Justin Phillips
cs325 spring HW4

Problem 1:(5points)
Class Scheduling:
Suppose you have a set of classes to schedule among a large number of lecture halls, where any class can class place in any lecture hall. Each class c_j has a start time s_j and finish time f_j. We wish to schedule all classes using as few lecture halls as possible.
Verbally describe an efficient greedy algorithm to determine which class should use which lecture hall at any given time.
What is the running time of your algorithm?

When scheduling we start with the class with the earliest start time (ordered by start) then move on to the next class, we then compare the starting time of the next class with the finish of the class already set if there is an overlap, we move the class to the next hall. After we set the class to the proper hall we move on to the next class and compare the next classes starting time to the finishes of the set classes until either we find a spot after a class already set if they're no spots available, we add a new hall. We continue to do this till all classes have a hall with no over lapping time.

The running time of the algorithm is Θ (n log n)

Problem 2:(5points)
Scheduling jobs with penalties: For each $1 \leq i <= n$ job j_i is given by two numbers d_i and p_i, where d_i the deadline and p_i is the penalty The length of each job is equal to 1 minute and once the job starts it cannot be stopped until completed. We want to schedule all jobs, but only one job can run at any given time. If job i does not complete on or before its deadline, we will pay its penalty p_i . Design a greedy algorithm to find a schedule such that all jobs are completed and the sum of all penalties is minimized. What is the running time of your algorithm?

It is stated in the problem that we must complete all jobs and since it is never determined how many machines, we are using I'm going to assume one. Since we can't start another job till one is finished and all jobs must be completed on our single machine. All that needs to be done is any job that is over our deadline of 1 min minute is assessed a penalty. There is really no minimum way to determine the penalty if the job is over a minute, we just add the penalty to our penalty sum.

input: J number of job j_i is given by two numbers d_i and p_i, where d_i the deadline and p_i is the penalty
output: if a job isn't completed by d_i a penalty is imposed p_i
while J !=0
    for int i =1; i <= J ; i++
        m[i]=j_i //set job to machine array
        if(d_i >1)
        totalPenalty += p_i
Based on the way the question is phrased we would just loop one time through the set and our running time would be 0(n)

Problem 3(5points) CLRS 16-1-2
Activity Selection Last-to-Start
Suppose that instead of always selecting the first activity to finish, we instead select the last activity to start that is compatible will all previously selected activities. Describe how this approach is a greedy algorithm, and prove that it yields an optimal solution

A greedy algorithm is defined as one that always makes the choice that looks best at the moment. Instead of the earliest finish time it chooses the task with last activity to start and work backwards. we are given a set S ={a_1,a_2…,a_n} of activities where a_i={s_i,f_i}. We need to find the optimal solution by selecting the activity which start time starts last. now lets create another set S' = {a_1',a_2'…,a_n'} of activities where a_i'={s_i',f_i'}. since a_i' is the reverse of the non-prime then the optimal solution for S' = the optimal solution for S.

4. verbal description: sort the activity in descending order based on start time for each activity in the order put it into the schedule if the finish times don't overlap the start time. if there are 2 with the same finish time then choose the one with the greater start time. every event that overlaps start times can be eliminated and we continue to do this till we reach the lowest start time.

pseudo: function Activity (start finish number schedule m)

sort the jobs in decreasing order based on start time

initiate schedule with last start time

track activities in the schedule

for loop to find latest compatible activity and add to the schedule

schedule and the m will be computed in this function

analysis of running time: O (n log n)