Introduction to Algorithms	**Problem Set 3 (3 problems)**
CS 4820 Fall 2019	**Due 11:59pm Thursday, September 26**

There are three questions on this homework, a coding problem and two written questions.

Your homework submissions for written questions need to be typeset (hand-drawn figures are OK). See the course web page for suggestions on typing formulas. The solution to each written question needs to be uploaded to CMS as a separate pdf file. To help provide anonymity in your grading, do not write your name on the homework (CMS will know it's your submission).

Solutions to the coding problem need to be submitted to the **online autograder** at `https://cs4820.cs.cornell.edu/`.

Collaboration is encouraged while solving the problems, but

1. list the names of those with whom you collaborated with (as a separate file submitted on CMS);
2. you must write up the solutions in your own words;
3. you must write your own code.

Remember that when a problem asks you to design an algorithm, you must also prove the algorithm's correctness and analyze its running time. The running time must be bounded by a polynomial function of the input size.

Some Comments on Dynamic Programming Problems. In writing up dynamic programming algorithms (for written questions) all previous advice applies; but there are some additional things needed for full credit as well. If your solution consists of a dynamic programming algorithm, you must clearly specify the set of sub-problems you are using and the recurrence you are using in addition to describing what they mean in English as well as any notation you define. You must also explain why your recurrence leads to the correct solution of the sub-problems: this is the heart of a correctness proof for a dynamic programming algorithm. Finally, you should describe the complete algorithm that finds the solution value as well as the solution itself, while making use of the recurrence and sub-problems. A description of a DP algorithm consisting of a piece of pseudo-code **without these explanations will not get full credit**.

(1) Highway Planning. You are helping plan a new stretch of highway that is M miles long. One issue that came up is the placements of gas-stations. To help drivers not get off the road, you agreed to have a gas station at least every m miles (that is, you need to have a gas station with m miles of the start and end of the highway stretch, and two consecutive stations cannot be more than m miles apart. You also have estimates on costs of building gas-stations at various points along the high way. The costs vary due to differences in the land around the highway, and the proximity of town. The possible options (measured from the start of the highway) are at miles $x_1, \ldots, x_n$, and location x_i has cost estimate $c_i > 0$. You may assume that you have enough possibilities to satisfy the requirements, that is $x_1 \leq m$, $M - x_n \leq m$, and $x_{i+1} - x_i \leq m$ for all $i \leq n-1$. Design a $O(n \log n)$ dynamic programming algorithm for finding a set of gas stations of minimum cost satisfying the requirements, and implement the algorithm in Java. We suggest using subproblem $Opt[i]$ to be the minimum cost solution for the stretch of highway up to position x_i **with a gas-station at location** x_i. Your algorithm needs to output the total cost (sum of costs of the selected locations), as well as the list of locations selected in increasing order of distances.[1] Algorithms that find the correct solution, but run in $O(n^2)$ time will be graded out of 7 points. Algorithms that are faster than $O(n \log n)$ can earn 3 bonus points.

The only libraries you are allowed to `import` are the ones in `java.util.*`, and ones dealing with reading and writing inputs, such as `java.io.Reader`, `java.io.Writer`, etc. We recommend using BufferedReader and BufferedWriter for reading/writing standard input/output.

[1] You can assume that solution with optimum cost is unique.

Warning: be aware that the running time of calling a method of a built-in Java class is usually not constant time, and take this into account when you think about the overall running time of your code. For instance, if you used a LinkedList, and use the `indexOf` method, this will take time linear in the number of elements in the list.

We have set up an **online autograder** at `https://cs4820.cs.cornell.edu/`. You can upload solutions as many times as you like before the homework deadline.[2] **You need to have the main method of your code named Main, must not be "public", must not be part of a package**. When you upload a submission, the autograder will automatically compile and run it on a number of public test-cases that we have prepared for you, checking the result for correctness and outputting any problems that may occur. You should use this facility to verify that you have interpreted the assignment specification correctly. After the deadline elapses, your last submission will be tested against a new set of *private* test cases, which your grade for the assignment will be based on.

Input / output formatting and requirements
Your algorithm is to read data from `stdin` in the following format:

- The first line has three integers, $1 \leq n \leq 10^6, 1 \leq M \leq 10^9, 1 \leq m \leq M$, representing the number of possible options for gas-stations, the miles of the highway, and the distance constraint between two consecutive gas-stations.
- Each of the following n lines contains 2 integers, $1 \leq x_i \leq M, 0 \leq c_i \leq 10^9$ in the i-th line, describing a potential option for a gas-station locating at x_i with a cost of c_i. You may assume locations are given in an increasing order. (I.e., $x_i \leq x_{i+1}$ for all $i \leq n-1$.)

Your algorithm should output data to `stdout` in the following format:

- The first line contains the minimum total cost s for building gas-stations on the highway.
- Note that s could be as large as $10^6 \times 10^9 = 10^{15}$, so be careful with the integral type you stored $Opt[i]$. Typically, it should be `long` instead of `int`.
- The second line contains the list of locations where gas-stations should be placed, separated by a space and in increasing order. You can assume that the solution with optimum cost is unique. Please use $\backslash n$ for your line endings.

(2) Matrix Multiplication. Consider the problem of multiplying a sequence of matrices $A_1, A_2, \ldots, A_m$ that was mentioned at the beginning of class on Wednesday, September 18th. Assume matrix A_i is an $n_{i-1} \times n_i$ matrix. To multiply two matrices of size $n \times m$ and $m \times k$, the resulting matrix will be $n \times k$, and we get each entry by m multiplications and $m-1$ additions, so this requires a total of mnk multiplications and $\leq mnk$ additions. In multiplying a sequence of matrices, by associativity of multiplication, we have many orders we can do the multiplication. For example $(A_1 \cdot A_2) \cdot A_3 = A_1 \cdot (A_2 \cdot A_3)$, and doing it different order has different running times. For example, suppose A_1 is 2×4, A_2 is 4×10, and A_3 is 10×2. If we multiply $A_1 \cdot A_2$ and then multiply the result by A_3 the number of multiplications is $2 \times 4 \times 10 = 80$ for the $A_1 \cdot A_2$ and then $2 \times 10 \times 2 = 40$ for multiplying the result by A_3 for a total of 120 multiplications. In contrast, multiplying $A_2 \cdot A_3$ first is $4 \times 10 \times 2 = 80$ multiplication, and then A_1 times this product is another $2 \times 4 \times 2 = 16$ for a total of only 96 multiplications, so this is a better order. Given a sequence of m matrices, $A_1, A_2, \ldots, A_m$ (where A_i is an $n_{i-1} \times n_i$ matrix), give a polynomial time algorithm to find the order to do the multiplications that require the fewest multiplications. Your algorithm needs to output the order of multiplications, as well as the number of multiplications needed.

(3) Min-cost path with negative costs. Consider a directed graph $G = (V, E)$ where the cost of edge e, c_e, can be negative, and this graph may have negative cycles. Given two nodes s and t, we have

[2]The deadline shown on the auto-grader is the latest time at which you will be allowed to submit your code, including the maximum number of slip days (3). If you submit your code past the actual deadline, you will automatically be charged slip days accordingly, so please be careful.

seen in class how to compute the length of the min-cost path in G assuming G has no negative cost cycles. In this problem, we will want to find low cost path, and also detect if the graph has negative cycles.

(a) *(2 points)* Give a polynomial time algorithm that computes the minimum cost path between two nodes $s, t \in V$ using at most k edges. If the graph has negative cycles, you may want to use edges more than once, which is OK as long as you don't use more than k edges.

(b) *(4 points)* Give a polynomial time algorithm that either finds the minimum cost path from s to t or finds a negative cycle in the graph G.

(c) *(4 points)* To be more robust, you would like to know if the minimum cost path is unique. Modify the algorithms in (a+b) to also decide if there is only one min-cost path or are there multiple min-cost options.