SDE TRAINING PROGRAMMING TASK-4

Task-1

While other heroes from Marvel and DC Universe have resorted to competing in duels to establish their supremacy, Batman and Dr. Strange decided to find the winner by playing a stratergy board game. The game consists of a **N x M** board that has either white or black stones in each cell. The game is simple, each player is allowed to select a **single** white stone anywhere on the board, and he owns that stone. However, he is also granted the ownership of every white stone present in the immediate adjacent cell of the current cell, in all directions. Repeat this for all white stones that a player owns. Cells having black stones cannot be owned. A player cannot change the configuration of the board. The rules are elementary, whosoever owns maximum white stones using the rules stated above, is declared the winner. Your task is to help Batman (Yeah right !! Batman needs your help !) in calculating the maximum number of stones he can own, if he starts first.

Input

The first line contains a positive integer **t**, denoting the number of test cases.
The first line of each test case contains two positive integers **N and M**, denoting the number of rows and coloumns on the board respectively.
Each line *i* of the **n** subsequent lines contain **m** space separate integers denoting i^{th} row of the board where each cell has a value of either 0 or 1, where 0 denotes a black stone and 1 denotes a white stone.

Output

For each test case, print a single positive integer denoting the maximum number of stones Batman can own if he starts first.

Constraints

- **1 ≤ t ≤ 250**
- **1 ≤ N,M ≤ 200**

Example

`Input:`

```
1

5 5

0 0 1 1 0

0 1 0 0 0

1 1 1 0 0

0 0 0 0 1

1 0 0 0 0
```

Output:

```
6
```

Explanation

Example case 1. The board configuration above has 3 regions with maximum white stones as 6,1 and 1 respectively. Hence if the batman starts from the cell (1,3) he can own a maximum of 6 stones

TASK-2

A real estate company undertakes to build a large township. The work is complex and requires the company to hire different categories of highly skilled laborers at different times for different parts of the work. An analysis of the whole project showed that **K** different groups of highly skilled laborers are needed for **K** subparts of the project, each of which takes a certain number of days. The subparts are numbered from 1 to **K**.

The whole work is supervised by a consultant of the company and he can supervise only one subpart at a time, so only one group of skilled laborers is actually utilized at one time. However, it is very difficult to find skilled labor just when you need them, so in order to avoid delays due to non-availability of required labor when it is needed, the company decides to hire all skilled laborers at the start of the project itself, paying them from the beginning of the project till their subpart is done, at which time, they are let go. Each such group costs the company a certain amount of money per day from the day they are hired till the day they finish their part.

The situation is compounded by the fact that some of these subparts have to be done in order. However, it is seen that any subpart can be dependent on at most one other subpart and can cause at most one other subpart to be dependent on it.

What is the minimum total amount the company has to pay to the laborers?

Input

The first line contains the number of test cases **N**.

For each test case, first line contains the number of subparts **K**. Second line contains a sequence of **K** integers indicating the number of days required to finish each subpart. Third line contains a sequence of **K** integers indicating the hiring cost per day for the labor group needed for each subpart. The fourth line contains an integer **D** indicating the number of pairs of subparts where one subpart is dependent on the other. This is followed by **D** lines, each of which contains two integers **x** and **y** (in that order), where subpart **y** depends on subpart **x** (i.e., **y** can start only after **x** has finished).

Output

For each test case, print the case number, followed by a colon, followed by a single space, followed by a single integer indicating the total labor cost paid by the company.

Constraints

0 < **N** ≤ 3

0 < **K** ≤ 200

Example

`Input:`

```
2

5

21 27 12 33 5

100 200 50 25 75

3

1 2

3 5

5 4

4
```

```
10 10 10 10

40 30 20 10

0
```

Output:

```
Case 1: 21125

Case 2: 2000
```

TASK-3

Mr. Yagami is playing a game of arrays. He is given two random arrays **A** and **B** consisting of **N** positive integer elements. Game starts by Mr. Yagami assigning **0 or 1** to each element in **A and B**.

After this assignment is done, a graph is constructed with a node for each element in the array A and B (**2N** nodes) and no edges. The game proceeds with a valid move being defined in the following way:

- One of the arrays from **A** or **B** is selected. From the selected array, an element which has been marked **0** is chosen. Let us call this element as **X**.
- A set of elements, **Y**, are chosen from the array, which was not chosen in the first step, such that all elements of **Y** should be marked as **1** and all elements of **Y** should be greater than **X** and no element of **Y** should be coprime to **X**.
- Finally an edge is drawn from the node labelled **X** to all the nodes corresponding to the elements in set **Y**. There can only be a single edge between any **2** nodes in the graph.

He can make as many valid moves. Mr. Yagami receives **1** point for each edge that is drawn in the graph.

Mr. Yagami is very clever, so he makes the initial assignment in such a way that it maximizes the number of points he receives in the game. You have to return the maximum number of points that Mr. Yagami can receive.

Input Format:

The first line of the input contains a single integer, **N (1 ≤ N ≤ 40)**
The second line of input contains **N** integers separated by a single space character, which represent the elements of the array **A. (2 ≤ A[i] ≤ 10^9)**
Similarly, the last line of input also contains **N** integers separated by a single space character, which represent the elements of the array **B. (2 ≤ B[i] ≤ 10^9)**

Output Format:

A single integer representing the maximum score which Mr. Yagami can receive.

Sample Input:

```
4

16 3 2 9

12 18 13 4
```

Sample Output:

```
8
```

Explanation:

He picks 2 from first array. So he gets to put 3 edges ie. 2->4, 2->12, 2->18.
Next he picks 3 from the first array. So he gets to put 2 edges ie. 3->12, 3->18.
Next he picks 9 from the first array. So he gets to put 2 edges ie. 9->12, 9->18.
Next he picks 16 from the first array. So he gets to put 1 edge ie. 16->18. Total edges=8.

TASK-4

Anakin's spacecraft is being attacked by Buzz droids. Again. Buzz droids intend to disable rather than destroy their targets. They achieve this by introducing bugs in the program of the underlying software.

One important module of every navigation controller is the Floyd-Warshall algorithm. Here is how the code for Floyd-Warshall looks like:

```
n <- number of vertices in the graph

adj[n][n] <- adjacency matrix(0/1) of a simple undirected graph

connected[n][n] <- 2D matrix of size n * n

for (i=1; i<=n; i++)
```

```
  for (j=1; j<=n; j++)

    connected[i][j] = adj[i][j]



for (p=1; p<=n; p++)    // runs only up to n-k after reprogramming

  for (i=1; i<=n; i++)

    for (j=1; j<=n; j++)

      connected[i][j] = (connected[i][j])

                        || (i!=j && connected[i][p] && connected[p][j])



return connected
```

Due to reprogramming by Buzz droids, now the outer loop on ***p*** runs only from ***1*** to ***n-k*** (both inclusive). Your task is to find the number of *simple undirected unweighted graphs* for which the reprogrammed code still returns the correct answer. Two graphs are considered different if their adjacency matrices are not identical.

Input

The first line of the input contains an integer **T** denoting the number of test cases. The description of **T** test cases follows. Each test case consists of single line containing two integers, **n** and **k**.

Output

For each test case, output a single line containing the number of graphs (modulo 10^9+7) for which the buggy code returns correct answer.

Constraints

- **1 ≤ T ≤ 17000**
- **1 ≤ n ≤ 180**
- **0 ≤ k ≤ n**

Example

Input:

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

Output:

```
1
2
7
```

Explanation

All graphs of size 1 or 2 produce correct result because the matrix does not change during any iteration. For **n=3, k=1**, out of a total of 8 graphs, there is only one graph for which the result is wrong: The one with the edge set {{1,3} , {2,3}}.