Day 99 coding Statement :

There is a hallway of length N−1 and you have M workers to clean the floor. Each worker is responsible for
segment [Li?,Ri?], i.e., the segment starting at Li? and ending at Ri?. The segments might overlap.

Every unit of length of the floor should be cleaned by at least one worker. A worker can clean 1 unit of
length of the floor in 1 unit of time and can start from any position within their segment. A worker can also
choose to move in any direction. However, the flow of each worker should be continuous, i.e, they can’t skip any
portion and jump to the next one, though they can change their direction.
What’s the minimum amount of time required to clean the floor, if the workers work simultaneously?

```
import java.io.*;
import java.lang.reflect.Array;
import java.util.*;
import java.lang.*;
class Main {
 int n,m;
 public boolean check(Segment[]ss,int x){
 int curLoc = 1;
 PriorityQueue<Integer> pq = new PriorityQueue<>();
 int i=0;
 while ( curLoc < n ){
 while ( i < m && ss[i].l <= curLoc ){
 pq.add(ss[i].r);
 i++;
 }
 int cur = curLoc;
 while ( cur == curLoc && !pq.isEmpty() ){
 int r = pq.poll();

 curLoc = Math.max(curLoc,Math.min(r,curLoc+x));
 }
 if( cur == curLoc)break;
 }
 return curLoc==n;
 }
 public void solve() {
 FastScanner fs = new FastScanner();
```

```
PrintWriter out = new PrintWriter(System.out);
int tc = fs.nextInt();
while(tc-- > 0 ){
n = fs.nextInt(); m = fs.nextInt();
Segment[]ss = new Segment[m];
for(int i=0;i<m;i++){
int l = fs.nextInt(), r = fs.nextInt();
ss[i] = new Segment(l,r);
}
Arrays.sort(ss,Comparator.comparingInt(s -> s.l));
int ans= -1 ,l=1,r= (int)1e9;
while ( l <= r ){
int mid = (l+r)/2;
if( check(ss,mid)){
ans = mid;
r = mid-1;
}else{
l = mid+1;
}
}
out.println(ans);

}
out.flush();
}
class Segment{
int l,r;
Segment(int l,int r){
this.l = l;
this.r = r;
}
}
public static void main(String[]args){
try{
new Codechef().solve();
} catch (Exception e){
e.printStackTrace();
}
}
class FastScanner {
BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
StringTokenizer st=new StringTokenizer("");
String next() {
while (!st.hasMoreTokens())
try {
st=new StringTokenizer(br.readLine());
} catch (IOException e) {
e.printStackTrace();
```

```
 }
 return st.nextToken();
 }

 String nextLine()
 {
 String str = "";
 try
 {
 str = br.readLine();
 }catch (IOException e)
 {
 e.printStackTrace();
 }
 return str;
 }
 int nextInt() {return Integer.parseInt(next()); }
 long nextLong() {return Long.parseLong(next());}
 }
}
```