CISC/CMPE-365
Fall 2015

Marked Assignment 1

Due Date: October 30, 2015

The Bader Foundation has presented Queen's with its very own space station, the ABQSS, currently orbiting the Earth in a high-altitude orbit. ABQSS is full of sophisticated equipment and staffed with ace Queen's research personnel. You are personally in charge of selecting the research projects that will be carried out. You have a list of proposed projects, $A = \{a_1, \ldots, a_n\}$ where each project will run from a specified start time s_i to a specified finish time f_i. Due to the nature of the projects, these times cannot be adjusted. The times may overlap but that is not a problem – the space station staff are capable of multi-tasking.

Selecting and approving a research project for the space station requires you to complete a huge amount of paperwork, so your goal is to approve as few projects as possible.

However, what most people don't know is that you are actually an alien spy from The Planet of the Slime Creatures (that's what you call your own planet – nice!) Your planetary leaders are planning an imminent invasion of Earth. Your attack fleet is approaching and there is a time period in the near future, starting at time S and ending at time F, during which your ships will be easily visible from the ABQSS. You need to select a set of projects for the crew so that they will be continually busy during this critical period - they won't have any free time to look out the window and spot your fleet.

Thus your task is to select a minimum-size set of projects so that at all times from S to F, **at least** one research project is in progress. You can choose projects that start before S and/or projects that end after F if you want to. (Hint: the solution to this is quite a lot like the Road Trip Problem: we know we have to cover time S – then we want to go as long as possible before we need to start another project.)

Example: Suppose S = 3, F = 10, and the projects have start and end times
p1: 2 – 5
p2: 1 – 2
p3: 3 – 4
p4: 5 – 9
p5: 6 – 8
p6: 7 – 9
p7: 7 – 13
p8: 4 – 9

In this example we could select {p1, p4, p7} **NOTE: p4 starts at the exact time p1 ends., and p4 and p7 overlap. Both of these situations are ok – the only requirement is that there must not be any gaps between the selected projects.**

There is a further complication to your job. Each project has an associated cost equal to its length, and the costs of the tasks you select must each be charged to one of two accounts. The trolls in Accounting insist that these two accounts must be kept as closely balanced as possible. Thus when you have decided which tasks to select to keep the space station crew busy, you need to divide them into two subsets so that the total times for the two subsets are as close as possible to each other.

Example: continuing with the example above, the three selected projects have costs {3, 4, 6}. The division of this set that comes closest to being equal is {3,4} , {6}

(Hint: this is obviously a version of the Partition Problem. The new bit is testing for the "as close to equal as possible" solution. I recommend checking for a perfect solution, then if there isn't one you can check for a solution that is off by 1, then if necessary check for a solution that is off by 2, etc.)

Input and Output:

The input to your problem will be found in a file named SpaceStation.txt, which will arrive on the course website “soon”.

In the text file provided, the first line gives S and F, the second line gives the number of projects, and each subsequent line gives the project number, and start and finish time of a potential project.

Input for the example problem given above would be

3	10	
8		
1	2	5
2	1	2
3	3	4
4	5	9
5	6	8
6	7	9
7	7	13
8	4	9

Output consists of the selected projects as a list, then divided into two sets with total times as equal as possible. For the example, it should look something like this:

Selected Projects:	1	4	7	
Group 1 Projects:	1	4		Total Time = 7
Group 2 Projects:	7			Total Time = 6

You can use this input to test your program, or create your own test data. However you must submit the result of running your program on the given input file data.

Deliverables:

You must submit:

- your source code, documented with your name and student number
- an output file showing the result of running your program on the provided input instance.
- a description of your algorithm – this can be included in your source code as a block comment, or as a separate document. You are not required to give a formal proof that your algorithm is correct.

Due Date:

Your solution must be submitted by midnight October 30, though Moodle.