CSCI 3104, Fall 2017 Problem Set 6
Adam Ten Hoeve (10/16) adamtenhoeve@colorado.edu

Collaborated with Erin Ruby and Andrea DeVore

Problem 1 (35 points)

An exhibition is going on in a town today from $t = 0$ (9 am) to $t = 720$ (6 pm), and celebrities will attend!! There are n celebrities $C_1, ..., C_n$ and each celebrity C_j will attend during a time interval $I_j : [\ell_j, u_j]$ where in $0 \leq \ell_j < u_j \leq 720$. Note: the ends of the interval are inclusive. An advertising company for high-end luxury goods wants to broadcast its ad on a large screen television in the exhibition, multiple times during the day. In particular, each celebrity must see the ad but the company also wants to minimize the number of times the ad must be shown.

For example:

Celebrity	$[\ell, u]$
1	*[2,53]*
2	*[5,50]*
3	*[5,98]*
4	*[102,150]*
5	*[120,186]*
6	*[85,190]*

Then, if the ad is shown at times $t_1 = 50$ and $t_2 = 150$, then all 6 of the celebrities will see the ad.

(a) *Greedy algorithm $\mathcal{A}$ selects a time instance when the maximum number of celebrities are present simultaneously. An ad is scheduled at this time and the celebrities covered by this ad are then removed from further consideration. The algorithm $\mathcal{A}$ is then applied recursively to the remaining celebrities.*
Give an example where this algorithm shows more ads than is necessary.

An example where the greedy algorithm would fail to be optimal:

Celebrity	$[\ell, u]$
1	[0, 100]
2	[200, 300]
3	[400, 500]
4	[600, 700]
5	[50, 380]
6	[250, 380]
7	[340, 450]
8	[340, 650]

The first 4 celebrities ($C_1 ... C_4$) have no time in common with each other and none fall between [340, 380]. The last 4 ($C_5 ... C_8$) all share the [340,380] time slot, which the algorithm sees as the time where the number of celebrities present is at a maximum. The greedy algorithm will thus show a single add for the 4 celebrities between [340, 380] and remove them. However, the algorithm would then have to show an add for the first 4 celebrities individually. The total ads shown would be 5. However C_1 shares a time slot with C_5, C_2 with C_6, C_3 with C_7, and C_4 with C_8. By showing the adds during a time when the pairs share a time slot, the company would only show 4 adds and still get all of the celebrities. Therefor, the greedy algorithm showed more adds than necessary.

(b) *Let C_j represent the celebrity who* **leaves** *first and let $[\ell_j, u_j]$ be the time interval for C_j. Suppose we have some solution $t_1, t_2, ..., t_k$ for the ad times that cover all celebrities. Let t_1 be the earliest ad time.*
Prove the following facts for the earliest scheduled at t_1. For each part, your proof must clearly spell out the argument. Overly long explanations or proofs by examples will receive no credit. You may as the course staff for help with the wordings of your proofs.

S1. *Prove $t_1 \leq u_j$. (Three sentences. Hint: proof by contradiction.)*

Assume $t_1 > u_j$ (contradiction):
This means the first add plays at time t_1 which is after the first celebrity has already left at time u_j.
This can not be true because all celebrities must encounter at least one ad for the algorithm to be correct.
Therefor, $t_1 \leq u_j$.

S2. *If $t_1 < \ell_j$, then t_1 can be deleted, and the remaining ads still form a valid solution. (Five sentences. Hint: suppose deleting t_1 leaves a celebrity uncovered, when should that celebrity have arrived and left? Prove a contradiction.)*

Assume $t_1 < \ell_j$ then t_1 can be deleted but the remaining ads do not form a valid solution because not all celebrities are covered by $t_2 ... t_k$ (contradiction) .
Then there are two cases:

1. Celebrity C_j is the first to arrive so because t_1 is a time before ℓ_j, t_1 does not cover C_j or any celebrity.
2. Celebrity C_j is the first to leave but there may be a celebrity C_i that arrives before C_j and leaves afterwards. t_1 is a time before ℓ_j so it may cover C_i but not C_j, meaning there is another time t_n that covers C_j that would then also cover C_i.

In both cases, $t_2 ... t_n$ is still a solution when t_1 is removed, so it is proved through contradiction.

S3. *If $t_1 < u_j$, then t_1 can be modified to be equal to u_j, while still remaining a valid solution. (Three sentences. Hint: suppose setting $t_1 := u_j$ leaves a celebrity uncovered, then when should that celebrity have arrived and left? Prove a contradiction.)*

Assume $t_1 < u_j$, t_1 can be modified to equal u_j does not make a valid solution.
This would mean t_1 does not cover the first celebrity to leave, meaning a celebrity C_i left before C_j.
This can not be true because C_j has to be the first celebrity to leave, thus proven by contradiction.

(c) *Use the insight from (1b) to design a greedy algorithm that is optimal*

(i) *Write pseudocode for your algorithm.*

```
findAds3(celebs[]) // Array of celebs arrive and leave times.
  sort(celebs)    //Sorted by leaving times
  t = celebs(0).leaveTime    // t = leave time of first celeb
  times[]
  times.append(t)
  for i in celeb    //Loops through each celeb
    if t < i[arriveTime] // If the celeb arrives before t
      t = i[leaveTime] // Updates t to leavetime
      times.append(t)
  return times    // where len(times) is optimal number of ads
```

(ii) *Prove that your algorithm is correct and give its running time complexity.*

Loop Invariant: The time array hold times that cover all celebrities up to i-1.
Initialization: The base cases are:

1. If the celeb array is empty. In this case, the times array will also be an empty array when initialized because there are no times to be shown.
2. If the celeb array has one celebrity in it. The algorithm returns the leavetime of the one celebrity.

Maintenance: The loop iterates through the array of celebrities. We assume that not all celebrities are covered yet. If the arrival time of the i^{th} celebrity is greater than the showing time of the most recent ad, then there must be another time that an ad must be shown. The next time is set to the leave time of the i^{th} celebrity.

Termination: The loop terminates after all the celebrities have been shown ads. We know all the celebrities have been covered up to i-1. Therefor all the celebrities are shown ads.

The sort function is $\Theta(n \log n)$. The rest of the algorithm runs at $\Theta(n)$ because it only has to iterate through a single loop. The steps inside the loop run at constant time. The overall run time of the algorithm is $T(n) = \Theta(n \log n) + \Theta(n)$ which can be reduced to $T(n) = \Theta(n \log n)$.

(iii) *Demonstrate the solution your algorithm yields when applied to the $n = 6$ example above.*

The given celebrities will be on the time intervals [2,53], [5,50], [5,98], [102,150], [120,186] and [85,190]. The algorithm will then sort these time intervals based on leave time in ascending order. The celebrities will be sorted as [5,50], [2,53], [5,98], [102,150], [120,186] and [85, 190]. We will then assign t to the leave time of the first element in the sorted array, so $t = 50$. We will then iterate through the sorted array to find all of the celebrities that have arrived by the time that celebrity has left. When $t = 50$, the first 3 celebrities will be covered. The fourth celebrity arrives after $t = 50$ so we change t to equal the leave time of the fourth celebrity, so $t = 150$. We will then iterate over the rest of the celebrities to find that the rest arrive before $t = 150$. All of the celebrities have been shown an ad so the algorithm terminates. The final solution is that two ads were shown at $t_1 = 50$ and $t_2 = 150$.