CS 577 – HW 4

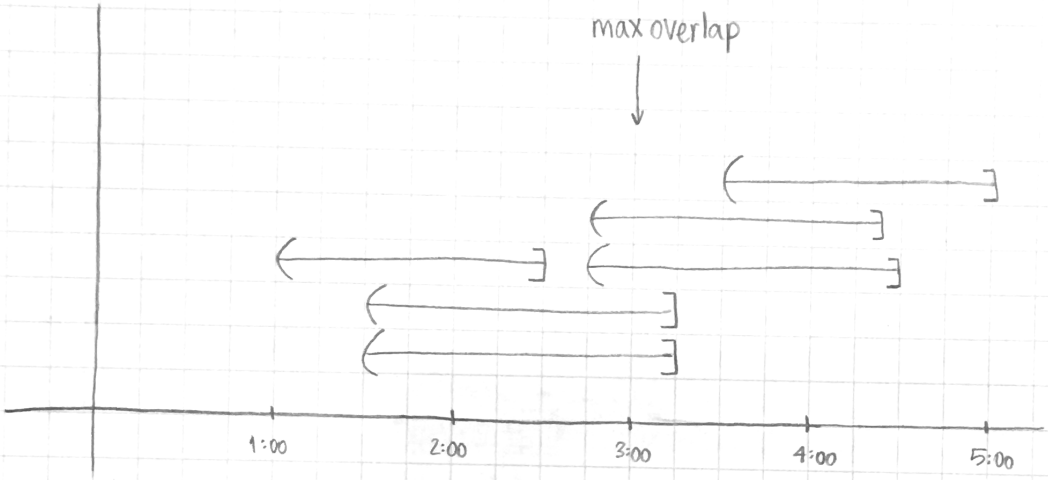
Abigail Rechkin, Alexandra Yavnilovitch

HOMEWORK 4

ABIGAIL RECHKIN
ALEXANDRA YAVNILOVITCH

(a) The greedy approach of picking visit times by the maximum number of overlaps fails to return an optimal solution for the following set of intervals:

$\{ (1.5, 3.25], (1.5, 3.25], (1, 2.5], (2.75, 4.5], (2.75, 4.3], (3.5, 5] \}$



When using the max-overlap approach with this counterexample, the returned number of visits is 3: $\{3.0, 2.0, 4.0\}$
However, the optimal result would be 2 visits:
$\{2.5, 4.3\}$

(b)

Input s[1,...,n] : Array of nonnegative rational numbers
e[1,...,n] : Array of nonnegative rational numbers,
where $I_i = (s[i], e[i]]$
n: number of intervals

Output A nonnegative integer for the minimum number of visits.

```
1        Procedure MinVisits(s[1,...,n], e[1,...,n], n)
2    SORT INTERVALS by earliest finish time.
3    i ← 1
4    numVisits ← 0
5    endTime ← -∞
6    While ( intervalIndex ≤ n)
7            if Overlap(endTime, s[i], e[i]) then
8                i ← i+1
9            else
10               endTime = e[i]
11               numVisits ← numVisits+1
12               i ← i+1
13   return numVisits
```

Helper function: Input cmpValue: nonnegative rational number
startLimit: " " " "
endLimit: " " " "

Output True if cmpValue is within range of start - end

```
1        Procedure: OVERLAP(cmpValue, startLimit, endLimit)
2            if cmpValue > startLimit
3                AND cmpValue ≤ end Limit
4                return true
5            return false
```

TIME COMPLEXITY The Sorting call on line 2 of MinVisits has time complexity of $O(n \log n)$. While loop on line 6 iterates through every interval of the input set, and makes a call to Overlap n time. Since OVERLAP is doing only a constant time $O(1)$ comparison, our while loop total complexity is $O(n)$. Finally, our resulting Total complexity for the greedy algorithm is:

$O(n \log n) + O(n) = O(n \log n)$

GREEDY STAYS AHEAD

We would like to show that our greedy approach finds the minimum number of visits necessary to cover all lecture intervals at the university.
$I_1, I_2, I_3, \ldots I_n$ represent the n given intervals (lectures) in sorted order by EARLIEST FINISH TIME.
Consider only one interval I_1 in the input set. Logically, only one visit time is required to cover the interval. Thus, we know that the greedy algorithm, which returns 1, is at least as good as any arbitrary algorithm S.
If, however, there are two intervals I_1, I_2, logically we conclude that there may either be one or two visits necessary. If $s[I_2] < e[I_1]$ (they overlap), G returns 1. Conversely, if the intervals do not overlap, G return 2. Thus, the greedy approach returns at least as good of a result as S. Continuing with this reasoning, we form our inductive argument.

We will prove the correctness of our greedy approach by Induction on K: The number of visits to the university.
PROOF
We want to show that our greedy algorithm returns at most as many visits as any algorithm S.
Induction Hypothesis
We may gather that the last (end time for) interval included in the kth visit for the greedy approach is greater than or equal to the end time of the last interval in the kth visit of S:

Claim 1. $e(S_K)_{\text{Last}} \leq e(G_K)_{\text{Last}}$

This relationship implies the assumption that on the kth visit for each algorithm respectively, G covers at least as many intervals as S.

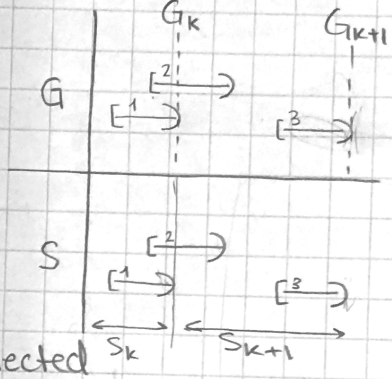
Claim 2. $S_K \leq G_K$

Base case $K=0$

if there are 0 intervals in the set, both S and G algorithms will return 0 visits.
Thus $S_0 = G_0 = 0$ ✓ □

Inductive Step

Consider the $(k+1)$st visit. Now, we show that $S_{k+1} \leq G_{k+1}$.



In order to consider a $(k+1)$st visit for G, we must know that the end time of the first interval in the kth visit PRECEEDS the start time of the first interval in the $(k+1)$st visit:

$$e(G_K^{first}) \leq s(G_{K+1}^{first})$$

By the I.H., we observe that the selected visit time K by the S algorithm MUST preceed the first interval of the G_{K+1} visit.

$$e(G_K^{last}) \leq s(G_{K+1}^{first})$$

Substitute into claim 1: $e(S_k^{last}) \leq e(G_{K+1}^{last})$

For the $(k+1)$st visit, G will select the end time of the first interval (in K+1) as the added visit time. We conclude that G_{K+1} will certainly include the first interval (3 in illustration), and potentially other overlapping intervals, with later end times.
On the other hand, the S algorithm is limited by the relationship from the I.H.: Depending on the kth visit time, S will either cover exactly as many intervals as G, or less. Thus, we know that the visit time for S_{K+1} is at most the visit time of G_{K+1}. Therefore, S_{K+1} covers at most as many intervals as G_{K+1} (see illustration). Finally, we may conclude that

$$S_{K+1} \leq G_{K+1}$$

■