Homework 4

CS 577
Jason Jiang

February 20, 2019

1. The year is 1922. You are organizing a big dance in your university and you want to invite everyone. Unfortunately, emails and cellphones were not available that time so you have to visit all classes during the lectures and invite the students of each class to the event. The lecture time of each class is a half-closed interval (si; fi] of a start and nish time. To minimize your trips to the university you try to visit it during times where many classes overlap and invite the students of all these classes. For simplicity we assume that the start 1 and nish times of the lectures are non-negative rational numbers. One possible schedule of lectures could be

$$(1, 3.1]; (2.2, 4]; (1.7, 5]; (4, 5]$$

In this case the smallest number of visits to the university is 2, for example one visit at 2:5 and one at 4:5. Visiting at 4 does not cover the (4, 5] class, however does cover the (2.2, 4] class.
a)You decide that you should pick your first visit to be a time that overlaps with the maximum number of lectures, that is a number t that is contained in the maximum number of intervals (si; fi]. Then you remove these intervals and pick your second visit with the same rule. You continue until no more lectures are uncovered. Construct a counter-example where this greedy strategy fails to compute an optimal solution.
b)Design a greedy algorithm that computes the smallest number of visits to the university that cover all lectures and runs in time O(n log n).

- In order to find a counter-example, we have to find a scenairo that picking a time with not the maximum number of overlaps that will eventually lead to a better result. Say, here is an example:

$$(1, 3], (2, 5], (2, 5], (4, 7], (4, 7], (6, 8]$$

  Given those 6 time intervals, by the naive greedy alogrithm, we should pick (4,7], (4,7], (2,5], (2,5] at the first time, say we pick 4 lectures at the same time. Then we pick (1,3] for the second time, and finally, (6,8] for the last time. Total 3 visits.
  However, if we go (1,3], (2,5] and (2,5] at the first time, with 3 lectures at the same time, and the next one we just go (4,7], (4,7] and (6,8] for the second time then we are done. Total is just 2 visits.

This algorithm has a complexity of $O(nlogn)$. Two priority queue will take $2nlogn$time, and for the while loops, although there are two embeded while loops, each node is at most visited twice so the total time is $2n$. Then the overall time complexity is $O(nlogn)$

-
  ```
  struct invterval contains
   int B;
   int E;
   bool isVisited;
  end

  Input:   $B[B_1, ..., B_n$ $E[E_1, ..., E_n$ Two arrays of Begin and corresponding Ending time for each lectures with n > 0
  Output:   Number of minimum visits
  procedure COUNT_MIN_VISIT()
      interval_list ← ∅
      Initialize Priority Queue pq_end according to E by ascending order
      Initialize Priority Queue pq_begin according to B by ascending order
      count ← 0
      for $i = 0$ to $n$ do
          new_interval.B ← $B_i$
          new_interval.E ← $E_i$
          new_interval.isVisited ← false
          pq_end.enque(new_interval)
          pq_begin.enque(new_interval)
      while pq_end.size > 0 do
          interval_1 ← pq_end.dequeue()
          if interval_1.isVisited = true then
              continue
          while pg_end.size > 0 do
              interval_2← pq_begin.peek()
              if interval_2.B < interval_1.E then
                  pq_begin.dequeue()
                  Mark interval_2.isVisited and corresponding node in pq_end as true
          count ← count + 1
      return count
  ```

- **Proof:**

  Now we assume we have 2 algorithms, one greedy algorithm G as I mentioned before, and another valid algorithm V of calculating same visit times.
  Say we denote k as the k^{th} visit, G_k is the total visit time after the k^{th} visit by using greedy algorihm and V_k for the total visit time for V. Also we denote L_k as the number of lectures marked as visited during one visit.
  Claim: $\forall k \geq 0, \sum_{i \in G_k} L_i \geq \sum_{i \in V_k} L_i$ Say for any time, the Greedy algorithm will always better or equal the algorithm V.
  Base Case:
  Before the first visit, the number of lecture covered is both 0 for G and V.
  Induction:
  Assume for the first to k^{th} visit G and V covers the same number of visits. What we have to prove is that the G algorithm will cover no less lectures than the V algorithm in V.

  Then for the $k+1^{th}$ visit, the next lecture we want to cover is $(B_{k+1}, E_{k+1}]$. In the algorithm G, we should visit at the time E_{k+1}, yet in the algorithm V, the visit is made during (B_{k+1}, E_{k+1}). The V cannot make visit on E_{k+1} since if so, G will equal to V. Then G will always pick equal number of lectures or even more lectures (if there are ovlaps during E_{k+1}) time.

  Thus, given the first k is the same, the whole G algorithm will return less or equal number of visits than the V algorithm. Hence the G algorihm is optimal.