Homework #1

Due Time: 2021/10/21 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.
- **Tips for programming problems.** Since the input files for some programming problems may be 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`.

Problem 1 - Tower of Hanoi (Programming) (10 points)

o

Problem Description

Tower of Hanoi is a classic mathematical game, which consists of three vertical pegs and several disks of different sizes. Robert, an extraordinarily handsome guy, is good at playing Tower of Hanoi optimally.
Once, he played the games repeatedly for a few days and nights until he was too tired and suddenly fell into sleep. After he woke up, he forgot what he had done for the current game. Moreover, he might even make some wrong moves.
You, a big fan of Robert, are eager to help him. Given the total number of disks (which can be from 1 to $998,244,353$) and the game's current state, please tell him whether he is on the right way. That is, there is an optimal solution containing the current state. Also, if he is on the right way, please tell him the minimum number of remaining steps to finish the game. Since the number of remaining steps might be extremely large, you should tell him the number modulo $998,244,353$.

Input

The first line of the input contains an integer n, indicating the number of disks in his current game. The disks are numbered from 1 to n (from small to big).
The following three lines contain $(k_i + 1)$ space-separated integers: the first integer k_i of the i^{th} line indicates the number of disks on the i^{th} peg, and the following k_i integers are the disks' numbers from top to bottom on this peg.

- $1 \leq n = \sum_{i=1}^{3} k_i \leq 100000$
- The input is guaranteed to be a valid state of the game.

Test Group 0 (0 %)

- Sample Input

Test Group 1 (20 %)

- $n \leq 10$

Test Group 2 (30 %)

- $n \leq 50$

Test Group 3 (20 %)

- The current state is on the right way.

Test Group 4 (30 %)

- No other constraints.

Output

If he had moved correctly, output an integer indicating the minimum number of remaining steps to finish the game modulo 998244353. Otherwise, output "-1" (without quotes).

Sample Input 1

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

Sample Output 1

```
4
```

Sample Input 2

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

Sample Output 2

```
-1
```

Sample Input 2

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

Sample Output 2

```
5
```

Problem 2 - Hellish Gag (Programming) (15 points)

Problem Description

The description is adapted from a true story.

Designing ADA Homework #1, the TA -10^{11} has come up with a dark story as a problem description. Because ADA is a course fulfilled with hopes and happiness, his depressing story is considered a hellish gag, and the TA is banished to the uttermost depths of the hell.

Upon noticing -10^{11}'s arrival, Hades, the king of the underworld, decides to grant him an opportunity to resurrect. "If you are able to solve this algorithm problem, you will get a chance to be back to the world.", said Hades, "Here comes the problem. 99% deceased cannot solve it..."

Given 2 arrays p, z of length N and 3 constants a, b, c. Let d_i be the number of j's satisfying all of the following conditions:

- $j \in \{1, 2, \ldots, N\}$
- $j \neq i$
- $p_j > \frac{b}{a} \cdot p_i + \frac{c}{a}$
- $z_j > z_i$

Please find the summation of d_i; that is, $\sum_{i=1}^{N} d_i$.

However, -10^{11} is getting pale and turns unable to solve the problem. To ask for your help, he tells you this story by setting the exact problem in your Homework #1. Please save -10^{11} from the hell (so he could get back to the world and set another problem in your next ADA Homework).

Input

The first line of the input contains 4 numbers N, a, b, c, denoting the the size of 2 arrays and the 3 constants used in the condition description.

Then, N lines follow, the i-th of which contains two non-negative integers, p_i and z_i, denoting the entries of two arrays.

- $1 \leq N \leq 2 \times 10^6$
- $1 \leq a \leq 10^9$
- $0 \leq b \leq 10^9$
- $-10^9 \leq c \leq 10^9$
- $0 \leq p_i, z_i \leq 10^9, \forall i = 1, 2, \ldots, N$

Test Group 0 (0 %)

- Sample Input

Test Group 1 (10 %)

- $N \leq 2000$

Test Group 2 (5 %)

- $b = 0$

Test Group 3 (15 %)

- $(a, b, c) = (1, 1, 0)$
- $[z_1, z_2, \ldots, z_N]$ is a permutation of $[0, 1, \ldots, N-1]$.

Test Group 4 (30 %)

- All z_i's are distinct.

Test Group 5 (40 %)

- No additional constraint.

Output

Output 1 integer, the summation $\sum_{i=1}^{N} d_i$.

Sample Input 1

```
4 1 0 0
0 0
0 0
0 0
0 0
```

Sample Input 2

```
4 1 1 2
1 47
3 22
7 81
5 65
```

Sample Input 3

```
4 2021 0 1111111
123 4
567 8
901 2
345 6
```

Sample Output 1

```
0
```

Sample Output 2

```
3
```

Sample Output 3

```
3
```

Explanation

- In the first test case, $(d_1, d_2, d_3, d_4) = (0, 0, 0, 0)$, so the required is $0 + 0 + 0 + 0 = 0$.
- In the second test case, $(d_1, d_2, d_3, d_4) = (2, 1, 0, 0)$, so the required is $2 + 1 + 0 + 0 = 3$.
- In the third test case, $(d_1, d_2, d_3, d_4) = (1, 0, 1, 1)$, so the required is $1 + 0 + 1 + 1 = 3$.

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

Problem Description

Given N points $(p_1, p_2 \cdots, p_n)$ on a 2-dimensional (2D) plane, we define two points p_i and p_j as a good pair if there exists a rectangle satisfying the following conditions:

1. The sides of the rectangle are parallel to the X-axis and Y-axis.
2. The rectangle contains these two points p_i, p_j only; no other points fall within this rectangle.

How many good pairs of points are there in total?

Input

The first line of the input contains one integer N, denoting the number of points.

The i^{th} of the following N lines contains two integers x_i, y_i, indicating the coordinates of the i^{th} point.

Test Group 0 (0 %)

- Sample Input.

Test Group 1 (20 %)

- $1 \leq N \leq 5 \times 10^2$
- $1 \leq x_i \leq N$
- $1 \leq y_i \leq N$
- All x_i are distinct.
- All y_i are distinct.

Test Group 2 (35 %)

- $1 \leq N \leq 3 \times 10^3$
- $1 \leq x_i \leq N$
- $1 \leq y_i \leq N$
- All x_i are distinct.
- All y_i are distinct.

Test Group 3 (45 %)

- $1 \leq N \leq 2 \times 10^5$
- $1 \leq x_i \leq N$
- $1 \leq y_i \leq N$
- All x_i are distinct.
- All y_i are distinct.

Output

Print the number of good pairs in a single line.

Sample Input 1

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

Sample Output 1

```
8
```

Sample Input 2

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

Sample Output 2

```
4
```

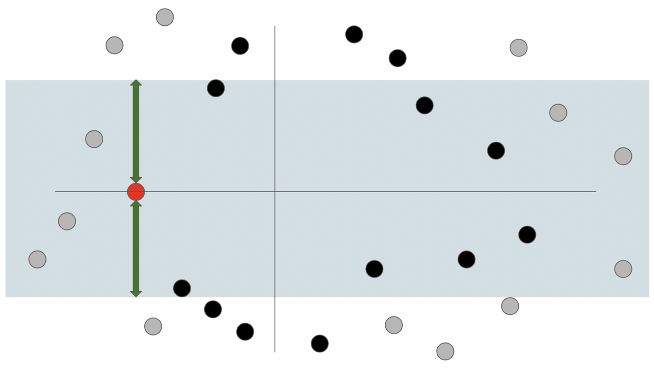
Sample Input 3

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

Sample Output 3

```
3
```

Hint



1. The good pairs in Sample Input 1 are (p_1, p_2), (p_1, p_3), (p_1, p_5), (p_2, p_4), (p_2, p_5), (p_3, p_4), (p_3, p_5), (p_4, p_5).
2. You are also encouraged to use the above picture to find inspiration. When conquering, you can use stacks to maintain the black points, and use binary search to identify how many black points are in the gray scope.
3. You can maintain a stack on each of the left and right sides to keep the "needed points", which are the points blocking the current point or forming a good pair with the current point you are focusing on.
4. Instead of std::stack, you can use std::vector to maintain stack and perform binary searches on std::vector directly using std::lower_bound.
5. GL & HF (Good Luck and Have Fun).

Problem 4 - Candies (Programming) (10 points)

Problem Description

Baluteshih brings N candies to his friend, Waynetu. Those candies are lined on the table. Each candy has its own sweetness indicated by an integer. The sweetness of the candies from left to right are a_1, a_2, …, a_N, respectively.

Baluteshih assigns Waynetu an interesting mission. He asks Waynetu to remove some candies from the table, so that the remaining candies on the table are *alternative*. Formally, we say that a sequence of candies with sweetness b_1, b_2, …, b_k is *alternative* if $b_i \times b_{i+1} \leq 0$ holds for all $1 \leq i < k$.

With the above rule, Waynetu hopes to maximize the sum of the sweetness of the candies on the table. Please help Waynetu find the maximum possible sum of the remaining candies' sweetness and provide a solution to reach the optimal case.

Input

The first line contains two integers T, representing the number of testcases, and $flag$ ($flag \in \{0, 1\}$), which will be described in the output section.

Each testcase includes two lines: the first line contains an integer N ($1 \leq N \leq 10^5$), and the second line contains N integers a_1, a_2, …, a_N ($|a_i| \leq 10^9$).

It is guaranteed that the sum of N does not exceed 10^5.

Output

For each testcase, please print an integer representing the maximum possible sum of the sweetness in the first line. If $flag$ equals 1 mentioned in the input section, please furthermore print out one optimal way in the second line: an integer k representing the number of candies remaining on the table, followed by k integers i_1, i_2, …, i_k, representing the indices of candies.

Test Group 0 (0 %)

- Sample Input.

Test Group 1 (20 %)

- $flag = 0$.
- $\sum N \leq 1000$.

Test Group 2 (50 %)

- $flag = 0$.

Test Group 3 (10 %)

- $flag = 1$.
- $\sum N \leq 1000$.

Test Group 4 (20 %)

- $flag = 1$.

Sample Input 1

```
1 0
5
3 -1 6 -7 4
```

Sample Output 1

```
8
```

Sample Input 2

```
2 0
3
1 2 3
4
1 -2 3 -4
```

Sample Output 2

```
3
3
```

Sample Input 3

```
3 1
1
-1
3
5 0 1
4
-1 -2 -3 -4
```

Sample Output 3

```
-1
1 1
6
3 1 2 3
-1
1 1
```

Hint

1. Since each input includes several independent testcases, please carefully clear all results of the current testcase before dealing with the next testcase.

Problem 5 - Time Complexity & Recurrence (Hand-Written) (25 points)

Note: In this problem, if you use any theorem not covered by the lectures, slides, and the textbook, you should prove it first.
Note: In this problem, you may assume the base, b, for logarithm has constraints $b \in \mathbb{R}, b > 1$.

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

Prove or *Disprove*.
If you think the statement is correct, you should provide a comprehensive proof.
If you think the statement is incorrect, you should disprove it or give a counterexample.

(a) (2%) $\ln n! = O(\ln n^n)$
(b) (2%) $n^{\ln c} = \Theta(c^{\ln n})$
(c) (3%) $\sqrt{n} = O(n^{\sin n})$
(d) (3%) $(\ln n)^3 = o(n)$

(2) **Solve Recurrences** (15%)

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

Assume: $T(n) = 1, \forall n \leq 2$

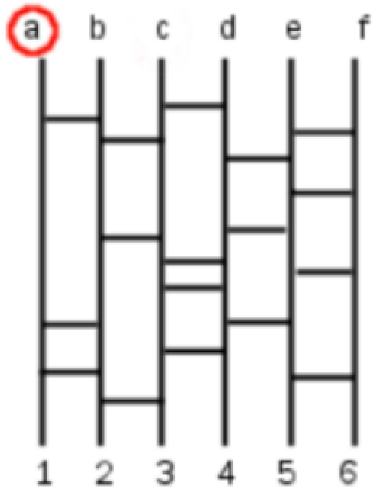
Note: Show your derivation thoroughly by using the assigned method.

(a) (2%) $T(n) = 2T(n-1) + 1$
Please use **brute force** method.
(b) (4%) $T(n) = T(\frac{n}{2}) + T(\frac{n}{4}) + T(\frac{n}{8}) + n \log n$
Please use **recursion tree** method.
(c) (4%) $T(n) = 4T(\frac{n}{2}) + n \log(n)$
Please use **master theorem** method.
(d) (5%) $T(n) = \sqrt{n}T(\sqrt{n}) + n$
Please use **variable transformation** method.

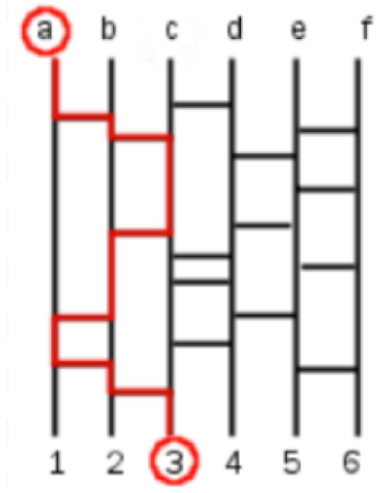
Problem 6 - Ghost Leg (Hand-Written) (15 points)

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

"Ghost Leg", known in Japan as Amidakuji ("Amida lottery" because the paper was folded into a fan shape resembling Amida's halo), is a method of lottery designed to create random pairings between two sets of the same number of prizes. This is often used to distribute prizes to people, where the number of prizes is the same as the number of people. Ghost Leg consists of vertical lines with horizontal lines connecting two adjacent vertical lines; the horizontal lines are called "legs". Note that "legs" cannot touch other legs. The number of vertical lines equals the number of people, and at the bottom of each line there is an item - a prize that will be paired with a player. The general rule for this game is: choosing a line on the top, and following it downwards. When a horizontal line is encountered, the player needs to transit to another vertical line and then continue going down. The procedure is repeated until reaching the end of the vertical line, and then the player gets the corresponding prize. Therefore, choosing the line decides which prize you get. You can refer to the link for more details: `https://en.wikipedia.org/wiki/Ghost_Leg`.



The example of one ghost leg.



Example of finding the prize if A is chosen.

Alan is a party host, and he distributes N prizes to N players by a ghost leg. The party attendees can get a gift based on the game result. However, Alan is not as generous as others think. Instead, he is a cunning person. Alan does not want to give any precious presents to attendees at all, so he assigns his employees to attend the party to collect good gifts and leave consolation prizes to other ignorant players.

You, as Alan's personal assistant, please help him manipulate the gift distribution result. You have to design an efficient algorithm for this black-box trick to give all good prizes to the designated people. The specific start must reach the planned destination as expected.

Here comes a problem: Now, given a ghost leg board without any horizontal lines, you have k constraints, which assign k starting points with their corresponding prizes. What is the minimum number of horizontal lines to be added in order to satisfy all k constraints?

For example, you have a ghost leg without any horizontal lines as Fig. 1, and the constraints are (c should go to 1), (b should go to 4). Thus, the answer should be 3, and one of the solutions is illustrated in Fig. 2.

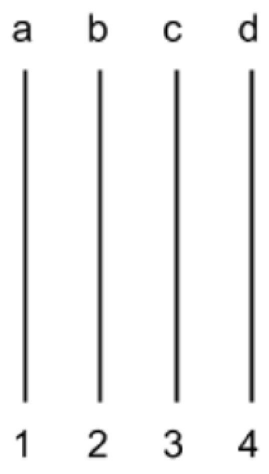


Fig 1. An empty ghost leg if $N = 4$

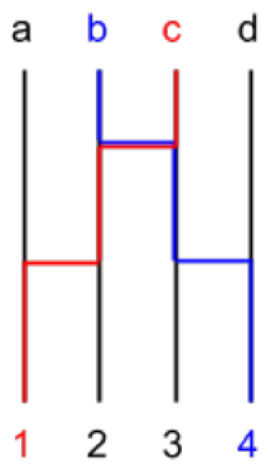


Fig 2. A solution meets all the constraints

To help you solve the problem, we define the property of "inversion".

Inversion: Given a sequence of unique numbers $B = b_1, b_2, ..., b_n$, an inversion is a pair of numbers b_i and b_j in this sequence such that $i < j$ and $b_i > b_j$. Let $I(B)$ be the set of inversions in B. For example, if $B = 1, 3, 5, 2$, $I(B) = \{(3, 2), (5, 2)\}$, and the number of inversions is 2 ($|I(B)| = 2$).

Please answer the questions below.

(1) (4pt) Given an unsorted and unique sequence B with n numbers, please design an efficient algorithm to calculate the number of inversions $|I(B)|$ in $O(N \log N)$ time.

(2) (3pt) Explain why your algorithm runs in $O(N \log N)$.

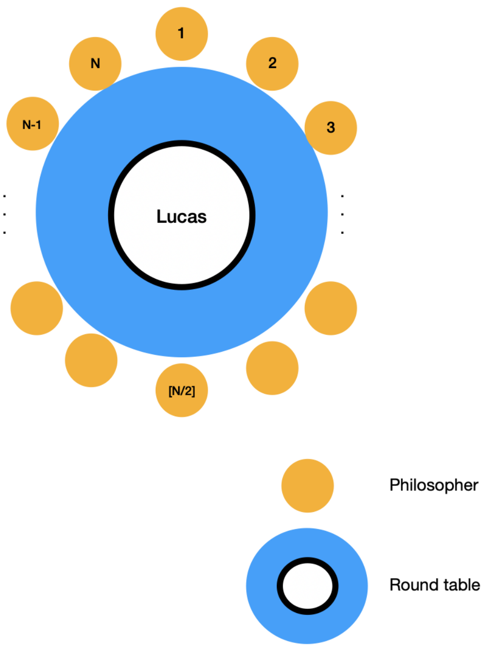
(3) (2pt) Prove that the number of exchanges when performing bubble sort on the sequence S equals the number of inversions in S.

(4) (4pt) Describe an $O(N \log N)$ algorithm that calculates the minimum number of horizontal lines in the ghost leg when $|constraints| \leq N$.

(5) (2pt) Prove the correctness of your algorithm in the previous problem.

Problem 7 - Unfair Lucas (Hand-Written) (10 points)

Far Far Away Kingdom has a custom that all philosophers gather at a round table to keep their friendship, and they hire a chef to prepare some dishes for them. Supposed that the round table is big enough to fit N philosophers, and their indices are shown in the picture below. Lucas is a chef standing in the center of the round table, and he needs to prepare some dishes for all philosophers.



Because Lucas is not familiar with all philosophers, he decides to prepare some tasty dishes for the philosophers who are friendly and awful dishes for other philosophers. However, Lucas is afraid of being accused of his unfair treatment, so he decides to choose a *contiguous* part of philosophers to get tasty dishes, such that the possibility of being charged disparate treatment is minimized. (Note: The contiguous part of philosophers means that you can choose the philosophers whose indices are $n, 1, 2, 3$ to eat tasty dishes, but you cannot choose the philosophers whose indices are $n, 1, 3, 4$ because of skipping the 2^{nd} philosopher.)

Given the friendliness value f_i for $i = 1, 2, 3, \ldots, N$, of the i^{th} philosopher, please help Lucas find out a contiguous part of philosophers such that the total friendliness value of those chosen philosophers is maximized. Moreover, he hopes that at least one philosopher can have tasty dishes.

(1) (2%) Consider 10 philosophers and their friendliness values $f = [-3, 0, 6, 4, 0, -1, -2, 3, -3, 9]$. What is the maximum total of friendliness values?

(2) (3%) Please design an algorithm with both time complexity and space complexity in $O(N)$ in the worst case. Briefly explain your algorithm and prove its time complexity.

(3) (5%) Now, Lucas wants to make his friends happier, so he decides to skip *at most* one philosopher in the contiguous part to give tasty dishes such that the maximum total friendliness value can be increased. After applying this policy, please design an algorithm with both time complexity and space complexity in $O(N)$ in the worst case. Briefly explain your algorithm and prove its time complexity.