CSCI3160: Regular Exercise Set 3

Prepared by Yufei Tao

Problem 1. Let S be a set of n intervals $\{[s_i, f_i] \mid 1 \leq i \leq n\}$, satisfying $f_1 \leq f_2 \leq ... \leq f_n$. Denote by S' the set of intervals in S that are disjoint with $[s_1, f_1]$. Prove: if $T' \subseteq S'$ is an optimal solution to the activity selection problem on S', then $T' \cup \{[s_1, f_1]\}$ is an optimal solution to the activity selection problem on S.

Solution. We will prove the claim by contradiction. Suppose that $T' \cup \{[s_1, f_1]\}$ is not an optimal solution to the activity selection problem on S. As proved in the class, there exists an optimal solution T (to the activity selection problem on S) which includes $[s_1, f_1]$. It thus follows that $|T' \cup \{[s_1, f_1]\}| < |T|$ (otherwise, $T' \cup \{[s_1, f_1]\}$ would be an optimal solution to the activity selection problem on S).

Since every interval in $T \setminus \{[s_1, f_1]\}$ is disjoint with $[s_1, f_1]$, all the intervals in $T \setminus \{[s_1, f_1]\}$ must come from S'. As T' is an *optimal* solution the activity selection problem on S', we know:

$$\begin{aligned} |T'| &\geq |T \setminus \{[s_1, f_1]\}| \\ \Rightarrow |T' \cup \{[s_1, f_1]\}| &\geq |T| \end{aligned}$$

thus causing a contradiction.

Problem 2. Describe how to implement the activity selection algorithm discussed in the lecture in $O(n \log n)$ time, where n is the number of input intervals.

Solution. Let S be the set of n intervals given, where each interval has the form $[s, f]$. Sort the intervals in ascending order the f-value. Denote the sorted order as $[s_1, f_1], [s_2, f_2], ..., [s_n, f_n]$ where $f_1 \leq f_2 \leq ... \leq f_n$. Proceed as follows:

```
1. $T = \{[s_1, f_1]\}$; $last = 1$
2. for $i = 2$ to $n$
3.     if $s_i > f_{last}$ then
4.         add $[s_i, f_i]$ into $T$; $last = i$
```

After sorting, the above algorithm runs in $O(n)$ time.

Problem 3. Prof. Goofy proposes the following greedy algorithm to "solve" the activity selection problem. Let S be the input set of intervals. Initialize an empty T, and then repeat the following steps until S is empty:

- (Step 1) Add to T the interval $I = [s, f]$ in S that has the smallest s-value.
- (Step 2) Remove from S all the intervals overlapping with I (including I itself).

Finally, return T as the answer.

Prove: the above algorithm does not guarantee an optimal solution.

Solution. Here is a counterexample: $S = \{[1, 10], [2, 3], [4, 5]\}$. Prof. Goofy's algorithm returns $\{[1, 10]\}$, while the optimal solution is $S = \{[2, 3], [4, 5]\}$.

Problem 4.** Prof. Goofy just won't give up! This time he proposes a more sophisticated greedy algorithm. Again, let S be the input set of intervals. Initialize an empty T, and then repeat the following steps until S is empty:

- (Step 1) Add to T the interval $I \in S$ that overlaps with the *fewest* other intervals in S.
- (Step 2) Remove from S the interval I as well as all the intervals that overlap with I.

Finally, return T as the answer.

Prove: the above algorithm does not guarantee an optimal solution.

Solution. The following nice counterexample is by courtesy of the site *http://mypathtothe4.blogspot.com/2013/03/greedy-algorithms-activity-selection.html.*

$S = \{[1, 10], [2, 22], [3, 23], [20, 30], [25, 45], [40, 50], [47, 62], [48, 63], [60, 70]\}$

Prof. Goofy's algorithm returns 3 intervals (one of them must be $[25, 45]$), while the optimal solution consists of 4 intervals.

Problem 5* (Fractional Knapsack). Let (w_1, v_1), (w_2, v_2), ..., (w_n, v_n) be n pairs of positive real values. Given a real value $W \leq \sum_{i=1}^{n} w_i$, design an algorithm to find $x_1, x_2, ..., x_n$ to maximize the *objective function*

$$\sum_{i=1} \frac{x_i}{w_i} \cdot v_i$$

subject to

- $0 \leq x_i \leq w_i$ for every $i \in [1, n]$;
- $\sum_{i=1}^{n} x_i \leq W$.

Remark: You can imagine that, for each $i \in [1, n]$, the value w_i is the 'weight' of a certain item, and v_i is the item's 'value'. The goal is to maximize the total value of the items we collect, subject to the constraint that all the items must weight no more than W in total. For each item, we are allowed to take only a fraction of it, which reduces its weight and value by proportion.

Solution. Assume, w.l.o.g., that $\frac{v_1}{w_1} \geq \frac{v_2}{w_2} \geq ... \geq \frac{v_n}{w_n}$. Our algorithm runs as follows:

```
1. for $i \leftarrow 1$ to $n$ do
2.     $x_i \leftarrow \min\{W, w_i\}$
3.     $W \leftarrow W - x_i$
```

Next, we prove the algorithm returns an optimal solution. Consider an arbitrary optimal solution $x_1^*, x_2^*, ..., x_n^*$. Observe that $\sum_{i=1}^{n} x_i^*$ must be exactly W (think: why?).

Suppose that the optimal solution differs from the solution returned by our algorithm. Let t be the smallest integer such that $x_t \neq x_t^*$ (this means $x_1 = x_1^*, ..., x_{t-1} = x_{t-1}^*$). By how our algorithm runs, we know $x_t > x_t^*$. Define $\Delta = x_t - x_t^*$.

We argue that $x^*_{t+1} + x^*_{t+2} + ... + x^*_n \geq \Delta$. If this is not true, then

$$\begin{aligned}
\Big(\sum_{i=1}^{t-1} x_i^*\Big) + \Big(\sum_{i=t}^{n} x_i^*\Big) &= \Big(\sum_{i=1}^{t-1} x_i\Big) + (x_t - \Delta) + \Big(\sum_{i=t+1}^{n} x_i^*\Big) \\
&< \Big(\sum_{i=1}^{t-1} x_i\Big) + (x_t - \Delta) + \Delta \\
&= \Big(\sum_{i=1}^{t-1} x_i\Big) + x_t \\
&\leq W
\end{aligned}$$

This means $\sum_{i=1}^{n} x_i^*$ is *strictly* less than W, giving a contradiction.

We now adjust the optimal solution as follows:

- First, increase x^*_t by Δ to make $x^*_t = x_t$.
- Second, reduce a total amount of Δ arbitrarily from x^*_{t+1}, x^*_{t+2}, ..., x^*_n. This is possible because $x^*_{t+1} + x^*_{t+2} + ... + x^*_n \geq \Delta$.

Because $\frac{v_t}{w_t} \geq \frac{v_i}{w_i}$ for any $i > t$, the new solution achieves *at least the same value* for the objective function

$$\sum_{i=1} \frac{x_i^*}{w_i} \cdot v_i.$$

compared to the original solution and therefore must also be optimal.

We now have obtained an optimal solution that agrees with our solution on the first t numbers, i.e., one more than before. By repeating the above argument, we can obtain an optimal solution that agrees with our solution on the first $t+1$ numbers, then another optimal solution agreeing with ours on the first $t+2$ numbers and so on. Eventually, we obtain an optimal solution that is completely the same as our solution. This proves the optimality of our solution.