//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 bookin 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 en er 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 ea h 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 conta ning two integers N and Q, the number of seats and the number of bookings, respectively. Then, there are Q more li es, the i-th of which contains the two integers Li and Ri, indicating that the i-th booking would like to book all the se ts 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 th 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 ans er 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 ne 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 he 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 an wer is 2.

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

```
// In the name of God

#include <iostream>
#include <algorithm>
#include <fstream>
#include <vector>
#include <deque>
#include <assert.h>
#include <queue>
#include <stack>
#include <set>
#include <map>
#include <stdio.h>
#include <string.h>
#include <utility>
#include <math.h>
```

```
#include <bitset>
#include <iomanip>
#include <complex>

using namespace std;

#define rep(i, a, b) for (int i = (a), i##_end_ = (b); i < i##_end_; ++i)
#define debug(...) fprintf(stderr, __VA_ARGS__)
#define mp make_pair
#define x first
#define y second
#define pb push_back
#define SZ(x) (int((x).size()))
#define ALL(x) (x).begin(), (x).end()

template<typename T> inline bool chkmin(T &a, const T &b) { return a > b ? a = b, 1 : 0; }
template<typename T> inline bool chkmax(T &a, const T &b) { return a < b ? a = b, 1 : 0; }
template<typename T> inline bool smin(T &a, const T &b)   { return a > b ? a = b : a;    }
template<typename T> inline bool smax(T &a, const T &b)   { return a < b ? a = b : a;    }

typedef long long LL;

const int N = (int) 1e6 + 6, mod = (int) 0;
int n, q, xl[N], xr[N], mvl[N];
pair<int, int> sr[N];
int check(int k) {
 for (int j = 0; j < q; ++j)
  xl[j] = sr[j].first, xr[j] = -sr[j].second, mvl[j] = xl[j];
 for (int j = 0; j < q; ++j) {
  int l = xl[j], r = xr[j];
  int st = mvl[j];
  int allowed_after = r;
  int cnt = 0;
  for (int i = j + 1; i < q; ++i) {
   if (xr[i] <= r) {
    if (xl[i] <= st) {
     st = max(st, xr[i]);
    } else {
     cnt += xl[i] - st;
     st = max(st, xr[i]);
     if (cnt >= k) {
      allowed_after = xl[i] - (cnt - k);
      break;
     }
    }
   }
  }
  if (cnt < k) {
   cnt += r - st;
   if (cnt < k) return 0;
   allowed_after = r - (cnt - k);
  }
  for (int i = j + 1; i < q; ++i) {
   if (xl[i] >= allowed_after) break;
   if (xr[i] > r) {
    mvl[i] = max(mvl[i], r);
   }
  }
 }
 return 1;
}
int main() {
 int tc;
 cin >> tc;
 for (int tt = 1; tt <= tc; ++tt) {
  cout << "Case #" << tt << ": ";
  cin >> n >> q;
  for (int j = 0; j < q; ++j) {
   cin >> xl[j] >> xr[j], --xl[j];
   sr[j] = mp(xl[j], -xr[j]);
  }
  sort(sr, sr + q);
  int bl = 0, br = n + 1;
  while (bl < br - 1) {
   int bm = bl + br >> 1;
   if (check(bm)) {
    bl = bm;
   } else {
    br = bm;
   }
  }
  cout << bl << '\n';

 }
}
```