CSE221_LabAssignment06_Fall2022

Submission Guidelines

1. You can code all of them either in Python or Java or any language you want. But you should choose one specific language for all tasks.
2. For **each task**, please write separate python files like task1.py, task2.py and so on.
3. For each problem, you must take input from files called "inputX.txt" and write output at "outputX.txt", where X is the task number. So, for problem 1, the input file name should be- "input1.txt" & the output file name should be - “output1.txt” and so on.

4. For each task, please include the input files (if any) in the submission folder. 5. All files **MUST** be put in a folder. The folder **MUST** be named as per the following format: **LabSectionNo_ID_CSE221LabAssignmentNo_Fall2022** [Example : **LabSection01_21101XXX_CSE221LabAssignment06_Fall2022**]. 6. Please Zip this folder and rename it as per the following format: **LabSectionNo_ID_CSE221LabAssignmentNo_Fall2022.zip**. [Example : **LabSection01_21101XXX_CSE221LabAssignment06_Fall2022.zip**] 7. Please Submit this zip file via designated google classrooms.

8. You **MUST** follow all the guidelines, naming / file / zipping convention stated above.

Failure to follow instructions will result in straight 50% mark

deduction. **Problem Descriptions**

The following tasks need to be solved using Greedy Algorithm Strategy. Greedy is an algorithm design strategy used in optimization problems where the next available choice with maximum benefit is chosen in each step to reach the final optimal solution. You have learned several greedy algorithms such as dijkstra’s shortest path, prim’s and kruskal’s MST, fractional knapsack, interval scheduling, huffman coding etc. The problems below are related or similar to some of the greedy algorithms you learned in class. Read the task descriptions carefully and implement the algorithm using either Java or Python. The output format **MUST** match exactly as shown in each task.

Task 1 [10 Marks]

Suppose, you have N number of assignments with their time intervals- i.e. starting and

ending times. As you are a student, you need to find out how you can finish the maximum number of assignments. Now, Implement a greedy algorithm to find the maximum number of assignments that can be completed by you.The following conditions must be met when writing the code: A student can only work on a single assignment at a time.
The input will contain N assignments, and then N lines with the starting time and ending time in the format given below:

N
S_1 E_1
S_2 E_2
…….
S�� E��

You have to read input from a file. The output will contain the maximum number of assignments that can be completed followed by the intervals of the selected assignment. Sample input and output is given below. Name your input file “task1_input.txt”. Make sure to try out different input examples to ensure that your code is working for different cases. Include the input file in your zipped submission folder.

Input 1:

```
7
0 4
3 4
1 5
9 10
6 9
2 3
1 2
```

Input 2:

```
6
1 5
1 2
2 4
6 8
5 7
8 9
```

Output 1:

```
5
1 2
2 3
3 4
6 9
9 10
```

Output 2:

```
4
1 2
```

```
2 4                  8 9
5 7
```

A greedy algorithm for the above problem is discussed below. **Before you look at the algorithm it is recommended that you spend some time and try to think of a solution yourself.**

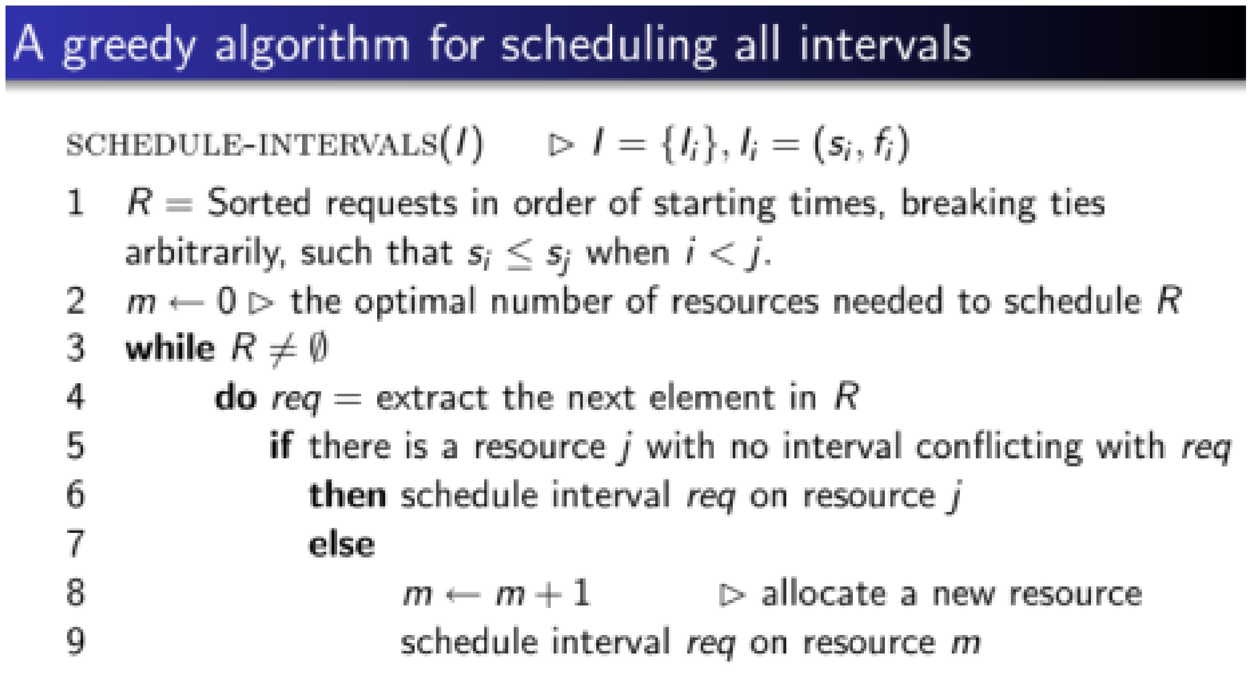
```
//arr[ ][ ] is a 2D array-each index contains the start and finish time of each assignment
Assignment_Selection (arr[ ][ ], n)
    1. Add the 1st assignment from the sorted array in a queue/list "selected"
    2. initialize a variable count = 1.
    3. Current finish time f = arr[0][1] //(finish time of first assignment)
    4. For each remaining assignment in index c:
        a. If the starting time of this assignment is greater or equal to the
           ending time of previously selected assignment, f
            i. then count++
            ii. f=arr[c][1]
            iii. Add the start and finish time of assignment in index c to
                 "selected"
    5. Print count.
    6. Print the queue/list selected
```

Task 2 [5 Marks]

Each activity has a time interval- i.e. a start and a finish time. The activities can be distributed between M people. Implement a greedy algorithm to find the total number of activities that can be completed by M people. The following conditions must be met when writing the code:

1. One person cannot do activities with overlapping time intervals, so each person can do only one activity at a given time interval.
2. Each activity can be completed by only one person.

For task 2 the pseudocode is provided below. You have to modify the pseudocode to complete Task 2.



A greedy algorithm for scheduling all intervals

```
SCHEDULE-INTERVALS($I$)      ▷ $I = \{I_i\}, I_i = (s_i, f_i)$
1  $R$ = Sorted requests in order of starting times, breaking ties
   arbitrarily, such that $s_i \leq s_j$ when $i < j$.
2  $m \leftarrow 0$ ▷ the optimal number of resources needed to schedule $R$
3  while $R \neq \emptyset$
4      do $req$ = extract the next element in $R$
5        if there is a resource $j$ with no interval conflicting with $req$
6          then schedule interval $req$ on resource $j$
7          else
8              $m \leftarrow m + 1$          ▷ allocate a new resource
9              schedule interval $req$ on resource $m$
```

The input will contain N and M, and then N lines with the start time and finish time in the format given below:

N M
S_1 F_1
S_2 F_2
…….
S�� F��

You have to read input from a file. The output will contain the total number of activities that can be completed. Sample input and output is given below. Name your input file "task2_input.txt". Make sure to try out different input examples to ensure that your code is working for different cases. Include the input file in your zipped submission folder.

Input 1:

5 2

1 5

3 6

2 5

8 10

Input 2:

3 2

3 6

2 4

2 3

6 9	
Output 1: 4	Output 2: 3

Task 3 [5 Marks]

Jack and Jill’s parents decide to make their children do some house chores. So they list a set of activities that can be completed in a whole day. In order to complete each activity a certain amount of time (in hours) is required. The parents randomly call each of their children several times separately to choose from the given activities. In each call the children choose an activity based on the following conditions:

1. In each call, each of them chooses one activity
2. Jack has to choose an activity that has not been selected yet.
3. Jill, being very little, should choose an activity that Jack has already chosen so that she can help her older brother because she cannot do such tasks by herself.
4. Jack does not like doing chores, so he decides he would choose the activity that requires the shortest amount of time to complete among the remaining activities. 5. Jill really adores her brother and wants to help him out the most she can, so in each call she decides to choose the activity that needs the maximum time to complete out of the activities chosen by Jack so far.

Implement a greedy algorithm that will meet all above conditions. The input will be N the number of tasks, followed by a sequence of N numbers denoting the time it takes to complete each task and then a call string which is a string of characters only containing J or j representing who is called next Jack (J) or Jill (j). The input format will be:

N
T_1 T_2...... T��
JJjJjjJ

Assume that both the children will be called and Jack will be called either the same or greater number of times than Jill. You should output the sequence of tasks chosen and the total hours each of the children will be working. Sample input output is given below. You should read from a file. Name the file “task3_input.txt” and include it in the submission folder. Sample input output is given below:

Input 1:- 4 1 4 2 3 JJjJjjJ	Input 2:- 3 2 5 3 JjJJjj
Output1:- 1223314	Output 2: 223553 Jack will work for 10 hours
Jack will work for 10 hours Jill will work for 6 hours	Jill will work for 10 hours

A possible greedy algorithm for the above problem is discussed below. **Before you look at the algorithm it is recommended that you spend some time and try to think of a solution yourself.**

1. Sort the activities in ascending order of time in an array
2. Create a stack or priority queue for Jack's current chosen task.
3. Initialize index to 0.
4. Create variables for sequence, Jack's hours and Jill's hours
5. For each character c in the call string
    a. If (c==”J”)
        i. Push value of the index of sorted array in the stack/priority queue ii. Append the value in the sequence
        iii. Increment index
        iv. Add the value to Jack's hours
    b. else if (c==”j”)
        i. Pop the top of the stack or highest value from priority queue
        ii. Append the popped value in the sequence
        iii. Add the popped value to Jill's hours
6. Print sequence, Jack's hours and Jill's hours

Task 4 [5 Marks]

A square number is an integer number whose square root is also an integer. For example 1, 4, 81 are some square numbers. Given two numbers a and b you will have to find out how many square numbers are there between a and b (inclusive).

Input
Each line contains two integers a and b ($0 < a \le b \le 100000$). Input is terminated by a line containing two zeroes. This line should not be processed.

Output
For each line of input produce one line of output. This line contains an integer which denotes how many square numbers are there between a and b (inclusive).

Sample Input
```
1 4
1 10
0 0
```

Sample Output
```
2
3
```