Comp 496 ALG Project #2 (30 pts) Spring 2017

Due: Thursday April 6

**GENERAL DIRECTIONS:** You can form a **team of two** to do this project or you can do it as an individual. You need to inform the instructor during class on March 9 who your team members will be. In this project you will create three Java classes to solve the job scheduling problem described below. Your source code must be in one file, called JobScheduler.java . Your Java classes and class methods must match the specifications in the classes in the template below . In particular, all specified methods must have the same signatures as specified. Otherwise my test program will not work. You may add any extra private methods needed. Neatness counts and so does indented code that is easy to read with helpful variable names and no wrap around lines in your printed code.

**Description of the job scheduling problem:** You are given n jobs, each with a job number, a length (i.e., time needed to complete) , a deadline, and a profit. Your problem is to schedule **all** the jobs on a single machine so as to maximize the total profit. The machine can handle only one job at a time. A job can be scheduled to start at any time from time = 0 on. Once started, it stays on the machine until completed. If it completes on or before its deadline, the job earns its profit; otherwise the job has profit 0. The problem is to find a schedule that maximizes the sum of the job profits earned. All inputs are positive integers.

Example **Input:** n = 4 jobs. The input to your program will be 3 arrays, corresponding to length, deadline and profit for each job. Jobs are numbered from 0 to n-1 by index.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Job# | 0 | 1 | 2 | 3 |
| length | 7 | 4 | 2 | 5 |
| deadline | 7 | 16 | 8 | 10 |
| profit | 10 | 9 | 14 | 13 |

**Typical Schedule Output:** Using Job printout notation:

job#: ( length, deadline, profit, start, finish) . Your program fills in the start and end time for each job.

Schedule Profit = 36

#3:(5,10,13,0,5)

#2:(2,8,14,5,7)

#1:(4,16,9,7,11)

#0:(7,7,10,11,18) <- no profit. Why?

1. **Construct the Job and the Schedule classes**, using the template below. You must use the exact signatures for all methods and exact names for all fields. You need to write all of the code that is missing. Do not change the code that is there. These are the basic data structures you will be using in the program. They are required. Test these classes.
2. **Construct the JobScheduler class** , using the template below. You must use the exact signatures for all methods and exact names for all fields. You need to write all of the code that is missing. Do not change the code that is there.
   1. EDF method: sort items by deadline from small to large. Schedule next job in deadline order, unless the job will finish after its deadline. In this latter case put the job at the end of the schedule since it won’t matter how late it is done. Its profit contribution will be 0.
   2. Similar method on SJF and HPF greedy solutions.
3. Test all of the methods in the JobScheduler class on sample input on page 1. Find the optimal schedule using brute force method: the earliest deadline first (EDF) method; the shortest job first(SJF) method; the highest profit first (HPF) method. Note that the EDF, SJF and HPF are only approximate solutions. They may sometimes produce an optimal schedule and sometimes not.
4. Continue to test your program on LOTS of random example inputs. Vary n between 2 and 11 and randomly vary the inputs. How do the profits given by the EDF, SJF and HPF compare with the profit given by the optimal solution?
5. Create a new approximate solution to the Job Scheduling problem. this approximate soluiton should run in O(n3) or better. Its signature should be  *Schedule newApproxSolution()* . How good is your new solution as compared with the known optimal solution for n <=11. Part of the grade on your newApproxSolution will be how well it performs for large n.

**Turn in:** ( Submit one report per team)

1. Report with
   * Cover Page with Project #, course, date and names of team members;
   * JobScheduler.java source code with a main program that tests your program using the Instructor’s Test Cases. These test cases will be posted by March 28.
   * Your program’s runtime results of the Instructor’s Test Cases from March 28.(labeled neatly)
   * A 2-3 page typewritten discussion the effectiveness and efficiency (Runtime analysis and space analysis of the brute force, EDF method, SJF method and the HPF method; and a discussion of the efficiency and effectiveness of your newApproxSolution(). Back up the discussion with numbers you got from running your code.
   * Due: Thursday April 6 in class
2. Electronic Copy: Submit one electronic copy per team. **Upload** a single source file called JobScheduler.java to Moodle. The file MUST not be a package nor a zip file. Your source file MUST contain your name, your team mate’s name ( if applicable) date and Project# as a comment. (Due Thursday April 6 at 8:00 am)

Template for the JobScheduler, Job and Schedule classes

public class JobScheduler  
{  
 private int nJobs;  
 private Job[] jobs;   
   
 public JobScheduler( int[] joblength, int[] deadline, int[] profit)  
 {  
 //Set nJobs

//Fill jobs array. The kth job entered has JobNo = k;  
 }  
   
 public void printJobs() //prints the array jobs  
 { }  
   
 //Brute force. Try all n! orderings. Return the schedule with the most profit  
 public Schedule bruteForceSolution()  
 { }  
  
   
 public Schedule makeScheduleEDF()

//earliest deadline first schedule. Schedule items contributing 0 to total profit last  
 { }  
  
 public Schedule makeScheduleSJF()

//shortest job first schedule. Schedule items contributing 0 to total profit last  
 { }

public Schedule makeScheduleHPF()

//highest profit first schedule. Schedule items contributing 0 to total profit last  
 { }

public Schedule newApproxSchedule() //Your own creation. Must be <= O(n3)

{ }  
  
}//end of JobScheduler class

//---------------------------Include Job and Schedule classes in JobScheduler. java-----------------------------  
class Job  
{

int jobNumber;  
 int length;  
 int deadline;  
 int profit;  
 int start;  
 int finish;   
   
   
 public Job( int jn , int len, int d, int p)  
 {  
 jobNumber = jn; length = len; deadline = d;  
 profit = p; start = -1; finish = -1;  
 }  
   
   
 public String toString()  
 {  
 return "#" + jobNumber + ":(" + length + ","   
 + deadline + "," + profit +   
 "," + start + "," + finish + ")";  
 }  
  
}//end of Job class

// ----------------------------------------------------  
class Schedule  
{  
 ArrayList<Job> schedule;  
 int profit;  
   
 public Schedule()  
 {  
 profit = 0;  
 schedule = new ArrayList<Job>();  
 }  
   
 public void add(Job job)  
 { }  
   
   
 public int getProfit()  
 { }  
   
 public String toString()  
 {  
 String s = "Schedule Profit = " + profit ;  
 for(int k = 0 ; k < schedule.size(); k++)  
 {  
 s = s + "\n" + schedule.get(k);  
   
 }  
   
 return s;  
 }   
}// end of Schedule class