CS 577: Introduction to Algorithms — Homework 4 Solutions

Problem 1

(a). Let $G' = (V, E')$ be a subgraph of G. Suppose G' is obtained by taking a spanning tree T and adding one edge e. Since T is a spanning tree, it is connected and contains no cycle. Then adding the edge e into T will form exactly one cycle and therefore G' is *special*. On the other hand, suppose G' is *special*, then it is connected and contains exactly one cycle. Pick an arbitrary edge e in this cycle and delete it from G, we obtain a new graph G_1. Since G_1 is still connected and contains no cycle, G_1 is a spanning tree of G.

(b). We first find a minimum spanning tree T of G. Let e denote the edge with minimum weight in $E \setminus T$. We conclude that $T \cup \{e\}$ is a minimum-weight special subgraph of G.

Algorithm 1: Pseudocode of MSS(G)

```
  Input: Graph $G = (V, E)$.
1 Find a minimum spanning tree $T$ of $G$;
2 Find the edge $e \in E \setminus T$ with minimum weight;
3 return $G' := T \cup \{e\}$.
```

Proof of Correctness:

Observe that by using the characterization in part (a), our algorithm returns a special subgraph G' of G. Now we prove that this special subgraph is optimal.

Let F be an arbitrary special subgraph of G. We will show that the weight of our solution G' is no more than the weight of F. By the definition of special subgraphs, we know that F is connected and contains exactly one cycle C. Let f be the edge with maximum weight in cycle C. From the proof of part (a), we have $F = T' \cup \{f\}$, where T' is a spanning tree of G.

Since T is an MST, the weight of T is no larger than the weight of T'. Meanwhile, since f has the maximum weight in the cycle C, we know that $f \notin T$ by the "cycle property". Therefore by our choice of edge e in the algorithm, we have $w_e \leq w_f$. Finally, we get $w_{G'} = w_T + w_e \leq w_{T'} + w_f = w_F$, as claimed.

This implies that our algorithm successfully returns a minimum-weight special subgraph of G.

Running time analysis:

To get a minimum spanning tree T, we can use Kruskal algorithm with running time $O(m \log n)$, and then we can find the edge e in time $O(m)$. Therefore the total running time of the algorithm is $O(m \log n)$.

An alternate proof of correctness:

It is possible to prove correctness using a more direct exchange argument. Consider, in particular, a supposedly optimal special subgraph F, and any edge $e' = (u, v)$ in our solution G' that does not belong to F. Suppose that e' is in the MST T. Then, there is a cut separating u and v (namely, the partition created by removing e' from T) such that e' is the cheapest edge crossing this cut in G. Then, F contains an edge f crossing this cut on the path in F from u to v. Then $F \setminus \{f\} \cup \{e\}$ is a special subgraph cheaper than F, contradicting the optimality of F. On the other hand, if $F = T \cup \{f\}$ for some edge $f \neq e$, then we can argue as above that $F \setminus \{f\} \cup \{e\}$ is a special subgraph cheaper than F.

Problem 2

(a). We need to show that the following statements are equivalent for any subset S of homeworks: (i) S is reasonable and (ii) for all integers $t \leq n$, the number of homeworks in S due within t days is at most t.

We first show by contrapositive that (i) implies (ii). In other words, we will show that if for some $t \leq n$, there are more than t homeworks in S due on or before day t, then S cannot be reasonable. Consider any such t. Given that we can complete at most one homework per day, it is impossible to complete more than t homeworks in t days. It means that there exists at least one homework in S we cannot finish on time and thus S is not reasonable.

Now we show that (ii) implies (i) by induction on the number of homeworks in S, say k. Specifically, given (ii) we will give a schedule to complete all homeworks in S before their deadline. For the base case $k = 0$, the argument is obviously correct. Suppose now there are $k + 1$ homeworks in S. Then there must exist a homework $h \in S$ with deadline $t \geq k + 1$, otherwise the number of homeworks in S due within k days is $k + 1$, which contradicts (ii). Now consider the set of homeworks $S \setminus \{h\}$. Note that this set satisfies (ii) for the same reason that S satisfies (ii): for all integers $t \leq n$, the number of homeworks in S due within t days is at most t. Therefore $S \setminus \{h\}$ is reasonable by the induction hypothesis. Thus we can first do all homeworks in S except h within k days and then finish homework h in the $(k + 1)$-th day. This proves that S is reasonable.

(b). We present a greedy algorithm that returns a reasonable set of homeworks S that maximizes the number of points. By part (a), completing the homeworks in S in order of their due date gives us a schedule for homeworks in S that meets all deadlines. The algorithm builds a reasonable set S by including homeworks in decreasing order of points while maintaining the invariant that the set S is reasonable.

Algorithm 2: Pseudocode of greedy

```
Input: number of homeworks $n$; each homework's deadline $t_i$ and points $p_i$.
1 Sort the homeworks by decreasing order of points;
2 Let $L$ represent this sorted list;
3 Initialize set $S = \emptyset$;
4 foreach $h \in L$ do
5     if $S \cup \{h\}$ is reasonable then
6         add $h$ to $S$;
7 return $S$.
```

To test the "If" condition, we maintain an array A that keeps track of number of homeworks in S due on or before day t for all t. That is, $A[t]$ keeps track of number of homeworks in S due on or before day t. When a homework with due date d is added to S, add 1 to $A[t]$ for all $t \geq d$.

Proof of Correctness:

By construction the set S that the algorithm outputs is reasonable. We show that S also attains maximum possible points.

We first prove this for the case in which each homework is worth a distinct number of points. For this case, we show that S is the unique optimal solution via an exchange argument. Let the homeworks in S listed in decreasing order of points be $\{h_1, h_2 \dots h_k\}$. Suppose that there is an optimal solution M that is different from S and that j is the smallest index such that h_j is not in M. That is, assume there is an optimal solution $M = \{h_1, h_2, \dots h_{j-1}, g_j, \dots g_{k'}\}$, where $g_j, \dots, g_{k'}$ are different from h_j. Note that since M is reasonable, so is its subset $\{h_1, \dots, h_{j-1}, g_j\}$. Now we observe that g_j cannot have larger value than h_j, otherwise our algorithm would have considered it before h_j and add it to our solutions. Thus g_j has value less than h_j and in fact all the other g's have fewer points than h_j.

We will now carry out an exchange argument. Specifically, we will exchange h_j with some other homework g in M such that the resulting set $M \cup \{h\} \setminus \{g\}$ is both reasonable and has more points than M. We first consider the set $M' = M \cup \{h_j\}$. This set may not be reasonable, however since M was reasonable, it must hold that for any day t, the number of homeworks in M' due on or before day t is $\leq t+1$. Let t_0 be the smallest t for which the number of homeworks due on or before day t is exactly $t+1$. Note that if we delete a homework from M' due on or before t_0, then the resultant set is reasonable. The key observation here is that there must exist a homework g, of value smaller than h_j, that is due before t_0. This is because, if the only homeworks due on or before t_0 in M' are the h_i's, then the number of homeworks due on or before t_0 must be no more than t_0 (because $\{h_1, \ldots h_j\}$ form a reasonable set). Therefore deleting the homework g from M' results in a reasonable set of total value greater than M. This is a contradiction, and thus S is the unique optimal set.

Now, consider the case in which the points are no longer distinct. In this case, we can use the argument above to prove correctness. If the points are not distinct S may not be the unique optimal solution, but S is still an optimal solution.

The argument in the prior case (points distinct case) shows that we can start with an optimal set M that contains the $j-1$ homeworks $h_1, h_2, \ldots h_{j-1}$, but not h_j, and transform it into a reasonable set of value equal to or greater than M. Repeating this gives us an optimal solution T that contains S. Since T must be reasonable, and adding any homework to S gives a set that is not reasonable, we have $T = S$. This completes the proof.

Running time analysis:

Sorting the list with respect to points takes $O(n \log n)$ time. To check whether the set S is reasonable at any given stage, we keep track of an array of length n in which the i-th entry stores the number of homeworks in S due on or before day i. Updating this array every time a homework is added to S and checking whether S remains reasonable requires $O(n)$ time. Since we need to do this n times, the entire algorithm requires $O(n^2)$ time. Finally sorting S with respect to due date requires $O(n \log n)$ time. Therefore the algorithm requires $O(n^2)$ time.