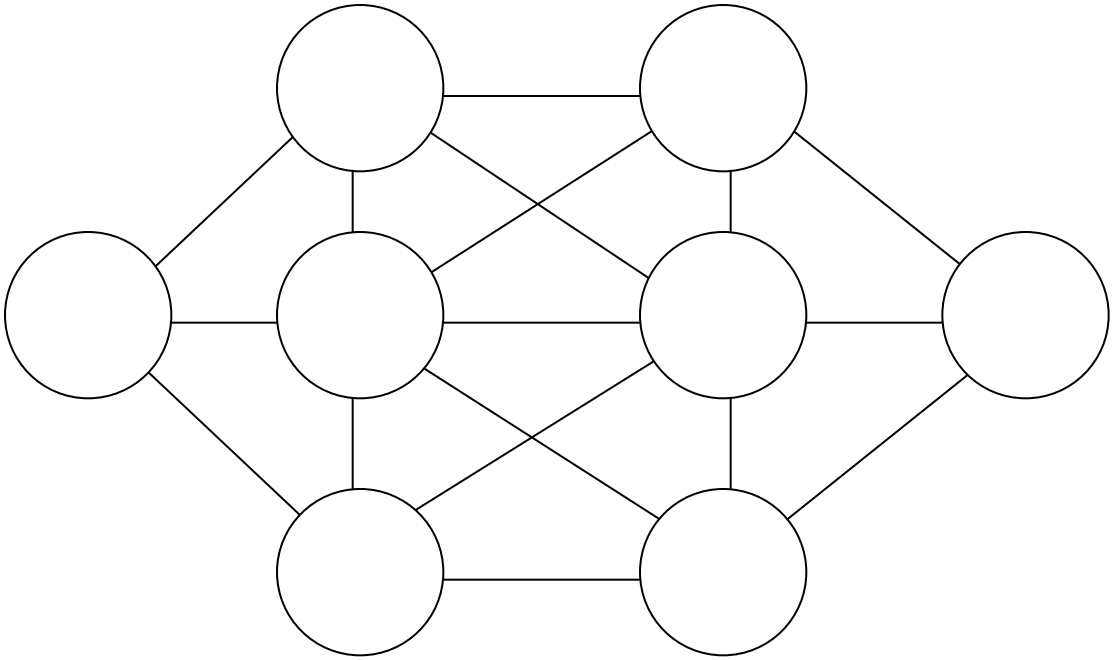
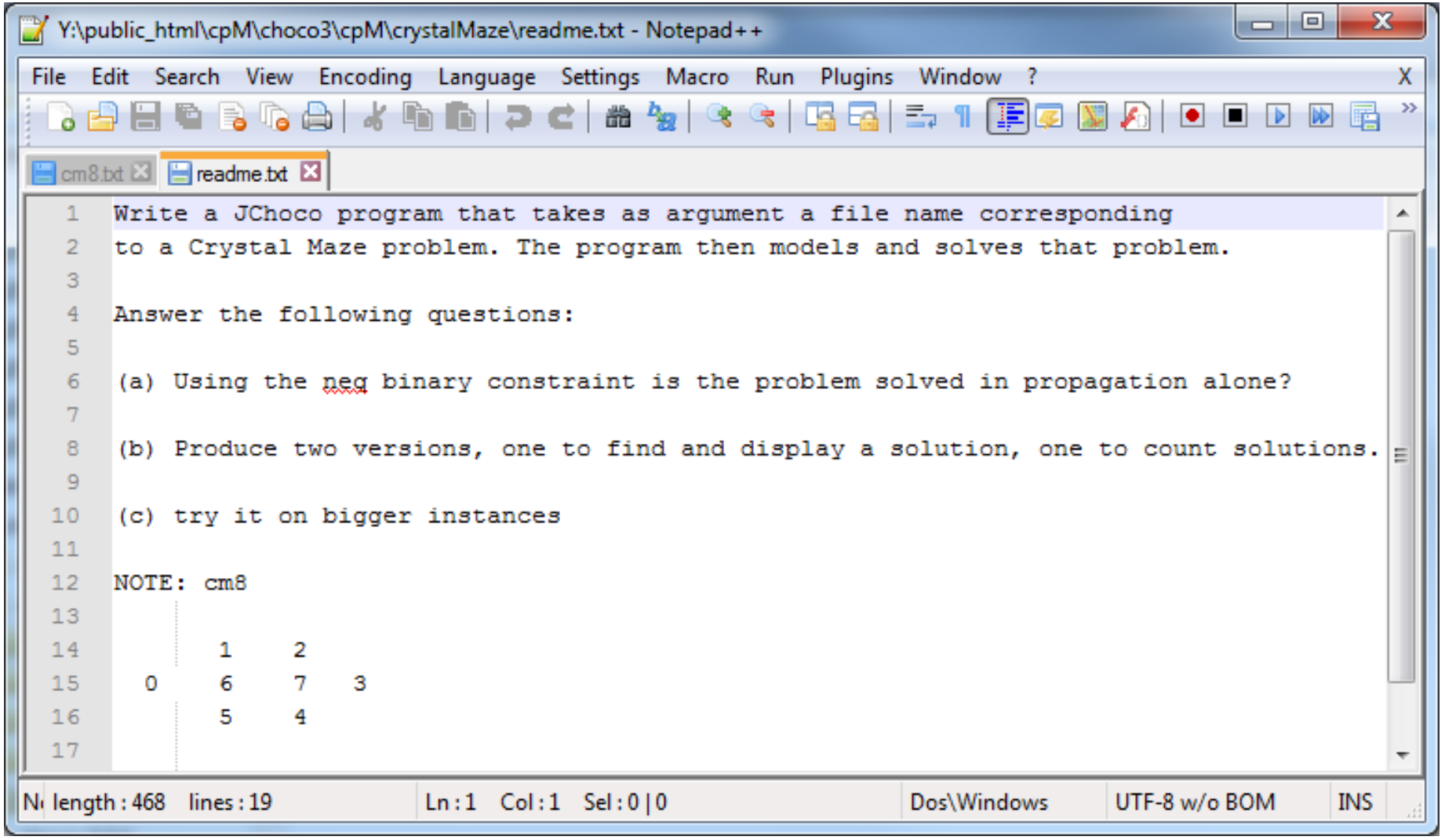
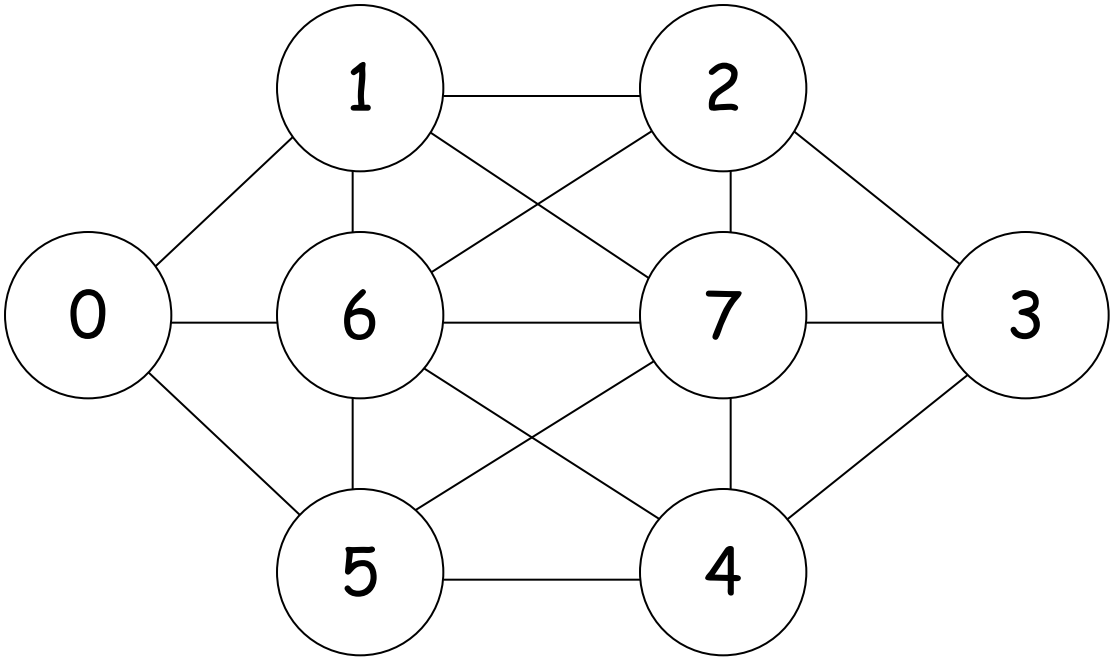
Crystal Maze

coded in choco3



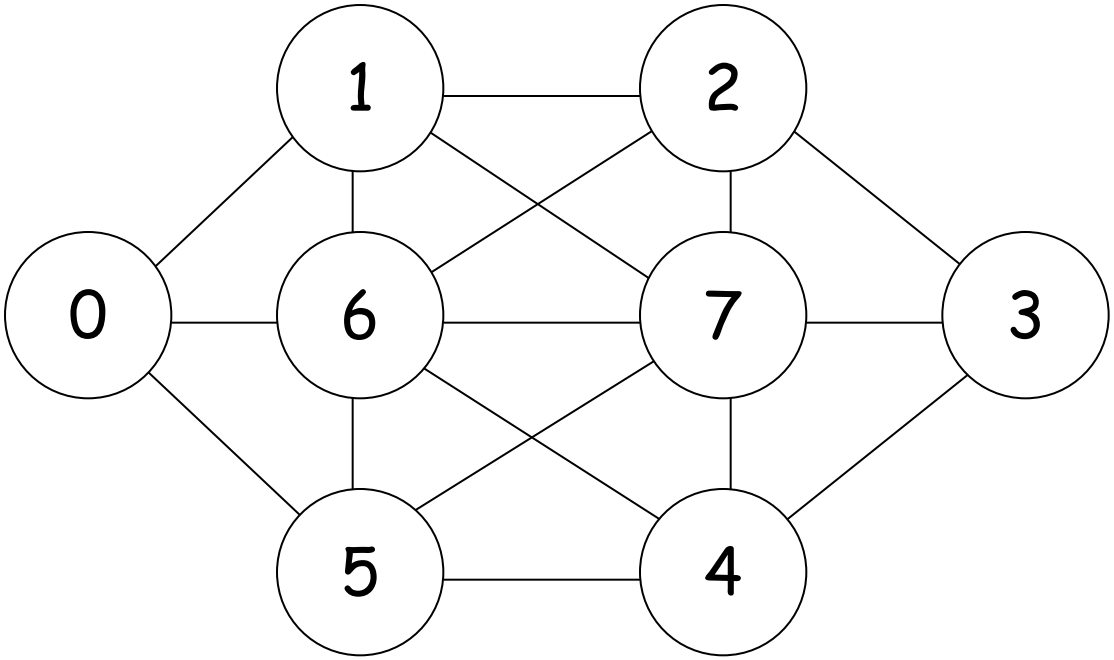
Put a different number in each circle (1 to 8) such that adjacent circles cannot take consecutive numbers


Y:\public_html\cpM\choco3\cpM\crystalMaze\readme.txt - Notepad++
File Edit Search View Encoding Language Settings Macro Run Plugins Window ?
cm8.txt
readme.txt
1 Write a JChoco program that takes as argument a file name corresponding
2 to a Crystal Maze problem. The program then models and solves that problem.
3
4 Answer the following questions:
5
6 (a) Using the neg binary constraint is the problem solved in propagation alone?
7
8 (b) Produce two versions, one to find and display a solution, one to count solutions.
9
10 (c) try it on bigger instances
11
12 NOTE: cm8
13
14 1 2
15 0 6 7 3
16 5 4
17
length : 468 lines : 19
Ln : 1 Col : 1 Sel : 0 | 0
Dos\Windows
UTF-8 w/o BOM
INS

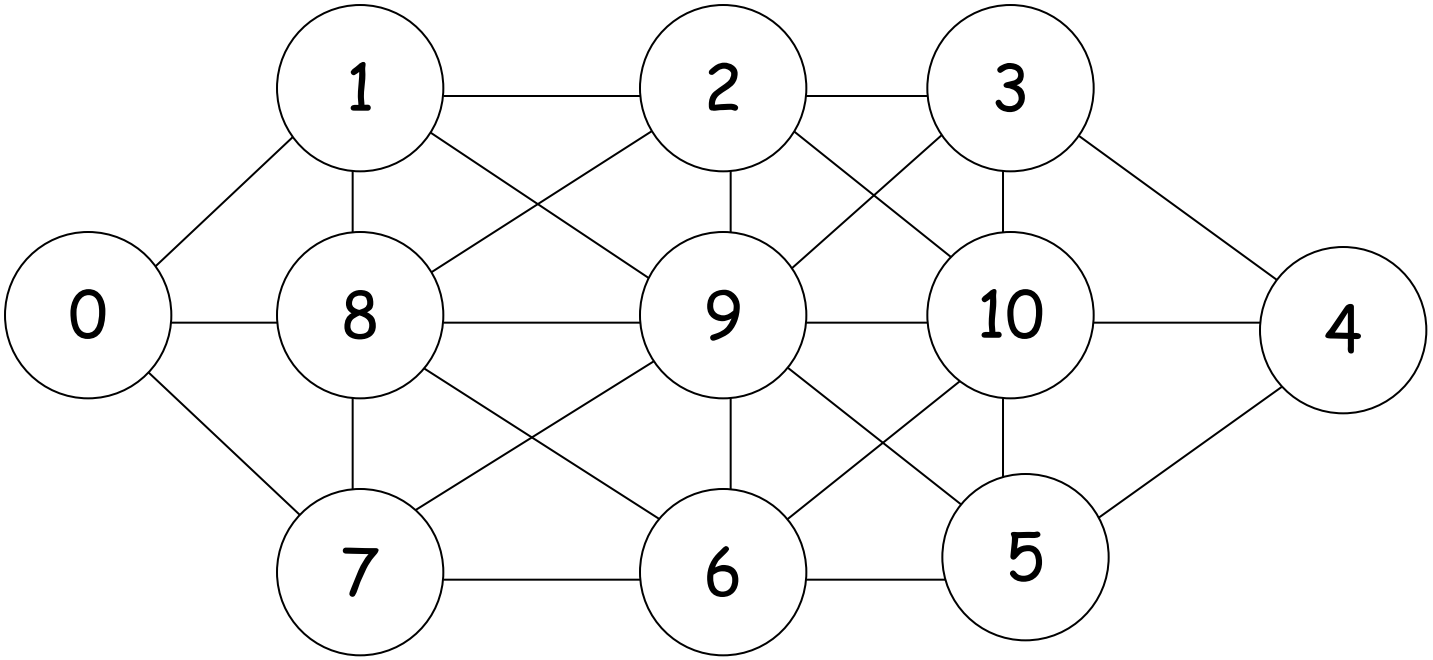


Put a different number in each circle (0 to 7) such that adjacent circles cannot take consecutive numbers

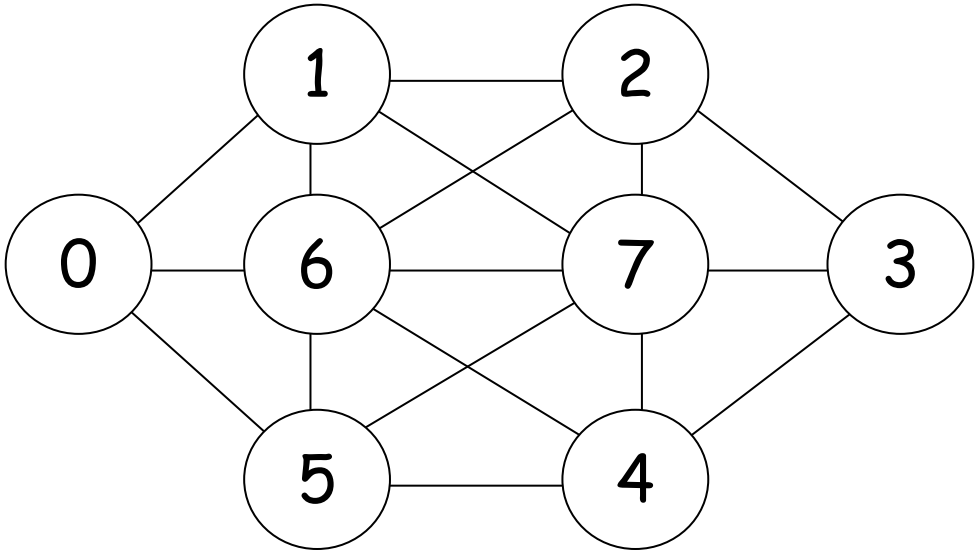
The numbers are the identification of a circle



Put a different number in each circle (0 to 7) such that adjacent circles cannot take consecutive numbers



Put a different number in each circle (0 to 10) such that adjacent circles cannot take consecutive numbers



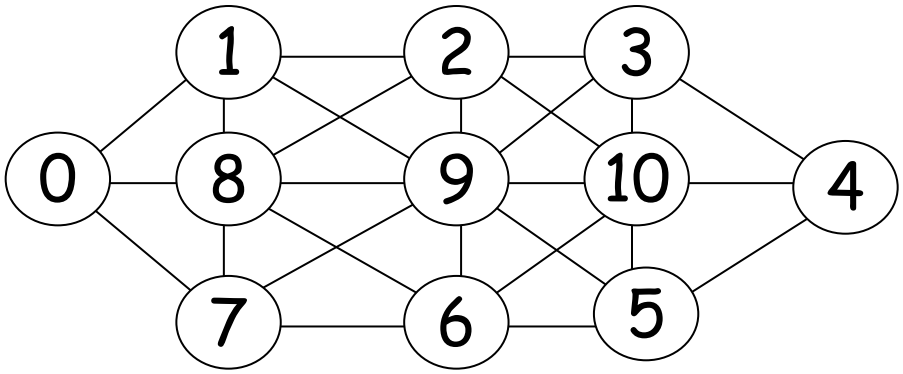
Y:\public_html\cpM\...

File Edit Search View Encoding
Language Settings Macro Run
Plugins Window ? X

cm8.txt

```
1  8
2  0 1
3  0 5
4  0 6
5  1 2
6  1 6
7  1 7
8  2 3
9  2 6
10 2 7
11 3 4
12 3 7
13 4 5
14 4 6
15 4 7
16 5 6
17 5 7
18 6 7
19
```

Dos\Window UTF-8 w/o BOM INS



Y:\public_html\cpM\choco3...

File Edit Search View Encoding
Language Settings Macro Run Plugins
Window ? X

cm8.txt cm11.txt

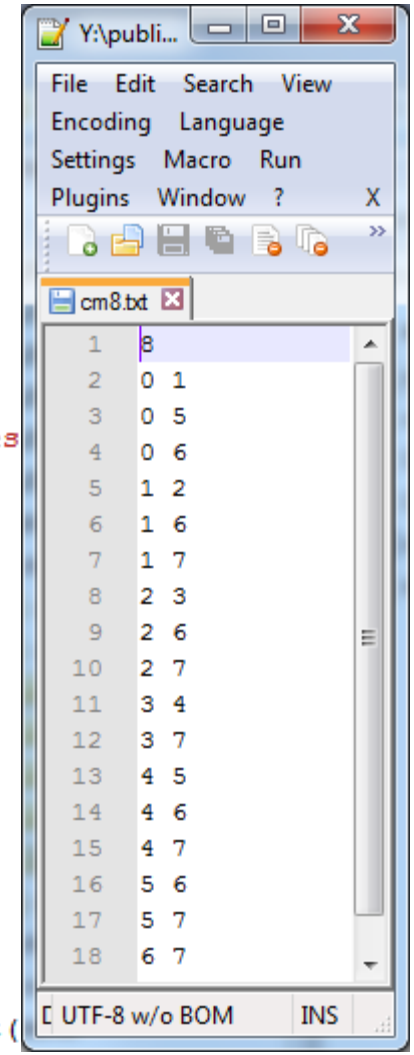
```
11
0 1
0 8
0 7
1 2
1 9
1 8
2 3
2 10
2 9
2 8
3 4
3 10
3 9
4 10
4 5
5 10
5 9
5 6
6 10
6 9
6 8
6 7
7 9
7 8
8 9
9 10
```

Dos\Windows UTF-8 w/o BOM INS

```java
import java.io.File;
import java.io.IOException;
import java.util.Scanner;
import org.chocosolver.solver.*;
import org.chocosolver.solver.variables.*;
import org.chocosolver.solver.constraints.*;


public class CrystalMaze {


    public static void main(String[] args) throws IOException {
        Scanner sc = new Scanner(new File(args[0]));
        int n = sc.nextInt(); // vertices
        Solver solver = new Solver("crystal maze");
        IntVar[] v = VF.enumeratedArray("v",n,1,n,solver); // n variables with values 1 to n

        while (sc.hasNext()){
            int i = sc.nextInt();
            int j = sc.nextInt();
            solver.post(ICF.distance(v[i],v[j],">",1));
        }

        sc.close();

        for (int i=0;i<n-1;i++)
            for (int j=i+1;j<n;j++)
                solver.post(ICF.arithm(v[i],"!=",v[j]));

        System.out.println(solver.findSolution());
        for (int i=0;i<n;i++) System.out.println(v[i].getValue());
        System.out.println("nodes: " + solver.getMeasures().getNodeCount() +
                           "   cpu: " + solver.getMeasures().getTimeCount());
    }
}
```

```java
import java.io.File;
import java.io.IOException;
import java.util.Scanner;
import org.chocosolver.solver.*;
import org.chocosolver.solver.variables.*;
import org.chocosolver.solver.constraints.*;

public class CrystalMaze {


    public static void main(String[] args) throws IOException {
        Scanner sc = new Scanner(new File(args[0]));
        int n = sc.nextInt(); // vertices
        Solver solver = new Solver("crystal maze");
        IntVar[] v = VF.enumeratedArray("v",n,1,n,solver); // n variables with values 1 to n

        while (sc.hasNext()){
            int i = sc.nextInt();
            int j = sc.nextInt();
            solver.post(ICF.distance(v[i],v[j],">",1));
        }

        sc.close();

        for (int i=0;i<n-1;i++)
            for (int j=i+1;j<n;j++)
                solver.post(ICF.arithm(v[i],"!=",v[j]));

        System.out.println(solver.findSolution());
        for (int i=0;i<n;i++) System.out.println(v[i].getValue());
        System.out.println("nodes: " + solver.getMeasures().getNodeCount() +
                           "   cpu: " + solver.getMeasures().getTimeCount());
    }
}
```

```java
import java.io.File;
import java.io.IOException;
import java.util.Scanner;
import org.chocosolver.solver.*;
import org.chocosolver.solver.variables.*;
import org.chocosolver.solver.constraints.*;


public class CrystalMaze {

    public static void main(String[] args) throws IOException {
        Scanner sc = new Scanner(new File(args[0]));
        int n = sc.nextInt(); // vertices
        Solver solver = new Solver("crystal maze");
        IntVar[] v = VF.enumeratedArray("v",n,1,n,solver); // n variables

        while (sc.hasNext()){
            int i = sc.nextInt();
            int j = sc.nextInt();
            solver.post(ICF.distance(v[i],v[j],">",1));
        }

        sc.close();

        for (int i=0;i<n-1;i++)
            for (int j=i+1;j<n;j++)
                solver.post(ICF.arithm(v[i],"!=",v[j]));

        System.out.println(solver.findSolution());
        for (int i=0;i<n;i++) System.out.println(v[i].getValue());
        System.out.println("nodes: " + solver.getMeasures().getNodeCount(
                           "    cpu: " + solver.getMeasures().getTimeCount());
    }
}
```



```java
import java.io.File;
import java.io.IOException;
import java.util.Scanner;
import org.chocosolver.solver.*;
import org.chocosolver.solver.variables.*;
import org.chocosolver.solver.constraints.*;


public class CrystalMaze {


    public static void main(String[] args) throws IOException {
        Scanner sc = new Scanner(new File(args[0]));
        int n = sc.nextInt(); // vertices
        Solver solver = new Solver("crystal maze");
        IntVar[] v = VF.enumeratedArray("v",n,1,n,solver); // n variables with values 1 to n

        while (sc.hasNext()){
            int i = sc.nextInt();
            int j = sc.nextInt();
            solver.post(ICF.distance(v[i],v[j],">",1));
        }

        sc.close();

        for (int i=0;i<n-1;i++)
            for (int j=i+1;j<n;j++)
                solver.post(ICF.arithm(v[i],"!=",v[j]));

        System.out.println(solver.findSolution());
        for (int i=0;i<n;i++) System.out.println(v[i].getValue());
        System.out.println("nodes: " + solver.getMeasures().getNodeCount() +
                           "   cpu: " + solver.getMeasures().getTimeCount());
    }
}
```

cm8.txt

```
8
0 1
0 5
0 6
1 2
1 6
1 7
2 3
2 6
2 7
3 4
3 7
4 5
4 6
4 7
5 6
5 7
6 7
```

```java
import java.io.File;
import java.io.IOException;
import java.util.Scanner;
import org.chocosolver.solver.*;
import org.chocosolver.solver.variables.*;
import org.chocosolver.solver.constraints.*;


public class CrystalMaze {


    public static void main(String[] args) throws IOException {
        Scanner sc = new Scanner(new File(args[0]));
        int n = sc.nextInt(); // vertices
        Solver solver = new Solver("crystal maze");
        IntVar[] v = VF.enumeratedArray("v",n,1,n,solver); // n variables with values 1 to n

        while (sc.hasNext()){
            int i = sc.nextInt();
            int j = sc.nextInt();
            solver.post(ICF.distance(v[i],v[j],">",1));
        }

        sc.close();

        for (int i=0;i<n-1;i++)
            for (int j=i+1;j<n;j++)
                solver.post(ICF.arithm(v[i],"!=",v[j]));

        System.out.println(solver.findSolution());
        for (int i=0;i<n;i++) System.out.println(v[i].getValue());
        System.out.println("nodes: " + solver.getMeasures().getNodeCount() +
                           "   cpu: " + solver.getMeasures().getTimeCount());
    }
}
```

```java
import java.io.File;
import java.io.IOException;
import java.util.Scanner;
import org.chocosolver.solver.*;
import org.chocosolver.solver.variables.*;
import org.chocosolver.solver.constraints.*;


public class CrystalMaze {


    public static void main(String[] args) throws IOException {
        Scanner sc = new Scanner(new File(args[0]));
        int n = sc.nextInt(); // vertices
        Solver solver = new Solver("crystal maze");
        IntVar[] v = VF.enumeratedArray("v",n,1,n,solver); // n variables with values 1 to n

        while (sc.hasNext()){
            int i = sc.nextInt();
            int j = sc.nextInt();
            solver.post(ICF.distance(v[i],v[j],">",1));
        }

        sc.close();

        for (int i=0;i<n-1;i++)
            for (int j=i+1;j<n;j++)
                solver.post(ICF.arithm(v[i],"!=",v[j]));

        System.out.println(solver.findSolution());
        for (int i=0;i<n;i++) System.out.println(v[i].getValue());
        System.out.println("nodes: " + solver.getMeasures().getNodeCount() +
                           "   cpu: " + solver.getMeasures().getTimeCount());
    }
}
```

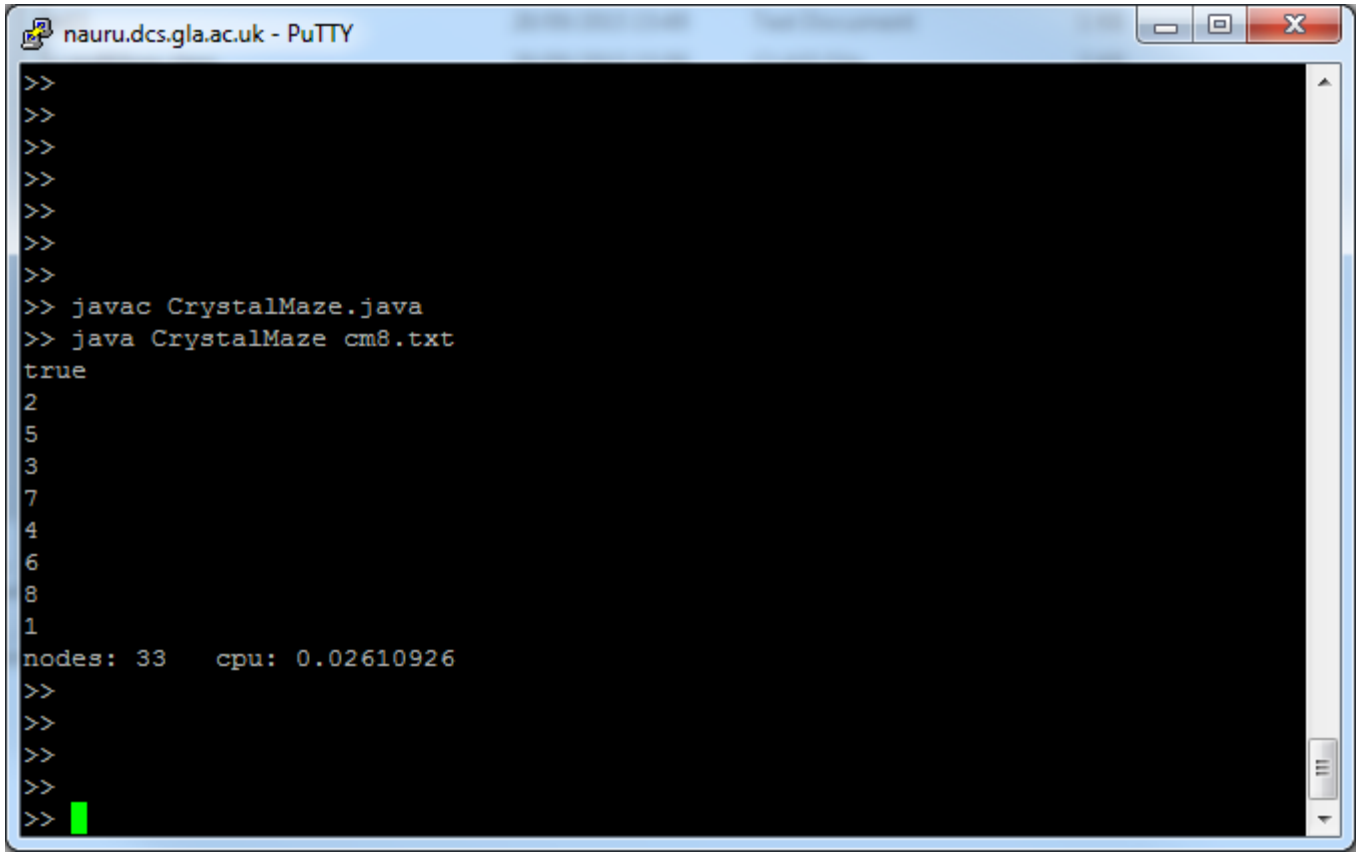
```java
import java.io.File;
import java.io.IOException;
import java.util.Scanner;
import org.chocosolver.solver.*;
import org.chocosolver.solver.variables.*;
import org.chocosolver.solver.constraints.*;


public class CrystalMaze {


    public static void main(String[] args) throws IOException {
        Scanner sc = new Scanner(new File(args[0]));
        int n = sc.nextInt(); // vertices
        Solver solver = new Solver("crystal maze");
        IntVar[] v = VF.enumeratedArray("v",n,1,n,solver); // n variables with values 1 to n

        while (sc.hasNext()){
            int i = sc.nextInt();
            int j = sc.nextInt();
            solver.post(ICF.distance(v[i],v[j],">",1));
        }

        sc.close();

        for (int i=0;i<n-1;i++)
            for (int j=i+1;j<n;j++)
                solver.post(ICF.arithm(v[i],"!=",v[j]));

        System.out.println(solver.findSolution());
        for (int i=0;i<n;i++) System.out.println(v[i].getValue());
        System.out.println("nodes: " + solver.getMeasures().getNodeCount() +
                           "   cpu: " + solver.getMeasures().getTimeCount());
    }
}
```

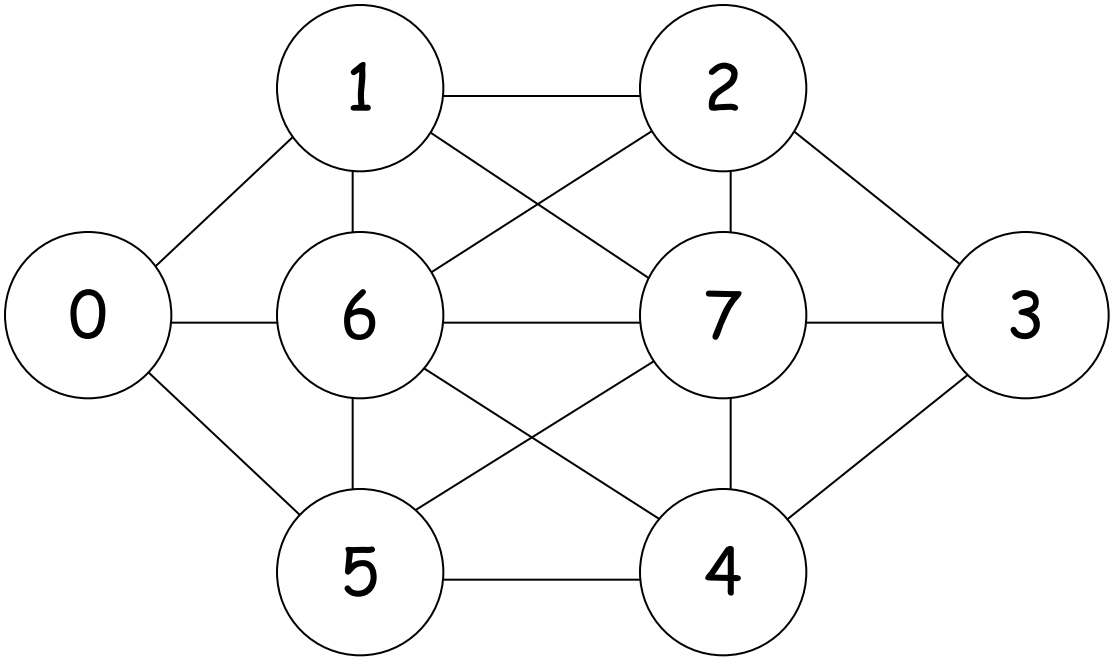
```java
import java.io.File;
import java.io.IOException;
import java.util.Scanner;
import org.chocosolver.solver.*;
import org.chocosolver.solver.variables.*;
import org.chocosolver.solver.constraints.*;


public class CrystalMaze {


    public static void main(String[] args) throws IOException {
        Scanner sc = new Scanner(new File(args[0]));
        int n = sc.nextInt(); // vertices
        Solver solver = new Solver("crystal maze");
        IntVar[] v = VF.enumeratedArray("v",n,1,n,solver); // n variables with values 1 to n

        while (sc.hasNext()){
            int i = sc.nextInt();
            int j = sc.nextInt();
            solver.post(ICF.distance(v[i],v[j],">",1));
        }

        sc.close();

        for (int i=0;i<n-1;i++)
            for (int j=i+1;j<n;j++)
                solver.post(ICF.arithm(v[i],"!=",v[j]));

        System.out.println(solver.findSolution());
        for (int i=0;i<n;i++) System.out.println(v[i].getValue());
        System.out.println("nodes: " + solver.getMeasures().getNodeCount() +
                           "   cpu: " + solver.getMeasures().getTimeCount());
    }
}
```

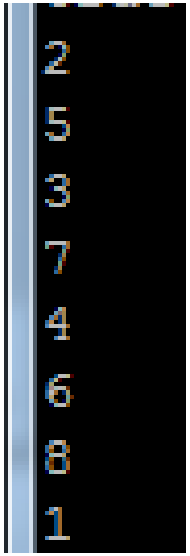
```java
import java.io.File;
import java.io.IOException;
import java.util.Scanner;
import org.chocosolver.solver.*;
import org.chocosolver.solver.variables.*;
import org.chocosolver.solver.constraints.*;


public class CrystalMaze {


    public static void main(String[] args) throws IOException {
        Scanner sc = new Scanner(new File(args[0]));
        int n = sc.nextInt(); // vertices
        Solver solver = new Solver("crystal maze");
        IntVar[] v = VF.enumeratedArray("v",n,1,n,solver); // n variables with values 1 to n

        while (sc.hasNext()){
            int i = sc.nextInt();
            int j = sc.nextInt();
            solver.post(ICF.distance(v[i],v[j],">",1));
        }

        sc.close();

        for (int i=0;i<n-1;i++)
            for (int j=i+1;j<n;j++)
                solver.post(ICF.arithm(v[i],"!=",v[j]));

        System.out.println(solver.findSolution());
        for (int i=0;i<n;i++) System.out.println(v[i].getValue());
        System.out.println("nodes: " + solver.getMeasures().getNodeCount() +
                           "   cpu: " + solver.getMeasures().getTimeCount());
    }
}
```

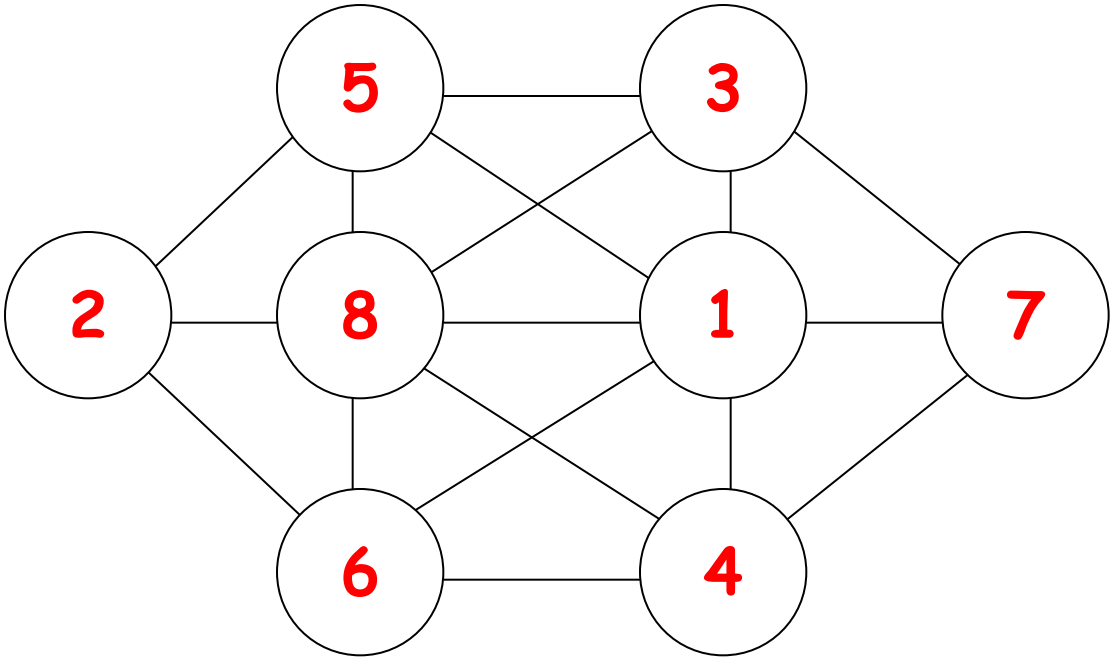
```java
import java.io.File;
import java.io.IOException;
import java.util.Scanner;
import org.chocosolver.solver.*;
import org.chocosolver.solver.variables.*;
import org.chocosolver.solver.constraints.*;


public class CrystalMaze {


    public static void main(String[] args) throws IOException {
        Scanner sc = new Scanner(new File(args[0]));
        int n = sc.nextInt(); // vertices
        Solver solver = new Solver("crystal maze");
        IntVar[] v = VF.enumeratedArray("v",n,1,n,solver); // n variables with values 1 to n

        while (sc.hasNext()){
            int i = sc.nextInt();
            int j = sc.nextInt();
            solver.post(ICF.distance(v[i],v[j],">",1));
        }

        sc.close();

        for (int i=0;i<n-1;i++)
            for (int j=i+1;j<n;j++)
                solver.post(ICF.arithm(v[i],"!=",v[j]));

        System.out.println(solver.findSolution());
        for (int i=0;i<n;i++) System.out.println(v[i].getValue());
        System.out.println("nodes: " + solver.getMeasures().getNodeCount() +
                           "   cpu: " + solver.getMeasures().getTimeCount());
    }
}
```

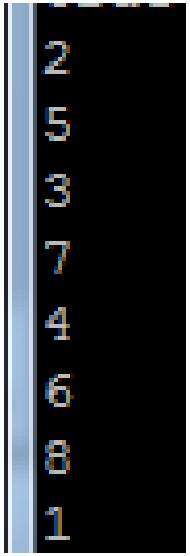
Compile & Run


nauru.dcs.gla.ac.uk - PuTTY
>>
>>
>>
>>
>>
>>
>>
>> javac CrystalMaze.java
>> java CrystalMaze cm8.txt
true
2
5
3
7
4
6
8
1
nodes: 33 cpu: 0.02610926
>>
>>
>>
>>
>>


1
2
0
6
7
3
5
4


2
5
3
7
4
6
8
1


5
3
2
8
1
7
6
4


2
5
3
7
4
6
8
1

So, what IS a constraint program?

Possible answers

It's a program that generates variables and constraints to represent a problem

It's a program that creates a model of a problem and then uses search and heuristics to solve the problem

It's a program that compiles some problem into a representation as CSP