Homework 3 - Contest planning for highest score

COP3503 — assigned Jul 25, 2017
Michael McAlpin, Instructor — due July xx, 2017

1 Objective

The rules for a new type of programming contest provides a list of problems, their respective score in integer points, and a statistically valid estimate of the time it takes to solve the problem. This duration or time is expressed in integer hours. To help with a solution strategy the contest organizers reveal there is a dynamic programming solution enabling all the contestants to maximize their score - and stay within the time limits for the contest. One of the rules' drawback is that the duration of the contest and the problem data is not announced until just before the contest start. Another drawback is that once a problem has been started it must be finished within the timeframe, otherwise there is no score credit for the problem being attempted.There is no penalty for finishing early and there is no benefit either, as the time gained cannot be applied to another problem. It is, however, acceptable to continue on to the next problem in the solution list generated from the dynamic program. The assignment is to build the dynamic program that creates the optimum problem list for the given number of problems and timeframe.

2 Requirements

Read the input file formatted as follows. The input file will contain the following data, in the order shown below.

Input Data	Description
5	Number of problems
10	Number of hours
2 3	Hours & points per problem per problem, repeating for *up to 10* problems

Table 1: Input file data layout

2.1 Functions

Input Read input file [1], passed as the first command line argument with one line each for the following:

- Number of problems - an integer between 1 and 10

[1]The data in the file will be in the order shown above, that is, # problems, # contest hours, and up to 10 occurences of the problem time and problem points.

- Number of hours for the contest - an integer between 1 and 12
- Number of hours for the problem following by the number of points for this problem. For example: `4 10` would be a four hour problem worth ten points.
  - These number pairs can occur up to 10 times.

Schedule This function will determine the problem set to be attempted to maximize the contest score.

Print list This function will print the optimum problem set as determined by the *Schedule* function described above.

Additional methods and properties may be required to successfully implement the functions specified above.

3 Testing

Make sure to test your code on Eustis **even if it works perfectly on your machine**. If your code does not compile on Eustis you will receive a 0 for the assignment. There will be three (3) input files and three (3) output files provided for testing your code, they are respectively shown in Table 2 and in Table 3.

Filename	Description
contestA.in	4 problems over 10 hours
contestB.in	6 problems over 12 hours
contestC.in	7 problems over 10 hours

Table 2: Input files

The expected output for these test cases will also be provided as defined in Table 3. To compare your output to the expected output you will first need to redirect *STDOUT* to a text file. Run your code with the following command (substitute the actual names of the input and output file appropriately):

```
java Hw03 inputFile > output.txt
```

The run the following command (substitute the actual name of the expected output file):

```
diff output.txt expectedOutputFile
```

If there are any differences the relevant lines will be displayed (note that even a single extra space will cause a difference to be detected). If nothing is displayed, then congratulations - the outputs match! For each of the three (3) test cases, your code will need to output

to *STDOUT* text that is identical to the corresponding *expectedOutputFile*. If your code crashes for a particular test case, you will not get credit for that case.
Please note that grading will be based on the input and output files described above, along with one or two other input files.

4 Submission - via WebCourses

The Java source file(s). Make sure that the *main* program is in Hw03.java.

Use reasonable and customary naming conventions for any classes you may create for this assignment.

Include a comment at the top of your main source file that contains the following statement (substitute your name and NID) - *I [name] ([NID]) affirm that this program is entirely my own work and that I have neither developed my code together with any another person, nor copied any code from any other person, nor permitted my code to be copied or otherwise used by any other person, nor have I copied, modified, or otherwise used program code that I have found in any external source, including but not limited to, online sources. I acknowledge that any violation of the above terms will be treated as academic dishonesty.*

5 Sample output

```
ff210377@eustis:~/COP3503$ java Hw03 contestA.in
contestA.in has 4 problems over 10 hours
Pr# Time Points
1   2    4
2   3    5
3   4    6
4   6    7

The selected  problems for the highest contest score are:
Problem Points
1       4
2       5
3       6

Target problem list yields 15 points
ff210377@eustis:~/COP3503$ java Hw03 >contestB.testOut
ff210377@eustis:~/COP3503$ diff contestB.testout contestB.output
ff210377@eustis:~/COP3503$
```

Assume the columnar data for the problem detail data shown above is separated by the tab character (`"\t"`). This will minimize troubleshooting data alignment in the `diff` processing.

Command	Output filenames
java Hw03 contestA.in	contestA.output
java Hw03 contestB.in	contestB.output
java Hw03 contestC.in	contestC.output

Table 3: Commands with input files and corresponding output files.

6 Grading

Grading will be based on the following rubric:

Percentage	Description
-100	Cannot compile on *Eustis.*
-100	Cannot read input files.
- 50	Does not execute in a reasonable time frame. *Infinite loop or exponential performance.*
- 25	Cannot produce an optimum problem sequence for the given input.
- 25	Only works with one input file.
- 10	Output does not match *expectedOutput.txt* exactly.

Table 4: Grading Rubric