Mo Felonda

577 HW4

and n, the total number of lectures

Input: L, an array of lecture times, in pairs of starting and finishing times.

Output: visits, a count of the minimum necessary visits for all lectures.

```
procedure MINIMUM-VISITS (L, n)
S ← Sort(Start times of lectures, sorted by earliest time)
F ← Sort(Finish times of lectures, sorted by earliest time)
if n == 0 then
    return 0
i = 0
visits = 0
for j = 0...n-1 do
    if(S[j] ≥ F[i]) then
        visits++
        i = j
return visits
```

Proof: Induction Hypothesis: Let G represent our greedy algorithm, and let A represent any valid algorithm sorted in the same manner as G. It is an important distinction that A and G are not the same algorithm. Let g represent how many lectures have yet to be visited using G, and a be the same in respect to A. Our claim is that after any given visit, the number of lectures remaining as done by G is less than that of A. In other words, G has some l_g lectures remaining while A has some l_a lectures remaining, and $g \le a$.

Base case: $n = 0$, or there are no lectures to be visited to begin with. In this case, $g = 0$ and $a = 0$, as there are no lectures to begin with, so $g \le a$.

Inductive Step: Assume that for visits-1: $g_{visits-1} \le a_{visits-1}$

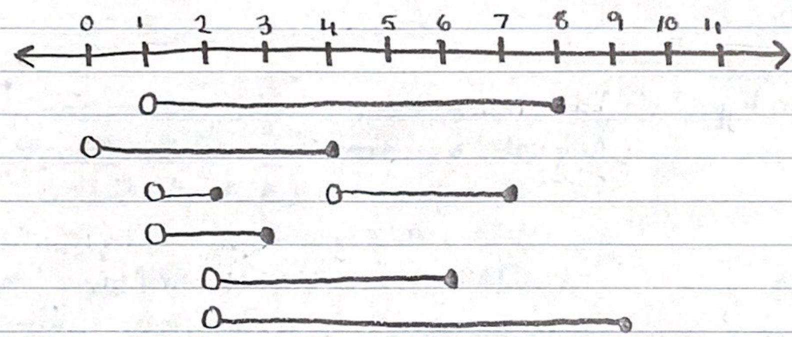
In the case of the given visit, G will continue until there are no lectures remaining. Since $g_{visits-1} \le a_{visits-1}$, A can go from a to G's current lecture (g_{visits}), which will always result in g_{visits} being equal to a_{visits}. However, in the case that A goes before or after that lecture, A will now have more lectures remaining than G because...

Inductive Step cont.: one of the lectures may not be overlapping with another that would be possible in G. This again results in $g_{visits} < a_{visits}$. Thus, the inductive step holds. Accordingly, this means that our greedy algorithm stays ahead of any other valid algorithm.

∴ MINIMUM-VISITS is an optimal algorithm.

Time complexity: As given in the greedy scribe notes, the sort can be done in nlog(n) time. All other steps in this algorithm can be done in linear or constant time, so this results in the overall time complexity being O(nlog(n)).

Counterexample:



1. (1,8]
2. (0,4]
3. (1,2]
4. (4,7]
5. (1,3]
6. (2,6]
7. (2,9]

The greedy algorithm would choose time 3, visiting lectures 1,2,5,6,7. Then it would need 2 more visits for lectures 3 and 4. This is not optimal as choosing times 2 and 6 results in only 2 visits as opposed to 3.