CS 4820, Spring 2018 **Homework 2, Problem 1**
Name: Rongguang Wang
NetID: rw564
Collaborators: Siyuao Liu (sl2928); Yihao Chen (yc2288)

(1.a) *(5 points)*
Consider the sceduling problem in Section 4.2 of the textbook. Suppose the goal is to minimize the sum of the latenesses of requests. Show that for this objective function, the earliest-deadline-first algorithm does not always find an optimal schedule.

Solution:

There are three jobs (s_1, t_1, d_1), (s_2, t_2, d_2)and (s_3, t_3, d_3). s represents start time, t represents duration and d represents deadline. Assume that l donates the lateness of interval and f donates the finishing time. Let $d_1 > d_2 > d_3$ as below:

Job 1: |————————————————| d_1
Job 2: |————————| d_2
Job 3: |——| d_3

When using Earliest Deadline First (EDF) Algorithm to schedule, job 3 will be scheduled before job 2 and job 2 will be scheduled before job 1.
3 2 1
|——||————||————————————|
$l_1 = f_1 - d_1$
$l_2 = f_2 - d_2$
$l_3 = s_3 + t_3 - d_3$
So, the total sum of lateness is $(l_1 + l_2 + l_3) = s_3 + t_3 - d_3 + f_1 + f_2 - d_1 - d_2$.
A counter example is that schedule job 2 before job 1 and job 1 before job 3.
2 1 3
|————||————————————||——|
$l'_1 = f'_1 - d_1$
$l'_2 = s'_2 + t_2 - d_2$
$l'_3 = f'_3 - d_3$
So, the total sum of lateness is $(l'_1 + l'_2 + l'_3) = s'_2 + t_2 - d_2 + f'_1 + f'_3 - d_1 - d_3$.
Since $s_3 = s'_2 = 0$ and $(t_3 + f_2 + f_1) > (t_2 + f'_3 + f'_1)$ for the two sum of lateness $s_3 + t_3 + f_2 + f_1 - (d_1 + d_2 + d_3)$ and $s'_2 + t_2 + f'_3 + f'_1 - (d_1 + d_2 + d_3)$, $s_3 + t_3 + f_2 + f_1 - (d_1 + d_2 + d_3) > s'_2 + t_2 + f'_3 + f'_1 - (d_1 + d_2 + d_3)$. Therefore, for the objective of minimizing the sum of the latenesses of requests, the earliest-deadline-first algorithm does not always find an optimal schedule.

(1.b) *(10 points)*
Again consider the scheduling problem in Section 4.2 of the textbook. Suppose every request has a positive weight w_i and the goal is to minimize the weighted sum of latenesses, $\sum_i w_i \ell_i$.

Give an efficient algorithm for the special case that all deadlines are equal to the time the resource becomes available (i.e., $d_i = s$ for all $i \in \{1, \ldots, n\}$).

Solution:

The goal is to minimize the weighted sum of latenesses. Given that all deadlines are equal to the time the resource (start time s) becomes available. w represents weight of the schedule, l represents the lateness, t represents the duration of the job, f represents the finishing time and d represents the deadline. So, $(f-d)$ is equals to l. The weighted sum of lateness is the summation of wl, which is the summation of $w(f-d)$.

Below is an observation of scheduling 2 jobs,

w_1t_1 w_2t_2

|——————||————| Schedule 1

w_2t_2 w_1t_1

|————||——————| Schedule 2

Weighted sum of lateness of schedule 1 is $w_1t_1 + w_2(t_1 + t_2)$.
Weighted sum of lateness of schedule 2 is $w_2t_2 + w_1(t_1 + t_2)$.

Difference between the weighted sum of lateness of the two schedules:
$w_1t_1 + w_2(t_1 + t_2) - [w_2t_2 + w_1(t_1 + t_2)] = w_2t_1 - w_1t_2 = 0$(if the weighted sum of lateness of schedule 1 equals to schedule 2)
$w_2t_1 = w_1t_2$
$w_2/t_2 = w_1/t_1$
$(w_2/t_2) - (w_1/t_1) = 0$
Therefore, there are 3 conditions:
1) $(w_2/t_2) - (w_1/t_1) = 0$ (Weighted sum of lateness of schedule 1 equals to schedule 2)[No difference between 2 schedules]
2) $(w_2/t_2) - (w_1/t_1) > 0$ (Weighted sum of lateness of schedule 1 greater than schedule 2)[schedule 2 is better than schedule 1]
3) $(w_2/t_2) - (w_1/t_1) < 0$ (Weighted sum of lateness of schedule 1 less than schedule 2)[schedule 1 is better than schedule 2]

The least finishing time should be attached to the maximum weight jobs. At the same time, the finishing time of the jobs should be reduced. So, the jobs with minimum time interval t should be run first. Therefore, a conflicting phenomena appears that the jobs should be scheduled in decreasing order of weights, from the weight point of view, but they should be scheduled in increasing order of weights in interval t point of view. According to the observation above, the ratio of weight over time interval (w/t) can be treated as a parameter of the algorithm.

The job with maximum weight w and minimum time interval t should be scheduled first. So, the job should be scheduled in decreasing order of the ratio of w/t. Therefore, the algorithm can start from sorting the jobs in decreasing order of w/t, and then, schedule the jobs in the order of previous steps result.

Algorithm's Correctness:
Let a pair of jobs m, n an inversion if m is scheduled before n and the ratio of w/t of m is smaller than that of n. Assume that the schedule discussed here has a consecutive inversion for any

inversion m, n. At the same time, assume that there is no gap between the optimal scheduled consecutive jobs although it minimized the objective function, which because a better solution can be achieved by removing the gap.

Consider 2 schedules $S1$ and $S2$ which have no gap between the jobs and the inversion between the jobs are consecutive. Schedule $S1$ is differ from schedule $S2$ which is caused by inversions of jobs. Let the consecutive inversion pair in both of the schedules i and j.

Weighted sum of lateness of schedule $S1$ is

$$\sum_{k=1}^{i-1} w_k * f_k + w_i * (f_{i-1} + t_i) + w_j * (f_{i-1} + t_i + t_j) + \sum_{k=j}^{n} w_k * f_k$$

Weighted sum of lateness of schedule $S2$ (an inversion happened between i and j) is

$$\sum_{k=1}^{i-1} w_k * f_k + w_j * (f_{j-1} + t_j) + w_i * (f_{i-1} + t_i + t_j) + \sum_{k=j}^{n} w_k * f_k$$

Compute the difference between the weighted sum of lateness of the two schedules and the result is

$$w_i * t_j - w_j * t_i$$

Since i and j is an inversion and

$$w_i/t_i < w_j/t_j$$

which means

$$w_i * t_j < w_j * t_i$$

So, the weighted sum of lateness of schedule $S1$ is smaller than that of the schedule $S2$. The schedule $S1$ is better than schedule $S2$ after inversion of i and j. To sum up, the greedy solution $S1$ is the optimal.

Running Time Analysis:
The sorting algorithm is used which takes $O(n \log n)$ time.