5. Implement an algorithm to determine the maximum matching in a bipartite graph, and, if that matching is perfect (all nodes are matched) in either C, C++, C#, Java, or Python. Be efficient and use your max-flow implementation from the previous week.

   The input will start with an positive integer, giving the number of instances that follow. For each instance, there will be 3 positive integers m, n, and q. Numbers m and n are the number of nodes in node set A and node set B. Number q is the number of edges in the bipartite graph. For each edge, there will be 2 more positive integers i, and j representing an edge between node $1 \leq i \leq m$ in A and node $1 < i < n$ in B.

   A sample input is the following:

```
3
2 2 4
1 1
1 2
2 1
2 2
2 3 4
2 3
2 1
1 2
2 2
5 5 10
1 1
1 3
2 1
2 2
2 3
2 4
3 4
4 4
5 4
5 5
```

   The sample input has 3 instances.

   For each instance, your program should output the size of the maximum matching, followed by a space, followed by an N if the matching is not perfect and a Y if the matching is perfect. Each output line should be terminated by a newline. The correct output to the sample input would be:

```
2 Y
2 N
4 N
```