CSE548 Fall 2018 Analysis of Algorithms

Ayush Garg (SBU ID 111870086)

October 1, 2018

1 Question 1

Given job J_i , Processing time p_i, Finishing time f_i for i=(1,2,....,n)

Algorithm:

1. Sort the jobs according to the finishing time f_i
2. If two jobs have same f_i, then choose the one with minimum p_i
3. Schedule the jobs

Proof:

Suppose in our greedy algorithm, a job J_i comes before another job J_j. That means $f_i > f_j$.

Then the total time taken will be

1. **case2:** If Job J_j completes before Job J_i.Then, total completion time would be T= p_i+f_i
2. **case1:** If Job J_j completes after Job J_i. Then, total completion time would be T= p_i+p_j+f_j

Now suppose there is an optimal algorithm in which job J_j comes before job J_i.

Then the total completion time would be:

1. **case 2:** There will be only one case. I.e. job J_j will always complete before Job J_i. Hence the total completion time would be T= p_j+ p_i+ f_i. [**Note** that f_i >f_j]

Now, If we compare the completion time of 2 algorithms, we will observe that T(Optimal) > T(Greedy case 1) [Since 2 terms p_i and f_i are equal and there is one additional term p_j which will increase the total time in optimal]

and T(Optimal) > T(Greedy case 2) [Since 2 terms p_i and p_j are equal and comparing the other terms $f_i > f_j$.]

Hence total time of our greedy algorithm will not be worse than the optimal algorithm and hence our proof is correct.

2 Question 2

Given n jobs. Finishing time of job i is C_i. Processing time is t_i and a Weight w_i. We need to minimize $\sum_{i=1}^{n} w_i C_i$

Algorithm:

1. Sort the jobs according to the decreasing order of w_i/t_i.
2. If two factors are same, choose any.
3. Schedule the jobs.

Proof:

Suppose in our greedy algorithm, a job i comes before another job j. Then, according to our algorithm, $w_i/t_i > w_j/t_j$ Then the total cost of the 2 jobs T would be $\sum_{i=1}^{n} w_i C_i$

T $= w_i C_i + w_j C_j$

If the jobs start at some initial time t_0, then, $C_i = t_0 + t_i$ and $C_j = t_0 + t_i + t_j$

Therefore, T $= w_i(t_0 + t_i) + w_j(t_0 + t_i + t_j)$

T$= w_i t_0 + w_i t_i + w_j t_0 + w_j t_i + w_j t_j$...equation (1)

Suppose there is an optimal algorithm in which job j comes before job i

Then the total cost of the 2 jobs would be $\sum_{i=1}^{n} w_i C_i$ which is equal to:

T $= w_j C_j + w_i C_i$

If the jobs start at some initial time t_0, then, $C_j = t_0 + t_j$ and $C_i = t_0 + t_j + t_i$

Therefore, T $= w_j(t_0 + t_j) + w_i(t_0 + t_j + t_i)$

T$= w_j t_0 + w_j t_j + w_i t_0 + w_i t_j + w_i t_i$...equation (2)

Now from equation (1) and (2), cancelling out the common terms, we get

T(Greedy) $= w_j t_i$ and T(Optimal) $= w_i t_j$

But we know that $w_i/t_i > w_j/t_j$

cross multiplying we get $w_i tj > w_j t_i$

Hence T(Optimal)$\geq$T(Greedy). Hence our Greedy algorithm is not worse than the optimal algorithm.

Hence we can exchange the order of jobs i and j in our optimal algorithm. Similarly if we keep on exchanging the jobs we can convert the optimal solution to our Greedy one. Hence proved that our algorithm is optimal

3 Question 3

Given n daily 24 hour jobs. Along with the *(start time,end time)* pairs of each job.

It is a different version of Interval Scheduling problem where the jobs can run over midnight and we have to find an algorithm in polynomial time to accept maximum number of jobs.

Basically we have to find an optimal start time to break the clock from where we can apply our normal Interval Scheduling Algorithm.

Algorithm:

1. Sort the jobs in increasing order of the *end time* of the jobs (earliest deadline first). Let max be the maximum number of jobs that could be scheduled.

2. For each of the job i in n jobs perform the following:

   (a) Take *start time* of the job i as the starting time of our Interval Scheduling problem over a period of 24 hours.

   (b) Take the job that has the earliest finishing time and schedule it. Remove all jobs that are overlapping with the scheduled job.

   (c) Stop when bag is empty

   (d) Suppose m is the number of jobs that could be scheduled for i. if m $>$ max, then update max = m.

3. Choose the Schedule where you can accept max number of jobs.

Running time complexity of step 1. is $nlogn$ and for step 2 is n^2. Hence total complexity is n^2 is in polynomial time.

4 Question 4

We are given a graph $G = (V, E)$. Each edge e is annotated with a highest altitude a_e

1. Yes the following conjecture is true for all choices of G and distinct altitudes a_e.

   Proof: We will prove this by contradiction.

   Suppose the Minimum spanning tree M of a graph is not the minimum altitude connected sub graph A. This means that there exists another path between two towns in A, that is not present in M, in which the maximum edge weight is less than that of the path in M.

   This means there exists a cycle in the graph G in which there are two paths between the towns where the path with maximum edge weight is included in the MST M and the path with lower maximum edge weight is included in the minimum altitude sub graph A but not M.

   However, by the cycle property of MST, we know that this is not possible. Cycle property of MST states that if in a graph there exists two paths (cycle) between any 2 points then we include that path which has minimum edge weight in our MST. Which is a Contradiction.

   Hence it is not possible to have a MST that is not the minimum altitude sub graph.

2. In the first part we proved that an MST of G is a minimum altitude connected sub graph. Now, we will prove the reverse that a sub graph cannot be a minimum altitude sub graph if it does not contain the edges

of the the MST. (**Note** that we have to prove if and only if, but one part of the question has already been proved above. So will prove the other remaining part here.)

Yes the second conjecture is also true for all choices of G and distinct altitudes a_e. We will prove this too by contradiction.

Suppose there exists a minimum altitude sub graph A of $G = (V, E)$ in which all the edges of MST M are not present. This means that there exists an edge connecting two towns i and j that is present in A but not in M. This means there exists 2 paths between i and j, One that is going through MST M and the other that is not going through MST M but through minimum altitude connected sub graph A. This means that there exists a cycle between i and j in the graph.

Now, Since A is the minimum altitude connected sub graph, this means that the path in A between i and j contains the edge with the lower maximum edge weight a_e and the path in MST M contains the higher maximum weight edge. However we know by the cycle property of MST that this is a contradiction.

Hence proved that sub graph of G is a minimum altitude sub graph if and only if it contains the edges of the MST.

5 Question 5

Given a set of points P=p1,p2,....,pn in graph. A distance function d on set P. A hierarchical metric $\tau(pi, pj) \leq d(pi, pj)$ Our polynomial time algorithm is as follows:

1. (a) We will sort the edges according to the increasing order of weight.
   (b) Then we start merging disconnected components into 1 component as we do in kruskal. As we merge, we will create a new intermediate node at τ value (height) h for edge weight h.
   (c) Since we are allocating node for each edge length and τ is allocated before, for any 2 points pi and pj before their direct edge comes in order. Hence it only makes sense that $\tau \leq d(pi, pj)$ [since its an increasing order]

   2. We will try to prove this by contradiction. Suppose $\tau'(pi, pj) > \tau(pi, pj)$ Suppose a node p comes before the first node of τ'. That means d(p,first node of τ') should be greater than τ since the least common ancestor will always lie above. But when, in our sorted order, edge (pi,pj) comes, all edges would be present and hence edge length should be smaller than $\tau(pi, pj)$ which is a contradiction to our definition we described above.

      Hence proved.

6 Question 6

Given a graph G=(v,E). Each edge e has a cost $f_e(t)$ which is a function of time t. Hence the total cost $C_G(t)$ will change with time t and even the MST will change. Now we have to find the value of t at which cost is minimized.

$f_e(t) = a_e t^2 + b_e t + c_e$

Now,

If we look at kruskal algorithm, it sorts the edges according to the increasing order of weights. Even if edges are the function of time t, the order of the edges will remain the same even if the weights keep on increasing or decreasing. Hence the MST formed by kruskal algorithm will remain same most of the time even if the edges weight keep on changing with time. time complexity would be $O(nlogn)$ for n edges for this step.

We observe that $f_e(t) = a_e t^2 + b_e t + c_e$ is a parabola.

The order of any two edges in the kruskal algorithm will vary only 2 times when the parabolas equation of the 2 edges intersect at 2 points. These are the points at which our MST will change.

Now, for all the edges,

Number of times MST can change = $\binom{m}{2}$ [for every edge pair].

We will apply kruskal algorithm to each of those points where MST order changes and we will make the new MST and will store the MST with minimum cost $C_G(t)$. [time complexity $O(n^2 * nlogn)$ for this step]

7 Question 7

We are given a Graph $G = (V, E)$ with positive edge lengths l_e. A denser sub graph $H = (V, F)$ where we will add an edge e to H if $3l_e < d_{uv}$.

1. **Proof:** Suppose there are two nodes, u,v in V. Now suppose there exists two paths between the nodes. One path is in H and the other path is in G. By definition path between the two nodes in G is smaller than that of H.

   Now, by the above kruskal algorithm, we know that we will only add the cycle to our MST if the resulting path between the two nodes is less than one-third of the distance of the existing path in H.

   Now, We will try to prove this by contradiction. Suppose there exists a shortest path in H which is more than 3 times the path in G between any two points u,v.

   Hence there exists a cycle in G between u,v whose shortest path is G is less than one third of the shortest path in H. However, we know by the definition of kruskal algorithm given in the question that this is a contradiction. Since, while adding the edges if we find that any edge makes $3l_e < d_{uv}$, then we add that edge to our H. Hence we would add such an edge (if present) into our G and thus it is not possible.

   Hence proved.

2. **Proof:** We know that the graph $G = (V, E)$ can have only short cycles (of length not more than 3). Since whenever there would be a long cycle of > 3, then according to our algorithm above, condition $3l_e < d_{uv}$ then that 4th edge should not be added because that would make the total cost d more than $3l_e$. And if the above condition is true then it is always the case that the number of edges will always be less than $O(n^2)$. [property of short cycles]. Hence proved $\sum_{i=1}^{n} f(n)/n^2 = 0$

8 References

1. Discussion with Classmates
2. Wikipedia.com