on. Proposed Solution: Let us exe-

cute Kruskal's Algorithm and connect any 2 components at a time and merging them into a single one, the resulting component will be a new Tree with a new node N as the root with height equal to the d(pi,pj),where edge pi-pj is edge selected by Kruskal's Algorithm to connect these two component. And lets do this thing recursively till we have a single connected Component.
Part A. τ is consistent with d:
Let's take 2 node pi and pj and prove this. Let us connect Component C1 having node pi and Component C2 having node pj, we will not select the value of τ(pi,pj) > d(pi,pj) because according to kruskal Algo, we will always select minimum edge connecting two Components of which they are part of as tau(pi,pj) which will be less than or equal to d(pi,pj). Also since we are recursively building a tree from bottom to top, any node u which is a parent of node v will always have h(u) > h(v) as edges are selected in sorted order in Kruskals Algorithm.
Part B. Let's say we have 3 components,
Distance (C1,C2) represent minimum distance between the edge connecting those two components. Lets say we have, D(C1,C2) < D(C2,C3) < D(C3,C1). The optimal way according to Kruskal would have been to connect components C1 and C2, and then C3.If we do something different, lets say connect component C2 and C3 first as Component c4, height h(c4) would be equal to D(C2,C3). Now if we connect component c1 to c4, the height value should be greater or equal to D(C1,C3) which will violate our condition for some nodes p1 in C1 and p2 in C2 who should have a $\tau(p1, p2)$ to be less than or equal to D(C1,C2). Hence our solution is optimal.

Problem-6

In Kruskal's Algorithm, if the relative order weight of edges is same , we will always have the same edges selected in MST. So now when will the relative order of our edges change? It will change before and after the time when values of two edges(quadratic equation) becomes same. Since every two quadratic equation (parabola) can intersect 2 time, m edges will intersect 2*$\binom{m}{2}$ times. So lets say this gives us K values. So if we sort them in ascending order we will have total of K+1 time intervals.Now for each time intervals we will put some random t values and find the edges that will be selected in that interval using kruskals Algorithm. After we find all the edges, we should now add the actual quadratic equations and find its minimum value for that interval and store at which time t are we getting that value. And then we can compare min values

across all these K+1 intervals to get our solution. How to find min value and also time t? The equation that we get by adding all the quadratic equation will itself be a quadratic equation of the form $ax^2 + bx + c$. Now we find t' = -b/2a, this value is the time t at which the equation will have a global minima. But if this value does not lie between that range we can just compare the end points of the intervals and use the minimum that we get.
Complexity m^2*$O(constant)$ to find points where all the quadratic equations intersect. And for each interval we apply kruskal's Algorithm i $O(mlogm)$ and $O(constant)$ to find time in that interval. total : $m^3log(m)$

Problem-7

Consider the shortest Path P between u and v , going through node N nodes. Now since this path is composed by connecting nodes such that the distance between adjacent nodes in Path is less than 3 times their shortest distance in G, the resulting path between u and v will also be less than 3 times the shortest distance between them in G. Suppose there existed a subpath p' in P which is greater than 3 times the path in G connecting those nodes, it would have been replaced by the appropriate path using the modified Kruskal's Algorithm.