**package** Networks;

**import** java.io.\*;

**import** java.util.\*;

**public** **class** RandomGraph {

HashMap<Integer,ArrayList<Integer>> adjList;

**int** n;

**double** m;

**double** c;

RandomGraph(){

adjList=**new** HashMap<Integer,ArrayList<Integer>>();

}

RandomGraph(**int** n, **double** c){

adjList=**new** HashMap<Integer,ArrayList<Integer>>();

**this**.n=n;

**this**.c=c;

**this**.m=0.5\*n\*c;

}

**public** HashMap<Integer,ArrayList<Integer>> populateAdjList(){

//boolean flag=false;

Random rand=**new** Random();

**for**(**int** i=0;i<m;i++){

**int** x=rand.nextInt(n);

**int** y=rand.nextInt(n);

**if**(x==y)**continue**;

**if**(!adjList.containsKey(x)){

ArrayList list=**new** ArrayList<Integer>();

list.add(**new** Integer(y));

adjList.put(**new** Integer(x), list);

list=adjList.get(y);

**if**(list!=**null** && !list.contains(x)){

list.add(x);

adjList.put(y, list);

}**else** **if**(list==**null**){

list=**new** ArrayList<Integer>();

list.add(x);

adjList.put(y, list);

}

}

**else** **if**(adjList.containsKey(x)){

ArrayList list=(ArrayList<Integer>)adjList.get(x);

**if**(list.contains(y)){

**continue**;

}**else**{

list.add(y);

adjList.put(x, list);

ArrayList l1=adjList.get(y);

**if**(l1!=**null** && !l1.contains(x)){

l1.add(x);

adjList.put(y, l1);

}**if**(l1==**null**){

l1=**new** ArrayList<Integer>();

l1.add(x);

adjList.put(y, l1);

}

}

}

}

**return** adjList;

}

**public** **int** findGiantComponentSize(){

//returns the maximum size of component in the network

**int** max=0;

**int** count=0;

//Keeps track of which nodes have been visited already and which not

HashSet<Integer> visited=**new** HashSet<Integer>();

**for**(**int** i=0;i<n;i++){

visited.add(i);

}

**for**(**int** k=0;k<n;k++){

**if**(visited.contains(k)){

count=0;

Queue<Integer> q=**new** LinkedList<Integer>();

**if**(adjList.containsKey(k)){

**for**(**int** j:adjList.get(k)){

q.add(j);count++;visited.remove(j);

**while**(!q.isEmpty()){

**int** t=q.remove();

ArrayList<Integer> l=adjList.get(t);

**for**(**int** h:l){

**if**(visited.contains(h)){

visited.remove(h);

q.add(h);

count++;

}

}

}

}

}

**if**(max<count){

max=count;

}

}

}

**return** max;

}

**public** **static** **void** main(String args[]) **throws** IOException{

RandomGraph g=**new** RandomGraph(1000000,1.3863);

g.adjList=g.populateAdjList();

System.*out*.println("Size of Mazimum Component: "+g.findGiantComponentSize());

}

}

/\*

Output:

Size of Mazimum Component: 500491

\*/