CS157 Homework 5

BY CHAOQIAN AND SILAO_XU

March 3, 2013

Problem 2

(10 points) You and your group of friends spend all your time hanging out in your dorm room unless there is a computer science event happening, in which you attend the event, and then immediately return to your dorm room. However, there may be overlapping events, in which case you aim to have at least one of you attend each event. Of course, you must arrive to computer science events on time, and it is rude to leave early.

Design a greedy algorithm so that you and your friends, between you, can attend every computer science event, if this is possible at all. As above, points on this problem will only be given for the proof that your algorithm is optimal; more points will be given for simpler and clearer proofs. Remember, if you are having trouble proving the correctness of your algorithm, consider a different algorithm; if your proof seems unwieldy and awkward, consider a different proof approach—your first idea might be correct, but might not be the best.

- ***Algorithm***

  Let the number of people be n and assume that there are at most n events overlapped at the same time.

```
n ← number of people
m ← number of events
label each people from 1 to n
label each event from 1 to m sorted by beginning time in ascending order
def Arrange-Events():
   construct a m array "Attends" in which each element i stores an empty
       set storing the people who are attending
   p ← NULL
   i ← 1
   S ← set(all people)
   E ← set(all events)
   while i ⩽ m:
       P ← set(events in E that will finish later than event_i's start time)
       V ← set(people who had attended events those are in P)
       F ← S − V, q ← |V|
       l ← number of events overlapped with event i
       e ← set(events that overlapped with event i)
       Attends[i] ← set(n − q − l number of people chosen from S)
       F ← F − Attends[i]
       for each event_j in e do:
           Attends[j] ← set(1 people chosen from F)
           F ← F − Attends[j]
       end
       i ← i + 1 + l, q ← 0
       E ← E − e − event_i
   end
   return Attends
end
```

- ***Correctness***

  - **Claim**. *Arrange-Event* terminates.

Proof: Since we have ordered the events by their staring time, the maximum index for those events would be m, m is the number of events. In line 23 the *Arrange-Event*'s index i (defined in line 9) would keep increasing and line 12 would become true at some point.

- **Claim**. *Arrange-Event* satisfies the feasibility that every event has at least one people to attend.

  Contradiction Hypothesis:

  Assume that there is an event, $event_k$, that has no people attend, which means there is no available people at the moment when it starts.

  So, in line 18, we get empty set for $event_k$, that is

$$\begin{aligned} n-q-l &\leqslant 0 \\ n &\leqslant l+q \end{aligned}$$

  which means $event_k$'s overlapping events number is $\geqslant$ the number of people n. Plusing $event_k$ itself, this sequence of overlapping events is $> n$. It contradicts with the prerequisite that the number of events occurs at the same time cannot exceed n.

  So we can conclude that *Arrange-Event* satisfies the feasibility that every event has at least one people to attend.

- **Claim**. *Arrange-Event* satisfies the optimality criteria that we achieve maximum total attendant number for all events.

  Proof: Let S^* be the optimal solution got from *Arrange-Event*. Assume that there is an optimal solution S which can arrange more attandants than solution S^* overall. There must be an event, $event_k$, such that

$$Attends^*[i] = Attends[i] \quad (\text{for } i < k)$$

  and

$$Attends^*[i] < Attends[i] \quad (\text{for } i = k)$$

  Case 1: there are some consecutive events that overlap $event_k$. Since for $i < k$, we had assigned the same number of people for $event_i$, in line 18, for $event_k$ we assign more people in solution S than in solution S^*, then for the following events that overlap with $event_k$, there will be one event cannot be assigned with people to attend. It contradicts with the feasibility that every event has at least one person to attend.

  Case 2: there is no consecutive overlapping event for $event_k$. Since for $i < k$, we had assigned the same number of people for $event_i$. In line 18, suppose we can assign more people than S^*, that is at this moment $n_{\text{free}} = n - q + c$, where c is a constant positive integer. Let $n_{\text{busy}} = q$. It's obvious since these people are still in events that would end after $event_k$'s ending time. So, for S, $n_{\text{free}} + n_{\text{busy}} = n + c$, it contradicts with the fact that we have n people for assigning.

  To sum this up, *Arrange-Event* satisfies the opitimality criteria that we would achieve the maximum total attendant number overall.