Homework #1

Due Time: 2020/10/22 14:20
Contact TAs: `ada-ta@csie.ntu.edu.tw`

Instructions and Announcements

- There are **four programming problems** and **two hand-written problems**.
- **Programming.** The judge system is located at `https://ada-judge.csie.ntu.edu.tw`. Please login and submit your code for the programming problems (i.e., those containing "Programming" in the problem title) by the deadline. NO LATE SUBMISSION IS ALLOWED.
- **Hand-written.** For other problems (also known as the "hand-written problems"), you should upload your answer to **Gradescope** as demonstrated in class. NO LATE SUBMISSION IS ALLOWED.
- **Collaboration policy.** Discussions with others are strongly encouraged. However, you should write down your solutions **in your own words**. In addition, for **each and every** problem you have to specify the references (e.g., the URL of the website you consulted or the people you discussed with) on the first page or comment in code of your solution to that problem. You may get zero points due to the lack of references.

Problem 1 - Bridge (Programming) (10 points)

Problem Description

There is a mysterious country famous for nuts and pies. The land of this country is rectangular, and the lower-left and upper-right corners are at $(0, 0)$ and $(10^9, 10^9)$ on a two-dimensional coordinate system, respectively. In this country, each city is located at a point of integer coordinates, and a city located at (x, y) can produce x nuts and y pies per unit time. We say that the city has x nut-productivity and y pie-productivity.

The capital O, located at $(0, 0)$, has no productivity itself, but is a trading center. Now you are asked to select two cities A and B from N candidate cities, and build two bridges: one between A and O, and the other between B and O. Constructing the bridges requires cost, but the trade after completion can bring profits. The cost of constructing a bridge is equal to the **square of** the Euclidean distance between two ends of the bridge. After finishing building these two bridges, the total profit will be **twice** the product of A and B's nut-productivity $(2A_x \cdot B_x)$ and pie-productivity $(2A_y \cdot B_y)$.

Given N candidate cities, please select two cities to build bridges in order to maximize the net revenue (that is, profits minus cost).

Input

The first line of the input file contains an integer indicating N.

The i-th of the next N lines contains two integers x_i, y_i — the coordinates of the i-th city.

Test Group 0 (0 %)

- Sample Input

Test Group 1 (30 %)

- $1 \leq N \leq 5 \times 10^3$
- $0 \leq X_i \leq 10^9$
- $0 \leq Y_i \leq 10^9$
- $X_i + Y_i \neq 0$

Test Group 2 (70 %)

- $1 \leq N \leq 2 \times 10^5$
- $0 \leq X_i \leq 10^9$
- $0 \leq Y_i \leq 10^9$
- $X_i + Y_i \neq 0$

Output

Please output an integer indicating the maximum net revenue.

Sample Input 1

```
2
1 1
1 1
```

Sample Output 1

```
0
```

Sample Input 2

```
2
0 1
1 0
```

Sample Output 2

```
-2
```

Problem 2 - Bomb Game (Programming) (15 points)

Problem Description

A helper header file is provided for this problem. Additional information is available at the end of the problem description.

BB Inc. recently launched a new online game called *the Bomb Game*. In this online game, players have to constantly upgrade their armors to protect themselves from bombing attacks.

The playfield can be seen as a one-dimensional grid containing N cells numbered from 1 to N. A round of *the Bomb Game* consists of a series of *events*. An *event* can be in one of the following two forms:

1. A new player enters the game. The player is located at cell c_i and has a *defence level* of d_i. (We assume that the player never changes location.)
2. A bombing attack occurs. The attack spans cells in the range l_i to r_i (inclusive). It has an *explosive power* of p_i and will deal k_i points of damage to all players in the range whose defence levels are **not greater than** p_i (i.e., the player j takes damage if $l_i \leq c_j \leq r_i$ and $d_j \leq p_i$).

Now, given all M events in a round of *the Bomb Game*, can you figure out the total damage dealt to each player?

Input

The first line of the input contains two integers N and M, representing the length of the field and the number of events in the round, respectively. Then M lines follow, the i-th of which describes the event i. Each line starts with a character T_i — the event type:

- If $T_i =$ `'P'`, the i-th event is a new player event. The line then contains two more integers c_i and d_i, representing the player's location and defence level, respectively.
- Otherwise $T_i =$ `'A'`, meaning the i-th event is a bombing attack. The line contains four more integers l_i, r_i, p_i, and k_i, representing the range of the attack ($[l_i, r_i]$), the attack's explosive power, and the damage dealt per player.

- $1 \leq N, M \leq 10^5$
- $1 \leq c_i \leq N$
- $1 \leq l_i \leq r_i \leq N$
- $1 \leq d_i, p_i \leq 10^9$
- $1 \leq k_i \leq 10^4$

Test Group 0 (0 %)

- Sample Input.

Test Group 1 (20 %)

- $M \leq 6000$

Test Group 2 (20 %)

- $d_i = p_i = 1$

Test Group 3 (60 %)

- No additional constraints.

Output

Output the total damage dealt to each player, one player per line, in the order they join the game.

Sample Input 1

```
10 10
P 3 5
A 2 8 15 5
P 7 10
A 4 10 5 3
A 1 9 10 7
P 6 20
P 5 1
A 4 9 17 2
A 1 2 20 4
P 9 5
```

Sample Input 2

```
10 6
P 5 1
P 2 1
A 3 8 1 3
P 6 1
A 2 6 1 6
A 2 4 1 4
```

Sample Output 1

```
12
9
0
2
0
```

Sample Output 2

```
9
10
6
```

Helper Header File

A helper header file is provided for this problem. You can download the header file from `https://www.csie.ntu.edu.tw/~b08902107/ADA/helper.h`, or simply include "`helper.h`" in your code submitted to the judge. **This header file provides a one-dimensional array that supports fast range addition.**

The array has $N = 200000$ cells, numbered from 1 to N. The values in these cells are initially all zeros. There are three functions to alter and access the array values:

- `void Memory::reset()`: Explicitly fill the whole array with zeros. This function runs in $\mathcal{O}(N)$ time.
- `void Memory::add(int l, int r, int k)`: Add k to each cell in the range $[l, r]$. The arguments should satisfy the condition $1 \leq l \leq r \leq N$, while k can be any signed 32-bit integer. This function runs in $\mathcal{O}(\log(N))$ time.
- `long long Memory::get(int x)`: This function returns the value currently stored in cell x. The argument x should be a positive integer not greater than N. This function also runs in $\mathcal{O}(\log(N))$ time.

Hint

- You may want to try divide-and-conquer on the **event timeline**.

Problem 3 - ADA Party (Programming) (15 points)

Problem Description

YP, BB, and their K classmates are going to the ADA party. To enjoy the party, they prepared N bags of candies, where the i-th bag contains a_i candies.

Now it's party time! These $(K+2)$ people decide to have fun together, so they play a game that involves eating candies. In this game, they select two integers ℓ, r and eat all the candies in the ℓ-th bag, $(\ell+1)$-th bag, ..., and the r-th bag (not necessarily in order) together based on the following rules: first, YP picks a bag with the **maximal** number of candies and eats every candy in it. Then BB picks a bag with the **minimal** number of candies and also eats the candies in it. After YP and BB finish the candies in those two bags, the other K people hope that they can eat the same number of candies in the remaining $(r-\ell-1)$ bags. It is okay that they don't eat any candies.

YP and BB think that (ℓ, r) is a good pair if the following conditions hold:

- $1 \leq \ell < r \leq n$
- The remaining K people can eat the same number of candies. Note that the candies in a bag can be split among multiple people, and each person can eat candies from multiple bags.

Please help YP and BB calculate the number of good pairs.

Input

The first line of the input contains two integers N and K, denoting the number of bags and the number of YP and BB's classmates respectively.

The second line of the input contains N space-separated integers $a_1, a_2, \ldots, a_N$, where the i-th integer denotes that the i-th bag contains a_i candies.

Test Group 0 (0 %)

- Sample Input.

Test Group 1 (10 %)

- $2 \leq N \leq 200$
- $1 \leq K \leq 200$
- $1 \leq a_i \leq 200$
- All a_i are distinct.

Test Group 2 (20 %)

- $2 \leq N \leq 5000$
- $1 \leq K \leq 5000$
- $1 \leq a_i \leq 10^9$
- All a_i are distinct.

Test Group 3 (50 %)

- $2 \leq N \leq 10^5$
- $1 \leq K \leq 10^5$
- $1 \leq a_i \leq 10^9$
- All a_i are distinct.

Test Group 4 (20 %)

- $2 \leq N \leq 5 \times 10^5$
- $1 \leq K \leq 5 \times 10^5$
- $1 \leq a_i \leq 10^9$

Output

Print the number of good pairs in a single line.

Sample Input 1

```
5 3
5 6 1 8 4
```

Sample Output 1

```
6
```

Sample Input 2

```
5 10
1 2 3 4 5
```

Sample Output 2

```
4
```

Sample Input 3

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

Sample Output 3

```
25
```

Sample Input 4

```
25 7
20 25 6 1 15 7 16 13 14 24 11 19 18 5 21 3 23 8 4 9 12 10 22 2 17
```

Sample Output 4

```
72
```

Hint

1. The good pairs in Sample Input 1 are $(1, 2)$, $(1, 5)$, $(2, 3)$, $(2, 4)$, $(3, 4)$, and $(3, 5)$.
2. Because the input files are large, please add
   - `std::ios_base::sync_with_stdio(false);`
   - `std::cin.tie(nullptr);`

   to the beginning of the main function if you are using `std::cin`.
3. You may need to use the *prefix sum* technique to solve this problem:
   - Define the prefix sum sequence $p_1, p_2, \ldots, p_n$ on sequence $a_1, a_2, \ldots, a_n$ as follows:
     - $p_1 = a_1$
     - $p_i = a_1 + a_2 + \cdots + a_i = p_{i-1} + a_i$, if $i > 1$
   - After constructing the prefix sum sequence, $(p_r - p_{\ell-1})$ is equal to $(a_\ell + a_{\ell+1} + \cdots + a_r)$.
4. After applying the prefix sum technique, the number of candies in the remaining $(r - \ell - 1)$ bags is equal to $(p_r - p_{\ell-1} - \max(a_\ell, a_{\ell+1}, \ldots, a_r) - \min(a_\ell, a_{\ell+1}, \ldots, a_r))$.
5. GL & HF (Good Luck and Have Fun).

Problem 4 - Robot (Programming) (10 points)

Problem Description

Robert is a farmer. He has a farm represented by an $n \times m$ grid. Each cell has unlimited nuts and each nut has its *sweetness*. Moreover, all nuts in the same cell have the same sweetness.

Robert would like to harvest some nuts, so he tries to control his robot named "Robot1003" to complete this mission. Initially, Robot1003 is at the top-left corner of the farm. Robert can only control Robot1003 to move right or down to the next cell in that direction. When Robot1003 is in a cell, it must take one nut from that cell. Besides, Robert has k *space machines* that can each be only used once. Once he uses up one space machine, he can make Robot1003 jump to any cell in the farm, including the current one.

However, every time a space machine is used, each nut in the farm will have its sweetness decreased by a constant c. Note that the sweetness may drop to a negative value. Robert can stop this process only when Robot1003 reaches the bottom-right corner of the farm. He hopes that the total sweetness harvested by Robot1003 can be as high as possible. Please help Robert calculate the optimal way of controlling Robot1003 to maximize the total sweetness.

Input

The first line contains four integers n, m, k, and c ($1 \leq n, m \leq 500,\ 0 \leq k \leq 10,\ 1 \leq c \leq 10^9$), denoting the numbers of rows and columns in the farm, the number of space machines Robert has, and the constant subtracted from the sweetness values when using a space machine.

In the next n lines, the i-th contains m integers. The j-th integer a_{ij} ($-10^9 \leq a_{ij} \leq 10^9$) represents the initial sweetness of nuts in the cell (i, j). $(0 \leq i < n, 0 \leq j < m)$

Output

The first line of the output contains an integer, denoting the maximal sweetness that can be harvested by Robot1003.

The second line contains an integer s, denoting the number of moves throughout the process. In the next s lines, the i-th line represents the i-th move. For each move, if Robert should use the space machine on that step, output the word "Jump"; otherwise, output "Move". After that, output two integers representing the coordinates Robot1003 will be at after the i-th move.

If there are multiple ways to control Robot1003 to maximize the total sweetness, you can output any one of them.

Test Group 1 (25 %)

- $k = 0$.

Test Group 2 (15 %)

- $k = 1$.

Test Group 3 (10 %)

- $n, m \leq 20$.

Test Group 4 (50 %)

- No additional constraints.

Sample Input 1

```
2 2 0 1
1 1
3 -1
```

Sample Output 1

```
3
2
Move 1 0
Move 1 1
```

Sample Input 2

```
2 2 1 1
1 1
3 -1
```

Sample Output 2

```
4
3
Move 1 0
Jump 1 0
Move 1 1
```

Sample Input 3

```
10 10 10 9
9 20 -15 -7 -5 -7 19 20 2 -13
7 8 -6 -14 0 -14 16 12 -7 19
2 1 -16 -12 -4 -9 11 -13 2 18
-1 -4 -11 -8 20 4 18 1 20 16
19 12 -4 0 4 -20 18 -11 -20 -5
19 0 -10 4 -16 -14 18 -5 -14 8
6 19 11 7 11 -16 -3 4 13 20
-2 19 11 17 -20 -5 9 17 -6 -3
-4 -19 7 13 -18 -20 15 2 -5 -13
17 19 -10 5 -17 -8 -1 -14 8 8
```

Sample Output 3

```
186
21
Move 1 0
Move 2 0
Move 3 0
Move 4 0
Move 5 0
Move 6 0
Move 6 1
Move 7 1
Move 7 2
Move 7 3
Move 8 3
Move 9 3
Jump 4 0
Move 5 0
Move 6 0
Move 6 1
Move 7 1
Move 7 2
Move 7 3
Move 8 3
Jump 9 9
```

Problem 5 - Time Complexity & Recurrence (Hand-Written) (20 points)

Note: In this problem, if you use any theorem not covered by the lectures, slides, and the textbook, you should prove it first.

(1) **Asymptotic Notations** (10%)

Prove or *Disprove*.
If you think the statement is correct, you should provide a comprehensive proof. For example, to prove that a big-O bound is correct, you should find constants c and n_0 as defined in class.
If you think the statement is incorrect, you should disprove it or give a counterexample.

Note: The functions $f(n)$, $g(n)$ mentioned below are **non-negative** and **increasing**. Furthermore, you may assume that for any $t \in \mathbb{R}$, there exists n_t such that for $n \geq n_t$, $f(n) \geq t$ and $g(n) \geq t$.

(a) (2%) $\sqrt{n} = O(n^{\sin n})$
(b) (2%) if $f(n) = \Theta(g(n))$, then $\log(f(n)) = \Theta(\log(g(n)))$
(c) (3%) if $f_1(n) = O(g_1(n))$, and $f_2(n) = O(g_2(n))$, then $(f_1 \circ f_2)(n) = O((g_1 \circ g_2)(n))$
(d) (3%) $(n+a)^b = \Theta(n^b)$, where a, b are real constants with $b > 0$.

(2) **Solve Recurrences** (10%)

Give the tight bound (Θ-bound, *e.g.* $T(n) = \Theta(n^3)$) of the following recurrence equations.

Assume: $T(n) = 1, \forall n \leq 1$

Note: Show your derivation thoroughly.

(a) (2%) $T(n) = T(n-127) + \frac{127}{\log(n)}$
(b) (2%) $T(n) = T(\frac{n}{2}) + T(\frac{n}{4}) + T(\frac{n}{8}) + n\log n$
(c) (3%) $T(n) = 4T(\frac{n}{2}) + n\log(n)$
(d) (3%) $T(n) = \sqrt{n}T(\sqrt{n}) + n$

Problem 6 - Viennese Waltz (Hand-Written) (30 points)

*In problem 6, please **briefly** explain your solution in text. **Do not** use pseudo code, or you will receive penalty.*

As a lover of Viennese Waltz, Lily likes to dance with her husband in their lovely house. Since Lily and her husband are masters of ballroom dance, they found it boring to rotate around the room in the standard way. Instead, they always adjust their steps, of course, elegantly, so the tiles they have stepped on form a special rectangle which satisfies several constraints.

The floor of Lily's house can be viewed as an $n \times n$ board. They gave each tile an integer weight. The perimeter of a rectangle is defined as the number of tiles its edges pass, and the weight of a rectangle is equal to the sum of the weight of tiles on the perimeter. For example, the following graph represents a 5×5 board with the pink cells denoting the ones that occupied by the rectangle. In this example, the rectangle has perimeter of 10 and the weight of 3. Now, Lily has set up 4 challenges, and she wondered if you can find the rectangle that she and her husband will dance on.

1	-1	0	4	5
-1	0	1	2	3
-5	2	0	1	0
-5	10	1	4	3
4	2	6	7	-9

(1) (5%) Given k ($k \gg n$) rectangles on the board, each of which is described by the its upper left cell a and its lower right cell b, please give a $O(n^2 + k)$ algorithm to calculate each of their weights.

(2) (5%) Please give a $O(n^4)$ algorithm to find a rectangle with the maximum weight whose perimeter is no greater than a given constant L.

(3) (10%) Please give a $O(n^3)$ algorithm to find a rectangle with the maximum weight.

(4) (10%) Please give a $O(n^3)$ algorithm to find a rectangle with the maximum weight whose perimeter is no greater than a given constant L, and briefly explain the time complexity and correctness of your solution.
If you complete (4) correctly, you can skip problems (1)(2)(3) as you will receive the full score of problem 6 automatically. However, if you give a wrong algorithm in (4), and skip (1)(2)(3) at the same time, you may receive 0 for this problem.

(5) (15%) Bonus time!! If you develop an $o(n^3)$ algorithm for (4), or prove that the problem is of $\Omega(n^3)$ complexity, please come to the TA hour (Wed. 15:30 to 16:30) and discuss with the TA. If you can prove the correctness of your idea, then you will receive the bonus points! :) (Or, if you're not available during the scheduled TA hour, you're welcome to email your idea to the TAs, but you may not receive response if your solution is incorrect.)