2025 Fall CSC6001 Assignment 2

Problem 1

Description

You are asked to perform a **join operation** between two tables, A and B, in a **distributed database system**.

- Table A is distributed across m machines in the first cluster. The i-th machine stores a_i rows from table A.
- Table B is distributed across n machines in the second cluster. The j-th machine stores b_j rows from table B.

In **one operation**, you may **copy one row** from any machine to any other machine — even across clusters. That is, you can duplicate a row and place it into any chosen machine. After all operations are completed, you must ensure that for every row in A and every row in B, there exists at least one machine that contains both of them. In other words, for every pair (x, y) where $x \in A$ and $y \in B$, there must exist a machine that stores both x and y. Compute the **minimal number of copy operations** required to achieve this goal.

Example.

- Table A is distributed over $m = 2$ machines: $a = [3, 2]$ (machine 1 stores 3 rows, machine 2 stores 2 rows).
- Table B is distributed over $n = 3$ machines: $b = [4, 1, 2]$ (machines store 4, 1, and 2 rows respectively).

Then

$$A_{\text{total}} = 3 + 2 = 5, \quad B_{\text{total}} = 4 + 1 + 2 = 7.$$

Input Format

The first line contains two integers m and n. The second line contains m integers $a_1 \cdots a_m$, split by space. The third line contains n integers $b_1 \cdots b_n$, split by space.

Output Format

Output one integer representing the minimal number of operations required.

Sample Input 1

```
2 2
3 7
50 2
```

Sample Output 1

```
12
```

Explanation 1

In the first example, it makes sense to move all the rows to the second machine of the second cluster, which is achieved in $2 + 6 + 3 = 11$ operations.

Sample Input 2

```
2 3
10 10
1 1 1
```

Sample Output 2

```
6
```

Explanation 2

In the second example, you can copy each row from B to both machines of the first cluster, which needs $2 \cdot 3 = 6$ copy operations.

Sample Input 3

```
3 4
337369924 278848730 654933675
866361693 732544605 890800310 350303294
```

Sample Output 3

```
3220361921
```

Explanation 3

The optimal plan is to merge all rows of B into its largest machine (with 890,800,310 rows) and copy all rows of A there. That needs 3,220,361,921 operations.

Data Constraints

- For 20% of the test cases, $1 \leq m, n \leq 10$
- For 40% of the test cases, $1 \leq m, n \leq 10^3$
- For 100% of the test cases, $1 \leq m, n \leq 10^5, 1 \leq a_i, b_i \leq 10^9$

Problem 2

Description

You are trapped on a chessboard designed by a god, and you must find your way out. The chessboard is an $n \times n$ grid. The god has placed exactly k obstacles on k distinct squares ($k \leq 10$). You cannot step on a square that contains an obstacle, but all other squares are accessible. You start at the **top-left corner** $(1, 1)$ and aim to reach the **bottom-right corner** (n, n). At each step, you may move **only one square** either **down** or **right**. (You cannot move left, up, or diagonally.) Your task is to determine the **number of distinct paths** from $(1, 1)$ to (n, n) that do **not** pass through any obstacle. Since the answer may be large, output it **modulo** 998244353.
Input

- The first line contains two integers n and k ($1 \leq n \leq 10^9$, $0 \leq k \leq 10$).
- The next k lines each contain two integers x_i, y_i ($1 \leq x_i, y_i \leq n$) — the coordinates of the obstacles.

It is guaranteed that all obstacles are at **distinct** positions and that $(1, 1)$ and (n, n) are **not blocked**.

Output

Print a single integer — the number of valid paths from $(1, 1)$ to (n, n) **modulo** 998244353.

Input Format

- The first line contains two integers n and k, representing the size of the chessboard and the number of obstacles, respectively.
- The next k lines contain two integers x and y, representing the coordinates of each obstacle.

Output Format

Output one integer, representing the number of valid paths modulo 998244353.

Sample Input 1

```
3 1
2 2
```

Sample Output 1

```
2
```

Explanation 1

Without obstacles, there are $\binom{4}{2} = 6$ paths from $(1, 1)$ to $(3, 3)$. Two of these paths go through the obstacle at $(2, 2)$, so only $6 - 4 = 2$ valid paths remain.

Sample Input 2

```
4 2
2 2
3 3
```

Sample Output 2

```
4
```

Explanation 2

Without obstacles, there are $\binom{6}{3} = 20$ paths from $(1, 1)$ to $(4, 4)$. After removing all paths that pass through either $(2, 2)$ or $(3, 3)$, exactly 4 valid paths remain.

Sample Input 3

```
7 3
1 4
5 3
3 6
```

Sample Output 3

```
540
```

Explanation 3

Among all $\binom{12}{6} = 924$ possible paths from $(1,1)$ to $(7,7)$, some intersect with the obstacles at $(1,4)$, $(5,3)$, or $(3,6)$. After excluding those blocked paths, 540 valid paths remain.

Data Constraints

- For 50% of the test cases, $1 \leq n \leq 10, 0 \leq k \leq 2$.
- For 80% of the test cases, $1 \leq n \leq 10^3, 0 \leq k \leq 10$.
- For 100% of the test cases, $1 \leq n \leq 10^7$, $0 \leq k \leq 10$, and $k \leq n \times n$.

Problem 3

You are given an array $a_0, a_1, \ldots, a_{n-1}$ consisting of n integers. You may choose **exactly one continuous subarray** and reverse it, or choose not to reverse any subarray. Formally, you may choose indices l and r $(0 \leq l \leq r < n)$ and reverse the segment $a_l, a_{l+1}, \ldots, a_r$ into $a_r, a_{r-1}, \ldots, a_l$.

Your goal is to make the sum of elements at **even positions** (i.e., indices $0, 2, 4, \ldots$) as large as possible after performing at most one reversal. You need to calculate this **maximum possible sum**.

Example for illustration. Suppose the array is $[1, 3, 2, 4, 1]$. If you reverse the subarray $[1, 3, 2, 4]$, the array becomes $[4, 2, 3, 1, 1]$, and the sum on even positions is $4 + 3 + 1 = 8$. This is the maximum possible value.

Input Format

- The first line contains one integer n $(1 \leq n \leq 2 \times 10^5)$.
- The second line contains n integers $a_0, a_1, \ldots, a_{n-1}$, separated by spaces.

Output Format

Output one integer — the maximum possible sum of elements on even positions after at most one reversal operation.

Sample Input 1

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

Sample Output 1

```
8
```

Explanation 1

Reversing the subarray $\{a_0, a_1, a_2, a_3\}$ gives the array $[4, 2, 3, 1, 1]$, where the even-index elements are $4, 3, 1$, summing to 8. It can be verified that 8 is the maximum possible value.

Sample Input 2

```
4
2 9 1 5
```

Sample Output 2

```
10
```

Explanation 2

Initially, the even-index elements are 2 and 1, giving a sum of 3. If we reverse the subarray $\{a_1, a_2, a_3\}$, the array becomes $[2, 5, 1, 9]$, and the even-index elements are 2 and $1 = 3$. But if we reverse $\{a_0, \cdots, a_3\}$ (the whole array), we get $[5, 1, 9, 2]$, whose even-index elements are $5 + 9 = 14$. Thus, the maximum possible sum is 14.

Sample Input 3

```
6
5 4 3 2 1 6
```

Sample Output 3

```
14
```

Explanation 3

Initially, the even-index elements are $5, 3, 1$, giving a sum of 9. Reversing the subarray $\{a_1, \cdots, a_5\}$ yields $[5, 6, 1, 2, 3, 4]$, and now the even-index elements are $5, 1, 3$, summing to 9. However, reversing $\{a_2, \cdots, a_5\}$ gives $[5, 4, 6, 1, 2, 3]$, with even-index elements $5, 6, 2 = 13$. The optimal choice is to reverse $\{a_0, \cdots, a_5\}$, producing $[6, 1, 2, 3, 4, 5]$, whose even-index elements are $6 + 2 + 4 = 12$. After checking all possible reversals, the best achievable sum is 14.

Data Constraints

- For 20% of the test cases, $1 \le n \le 500$
- For 40% of the test cases, $1 \le n \le 5 \times 10^3$
- For 100% of the test cases, $1 \le n \le 2 \times 10^5, 1 \le a_i \le 10^9$

Problem 4

Description

You manage a cargo port where multiple ships come to load goods. The i-th ship requires n_i days to complete loading within a certain period, from the x_i-th day to the y_i-th day. The loading days do not need to be continuous. Given a loading plan for L ships, you are required to find the minimum number of days the port needs to be open to accommodate all loading operations.

Input Format

1. The first line contains one integer L.
2. The remaining L lines represent loading arrangements for all the ships. In each line, x_i, y_i, n_i are included from left to right, splited by space.

Output Format

An integer representing to the minimum number of days the port needs to be open.

Sample Input 1

```
3
3 4 1
7 8 1
2 8 2
```

Sample Output 1

```
2
```

Explanation 1

In this example, the optimal arrangement is to load the first ship on day 4, the second ship on day 7, and the third ship on days 4 and 7. Therefore, the port only needs to be open for 2 days.

Sample Input 2

```
1
8 8 1
```

Sample Output 2

```
1
```

Explanation 2

In this example, there is only one ship that needs to be loaded on day 8, and it requires 1 loading day. Thus, the port only needs to be open for 1 day — specifically, day 8.

Sample Input 3

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

Sample Output 3

```
3
```

Explanation 3

In this example, an optimal schedule is to open the port on days 2, 4, and 6.

- The first ship loads on days 2 and 4 (within its period 1–3).
- The second ship loads on days 2 and 4 (within 2–5).
- The third ship loads on day 6 (within 4–6).
- The fourth ship loads on day 6 (within 6–7).

Thus, all ships can complete their loading with the port open for only 3 days.

Constraints

- $1 \leq n_i \leq y_i - x_i + 1$
- Case 1: $1 \leq L \leq 10$, $1 \leq x_i \leq y_i \leq 10$
- Case 2-4: $10 \leq L \leq 100$, $10 \leq x_i \leq y_i \leq 100$
- Case 5-7: $100 \leq L \leq 1000$, $100 \leq x_i \leq y_i \leq 1000$
- Case 8-10: $1000 \leq L \leq 10000$, $1000 \leq x_i \leq y_i \leq 10000$