Junhyeok Jeong

CS 325 – 400 F2019

October 28, 2019

Homework 4

1)

- To minimize the number of days, the optimization strategy is greedy algorithm because I should reach farthest hotel within fixed d miles in a day.

Algorithm steps

1. Set variables: the current hotel index ‘h’, the minimum number of stayed hotels ‘m’, hotel list ‘x’, and maximum miles which can go in a day ‘d’(fixed)

2. Start with h = 0 and m = 0

3. Go to farthest hotel x_i in the hotel list x within d miles

4. Set h = x_i and m += 1

5. repeat from 2 to 4 until h = x_n (n = the last hotel in the list x)

As there are n hotels on the road trip map, we need to walk through each element for searching available hotel within d miles. In this searching process, the worst case is searching 2 times for each element in the list ($\Theta(2n)$). Therefore, the running time is $\Theta(n)$.

2)

To minimize the sum of all penalties in the scheduling, the greedy algorithm can be used for this scheduling problem because it focused on maximizing the best result (= minimum the penalty variable p)

At first, I should think about I can finish all jobs after n minutes. ($1 \leq i \leq n$)

However, I can’t finish all jobs within all deadlines. Therefore, I should finish large penalty job first.

Let make an interval list T with an index i, which is in $1 \leq i \leq n$ (starts at i -1 minutes and finishes at minute i)

Algorithm steps

1. sort the job list in descending order of the penalty value p

2. For adding job j_i into the interval list T, check the interval slot T_s ($1 \leq s \leq d_i$ deadline of j_i)

3. If the interval slot is available, then put j_i

4. If it is not available, then check the interval from T_n to T_1

5. repeat until fil the interval list T

6. and then figure out the total penalty value of unfiled tasks

The running time of this algorithm is $\Theta(n^2)$

At first, the sorting step takes $\Theta(n \log n)$ and the searching empty slot step could be $\Theta(n^2)$ in the worst case when I used the two 1-D arrays (time slots and jobs) for searching. Therefore, $\Theta(n \log n + n^2)$ is $\Theta(n^2)$

3)

If I should choose the last activity to start instead of first activity to finish, I should start from end of schedule. In other words, I should reverse the schedule which change S = $\{j_1, j_2, \ldots, j_n\}$ where j_i = start_i to finish_i to reversed S = $\{j_1, j_2, \ldots, j_n\}$ where j_i = finish_i to start_i

Anyway, reversed S's subset is compatible like normal S's subset. Therefore, choosing the last activity to start is greedy algorithm. It means that if the optimal solutions from normal S can be applied on reversed S as optimal solution.

4)

To solve this problem, I can use reversed the original activity selection algorithm because the last activity to start means reversed version of the first activity to finish. From problem 3, it proved the optimal solution is compatible to reversed one.

To apply reversed version, I should change a for-loop in the algorithm, which ranged from m =2 to n-1, to reversed range from m= n-1 to 1 as looping from last to start.

Furthermore, I need to change the K value from 1 to n because n indicates the last activity.

Reversed greedy activity selector (s, f)'s pseudo code

```
n = s.length
A = {$a_1$}
k = n
for m = n-1 to 1
    if s[k] $\geq$ f[m]
```

```
        A = A and { $a_m$ }
        k = m
return A
```

The reversed version (last to start)'s running time is the same with the original version. Therefore, $\Theta(n \log n)$ is theoretical running time complexity for this algorithm because the input list is needed to be sorting ($\Theta(n \log n)$). If list is already sorted then $\Theta(n)$