Programming Assignment 2
Optimal Cheque Writing Problem

COMP 6651 – Winter 2019

Dr. Tiberiu Popa

Introduction

This programming assignment will help you acquire practical experience with dynamic programming. You are asked to implement a dynamic programming algorithm that solves the optimal cheque writing problem.

The input is a number of periods (N), a constant amount (A) that has to be paid each period to a creditor and an initial balance in your bank account (B). For each time period $0 \leq k < N$ you need to write a cheque for an amount m_k given the following rules:

- You cannot owe money to your creditor: $\sum_{i=0}^{i \leq k} m_i \geq A * (k+1)\,, \forall\, 0 \leq k < N$
- Your balance cannot be negative (but you can assume that your account gets credited by an amount equal to A every time period): $\sum_{i=0}^{i \leq k} m_i - A * (k+1) \geq B, \forall\, 0 \leq k < N$
- At the end of the N periods you paid the creditor exactly the amount owed: $\sum_{i=0}^{i<N} m_i = A * N$
- The cheque amount has to be positive. An amount of 0 as payment is allowed, but the cost is 0 as it does not require writing a cheque ($m_k \geq 0, \forall\, 0 \leq k < N$)

For example:

1. If N=3, A=100, B=0. The only way to pay is to write a cheque for 100 every time period
2. If N=3, A=5, B=1. You can pay: (5, 5, 5), (6, 5, 4) , (6, 4, 5), (5, 6, 4). But you cannot pay: (4, 5, 6) because in the first month you owe 5; you cannot pay: (6, 6, 3) because your account will go negative after the second period; you cannot pay (6, 5, 5) because at the end you paid too much.

We associate to writing each cheque a cost C[x] equal to the number of characters taken to write the amount x on the cheque. You are asked to write a program that finds a payment schedule that satisfies the constraints above and it is optimal: $(m_i)_{0 \leq i < N}$ *is optimal iff* $\sum_{i=0}^{i<N} C[m_i] \leq \sum_{i=0}^{i<N} C[m'_i]$ for any other valid payment schedule $(m'_i)_{0 \leq i < N}$. $S = \sum_{i=0}^{i<N} C[m_i]$ is the optimal cheque writing cost.

For example:

1. If N=2, A=100, B=0; the optimal schedule cost is 20: writing twice "one hundred"
2. If N=2, A= 100, B=100; the optimal schedule cost is 10:(writing one cheque for "two hundred"). Note that a payment schedule of (200,0) is valid.
3. If N=3, A=97, B=6; the optimal schedule is 28 for the schedule (99,96,96) corresponding to writing: "ninety nine" and twice "ninety six". Another valid schedule might be (97,97,97) but its cost is higher at 33.

You can further assume that all amounts are integers: $0 < N < 1001, 0 \leq B < 10{,}000, 0 < A < 10{,}000$. The cost for each positive integer number is given in a file and its format is explained below. The largest cheque that can be written regardless of any other parameters of the program is 10,000 (i.e. $(0 \leq m_i \leq 10{,}000)$) $\forall\, 0 \leq i < N$. A payment of 0 is allowed and it has 0 cost.

Specifications

The input is specified in a file whose name is the first argument of the program. The first line contains an integer M specifying how many datasets are in the file. The reminder of the file encodes the datasets. Each dataset is encoded in one line as 3 space separated integers (N, A, B) as defined above.

Here is an example:

```
3
2 100 0
2 100 100
3 97 6
```

The output is a file called whose name is the second argument of the program. Each line encodes the results of each test case. A valid solution always exists (i.e. $(m_i = A)\ \forall\ 0 \leq i < N)$ The algorithm should output the cost of an optimal payment schedule to the output file. Note that there should be no end of line after the last test case.

For example, the output corresponding to the input above is as follows:

```
20
10
28
```

(there should be no end of line after 28)

Additionally, the program should output to the standard output console, one line per test case, a payment schedule $(m_i)\ \forall\ 0 \leq i < N$ corresponding to the optimal cost. For the above example one output could be:

```
100 100
200 0
99 96 96
```

What is given

No code is given for this exercise. However, a file containing the costs of writing a cheque of a given amount is given. The name of this file (may include a full path) should be the third argument of your program.

The file contains 10,001 lines, and each line contains two space separated integers: D and F. D is an amount and F is the cost of writing D on a cheque (i.e the number of characters not including spaces or dashes or the "dollar or "dollars" words sometimes present in writing cheques).

As there could be disputes over spelling of certain numbers, for the purpose of this assignment, the cost should always be taken from the provided file. The amount D will always appear in increasing order from 0 to 10,000. The cost of 0 will be considered 0 as it does not require writing any cheques.

Submission

You have to implement your program in plain C/C++ in a file called *main.cpp* that has no dependency other than the standard C/C++ libraries available in a standard Linux system such as the one in the graduate labs. The code should compile using the command *g++ main.cpp* in any machine in the graduate labs.

You need to submit the file *main.cpp* **ONLY,** using the EAS system. If you submit anything else, you will receive a grade of 0. Submit the exercise under "**Programming Assignment 2**". The due-date is **March 22nd , 5pm. For this programming assignment, no late submission is allowed.**

This programming assignment is individual.

Originality and Plagiarism

Some of the problems proposed in the course are "classical" in the algorithm design literature and, therefore, solutions may be available on-line. Please note that you are expected to do this problem individually and you are expected to produce an original solution. We run plagiarism tests and if your submission is red-flagged you are expected to explain it in detail to the instructor. Failure to comply with this request or to adequately explain your own code may result in filing a plagiarism case with the Dean of the Faculty of Engineering and/or the Dean of Graduate Studies.

Evaluation and Testing

The code is evaluated automatically based on the output file. However, we do check if you implemented the right algorithm and if the code is original. Grades will be deducted if the output to the console is incorrect. You will receive 0 if the code does not compile. We run 3 test files, each file may have several test cases. You receive 1/10 marks if your code compiles, but it does not return the correct results on any of the test files; 4/10 if it compiles and runs correctly on one of the test files, etc. If your code does not provide the correct output for ALL test cases in a given test file, you will receive a grade of 0 for that test fie.

Therefore, we **strongly recommend** that you create many test cases/files to test your code.

You will receive a code of 0 if you did not implement a proper dynamic programming solution or if the code takes too long to run. The limit on the run-time of a give test file is twice the run-time of the official solution.

You may be asked to explain your code to the teaching assistant or the instructor. Failure to comprehensively explain your code and any design decision you took in writing your solution may lead to a decrease in your overall grade for this programming assignment.