Algorithms and Data Structures
Jacobs University Bremen
Dr. Kinga Lipskoch

Course: CH08-320201
April 26th, 2019

Homework 10

- Submit one ZIP file per homework sheet which contains one PDF file (including pictures, computations, formulas, explanations, etc.) and your source code file(s) with one makefile and without adding executable, object or temporary files.
- The implementations of algorithms has to be done using C, C++, Python or Java.
- The TAs are grading solutions to the problems according to the following criteria: https://grader.eecs.jacobs-university.de/courses/320201/2019_1/Grading_Criteria_ADS.pdf

Problem 10.1 *Longest ordered subarray* (4 points)

Consider the array $A = (a_1, a_2, ..., a_n)$. We call a subarray a succession of numbers in A where the position of any value in the subarray in the array is always bigger than the value of the previous one. Use dynamic programming to find a subarray of maximal ordered length of the array A. You can assume there are no duplicate values in the array. Your algorithm should find one optimal solution if multiple exist.

Example
For $A = (8, 3, 6, 50, 10, 8, 100, 30, 60, 40, 80)$ the solution is: $(3, 6, 10, 30, 60, 80)$

Your program should take an input and generate an output as follows:
Sample input

```
8 3 6 50 10 8 100 30 60 40 80
```

Sample output

```
3 6 10 30 60 80
```

Your solution will be graded using multiple testcases, therefore following the input and output format is mandatory.

Problem 10.2 *Sum in triangles* (7 points)

Consider a triangle formed from n lines ($1 < n \leq 100$), each line containing natural numbers in the interval $[1, 10000]$. If the input does not fall in the range from before your implementation should throw exceptions or print messages and repeatedly ask for valid input.

(a) (5 points) Use dynamic programming to determine the biggest sum of numbers existent on the path from the number in the first line and a number from the last line and print the respective path to the output. Each number in this path is seated to the left or to the right of the other value above it.

(b) (1 point) Analyze the runtime of your solution and compare it to the brute force approach.

(c) (1 point) Explain why a greedy algorithm does not work for this problem.

Example
The values are displayed for example in this manner:

```
          7
        3   8
      8   1   0
    2   7   4   4
4   5   2   6   5
```

For this example the resulting sum is 30.

Your program should take an input and generate an output as follows:
Sample input

```
7
3 8
8 1 0
2 7 4 4
4 5 2 6 5
```

Sample output

```
30
7 3 8 7 5
```

Your solution will be graded using multiple testcases, therefore following the input and output format is mandatory.

Problem 10.3 *Scuba Diver* (12 points)

A scuba diver uses a special equipment for diving. He has a cylinder with two containers: one with oxygen and the other with nitrogen. Depending on the time he wants to stay under water and the depth of diving the scuba diver needs various amount of oxygen and nitrogen. The scuba diver has at his disposal a certain number of cylinders. Each cylinder can be described by its weight and the volume of gas it contains. In order to complete his task the scuba diver needs specific amounts of oxygen and nitrogen. Theoretically, the diver can take as many cylinders as he wants/needs. Use dynamic programming to find the minimal total weight of cylinders he has to take to complete the task and which those cylinders are. In case of several acceptable solutions, printing just one of them is enough.

Example

The scuba diver has at his disposal 5 cylinders described below. Each description consists of: volume of oxygen, volume of nitrogen (both values are given in liters) and weight of the cylinder (given in decagrams):

```
3 36 120
10 25 129
5 50 250
1 45 130
4 20 119
```

If the scuba diver needs 5 liters of oxygen and 60 liters of nitrogen then he has to take two cylinders of total weight 249 (for example the first and the second ones or the fourth and the fifth ones).

Input

The number of test cases c is in the first line of input, then c test cases follow separated by an empty line. In the first line of a test case there are two integers t, a separated by a single space, $1 \leq t \leq 21$ and $1 \leq a \leq 79$. They denote volumes of oxygen and nitrogen respectively, needed to complete the task. The second line contains one integer n, $1 \leq n \leq 1000$, which is the number of accessible cylinders. The following n lines contain descriptions of cylinders; i^{th} line contains three integers t_i, a_i, w_i separated by single spaces, ($1 \leq t_i \leq 21$, $1 \leq a_i \leq 79$, $1 \leq w_i \leq 800$). These are respectively: volume of oxygen and nitrogen in the i^{th} cylinder and the weight of this cylinder.

Output

On the first line will be printed the total weight w_t and on the next line separated by spaces the index of the cylinders in order.

If the input values do not fall in the ranges from before your implementation should throw exceptions or print messages and repeatedly ask for valid input.

Sample input

```
1
5 60
5
3 36 120
10 25 129
5 50 250
1 45 130
4 20 119
```

Sample output

```
249
1 2
```

Your solution will be graded using multiple testcases, therefore following the input and output format is mandatory.

How to submit your solutions

You can submit your solutions via *Grader* at **`https://grader.eecs.jacobs-university.de`** as a generated PDF file and/or source code files.
If there are problems with *Grader* (but only then), you can submit the file by sending mail to `k.lipskoch@jacobs-university.de` **with a subject line that starts with CH08-320201**.

Please note, that after the deadline it will not be possible to submit solutions. It is useless to send solutions by mail, because they will not be graded.

This homework is due by Friday, May 3rd, 23:00.