CS157 Homework 5

BY CHAOQIAN AND SILAO_XU

March 4, 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***

  The pseudocode for the algorithm is as follows:

```
00 Arrange-Event(M /* the number of events */)
01     label each event from 1 to M ordered by beginning time in ascending order
02     S ← set(all students)
03     i ← 1 /* used to iterate all events labeled from 1 to M */
04     while i ⩽ M do
05         students who have finished attending any previous events come back to S
06         if S is empty then do nothing
07         else:
08             s ← 1 student chosen from S
09             let student s attend event i
10             i ← i + 1
11         end
12     end
13     if there exists an event that has no people to attend then do:
14         return FALSE
15     else
16         return TRUE
17     end
18 end
```

- ***Correctness***

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

  Proof: The maximum index for those events would be M, M is the number of events. Since the events number is finite and in line 07 the *Arrange-Event*'s index i (defined in line 3) would keep increasing and line 04 would become true at some point. So the algorithm can terminate.

  – **Claim**. *Arrange-Event* satisfies the feasibility that our algorithm's arrangement can make people arrive event, if possible, on time and won't leave early.

Proof: In line 04, for any incoming event, we would either assign a person from the set S or abort letting people taking sucessive events. So our algorithm would not let people attend event in the middle. In line 05, we would let people come back to the set S if and only if the event finishes. So our algorithm would not let people leave an event in the middle.

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

– **Claim**. *Arrange-Event* satisfies the optimality that our algorithm's arrangement, if there exists such an arrangement, we can let every event has at least one people 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 $event_k$ starts.

So, in line 06, we get empty set for $event_k$, that is

$$n - n_{\text{busy}} \leqslant 0$$

where n_{busy} denotes the number of people who are busy attending other events currently, which means there doesn't exist an arrangement that at least one people can attend each event at the first place.

It contradicts with the assumption that there exists such an arrangement.

Proof completed.