Bipartite Graph Game

Time Limit	**Memory Limit**
1 second (See Below)	512 MB

Description

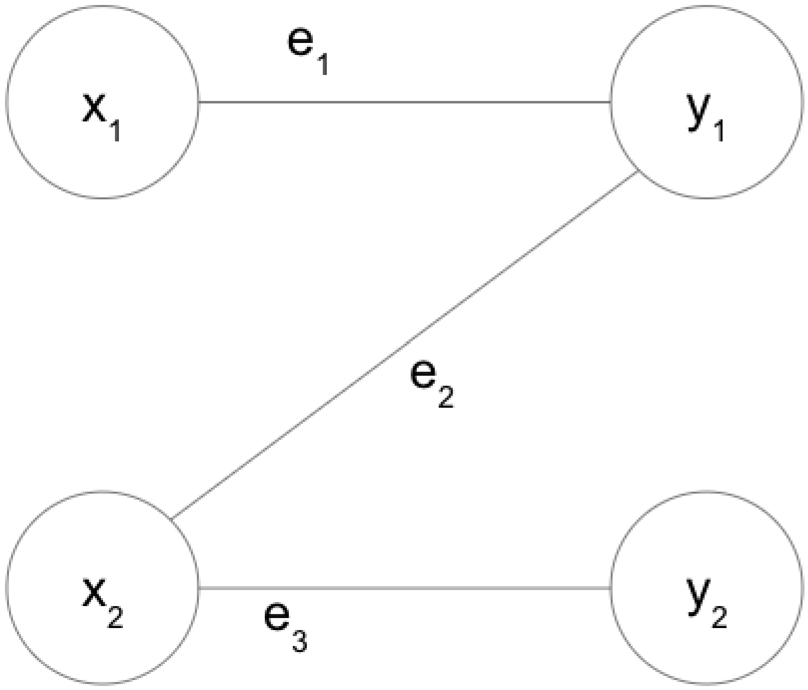
Bob likes to play a game using bipartite graphs. We define a bipartite graph $H=(X+Y,E)$ as follows:

- An n-node set $X=\{x1,x2,\ldots,xn\}$ and an m-node set $Y=\{y1,y2,\ldots,ym\}$ are disjoint, and thus every edge in H connects one node in X with one node in Y.
- The edge set is represented by $E=\{e1,e2,\ldots,ek\}$.

Bob plays the bipartite graph game as follows:

- To each node xi in X, a distinct integer weight vi between 1 and n is assigned. That is, $1 \le vi \le n$ & $vi \in Z$ for each i with $1 \le i \le n$ and $vi \ne vj$ when $i \ne j$.
- To each node yi in Y, a distinct integer weight wj between 1 and m is assigned. That is, $1 \le wj \le m$ & $wj \in Z$ for each j with $1 \le j \le m$and $wi \ne wj$ when $i \ne j$.
- Given this, the weight of an edge $e=(xi,yj)$ is defined as $vi+wj$.
- The "*score*" of H is defined as the sum of the weights of all edges.

For instance, suppose $X = \{x_1, x_2\}$, $Y = \{y_1, y_2\}$, and $E = \{(x_1, y_1), (x_2, y_1), (x_2, y_2)\}$ as shown below.

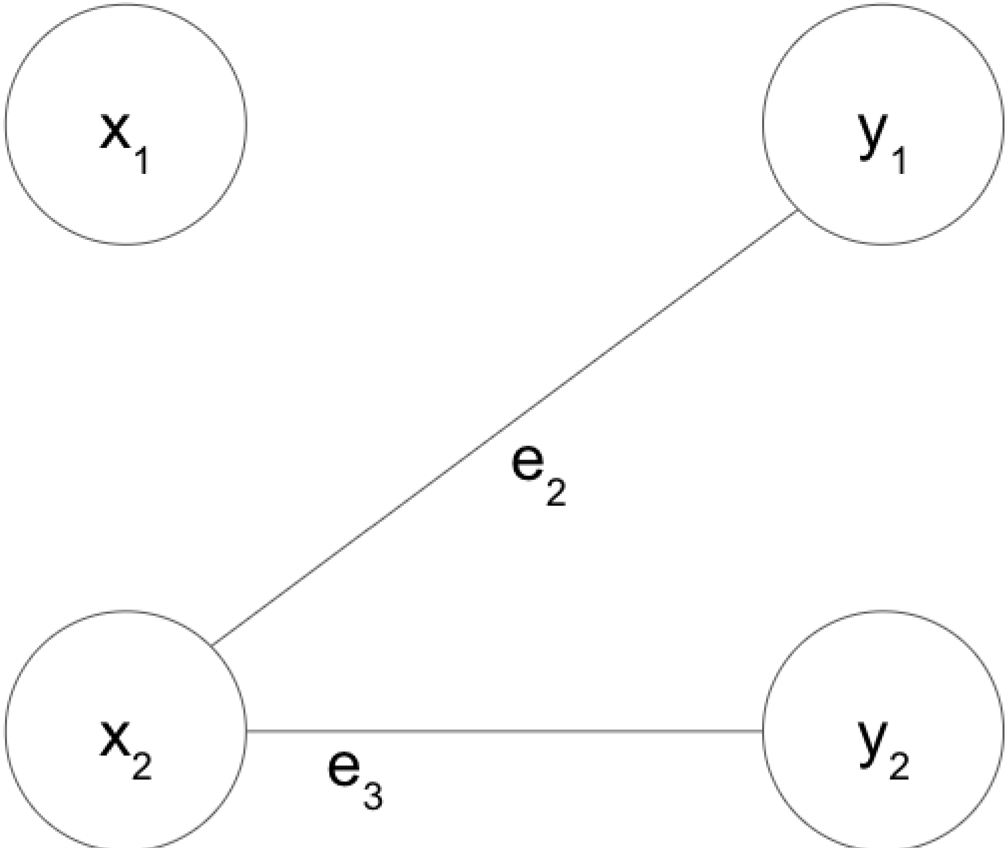


There are 2! × 2! = 4 ways to assign weights to the nodes as above.

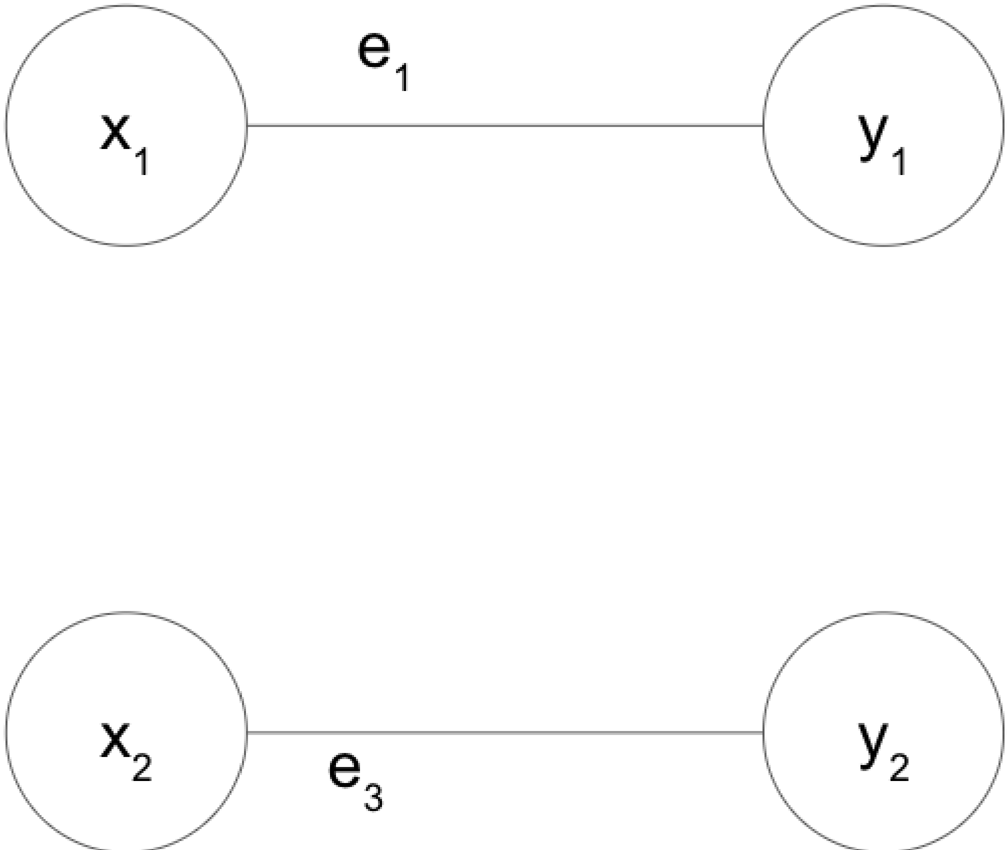
- If $v_1 = 1$, $v_2 = 2$ and $w_1 = 1$, $w_2 = 2$: $e_1 = v_1 + w_1 = 2$, $e_2 = v_2 + w_1 = 3$, $e_3 = v_2 + w_2 = 4$, and thus the score of H is 9.
- If $v_1 = 1$, $v_2 = 2$ and $w_1 = 2$, $w_2 = 1$: $e_1 = v_1 + w_1 = 3$, $e_2 = v_2 + w_1 = 4$, $e_3 = v_2 + w_2 = 3$, and thus the score of H is 10.
- If $v_1 = 2$, $v_2 = 1$ and $w_1 = 1$, $w_2 = 2$: $e_1 = v_1 + w_1 = 3$, $e_2 = v_2 + w_1 = 2$, $e_3 = v_2 + w_2 = 3$, and thus the score of H is 8.
- If $v_1 = 2$, $v_2 = 1$ and $w_1 = 2$, $w_2 = 1$: $e_1 = v_1 + w_1 = 4$, $e_2 = v_2 + w_1 = 3$, $e_3 = v_2 + w_2 = 2$, and thus the score of H is 9.

Let S(H) be the largest score of H we can obtain by assigning the weights. Bob is getting bored from computing S(H) as he is too good at it.

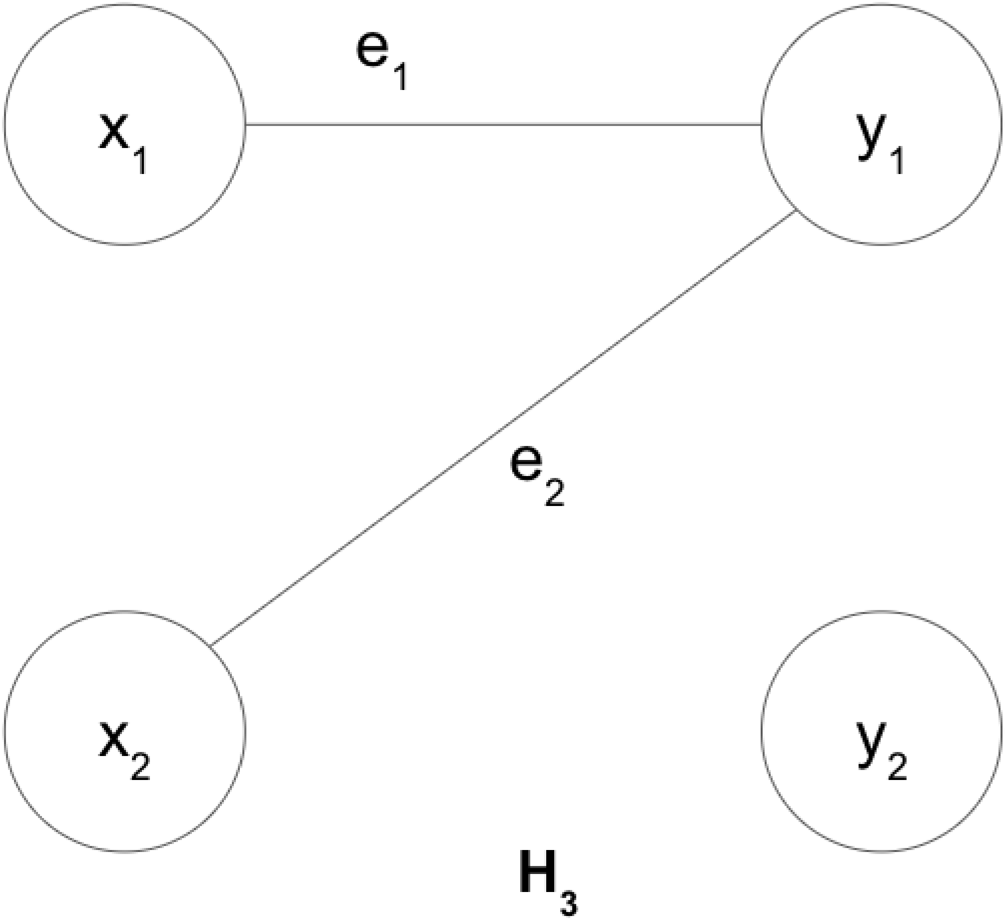
Alice, noticing this, proposed a new game. Let H_i be the graph you obtain by removing the i-th edge from H (see below).



H_1



H_2



$S(H_i)$ is then the highest score achievable for H_i, analogously. In the example above, H_1 is obtained by removing (x_1, y_1) in H, and $S(H_1) = 7$ (to achieve this, Bob should assign $v_1 = 1$, $v_2 = 2$). In the same example, H_2 is obtained by removing (x_2, y_1) in H, and $S(H_2) = 6$ regardless of the weights (v_1, v_2, w_1, w_2). Lastly, $S(H_3) = 7$, and thus the maximum of $S(H_1)$, $S(H_2)$, $S(H_3)$ is 7.

Per Alice's suggestion, Bob wants to compute S(H) as well as the maximum of $S(H_1)$, $S(H_2)$, ..., $S(H_k)$. Let's help Bob.

Input

The first line of the input will contain T, the number of test cases.

The first line of each test case will contain n, m, k, separated by whitespace. The next k lines will contain two integers (a node in X and a node in Y) in each line, describing an edge. If an edge connects x_i and y_j, then the input will be given as "i j".

Output

Output each test case's answer (two integers) in each line. The first integer must be S(H) and the second the maximum of $S(H_1)$, ..., $S(H_k)$.

Limit

- 1 ≤ T ≤ 10
- 1 ≤ n, m ≤ 10,000
- 1 ≤ k ≤ 100,000

Sample Input 1 Copy

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

```
3 2
```

Sample Output 1 Copy

```
11 6

10 7

15 13
```

Case 1: One way is to assign v_1 = 1, w_4 = 4, w_5 = 5 to obtain S(H) = 11. S(H_1) = S(H_2) = 6 in this example.

Case 2: Discussed in the problem statement.

Case 3: No explanation.

Time Limit

- Java 8: 6 seconds
- Python 3: 3 seconds
- PyPy3: 3 seconds
- Java 8 (OpenJDK): 6 seconds
- Java 11: 6 seconds
- Kotlin (JVM): 6 seconds
- Java 15: 6 seconds