Road Trips and Museums

Lavanya and Nikhil have **K** months of holidays ahead of them, and they want to go on exactly **K** road trips, one a month. They have a map of the various cities in the world with the roads that connect them. There are **N** cities, numbered from 1 to **N**. We say that you can reach city B from city A if there is a sequence of roads that starts from city A and ends at city B. Note that the roads are bidirectional. Hence, if you can reach city B from city A, you can also reach city A from city B.

Lavanya first decides which city to start from. In the first month, they will start from that city, and they will visit every city that they can reach by road from that particular city, even if it means that they have to pass through cities that they have already visited previously. Then, at the beginning of the second month, Nikhil picks a city that they haven't visited till then. In the second month, they first fly to that city and visit all the cities that they can reach from that city by road. Then, in the third month, Lavanya identifies a city, and they fly there and visit all cities reachable from there by road. Then in the fourth month it is Nikhil's turn to choose an unvisited city to start a road trip, and they alternate like this. Note that the city that they fly to (that is, the city from where they start each month's road trip) is also considered as being visited.

Each city has some museums, and when they visit a city for the first time, Lavanya makes them visit each of the museums there. Lavanya loves going to museums, but Nikhil hates them. Lavanya always makes her decisions so that they visit the maximum number of museums possible that month, while Nikhil picks cities so that the number of museums visited that month is minimized.

Given a map of the roads, the number of museums in each city, and the number **K**, find the total number of museums that they will end up visiting at the end of **K** months. Print -1 if they will have visited all the cities before the beginning of the $\mathbf{K}^{th}$ month, and hence they will be left bored at home for some of the **K** months.

Input

- The first line contains a single integer, **T**, which is the number of testcases. The description of each testcase follows.
- The first line of each testcase contains three integers: **N**, **M** and **K**, which represents the number of cities, number of roads and the number of months.
- The i^{th} of the next **M** lines contains two integers, $\mathbf{u_i}$ and $\mathbf{v_i}$. This denotes that there is a direct road between city $\mathbf{u_i}$ and city $\mathbf{v_i}$.
- The next line contains **N** integers, the i^{th} of which represents the number of museums in city i.

Output

For each test case, if they can go on **K** road trips, output a single line containing a single integer which should be the total number of museums they visit in the **K** months. Output -1 if they can't go on **K** road trips.

Constraints

- $1 \leq \mathbf{T} \leq 3$
- $1 \leq \mathbf{N} \leq 10^6$
- $0 \leq \mathbf{M} \leq 10^6$
- $1 \leq \mathbf{K} \leq 10^6$
- $1 \leq \mathbf{u_i}, \mathbf{v_i} \leq \mathbf{N}$

- There is no road which goes from one city to itself. ie. $\mathbf{u_i} \neq \mathbf{v_i}$.
- There is at most one direct road between a pair of cities.
- 0 ≤ Number of museums in each city ≤ 1000
- Sum of **N** over all testcases in a file will be ≤ $1.5 * 10^6$

Subtasks

- **Subtask 1** (11 points): **M** = 0
- **Subtask 2** (21 points): Each city has at most two roads of which it is an end point. That is, for every i, there are at most two roads (u, v) in the input, such that u = i or v = i.
- **Subtask 3** (68 points): Original constraints.

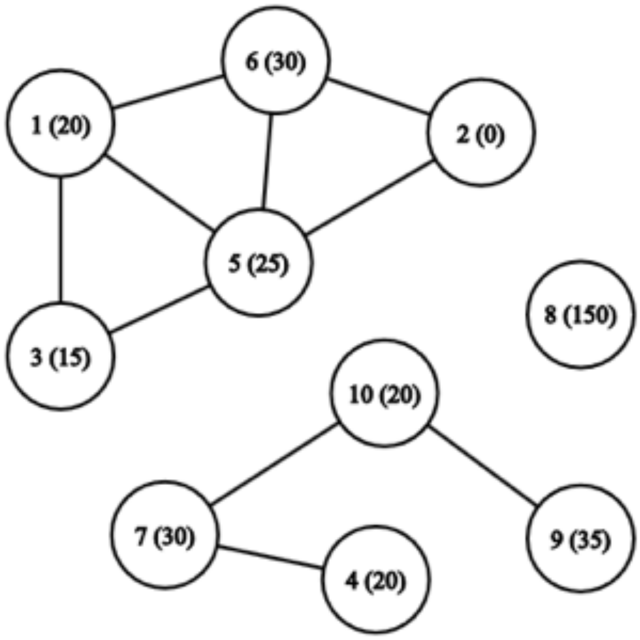
Example

```
Input:
3
10 10 3
1 3
3 5
5 1
1 6
6 2
5 6
2 5
7 10
4 7
10 9
20 0 15 20 25 30 30 150 35 20
10 10 2
1 3
3 5
5 1
1 6
6 2
5 6
2 5
7 10
4 7
10 9
20 0 15 20 25 30 30 150 35 20
10 10 5
1 3
3 5
5 1
1 6
6 2
5 6
2 5
7 10
4 7
10 9
20 0 15 20 25 30 30 150 35 20

Output:
345
240
-1
```

Explanation

Notice that in all the three testcases, everything is the same, except for the value of **K**. The following figure represents the road map in these testcases. Each node denotes a city, with a label of the form "*n* (*m*)", where *n* is the city number, between 1 and **N**, and *m* is the number of museums in this city. For example, the node with label "5 (25)" represents city 5, which has 25 museums.



Testcase 1: Lavanya will first choose to fly to city 8. In the first month, they visit only that city, but they visit 150 museums.

Then in the second month, Nikhil could choose to fly to city 3, and they visit the cities 1, 2, 3, 5 and 6, and visit 20 + 0 + 15 + 25 + 30 = 90 museums that month. Note that Nikhil could have instead chosen to fly to city 1 or 2 or 5 or 6, and they would all result in the same scenario.

Then, Lavanya could choose city 7, and in the third month they will visit the cities 7, 4, 10 and 9. Note that Lavanya could have chosen to fly to city 4 or 10 or 9, and they would all result in the same scenario.

In total, they have visited 345 museums in the three months (which is in fact all the museums), and that is the answer.

Testcase 2: It is same as the previous testcase, but now they have only 2 months. So they visit only 150 + 90 = 240 museums in total.

Testcase 3: It is same as the previous testcase, but now they have 5 months of holidays. But sadly, they finish visiting all the cities within the first three months itself, and hence the answer is -1.

Two Paths

You are given an **N** * **M** grid, **A** [1, 2, …, **N**] [1, 2, …, **M**], that has an integer, which may be negative, in every cell. The grid has **N** rows and **M** columns. (1,1) is the top left corner cell and (**N**, **M**) is the bottom right corner cell. You are also given an integer **K**.

A *Path* is a sequence of **N** cells that starts at any cell at the bottom-most row and goes up one row at a time. When you move from a row to the one above, you can only move straight-up, or up-and-left. That is, if you are at cell (i, j) you can move straight-up to the cell (i - 1, j) or up-and-left to the cell (i - 1, j - 1). You have to stay within the grid, always. Also, you are allowed to move up-and-left at most **K** times in a single Path.

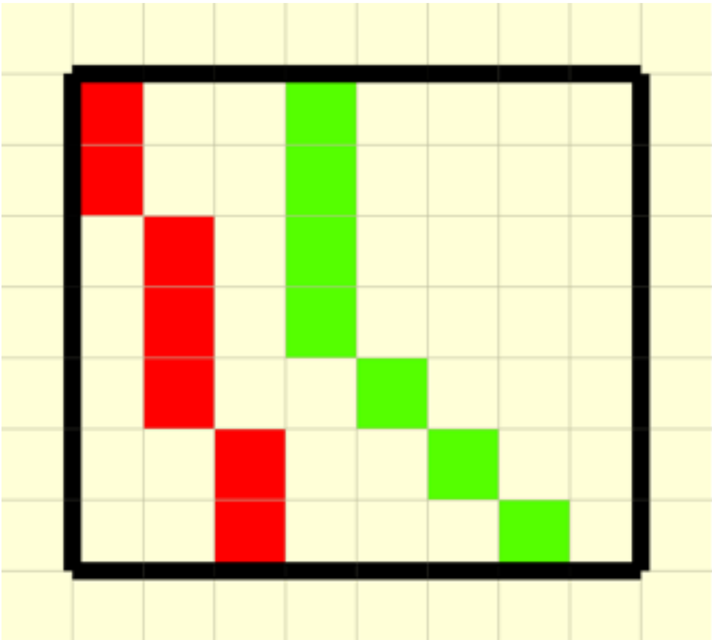
You need to choose two Paths with the following constraints.

- The starting cells are different and have at least **K** other cells in between them. That is, if the starting cells are (**N**, a) and (**N**, b), then | a - b | should be greater than or equal to (**K** + 1).
- The Paths are non-intersecting. Two Paths are said to be intersecting if they have at least one cell in common.

The *score* associated with any such pair of Paths is the sum of all the values between these two Paths (including the cells on the two Paths). That is, take every cell that is on, or in between these two Paths, and add up their values.

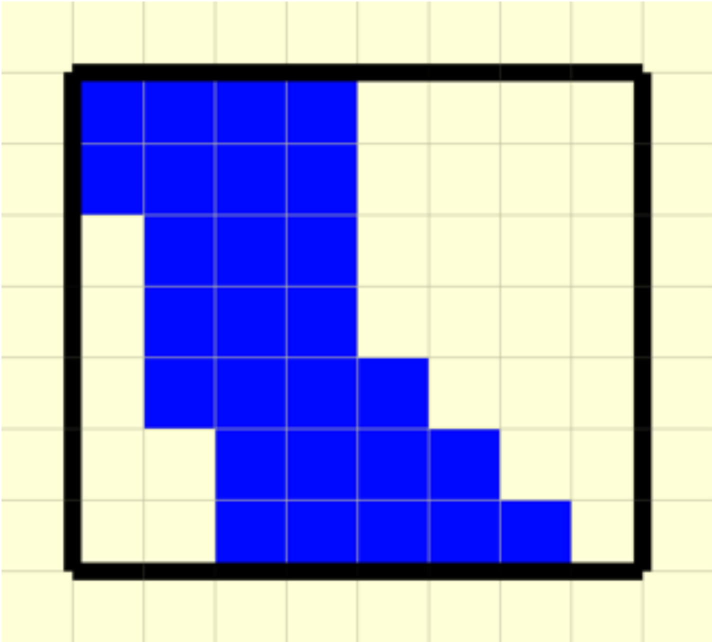
Formally, the score for a pair of non-intersecting Paths is defined as follows. On row i, suppose one path passes through cell (i, a) and the other path passes through cell (i, b), where $a < b$. Then, Score[i], the score for row i, is the sum of all entries **A**[i][j] for $a \le j \le b$. The overall score is the sum of all row scores Score[i] for $1 \le i \le N$.

For example, suppose **N** = 7, **M** = 8 and **K** = 3. Then, here are a pair of valid Paths that you might choose, coloured red and green.



Notice that the starting points are (7, 3) and (7, 7) and | 3 - 7 | = 4 ≥ (**K** + 1). The red Path makes only 2 up-and-left moves, and the green Path makes 3 up-and-left moves, both of which are less than or equal to **K**.

If these were the two Paths that you had chosen, then your score would be the sum of all the values in the cells which are coloured blue:



You need to choose two valid Paths such that the score is maximized, and you need to print this maximum score.

Input

- The first line contains a single integer, **T**, which is the number of testcases. The description of each testcase follows.
- The first line of each testcase contains three integers: **N**, **M** and **K**.
- The i^{th} of the next **N** lines contains **M** integers, denoting the values of the i^{th} row. That is, the j^{th} integer in this line represents **A**[i][j].

Output

For each test case, output a single line containing a single integer which should be the maximum score attainable.

Constraints

- $1 \le \mathbf{T} \le 3$
- $2 \le \mathbf{N}, \mathbf{M} \le 1000$
- $0 \le \mathbf{K} \le 20$
- $0 \le \mathbf{K} \le \mathbf{M} - 2$
- $-1000 \le \mathbf{A}[i][j] \le 1000$

Subtasks

- **Subtask 1** (15 points):
  - $2 \le \mathbf{N}, \mathbf{M} \le 30$
  - $0 \le \mathbf{K} \le 5$
- **Subtask 2** (5 points):
  - $2 \le \mathbf{N}, \mathbf{M} \le 110$
  - $0 \le \mathbf{K} \le 5$

- **Subtask 3** (12 points):
    - **K** = 0
- **Subtask 4** (16 points):
    - **K** = M - 2
- **Subtask 5** (52 points):
    - Original constraints

Example

```
Input:
1
7 8 3
2 1 1 1 -1 -1 -1 -1
2 1 1 1 -1 -1 -1 -1
-1 1 1 1 -1 -1 -1 -1
-1 1 1 1 -1 -1 -1 -1
-1 1 1 1 1 -1 -1 -1
-1 -1 1 1 1 1 -1 -1
-1 -1 1 1 1 1 1 -1

Output:
29
```

Explanation

The maximum score is obtained by choosing the Paths as shown in the figures in the statement. The blue area sums up all the positive integers and none of the negative integers and hence that is the maximum possible. The score is 29, and hence that is the output.

INOI 2018

Cutoff scores to qualify for IOITC 2018

- Class 12, 120/200
- Class 11, 116/200
- Class 10, 112/200
- Class 9 and below, 100/200