Quiz 9: Supplementary question

Praveen Venkatesh
EE10B028

April 7, 2013

Let $\underline{\mathtt{p}}$ be the vector of payoffs $\{p_i\}_{i=1}^{N}$. $\{t_i\}$ are the respective deadlines. The index i denotes the task number. The optimal algorithm is described below.

```
todo = zero vector // Final schedule: Do task todo[i] in time step i.
while p is not empty
    i = index{extractmax_i(p)}
    if todo[t_i] is empty:
       todo[t_i] = i
    else:

       insert i in the first empty slot to the left of t_i in todo
       if no empty slots:
          // Cannot schedule. Push to last.
          insert i in the first empty slot from the end of todo
```

Since tasks of any given payoff take the same amount of time to complete, we would always prefer to do a task with higher payoff. In other words, out of the tasks that are yet to be scheduled, we would always want to try to perform the task with the highest payoff.

The algorithm, therefore, greedily picks the task with the largest payoff and places it as close to that task's deadline as possible. Thus, at any stage, the tasks that have already been scheduled (tasks in `todo`) are of higher payoff than tasks that are yet to be scheduled (tasks in $\underline{\mathtt{p}}$). Each of these tasks has been scheduled by attempting to place it to the left of its deadline. The only way that it could not have found a place before its deadline is if all of these slots were already filled. But that implies that there were tasks of higher payoff than the one which we were trying to schedule. The algorithm has therefore correctly scheduled tasks with higher payoff, putting off ones with a lower payoff to the end.

This is evidence of optimal sub-structure: tasks that are scheduled are always of the highest payoff. Clearly, the locally greedy approach maintains the optimal sub-structure by picking the highest available paying task for scheduling next.