Day 92 coding Statement :

Let X be the set of all integers between 0 and n-1. Suppose we have a collection S1, S2, ..., Sm of subsets of X. Say an atom A is a subset of X such that for each Si we have either A is a
subset of Si or A and Si do not have any common elements.

Your task is to find a collection A1, ..., Ak of atoms such that every item in X is in some Ai and no two Ai, Aj with i ≠ j share a common item. Surely such a collection exists as we could create a single set {x} for each x in X.
A more interesting question is to minimize k, the number of atoms.

```
import java.io.*;
import java.util.StringTokenizer;
public class Main {
BufferedReader in;
StringTokenizer str;
PrintWriter out;
String next() throws IOException {
while ((str == null) || (!str.hasMoreTokens())) {
str = new StringTokenizer(in.readLine());
}
;
return str.nextToken();
};
int nextInt() throws IOException {
return Integer.parseInt(next());

};
double nextDouble() throws IOException {
return Double.parseDouble(next());
};
double nextLong() throws IOException {
return Long.parseLong(next());
};
int n, m;
int[][] a;
int[] buv;
int[] kilk;
 void dfs(int v)
 {
 buv[v]=1;
 for (int i=0; i<n; i++)
 if ((a[v][i]==0)&&(buv[i]==0))
 {
 dfs(i);
 };
 };
```

```
void solve() throws IOException {
n = nextInt();
m = nextInt();
a = new int[n][n];
buv = new int[n];
kilk = new int[n];

for (int i = 0; i < m; i++) {
int t = nextInt();
int now[] = new int[n ];
for (int j = 0; j < t; j++) {
int k = nextInt();
now[k] = 1;
}
for (int j = 0; j < n; j++)
for (int l = 0; l < n; l++) {
if ((now[j] ^ now[l]) == 1) {
a[j][l] = 1;
a[l][j] = 1;
}
}
};
int res=0;
for (int i=0; i<n; i++)
if (buv[i]==0)
{
res++;
dfs(i);
};
out.println(res);
};
void run() throws IOException {
in = new BufferedReader(new InputStreamReader(System.in));

out = new PrintWriter(System.out);
int n = nextInt();
for (int i = 0; i < n; i++) {
solve();
}
;
out.close();
}
public static void main(String[] args) throws IOException {
new Main().run();
}
}
```