CIS 502 - Algorithms

Fall 2015 Homework 2

Problem 1 *Problem 17, page 197, of textbook.*

Solution: Suppose we were told which job was running at midnight (also including the case that no job was running at midnight). Then contingent on scheduling this job we have a linear interval on which we have to schedule the rest of the jobs (the overlap issues do not exist, because all jobs have to finish before the midnight job needs to be started). We have an $O(n \log n)$ algorithm based on earliest finish time.

We simply guess the midnight job. There are n guesses and for each guess we run the $O(n \log n)$ algorithm for the remainder of the schedule. Naively, this takes $O(n^2 \log n)$ time. However the sorting does not need to be performed for each of the guesses separately, in particular if the set of jobs running at midnight were S then we need to sort the $n - |S|$ jobs based on their finish time exactly once. Now for a fixed guess that optimum is running job $i \in S$ at midnight, we need to discard the jobs that start before i finishes, but that can be done in $O(n)$ time. So the algorithm is

---

1: Pick an arbitrary time t.
2: Let the set of jobs which can be running at time t be S.
3: Sort the remaining $n - |S|$ jobs based on their finish times.
4: **for** Each $i \in S$ **do**
5: Let $i' = i$.
6: Consider the job that finishes first. If this overlaps with i' then ignore the job. Otherwise schedule it. And i' is now this scheduled job.
7: **end for**
8: Also consider the case $i' = -1$ (no job is scheduled at time t).
9: Output the best schedule of the $|S| + 1$ schedules considered.

Problem 2 *Problem 20, page 199-200, of textbook.*

Solution: We only need to prove Part (b). We show that every edge added by Kruskal's algorithm has to be added to the min-altitude subgraph.

Suppose not. Let (u, v) be the first time we add an edge in Kruskal's algorithm which is not in our min-altitude subgraph. Consider the cut defined by the edge (u, v), say $S, V \setminus S$ (where S corresponds to the nodes in the subtree connected to u). Since every other edge that spans the cut has altitude more than that of (u, v), the minimum altitude path between u and v itself must be the edge (u, v). Thus we have a contradiction to the min-altitude subgraph property for this pair.

We now show that the min-altitude subgraph is a tree, provided that the weights are distinct. Suppose not. Consider a cycle – the highest altitude edge can be deleted without affecting the connectivity in any way. Therefore the min-altitude subgraph is exactly the minimum spanning tree.

Problem 3 *Problem 21, page 200, of textbook.*

Solution: We will use the cycle property: that given any cycle there exists a minimum spanning tree that does not use the heaviest edge in that cycle.

We first ensure that the graph is connected in $O(n+n+8) = O(n)$ time. We use our favourite algorithm to find a cycle (BFS/DFS) and can find a cycle in $O(n + m) = O(n + n + 8) = O(n)$ time. We then delete one edge. We now have $n + 7$ edges. If we repeat this step for 8 more steps, we will have $n - 1$ edges left. At this point we must have a tree since the graph is connected. The total running time is $9 \cdot O(n) = O(n)$.

Problem 4 *Problem 28, page 203, of textbook.*

Solution: Let G_X be the graph on the X edges and G_Y be the graph on Y edges. Suppose both G_X and G_Y are individually connected. Then we can choose a connected subtree from G_X with $k+1$ nodes (we can start from an spanning tree of X and start deleting leaves). If we compress these nodes nodes into one supervertex X_0.

Define a graph G'_Y such that: If (u, v) was an edge in G_Y and $u \in X_0, v \notin X_0$ then we add the edge (X_0, v) in G'_Y. If (u, v) was an edge in G_Y and $u, v \notin X_0$ then we add the edge to G'_Y. Observe that G'_Y is connected (since G_Y was connected). We now pick a spanning tree in G'_Y which has only $n-k-1$ edges from Y. This runs in time $O(n+m)$.

Therefore the question reduces to: What do we do when either G_X or G_Y are not individually connected?

Suppose G_X is not connected. We first shrink the connected components of G_X into supervertices — if there is more than $n-k$ supervertices then we cannot choose $n-k-1$ edges from Y and connect the supervertices (and therefore connect the graph), we output "not possible". Suppose the number of supervertices is $z \leq n-k$. We now consider the graph G'_Y where these supervertices are shrunk (defined as above) and find a spanning tree – if no such spanning tree exists then we output "not possible". Otherwise we have chosen $z-1$ edges from Y, say this set is S_1.

Now focus on the graph $G_1(X), G_1(Y)$ where we collapse the vertices connected by S_1 into connected components. Observe that $G_1(X)$ is now connected. We now have reduced our problem to $n' = n - |S_1|$ and $G = G_1$.

Therefore, we only need to assume that G_Y is not connected. We now consider the connected components of G_Y. Again, these cannot be more than $k+1$ in number and if we shrink them, they must be connected by a spanning tree in G'_X. In either case we output "not possible". We can again choose S_2 edges from X such that $|S_2| \leq k$ and adding these edges will connect G_Y.

Now we collapse *both* edge sets S_1, S_2 and have a problem where $n' = n - |S_1| - |S_2|$ and $k' = k - |S_2|$. We can now use the first solution which assumes G_X, G_Y are both connected. Overall running time: $O(n+m)$.