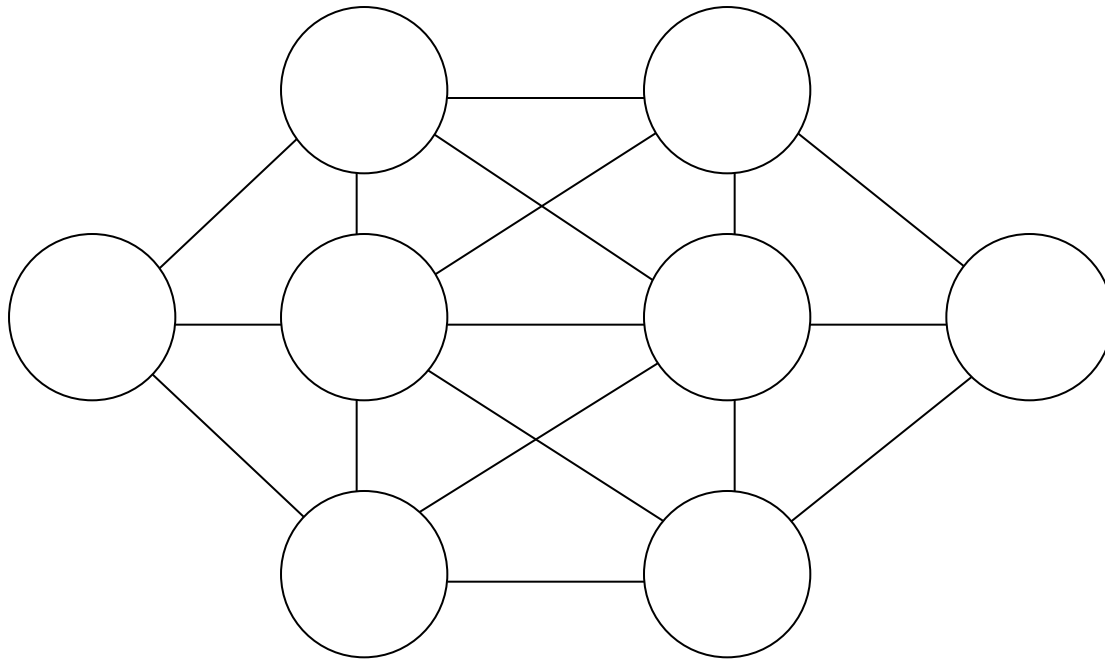
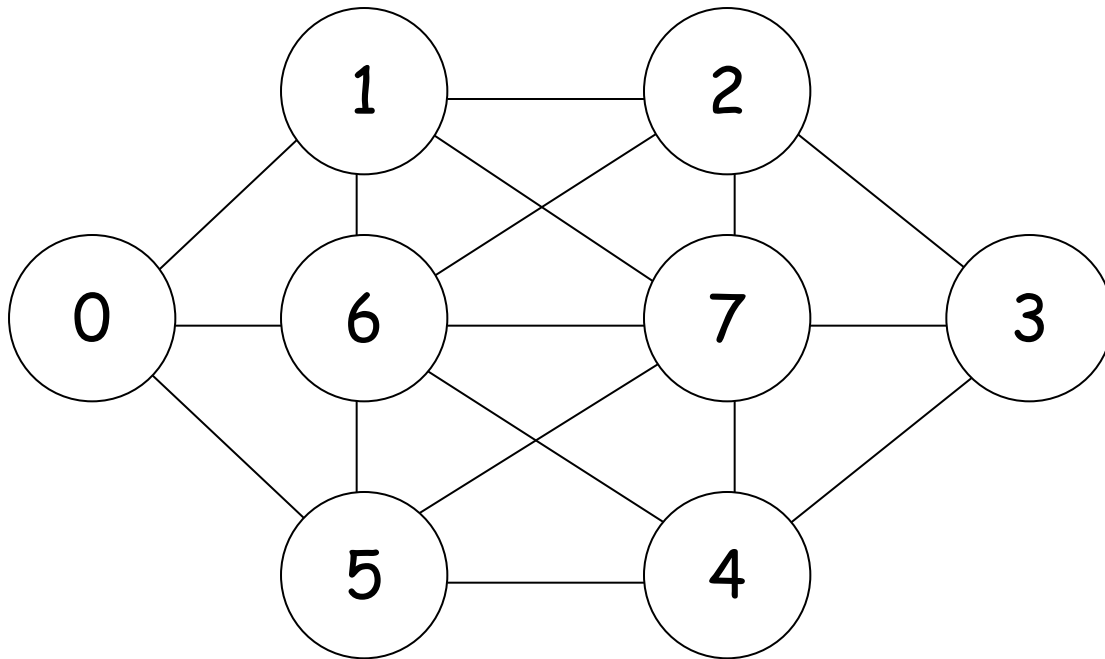


Crystal Maze

coded in minizinc

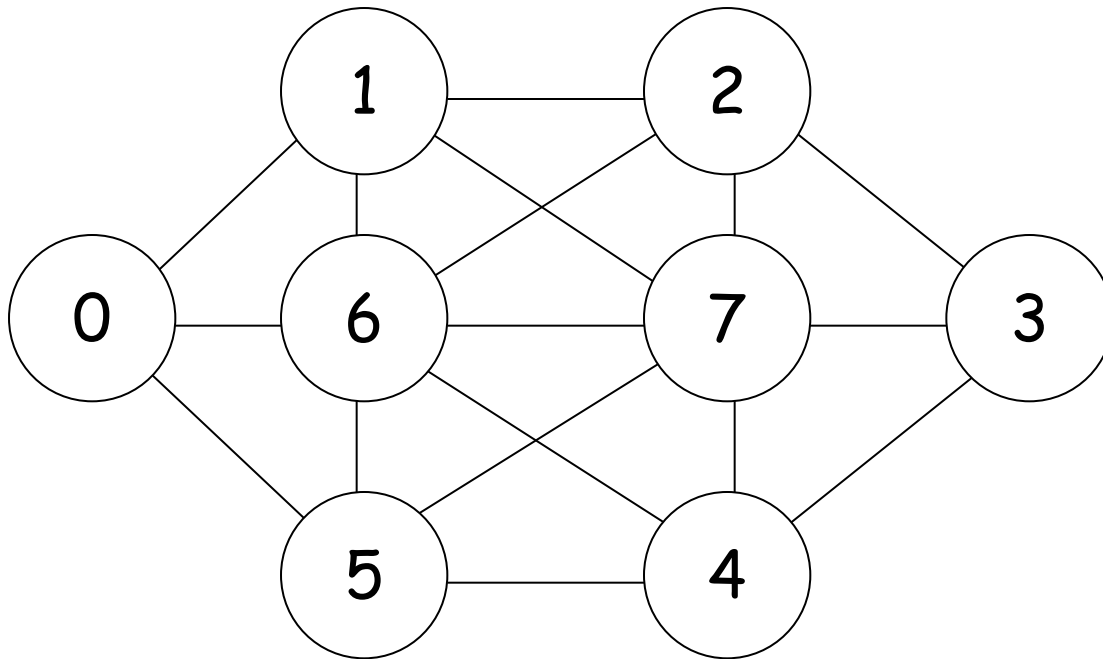


Put a different number in each circle (1 to 8) such that adjacent circles cannot take consecutive numbers



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



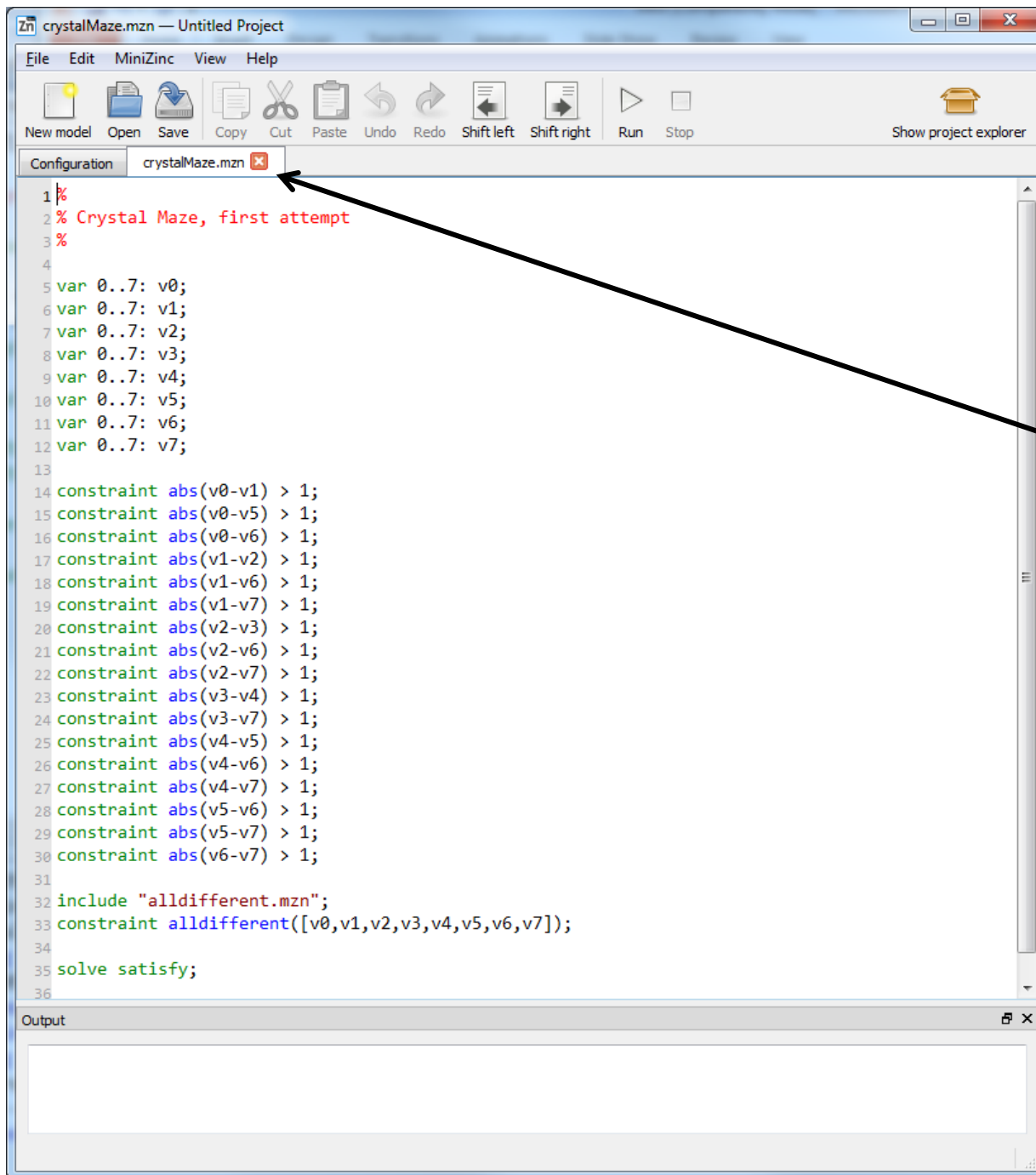
Put a different number in each circle (0 to 7) such that adjacent circles cannot take consecutive numbers

```
1 %  
2 % Crystal Maze, first attempt  
3 %  
4  
5 var 0..7: v0;  
6 var 0..7: v1;  
7 var 0..7: v2;  
8 var 0..7: v3;  
9 var 0..7: v4;  
10 var 0..7: v5;  
11 var 0..7: v6;  
12 var 0..7: v7;  
13  
14 constraint abs(v0-v1) > 1;  
15 constraint abs(v0-v5) > 1;  
16 constraint abs(v0-v6) > 1;  
17 constraint abs(v1-v2) > 1;  
18 constraint abs(v1-v6) > 1;  
19 constraint abs(v1-v7) > 1;  
20 constraint abs(v2-v3) > 1;  
21 constraint abs(v2-v6) > 1;  
22 constraint abs(v2-v7) > 1;  
23 constraint abs(v3-v4) > 1;  
24 constraint abs(v3-v7) > 1;  
25 constraint abs(v4-v5) > 1;  
26 constraint abs(v4-v6) > 1;  
27 constraint abs(v4-v7) > 1;  
28 constraint abs(v5-v6) > 1;  
29 constraint abs(v5-v7) > 1;  
30 constraint abs(v6-v7) > 1;  
31  
32 include "alldifferent.mzn";  
33 constraint alldifferent([v0,v1,v2,v3,v4,v5,v6,v7]);  
34  
35 solve satisfy;  
36
```

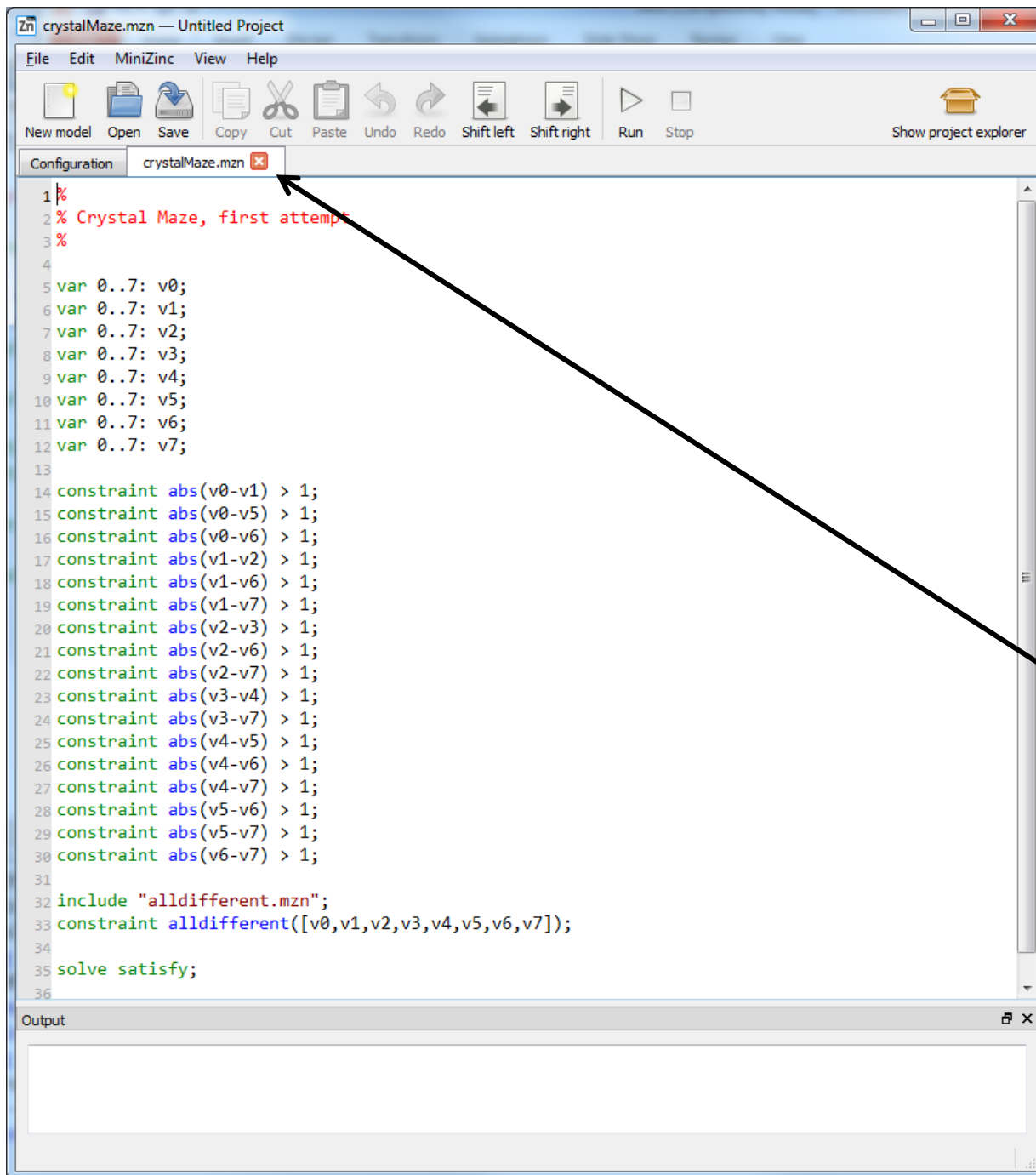
1st stab ...

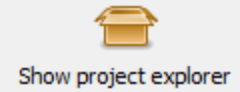
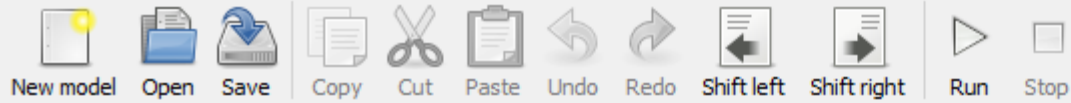
```
1 %  
2 % Crystal Maze, first attempt  
3 %  
4  
5 var 0..7: v0;  
6 var 0..7: v1;  
7 var 0..7: v2;  
8 var 0..7: v3;  
9 var 0..7: v4;  
10 var 0..7: v5;  
11 var 0..7: v6;  
12 var 0..7: v7;  
13  
14 constraint abs(v0-v1) > 1;  
15 constraint abs(v0-v5) > 1;  
16 constraint abs(v0-v6) > 1;  
17 constraint abs(v1-v2) > 1;  
18 constraint abs(v1-v6) > 1;  
19 constraint abs(v1-v7) > 1;  
20 constraint abs(v2-v3) > 1;  
21 constraint abs(v2-v6) > 1;  
22 constraint abs(v2-v7) > 1;  
23 constraint abs(v3-v4) > 1;  
24 constraint abs(v3-v7) > 1;  
25 constraint abs(v4-v5) > 1;  
26 constraint abs(v4-v6) > 1;  
27 constraint abs(v4-v7) > 1;  
28 constraint abs(v5-v6) > 1;  
29 constraint abs(v5-v7) > 1;  
30 constraint abs(v6-v7) > 1;  
31  
32 include "alldifferent.mzn";  
33 constraint alldifferent([v0,v1,v2,v3,v4,v5,v6,v7]);  
34  
35 solve satisfy;  
36
```

We are in the IDE



crystalMaze.mzn



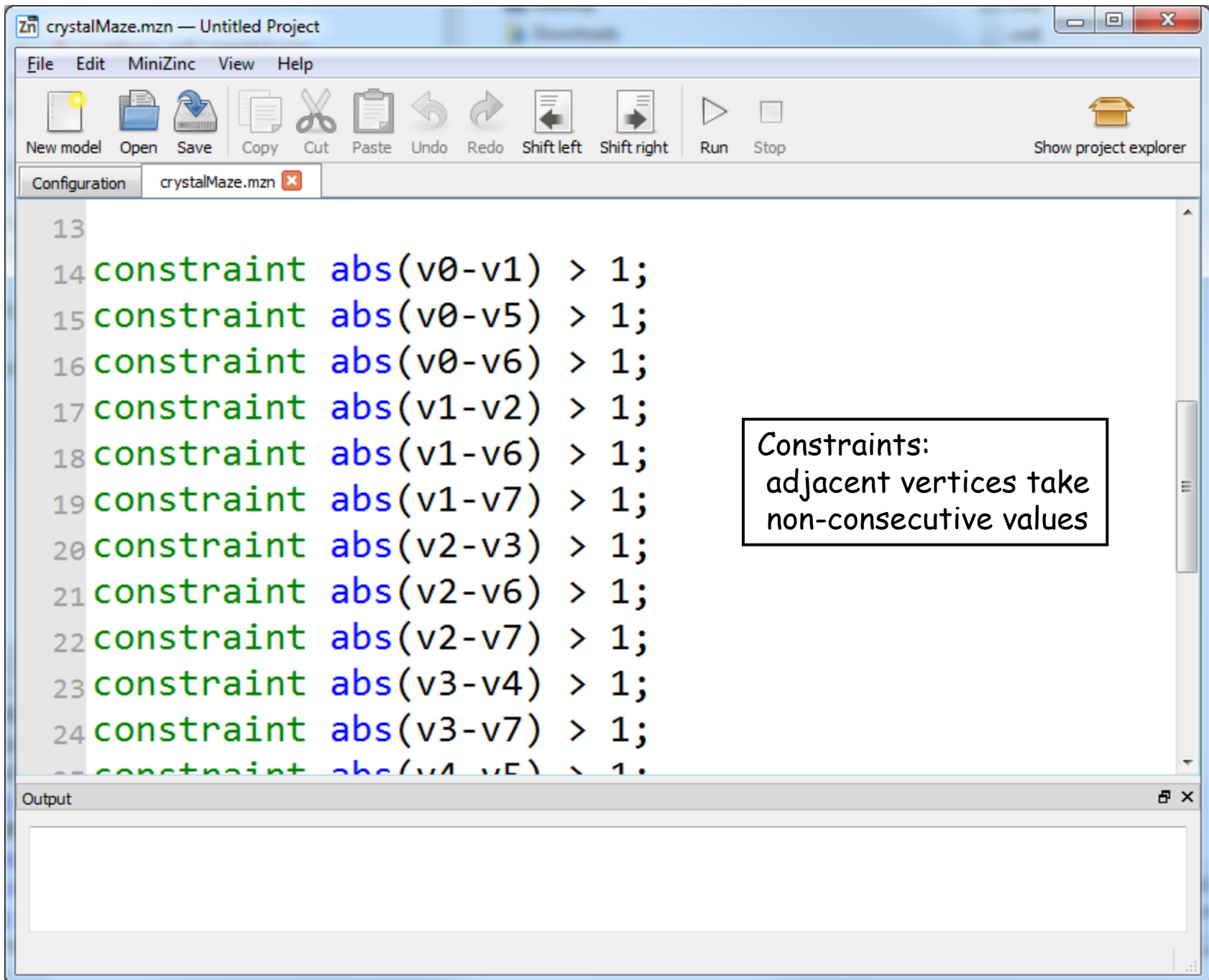


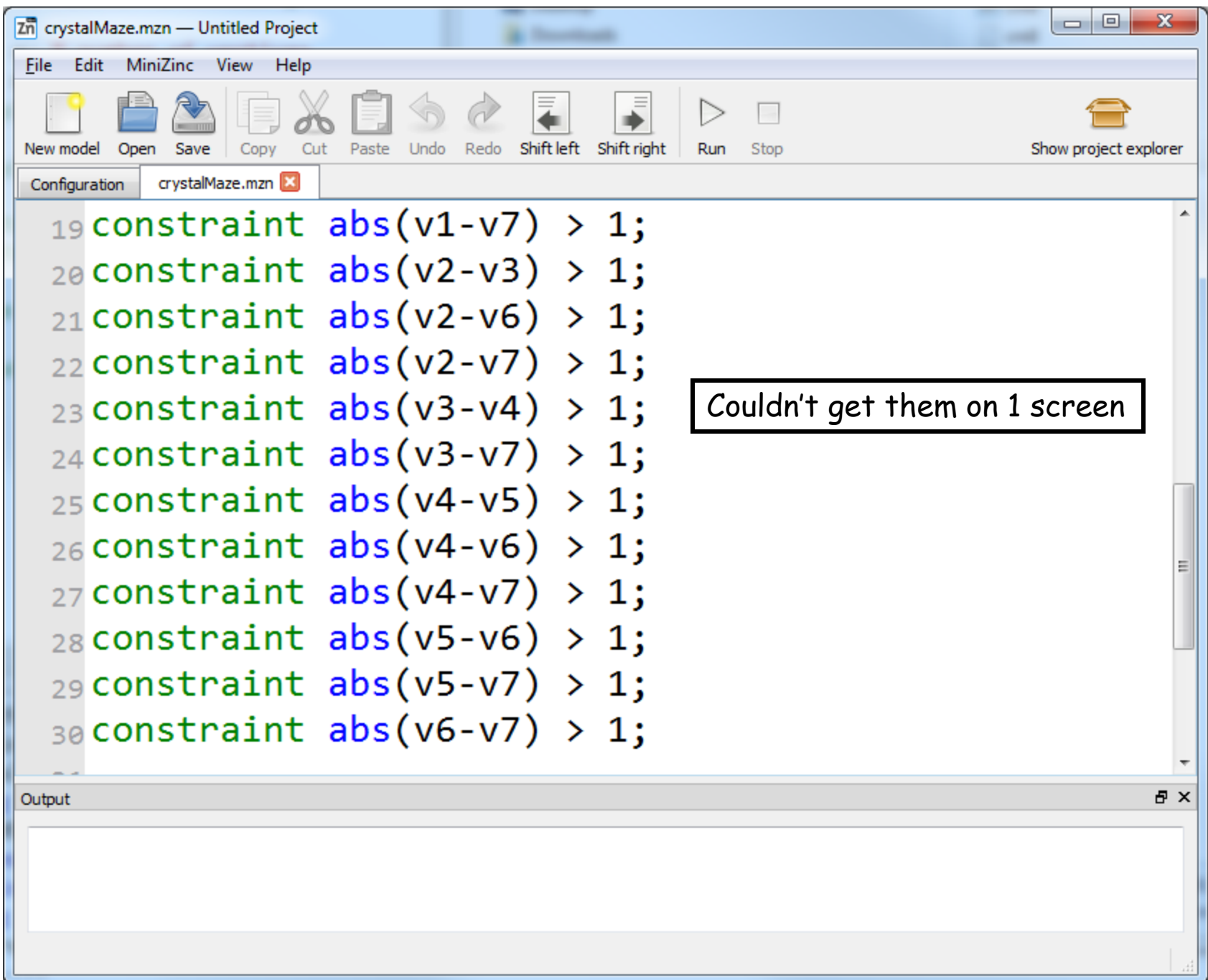
```
1 %  
2 % Crystal Maze, first attempt  
3 %  
4  
5 var 0..7: v0;  
6 var 0..7: v1;  
7 var 0..7: v2;  
8 var 0..7: v3;  
9 var 0..7: v4;  
10 var 0..7: v5;  
11 var 0..7: v6;  
12 var 0..7: v7;
```

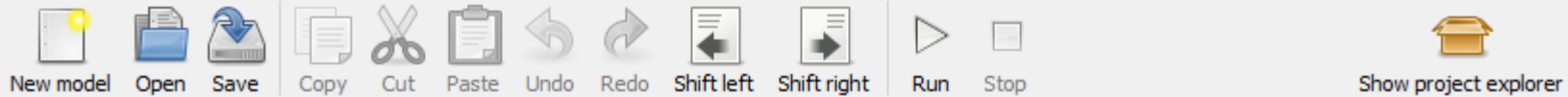
That's a comment

```
1 %  
2 % Crystal Maze, first attempt  
3 %  
4  
5 var 0..7: v0;  
6 var 0..7: v1;  
7 var 0..7: v2;  
8 var 0..7: v3;  
9 var 0..7: v4;  
10 var 0..7: v5;  
11 var 0..7: v6;  
12 var 0..7: v7;
```

Constrained integer variables:
v0 to v7
take values in the range 0 to 7

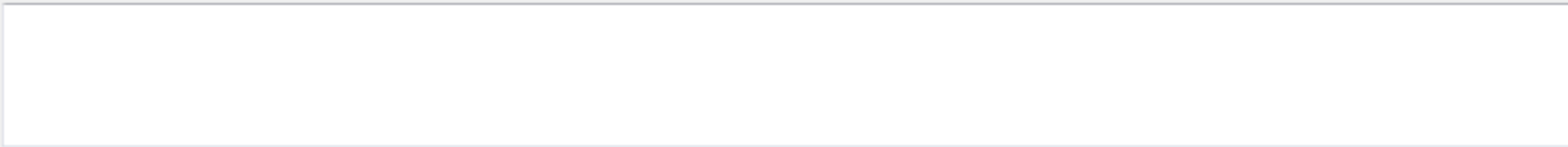
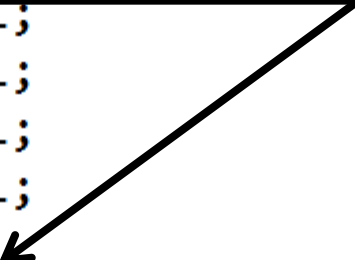


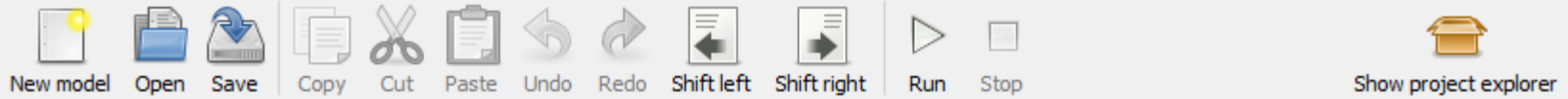




```
24 constraint abs(v3-v7) > 1;
25 constraint abs(v4-v5) > 1;
26 constraint abs(v4-v6) > 1;
27 constraint abs(v4-v7) > 1;
28 constraint abs(v5-v6) > 1;
29 constraint abs(v5-v7) > 1;
30 constraint abs(v6-v7) > 1;
31
32 include "alldifferent.mzn";
33 constraint alldifferent([v0,v1,v2,v3,v4,v5,v6,v7]);
34
35 solve satisfy;
36
```

All vertices take different values

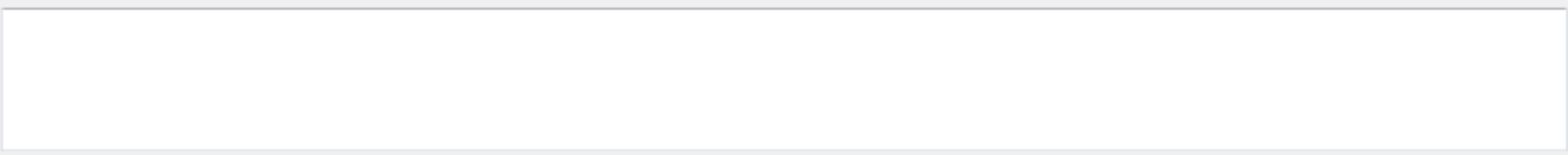


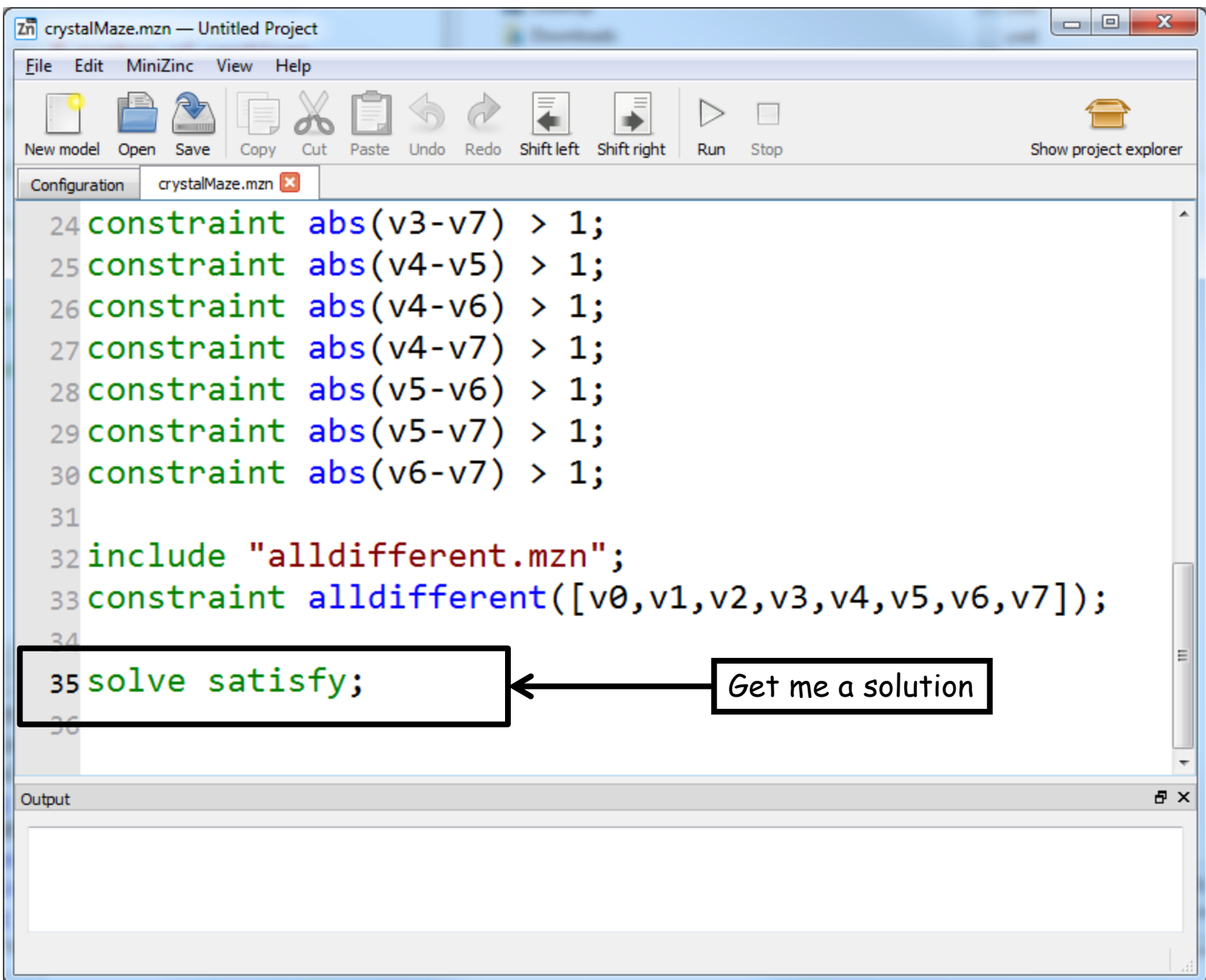


```
24 constraint abs(v3-v7) > 1;
25 constraint abs(v4-v5) > 1;
26 constraint abs(v4-v6) > 1;
27 constraint abs(v4-v7) > 1;
28 constraint abs(v5-v6) > 1;
29 constraint abs(v5-v7) > 1;
30 constraint abs(v6-v7) > 1;
31
32 include "alldifferent.mzn";
33 constraint alldifferent([v0,v1,v2,v3,v4,v5,v6,v7]);
34
35 solve satisfy;
36
```

That's an array!

[v0,v1,v2,v3,v4,v5,v6,v7]





Crystal Maze Solver Interface

File Edit MiniZinc View Help

New model Open Save Copy Cut Paste Undo Redo Shift left Shift right Run Stop Show project explorer

Configuration crystalMaze.mzn

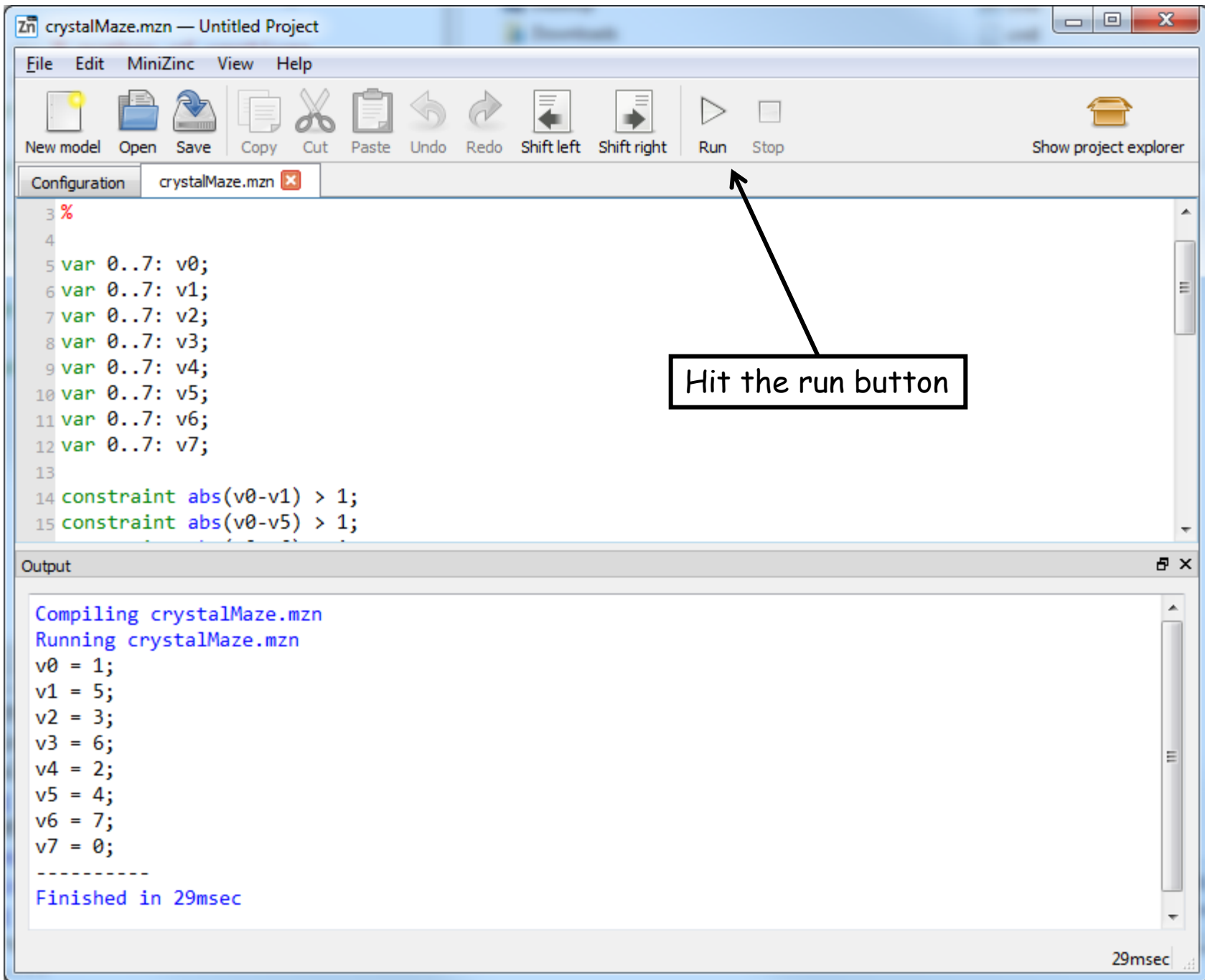
```
3 %
4
5 var 0..7: v0;
6 var 0..7: v1;
7 var 0..7: v2;
8 var 0..7: v3;
9 var 0..7: v4;
10 var 0..7: v5;
11 var 0..7: v6;
12 var 0..7: v7;
13
14 constraint abs(v0-v1) > 1;
15 constraint abs(v0-v5) > 1;
```

Hit the run button

Output

```
Compiling crystalMaze.mzn
Running crystalMaze.mzn
v0 = 1;
v1 = 5;
v2 = 3;
v3 = 6;
v4 = 2;
v5 = 4;
v6 = 7;
v7 = 0;
-----
Finished in 29msec
```

29msec

The image shows a screenshot of the MiniZinc IDE. The window title is "Zn crystalMaze.mzn — Untitled Project". The menu bar includes "File", "Edit", "MiniZinc", "View", and "Help". The toolbar contains icons for "New model", "Open", "Save", "Copy", "Cut", "Paste", "Undo", "Redo", "Shift left", "Shift right", "Run", and "Stop". The "Run" button is a play icon, and the "Stop" button is a square icon. A callout box with the text "Hit the run button" and an arrow points to the "Run" button. The main editor area shows the MiniZinc code for a maze solver. The code includes variable declarations for v0 through v7 and two constraints: abs(v0-v1) > 1 and abs(v0-v5) > 1. The output window at the bottom shows the compilation and execution of the code, resulting in a solution: v0 = 1, v1 = 5, v2 = 3, v3 = 6, v4 = 2, v5 = 4, v6 = 7, v7 = 0. The execution finished in 29msec.

Crystal Maze Solver Interface

File Edit MiniZinc View Help

New model Open Save Copy Cut Paste Undo Redo Shift left Shift right Run Stop Show project explorer

Configuration crystalMaze.mzn

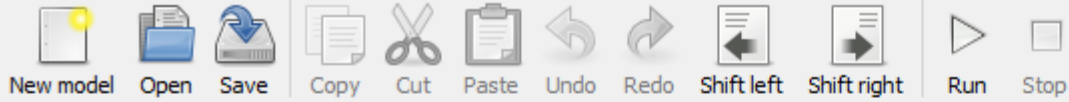
```
3 %
4
5 var 0..7: v0;
6 var 0..7: v1;
7 var 0..7: v2;
8 var 0..7: v3;
9 var 0..7: v4;
10 var 0..7: v5;
11 var 0..7: v6;
12 var 0..7: v7;
13
14 constraint abs(v0-v1) > 1;
15 constraint abs(v0-v5) > 1;
```

Output

```
Compiling crystalMaze.mzn
Running crystalMaze.mzn
v0 = 1;
v1 = 5;
v2 = 3;
v3 = 6;
v4 = 2;
v5 = 4;
v6 = 7;
v7 = 0;
-----
Finished in 29msec
```

29msec

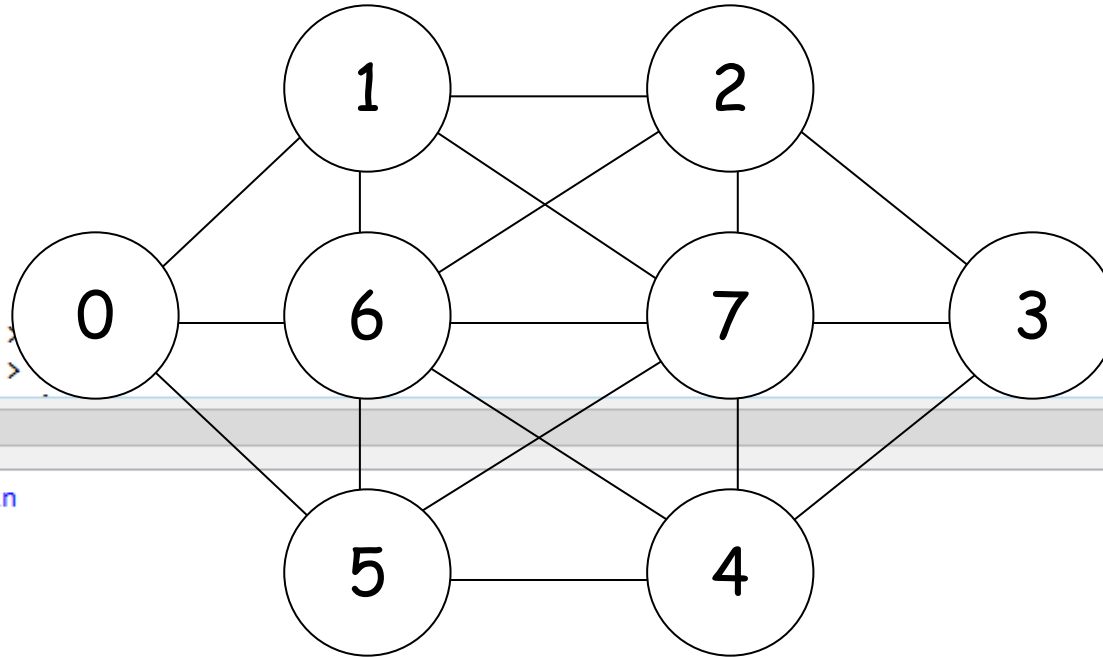
Here's a solution (not unique).



```

3 %
4
5 var 0..7: v0;
6 var 0..7: v1;
7 var 0..7: v2;
8 var 0..7: v3;
9 var 0..7: v4;
10 var 0..7: v5;
11 var 0..7: v6;
12 var 0..7: v7;
13
14 constraint abs(v0-v1) > 0;
15 constraint abs(v0-v5) > 0;

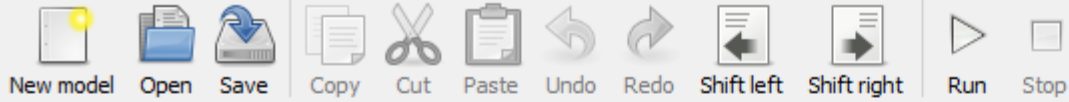
```



```

Compiling crystalMaze.mzn
Running crystalMaze.mzn
v0 = 1;
v1 = 5;
v2 = 3;
v3 = 6;
v4 = 2;
v5 = 4;
v6 = 7;
v7 = 0;
-----
Finished in 29msec

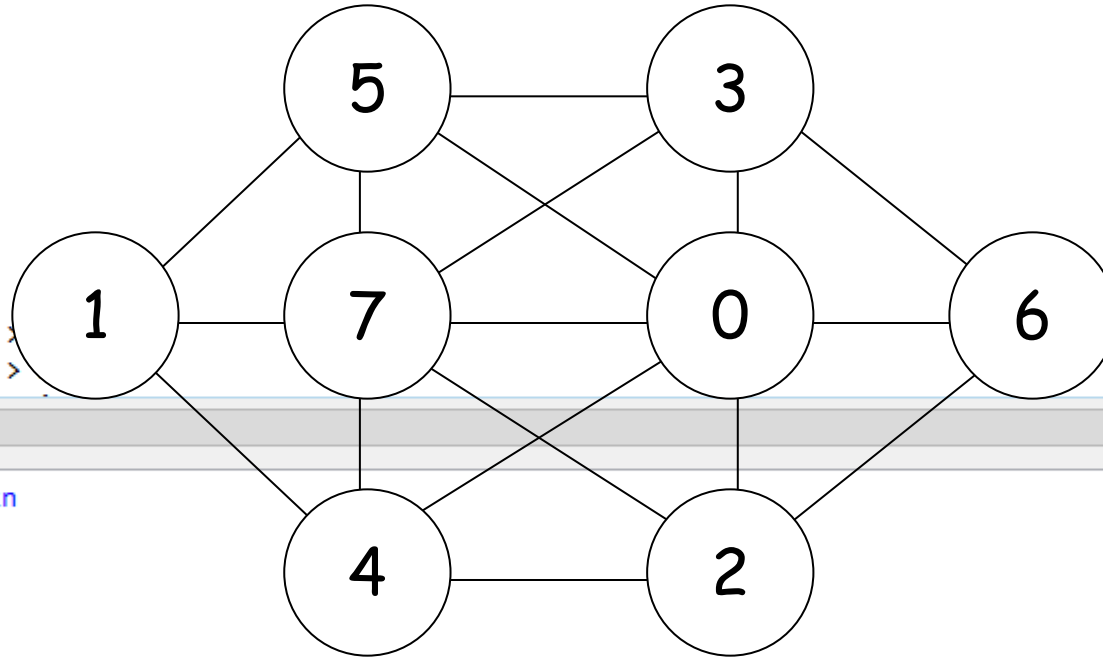
```



```

3 %
4
5 var 0..7: v0;
6 var 0..7: v1;
7 var 0..7: v2;
8 var 0..7: v3;
9 var 0..7: v4;
10 var 0..7: v5;
11 var 0..7: v6;
12 var 0..7: v7;
13
14 constraint abs(v0-v1) > 0;
15 constraint abs(v0-v5) > 0;

```

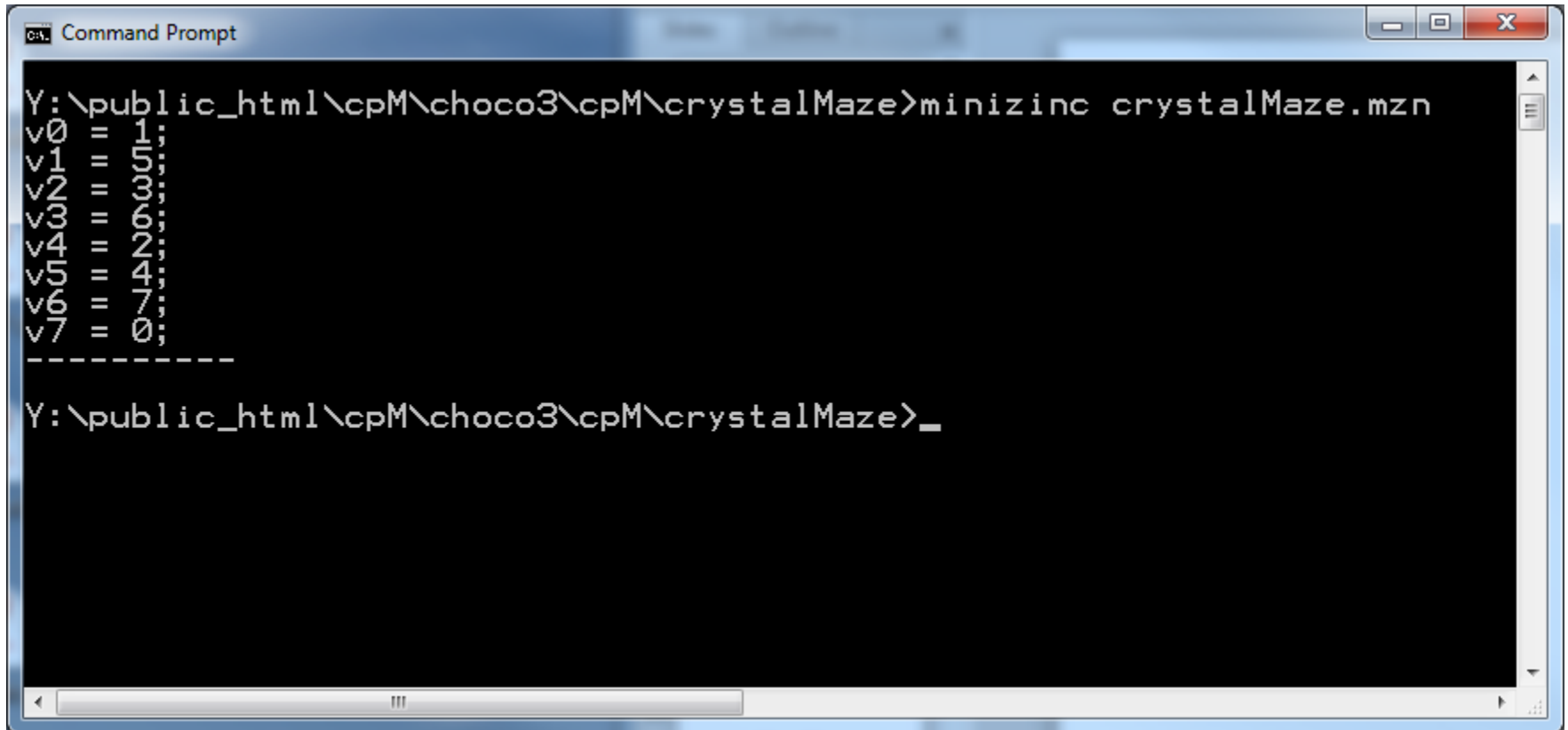


```

Compiling crystalMaze.mzn
Running crystalMaze.mzn
v0 = 1;
v1 = 5;
v2 = 3;
v3 = 6;
v4 = 2;
v5 = 4;
v6 = 7;
v7 = 0;
-----
Finished in 29msec

```

Can use command line



```
Command Prompt
Y:\public_html\cpM\choco3\cpM\crystalMaze>minizinc crystalMaze.mzn
v0 = 1;
v1 = 5;
v2 = 3;
v3 = 6;
v4 = 2;
v5 = 4;
v6 = 7;
v7 = 0;
-----
Y:\public_html\cpM\choco3\cpM\crystalMaze>_
```

Can get all solutions

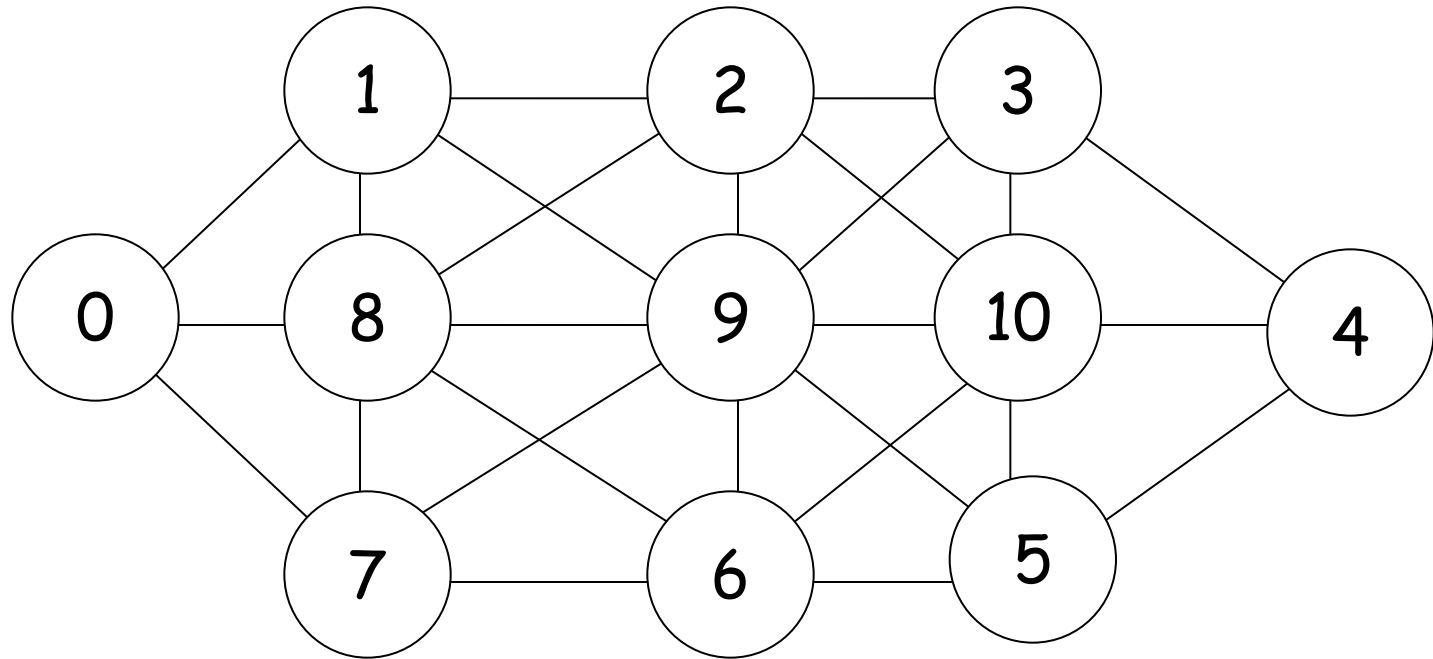
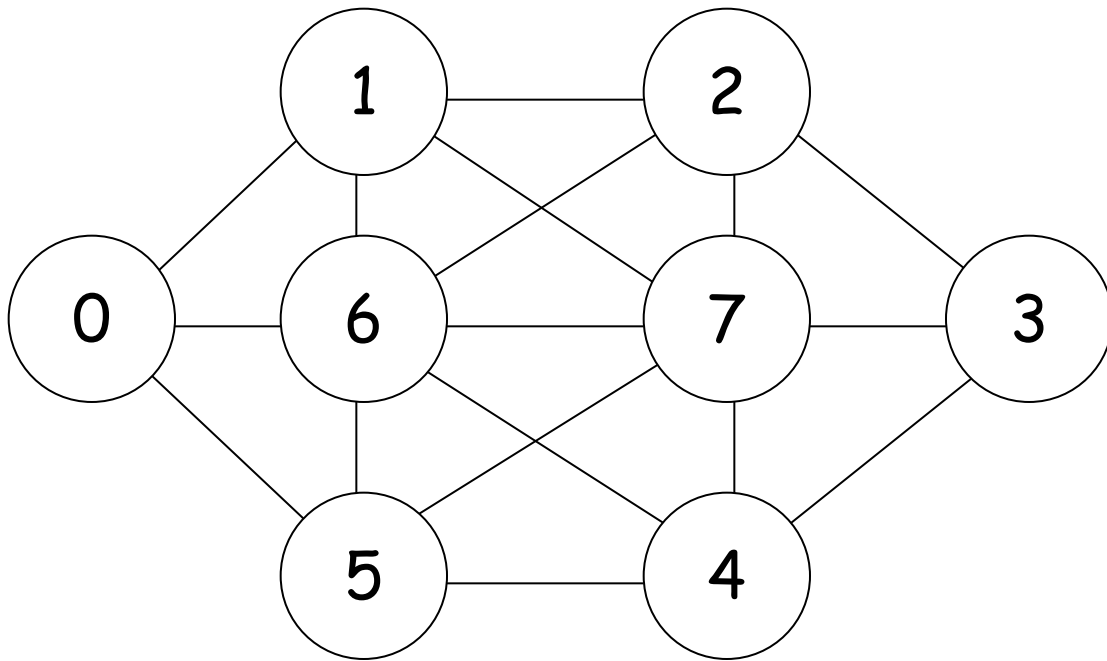
```
Y:\public_html\cpM\choco3\cpM\crystalMaze>minizinc crystalMaze.mzn -a
v0 == 1;
v1 == 5;
v2 == 3;
v3 == 6;
v4 == 2;
v5 == 4;
v6 == 7;
v7 == 0;
-----
v0 == 1;
v1 == 4;
v2 == 2;
v3 == 6;
v4 == 3;
v5 == 5;
v6 == 7;
v7 == 0;
-----
v0 == 6;
v1 == 3;
v2 == 5;
v3 == 1;
v4 == 4;
v5 == 2;
v6 == 0;
v7 == 7;
-----
v0 == 6;
v1 == 2;
v2 == 4;
v3 == 1;
v4 == 5;
v5 == 3;
v6 == 0;
v7 == 7;
-----
=====
Y:\public_html\cpM\choco3\cpM\crystalMaze>_
```

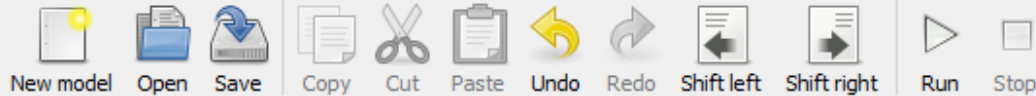
Can get statistics

```
cmd: Command Prompt
Y:\public_html\cpM\choco3\cpM\crystalMaze>minizinc crystalMaze.mzn -s
v0 = 1;
v1 = 5;
v2 = 3;
v3 = 6;
v4 = 2;
v5 = 4;
v6 = 7;
v7 = 0;
-----
%
% 74 choice points explored.
%
Y:\public_html\cpM\choco3\cpM\crystalMaze>
```

Let's make a more general model

One program to solve both

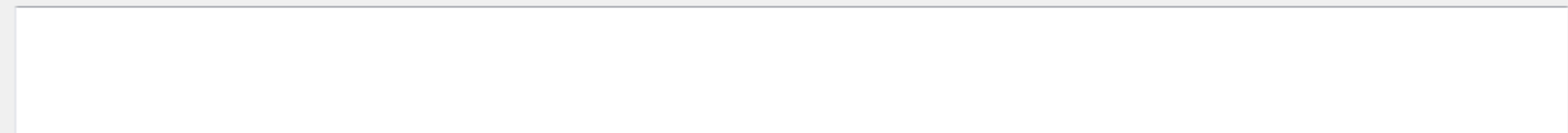




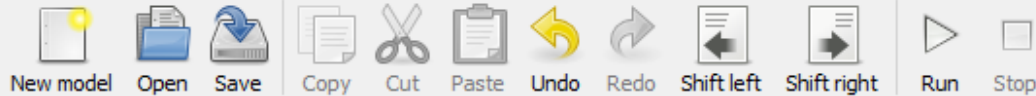
Configuration crystalMaze1.mzn

```
1 %
2 % Crystal Maze, second attempt, more general
3 %
4
5 include "alldifferent.mzn";
6
7 int: n; % number of vertices
8 int: m; % number of edges
9 array[1..m,1..2] of int: edge; % adjacency
10
11 array[0..n-1] of var 0..n-1: v; % vertices v[0] to v[n-1]
12
13 constraint forall(e in 1..m)(abs(v[edge[e,1]] - v[edge[e,2]]) > 1);
14 constraint alldifferent(v);
15
16 solve satisfy;
17
18 output ["v = \v");
19
```

Output



Ready.

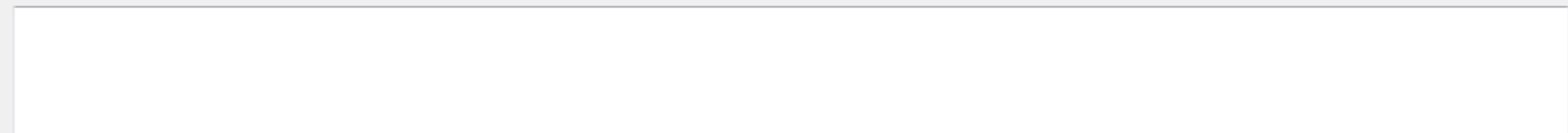


Configuration crystalMaze1.mzn

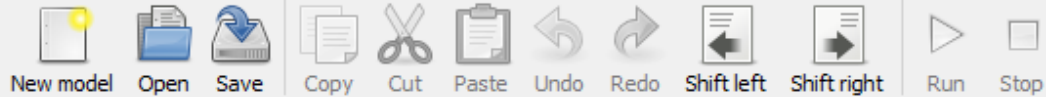
```
1 %  
2 % Crystal Maze, second attempt, more general  
3 %  
4  
5 include "alldifferent.mzn";  
6  
7 int: n; % number of vertices  
8 int: m; % number of edges  
9 array[1..m,1..2] of int: edge; % adjacency  
10  
11 array[0..n-1] of var 0..n-1: v; % vertices v[0] to v[n-1]  
12  
13 constraint forall(e in 1..m)(abs(v[edge[e,1]] - v[edge[e,2]]) > 1);  
14 constraint alldifferent(v);  
15  
16 solve satisfy;  
17  
18 output ["v = \"(v)\"];  
19
```

These are variables

Output



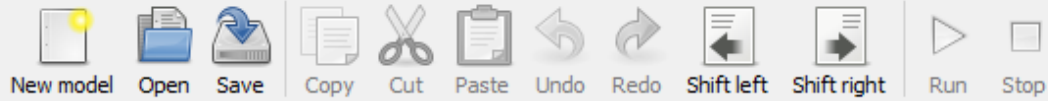
Ready.



```
1 n = 8;  
2 m = 17;  
3 edge = [ |0,1  
4           |0,5  
5           |0,6|  
6           |1,2  
7           |1,6  
8           |1,7  
9           |2,3  
10          |2,6  
11          |2,7  
12          |3,4  
13          |3,7  
14          |4,5  
15          |4,6  
16          |4,7  
17          |5,6  
18          |5,7  
19          |6,7| ]  
20
```

This is the data file, dzn extension

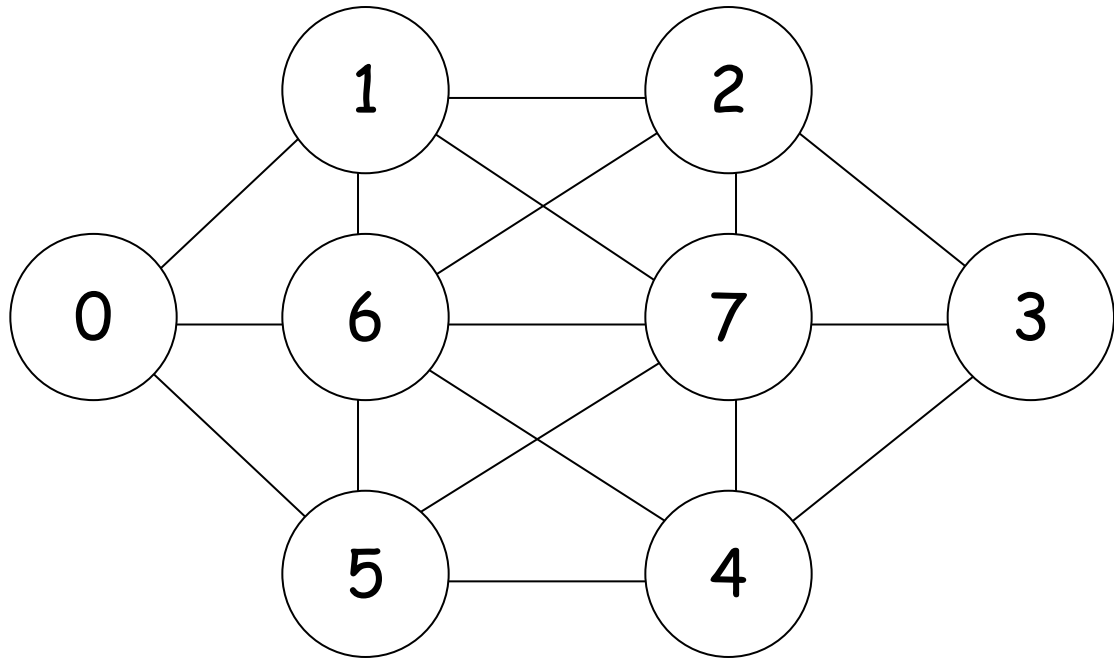


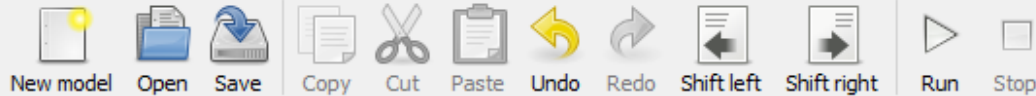


```

1 n = 8;
2 m = 17;
3 edge = [|0,1
4         |0,5
5         |0,6|
6         |1,2
7         |1,6
8         |1,7
9         |2,3
10        |2,6
11        |2,7
12        |3,4
13        |3,7
14        |4,5
15        |4,6
16        |4,7
17        |5,6
18        |5,7
19        |6,7|]
20

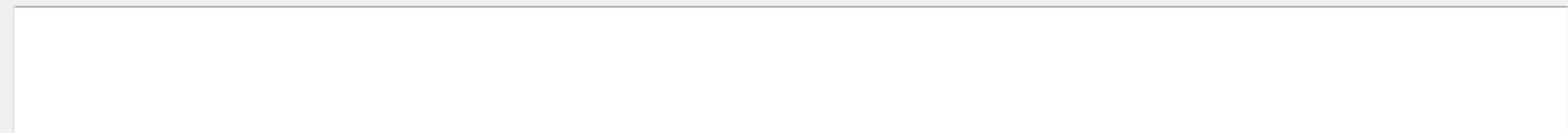
```

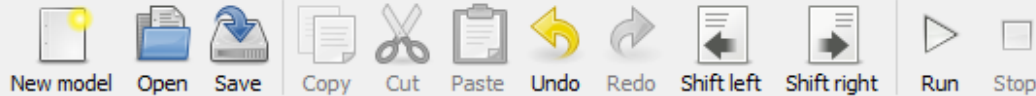




```
1 %
2 % Crystal Maze, second attempt, more general
3 %
4
5 include "alldifferent.mzn";
6
7 int: n; % number of vertices
8 int: m; % number of edges
9 array[1..m,1..2] of int: edge; % adjacency
10
11 array[0..n-1] of var 0..n-1: v; % vertices v[0] to v[n-1]
12
13 constraint forall(e in 1..m)(abs(v[edge[e,1]] - v[edge[e,2]]) > 1);
14 constraint alldifferent(v);
15
16 solve satisfy;
17
18 output ["v = \v");
19
```

I like to place this here





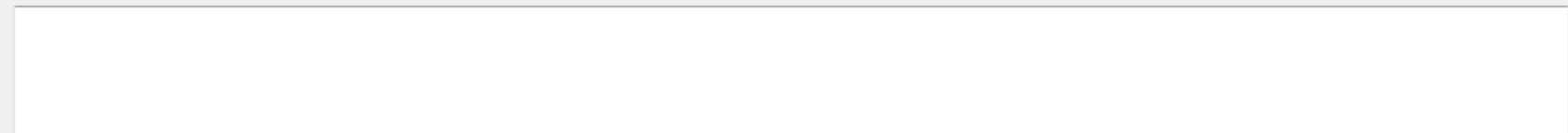
Configuration crystalMaze1.mzn

```
1 %  
2 % Crystal Maze, second attempt, more general  
3 %  
4  
5 include "alldifferent.mzn";  
6  
7 int: n; % number of vertices  
8 int: m; % number of edges  
9 array[1..m,1..2] of int: edge; % adjacency  
10  
11 array[0..n-1] of var 0..n-1: v; % vertices v[0] to v[n-1]  
12  
13 constraint forall(e in 1..m)(abs(v[edge[e,1]] - v[edge[e,2]]) > 1);  
14 constraint alldifferent(v);  
15  
16 solve satisfy;  
17  
18 output ["v = \v");  
19
```

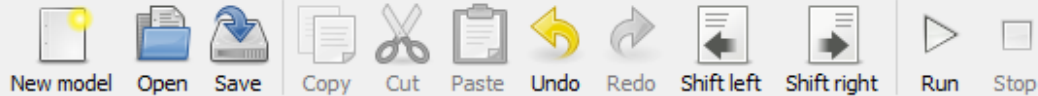
Array of constrained integer variables
 $v[0]$ to $v[n-1]$ where $v[i]$ is in $\{0..n-1\}$

array[0..n-1] of var 0..n-1: v; % vertices v[0] to v[n-1]

Output



Ready.

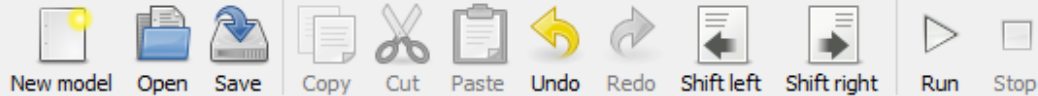


Configuration crystalMaze1.mzn

```
1 %  
2 % Crystal Maze, second attempt, more general  
3 %  
4  
5 include "alldifferent.mzn";  
6  
7 int: n; % number of vertices  
8 int: m; % number of edges  
9 array[1..m,1..2] of int: edge; % adjacency  
10  
11 array[0..n-1] of var 0..n-1: v; % vertices v[0] to v[n-1]  
12  
13 constraint forall(e in 1..m)(abs(v[edge[e,1]] - v[edge[e,2]]) > 1);  
14 constraint alldifferent(v);  
15  
16 solve satisfy;  
17  
18 output ["v = \v");  
19
```

forall ...

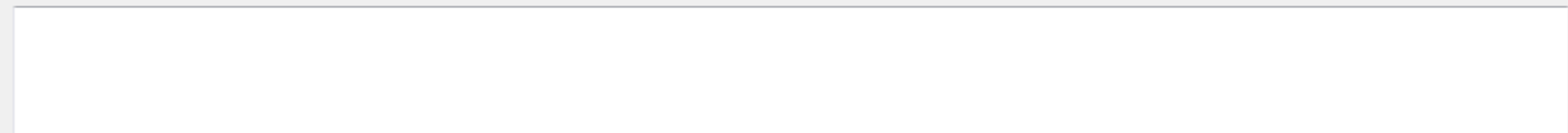
Output

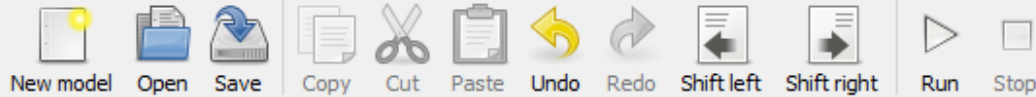


```
1 %
2 % Crystal Maze, second attempt, more general
3 %
4
5 include "alldifferent.mzn";
6
7 int: n; % number of vertices
8 int: m; % number of edges
9 array[1..m,1..2] of int: edge; % adjacency
10
11 array[0..n-1] of var 0..n-1: v; % vertices v[0] to v[n-1]
12
13 constraint forall(e in 1..m)(abs(v[edge[e,1]] - v[edge[e,2]]) > 1);
14 constraint alldifferent(v);
15
16 solve satisfy;
17
18 output ["v = \v");
19
```

(edge[e,1],edge[e,2]) is a pair of vertices

(v[edge[e,1]] - v[edge[e,2]]) > 1;

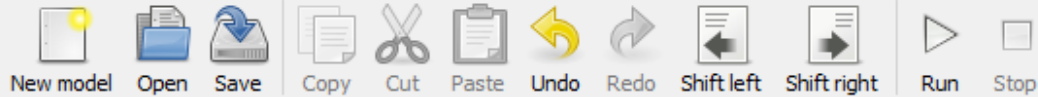




```
1 %  
2 % Crystal Maze, second attempt, more general  
3 %  
4 include "alldifferent.mzn";  
5  
6  
7 int: n; % number of vertices  
8 int: m; % number of edges  
9 array[1..m,1..2] of int: edge; % adjacency  
10  
11 array[0..n-1] of var 0..n-1: v; % vertices v[0] to v[n-1]  
12  
13 constraint forall(e in 1..m)(abs(v[edge[e,1]] - v[edge[e,2]]) > 1);  
14 constraint alldifferent(v);  
15  
16 solve satisfy;  
17  
18 output ["v = \v");  
19
```



All different



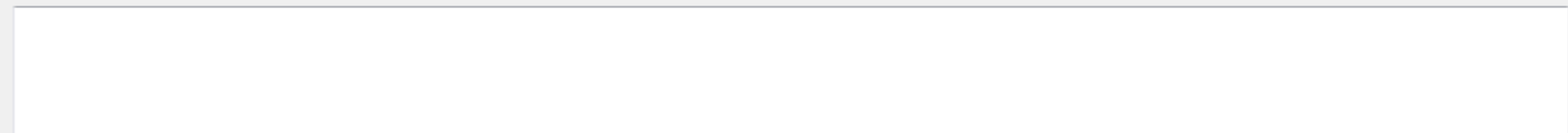
Configuration crystalMaze1.mzn

```
1 %
2 % Crystal Maze, second attempt, more general
3 %
4
5 include "alldifferent.mzn";
6
7 int: n; % number of vertices
8 int: m; % number of edges
9 array[1..m,1..2] of int: edge; % adjacency
10
11 array[0..n-1] of var 0..n-1: v; % vertices v[0] to v[n-1]
12
13 constraint forall(e in 1..m)(abs(v[edge[e,1]] - v[edge[e,2]]) > 1);
14 constraint alldifferent(v);
15
16 solve satisfy;
17
18 output ["v = \v");
19
```

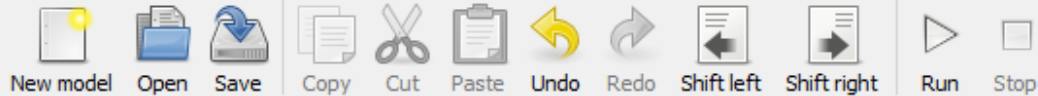
Get me a solution

solve satisfy;

Output



Ready.



Configuration crystalMaze1.mzn

```
1 %
2 % Crystal Maze, second attempt, more general
3 %
4
5 include "alldifferent.mzn";
6
7 int: n; % number of vertices
8 int: m; % number of edges
9 array[1..m,1..2] of int: edge; % adjacency
10
11 array[0..n-1] of var 0..n-1: v; % vertices v[0] to v[n-1]
12
13 constraint forall(e in 1..m)(abs(v[edge[e,1]] - v[edge[e,2]]) > 1);
14 constraint alldifferent(v);
15
16 solve satisfy;
17
18 output ["v = \v"];
19
```

Output solution

Output

Ready.



```

1 %
2 % Crystal Maze, second attempt.
3 %
4
5 include "alldifferent.mzn";
6
7 int: n; % number of v
8 int: m; % number of e
9 array[1..m,1..2] of
10
11 array[0..n-1] of var
12
13 constraint forall(e
14 constraint alldiffer
15
16 solve satisfy;
17
18 output ["v = \(v)"];
19

```

Hit run and it now allows me to select dzn file

Zn Model Parameters

Select data file or input values

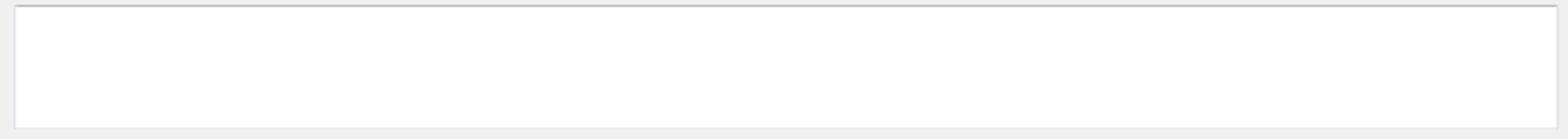
n =

m =

edge =

OK Cancel

> 1);





Configuration crystalMaze1.mzn cm8.dzn

```
1 %
2 % Crystal Maze, second attempt, more general
3 %
4
5 include "alldifferent.mzn";
6
7 int: n; % number of vertices
8 int: m; % number of edges
9 array[1..m,1..2] of int: edge; % adjacency
10
11 array[0..n-1] of var 0..n-1: v; % vertices v[0] to v[n-1]
12
13 constraint forall(e in 1..m)(abs(v[edge[e,1]] - v[edge[e,2]]) > 1);
14 constraint alldifferent(v);
15
16 solve satisfy;
17
18 output ["v = \v"];
19
```

Output

```
Compiling crystalMaze1.mzn, with additional data cm8.dzn
Running crystalMaze1.mzn
v = [1, 5, 3, 6, 2, 4, 7, 0]
-----
Finished in 29msec
```

A solution for cm8.dzn

Also on command line

Command Prompt

```
Y:\public_html\cpM\choco3\cpM\crystalMaze>minizinc crystalMaze1.mzn cm8.dzn  
v = [1, 5, 3, 6, 2, 4, 7, 0]  
-----
```

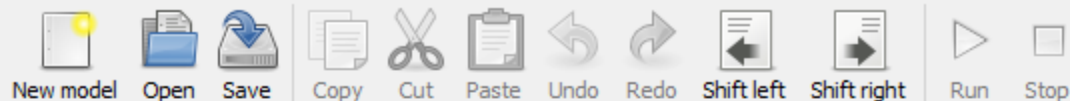
```
Y:\public_html\cpM\choco3\cpM\crystalMaze>_
```

Also on command line (all solutions)

```
Y:\public_html\cpM\choco3\cpM\crystalMaze>minizinc crystalMaze1.mzn cm8.dzn -a
v = [1, 5, 3, 6, 2, 4, 7, 0]
-----
v = [1, 4, 2, 6, 3, 5, 7, 0]
-----
v = [6, 2, 4, 1, 5, 3, 0, 7]
-----
v = [6, 3, 5, 1, 4, 2, 0, 7]
-----
=====
Y:\public_html\cpM\choco3\cpM\crystalMaze>
```

Compare two models ... cool

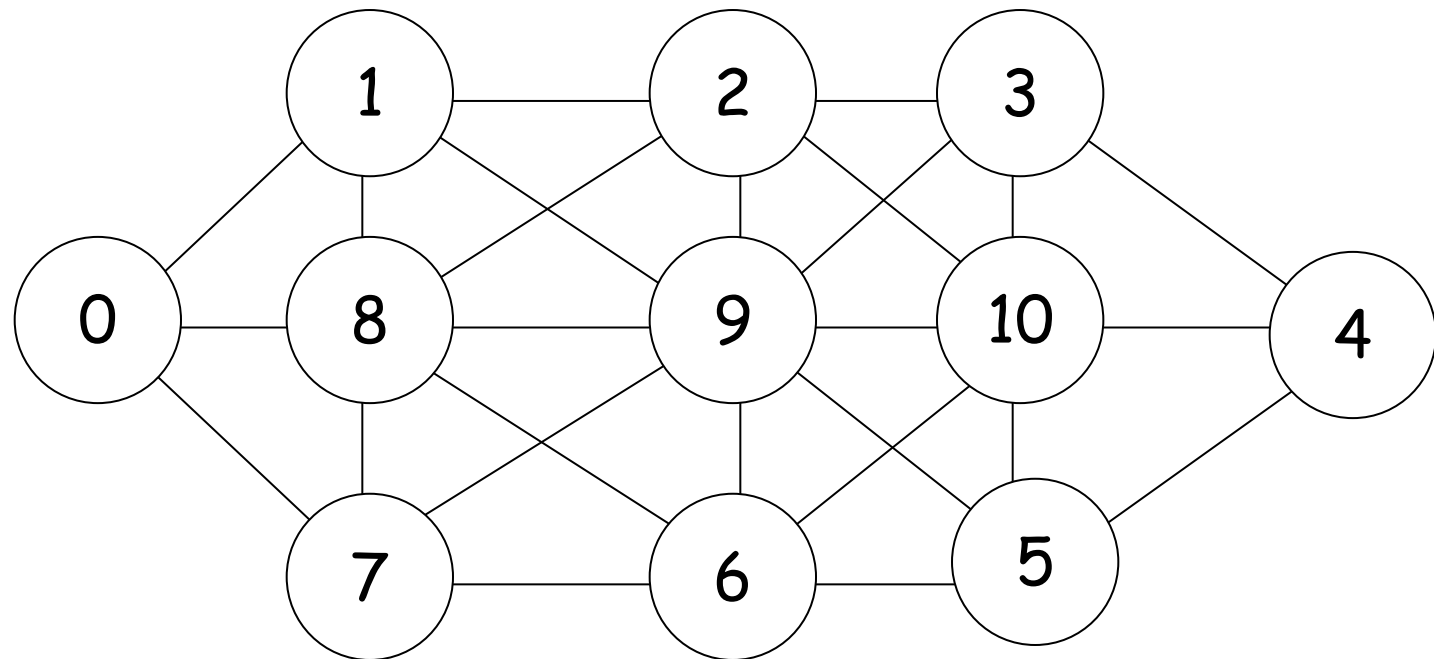
```
ca. Command Prompt
Y:\public_html\cpM\choco3\cpM\crystalMaze>minizinc crystalMaze.mzn -s
v0 = 1;
v1 = 5;
v2 = 3;
v3 = 6;
v4 = 2;
v5 = 4;
v6 = 7;
v7 = 0;
-----
%
% 74 choice points explored.
%
Y:\public_html\cpM\choco3\cpM\crystalMaze>minizinc crystalMaze1.mzn cm8.dzn -s
v = [1, 5, 3, 6, 2, 4, 7, 0]
-----
%
% 74 choice points explored.
%
Y:\public_html\cpM\choco3\cpM\crystalMaze>_
```

```

1 n = 11;
2 m = 26;
3 edge = [0,1
4         |0,8
5         |0,7
6         |1,2
7         |1,9
8         |1,8
9         |2,3
10        |2,10
11        |2,9
12        |2,8
13        |3,4
14        |3,10
15        |3,9
16        |4,10
17        |4,5
18        |5,10
19        |5,9
20        |5,6
21        |6,10
22        |6,9
23        |6,8
24        |6,7
25        |7,9
26        |7,8
27        |8,9
28        |9,10|]
29

```



```

-----
Finished in 29msec
Compiling crystalMaze1.mzn, with additional data cm11.dzn
Running crystalMaze1.mzn

```



```
1 %
2 % Crystal Maze, second attempt, more general
3 %
4
5 include "alldifferent.mzn";
6
7 int: n; % number of vertices
8 int: m; % number of edges
9 array[1..m,1..2] of int: edge; % adjacency
10
11 array[0..n-1] of var 0..n-1: v; % vertices v[0] to v[n-1]
12
13 constraint forall(e in 1..m)(abs(v[edge[e,1]] - v[edge[e,2]]) > 1);
14 constraint alldifferent(v);
15
16 solve satisfy;
17
18 output ["v = \"(v)\""];
19
```

```
v = [1, 5, 3, 6, 2, 4, 7, 0]
-----
Finished in 29msec
Compiling crystalMaze1.mzn, with additional data cm11.dzn
Running crystalMaze1.mzn
v = [1, 8, 6, 10, 4, 9, 5, 7, 3, 0, 2]
-----
Finished in 27msec
```

A solution for cm11.dzn

```
Command Prompt
Y:\public_html\cpM\choco3\cpM\crystalMaze>minizinc crystalMaze1.mzn cm11.dzn
v = [1, 10, 5, 8, 3, 6, 4, 9, 7, 2, 0]
-----
Y:\public_html\cpM\choco3\cpM\crystalMaze>_
```

Ouch! (-a on cm11)

```
Command Prompt
v = [3, 1, 10, 7, 5, 9, 2, 6, 8, 4, 0]
-----
v = [3, 1, 10, 7, 5, 9, 2, 8, 6, 4, 0]
-----
v = [3, 1, 10, 7, 5, 8, 2, 9, 6, 4, 0]
-----
v = [3, 8, 10, 7, 5, 2, 9, 6, 1, 4, 0]
-----
v = [3, 8, 10, 7, 5, 2, 9, 1, 6, 4, 0]
-----
v = [3, 8, 10, 7, 5, 2, 6, 9, 1, 4, 0]
-----
v = [5, 9, 2, 6, 3, 8, 10, 1, 7, 4, 0]
-----
v = [5, 8, 2, 6, 3, 9, 7, 1, 10, 4, 0]
-----
v = [3, 8, 6, 10, 5, 2, 7, 9, 1, 4, 0]
-----
solution received from solver:2:
v = array1d(0..10, [3, 1, 6, 10, 5, 8, 2, 7, 9, 4, 0]);
^
Error: syntax error, unexpected =, expecting ':'
MiniZinc: internal error: not sm.get(): solns2out_base: could not parse solution
flatzinc: I/O error: error writing to output file: Invalid argument
minizinc: evaluation killed by signal 1
Y:\public_html\cpM\choco3\cpM\crystalMaze>
```

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