University of Washington
Department of Computer Science and Engineering
CSE 417, Winter 2020
Yiliang Wang

February 5, 2020

Homework 4 Problem 1

Problem 1 (10 points):

Let S be a set of intervals, where $S = \{I_1, \ldots, I_n\}$ with $I_j = (s_j, f_j)$ and $s_j < f_j$. A set of points $P = \{p_1, \ldots, p_k\}$ is said to be a *cover* for S if every interval of S includes at least one point of P, or more formally: for every I_i in S, there is a p_j in P with $s_i \leq p_j \leq f_i$.

Describe an algorithm that finds a cover for S that is as small as possible. Argue that your algorithm finds a minimum size cover. You algorithm should be efficient. In this case $O(n \log n)$ is achievable but it is okay if your algorithm is $O(n^2)$. You may assume that the intervals are sorted in order of finishing time.

Answer:

Algorithm:

```
# S[i] := [s_i, f_i]
# sort S base on finishing time
def get_cover_set(S):
    cur_fi = S[0][1]
    P = []
    for I in S:
        if I[0] > cur_fi:
            P.append(I[1])
            cur_fi = I[1]
return P
```

Proof of correctness:
Let p_1 is the right bound of first finish interval I_1 (The first interval in sorted set S). Assume there is a $p'_1 < p_1$, if replace p_1 with p'_1, then what p_1 cover should at least equal or greater than p'_1 (that p_1 satisfies at least more I_i that $s_i \leq p_j \leq f_i$ than p'_1), since p_1 is further right than p'_1, it has at least equal possibity to cover overlap interval with first finish interval.
Assume $S = S - \{I_0$ and all intervals overlap $I_0\}$, the subproblem become exactly same problem as I_0. Let I_i is the first finish interval, and p_i is the right bound of I_i. And $p'_i < p_i$. And because of same exchange argument, what p_i cover should at least equal or greater than p'_i. This holds true for all I which finish first. Therefore, for each of k p_i, p_i should cover at least as much as interal as p'_i, and hence would use smallest k comparing to set P' with each exchange p'_i.
The time complexity is $O(n \log n)$, since the sorting set S cost is $O(n \log n)$, the algorithm only iterate through S once with $O(n)$ time. Hence the overall time complexity is $O(n \log n) + O(n) = O(n \log n)$