# Training (7pts, 13pts)

Problem

As the football coach at your local school, you have been tasked with picking a team of exactly **P** students to represent your school. There are **N**students for you to pick from. The i-th student has a *skill rating* **Si**, which is a positive integer indicating how skilled they are.

You have decided that a team is *fair* if it has exactly **P** students on it and they all have the same skill rating. That way, everyone plays as a team. Initially, it might not be possible to pick a fair team, so you will give some of the students one-on-one coaching. It takes one hour of coaching to increase the skill rating of any student by 1.

The competition season is starting very soon (in fact, the first match has already started!), so you'd like to find the minimum number of hours of coaching you need to give before you are able to pick a fair team.

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow. Each test case starts with a line containing the two integers **N** and **P**, the number of students and the number of students you need to pick, respectively. Then, another line follows containing **N** integers **Si**; the i-th of these is the skill of the i-th student.

Output

For each test case, output one line containing Case #x: y, where x is the test case number (starting from 1) and y is the minimum number of hours of coaching needed, before you can pick a fair team of **P** students.

Limits

Time limit: 15 seconds per test set.  
Memory limit: 1 GB.  
1 ≤ **T** ≤ 100.  
1 ≤ **Si** ≤ 10000, for all i.  
2 ≤ **P** ≤ **N**.

Test set 1 (Visible)

2 ≤ **N** ≤ 1000.

Test set 2 (Hidden)

2 ≤ **N** ≤ 105.

Sample

|  |  |
| --- | --- |
| Input | Output |
| 3  4 3  3 1 9 100  6 2  5 5 1 2 3 4  5 5  7 7 1 7 7 | Case #1: 14  Case #2: 0  Case #3: 6 |

In Sample Case #1, you can spend a total of 6 hours training the first student and 8 hours training the second one. This gives the first, second and third students a skill level of 9. This is the minimum time you can spend, so the answer is 14.

In Sample Case #2, you can already pick a fair team (the first and second student) without having to do any coaching, so the answer is 0.

In Sample Case #3, **P** = **N**, so every student will be on your team. You have to spend 6 hours training the third student, so that they have a skill of 7, like everyone else. This is the minimum time you can spend, so the answer is 6.

# Parcels (15pt, 20pts)

Competitive Submissions

You have not attempted this problem.

Last updated: Mar 24 2019, 00:02

Problem

You have been hired recently as the Chief Decision Maker (CDM) at a famous parcel delivery company, congratulations! Customers love speedy deliveries of their parcels and you have decided to decrease the time it takes to deliver parcels around the world to win customers. You have introduced this idea to the authorities and they have allocated you enough budget to build at most one new delivery office.

The world can be divided into an **R** × **C** grid of squares. Each square either contains a delivery office or it does not. You may pick a grid square that does not already contain a delivery office and build a new delivery office there.

The delivery time of a parcel to a square is 0 if that square contains a delivery office. Otherwise, it is defined as the minimum Manhattan distance between that square and any other square containing a delivery office. The overall delivery time is the maximum of delivery times of all the squares. What is the minimum overall delivery time you can obtain by building at most one new delivery office?

Note: The [Manhattan distance](https://en.wikipedia.org/wiki/Taxicab_geometry) between two squares (r1,c1) and (r2,c2) is defined as |r1 - r2| + |c1 - c2|, where |\*| operator denotes the absolute value.

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow. The first line of each test case contains the number of rows **R** and number of columns **C** of the grid. Each of the next **R** lines contains a string of **C** characters chosen from the set {0, 1}, where 0 denotes the absence of a delivery office and 1 denotes the presence of a delivery office in the square.

Output

For each test case, output one line containing Case #x: y, where x is the test case number (starting from 1) and y is the minimum overall delivery time you can obtain after adding at most one additional delivery office.

Limits

Time limit: 15 seconds per test set.  
Memory limit: 1GB.  
1 ≤ **T** ≤ 100.  
There is at least one delivery office in the initial grid.

Test set 1 (Visible)

1 ≤ **R** ≤ 10.  
1 ≤ **C** ≤ 10.

Test set 2 (Hidden)

1 ≤ **R** ≤ 250.  
1 ≤ **C** ≤ 250.

Sample

|  |  |
| --- | --- |
| Input | Output |
| 3  3 3  101  000  101  1 2  11  5 5  10001  00000  00000  00000  10001 | Case #1: 1  Case #2: 0  Case #3: 2 |

In Sample Case #1, you get a minimum overall delivery time of 1 by building a new delivery office in any one of the five squares without a delivery office.

In Sample Case #2, all the squares already have a delivery office and so the minimum overall delivery time is 0. Note you have to add **at most** one delivery office.

In Sample Case #3, to get a minimum overall delivery time of 2, you can build a new delivery office in any of these squares: (2, 3), (3, 2), (3, 3), (3, 4), or (4, 3). Any other possibility results in a higher overall delivery time than 2.

# Contention

### Problem

You are selling tickets for the front row of seats at a movie theater. The front row has **N** seats, numbered 1 to **N** from left to right. You have been out of the office the last week, and upon your return, **Q** *bookings* for seats have piled up! The i-th booking requests all the seats from **Li** to **Ri**inclusive. You now have the boring job of entering each booking into the system, one at a time.

Since some of the bookings may overlap, the system might not be able to fulfill each booking entirely. When you enter a booking into the system, it will assign every seat requested by the booking that hasn't already been assigned to a booking entered into the system earlier.

What is the largest integer k where there exists an order that you can enter the bookings into the system, such that each booking is assigned at least k seats?

### Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow. Each test case starts with a line containing two integers **N** and **Q**, the number of seats and the number of bookings, respectively. Then, there are **Q** more lines, the i-th of which contains the two integers **Li** and **Ri**, indicating that the i-th booking would like to book all the seats from **Li** to **Ri**, inclusive.

### Output

For each test case, output one line containing Case #x: y, where x is the test case number (starting from 1) and y is the largest value k, as described above.

### Limits

Time limit: 30 seconds per test set.  
Memory limit: 1GB.  
**T** = 100.  
1 ≤ **N** ≤ 106.  
1 ≤ **Li** ≤ **Ri** ≤ **N**.

#### Test set 1 (Visible)

1 ≤ **Q** ≤ 300.

#### Test set 2 (Hidden)

1 ≤ **Q** ≤ 30000.  
For at least 85 of the test cases, **Q** ≤ 3000.

### Sample

|  |  |
| --- | --- |
| Input | Output |
| 3  5 3  1 2  3 4  2 5  30 3  10 11  10 10  11 11  10 4  1 8  4 5  3 6  2 7 | Case #1: 1  Case #2: 0  Case #3: 2 |

In Sample Case #1, there are **N** = 5 seats and **Q** = 3 bookings. One possible order is:

* Put in the second booking, where the system will book 2 seats (3 and 4).
* Put in the first booking, where the system will book 2 seats (1 and 2).
* Put in the third booking, where the system will book 1 seat (only seat 5, since seats 1, 2, 3 and 4 are already booked).

Each booking is assigned at least 1 seat, and there is no order that assigns at least 2 seats to each booking, so the answer is 1.

In Sample Case #2, there are **N** = 30 seats and **Q** = 3 bookings. No matter what order you assign the seats in, at least one booking will have no seats assigned to it. So the answer is 0. Notice that there can be seats that are not part of any bookings!

In Sample Case #3, there are **N** = 10 seats and **Q** = 4 bookings. One possible order is:

* Put in the second booking, where the system will book 2 seats (4 and 5).
* Put in the third booking, where the system will book 2 seats (3 and 6, since 4 and 5 are already booked). Notice that the seats booked are not necessarily adjacent to each other.
* Put in the fourth booking, where the system will book 2 seats (2 and 7).
* Put in the first booking, where the system will book 2 seats (1 and 8).

Each booking is assigned at least 2 seats, and there is no order that assigns at least 3 seats to each booking, so the answer is 2.

**Note**: We do not recommend using interpreted/slower languages for the Large dataset of this problem.