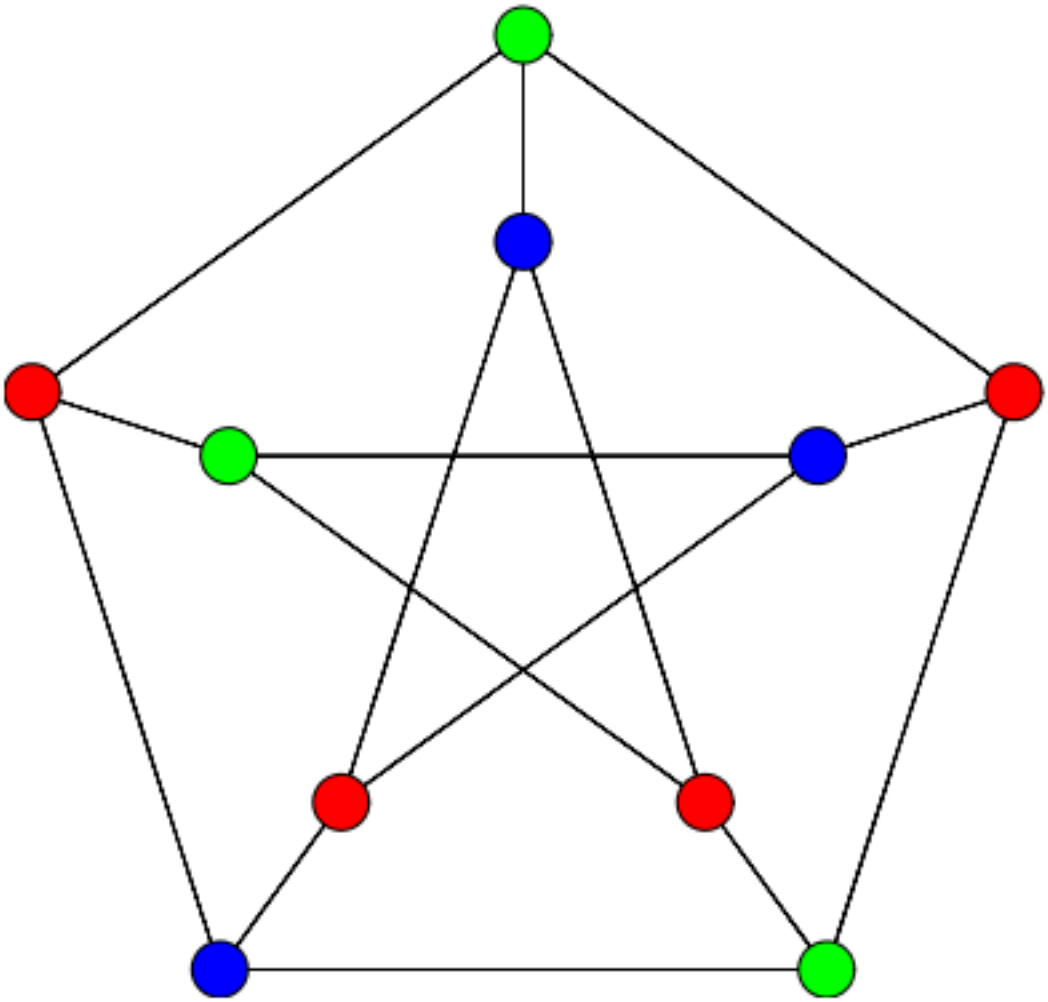
Graph Colouring with SAT

Curtis Bright, Maplesoft



- A *colouring* of a graph is an assignment of colours to its vertices such that every two adjacent vertices are coloured differently. Above is a colouring of the Petersen Graph using three colours.
- The minimum number of colours necessary to colour a graph is known as the *chromatic number* of the graph. It can be computed (along with a colouring using the minimum number of colours) using the ChromaticNumber function of the GraphTheory package.
- Since Maple 2018, the chromatic number function accepts a *method* parameter that controls how the chromatic number is computed. For example, by using *method* = *sat* the chromatic number is found by encoding the problem in Boolean logic and finding the chromatic number

using a SAT solver. This worksheet describes how the graph colouring problem can be encoded into Boolean logic.

- A SAT solver accepts a formula in Boolean logic and returns a satisfying assignment (if one exists). See the Satisfy command of the Logic package for more information.

Generating the constraints

- We'll use the Boolean variables $x_{i,j}$ where i is the index of a colour and j is a vertex of a graph G to represent that the vertex j is coloured with colour i.
- First, we need to specify that each vertex has been assigned a colour. In Boolean logic we express that vertex j has been coloured with one of C colours by the clause $x_{1,j} \vee \cdots \vee x_{C,j}$. These clauses are generated by the following function:

```
# Return constraints saying that each of the vertices 1, ..., N in a graph have been coloured with one of C colours
colourConstraints := proc(N, C)
    local i, j;
    return seq(&or(seq(x[i,j], i=1..C)), j=1..N)
;
end proc:
```

- Next, we need to specify that each vertex has not been assigned two colours. In Boolean logic we express that vertex k has not been assigned two distinct colours by the clauses $\neg x_{i,k} \vee \neg x_{j,k}$ for all distinct colour indices i and j. These clauses are generated by the following function:

```
# Return constraints saying that each of the vert
ices 1, ..., N in a graph have been coloured with
 at most one of C colours
distinctColourConstraints := proc(N, C)
     local i, j, k;
     return seq(seq(seq(&not(x[i,k]) &or &not(x[j
,k]), k=1..N), j=i+1..C), i=1..C);
end proc:
```

- Finally, we need to specify that adjacent vertices are not coloured using the same colour. In Boolean logic we express this through the clauses $\neg x_{i,j} \vee \neg x_{i,k}$ for all colour indices i and all pairs of connected vertices (j, k). These clauses are generated by the following function:

```
with(GraphTheory):
# Return constraints saying that each of the conn
ected vertices in G are coloured differently (usi
ng C distinct colours)
graphConstraints := proc(G, C)
     local i, j, l;
     return seq(seq(&not(x[i,Edges(G)[l][1]]) &or
&not(x[i, Edges(G)[l][2]]), l=1..NumberOfEdges(G)
), i=1..C);
end proc:
```

Generating and visualizing queen graphs

- We now demonstrate how the above encoding is used to colour the "queen graphs".
- The queen graph Q_n encodes the moves that queens can make on an $n \times n$ chessboard. The vertices of the graph are the squares of the chessboard and two vertices are connected exactly when a queen can move between the two squares in a single move, i.e., when the squares are in the same row, column, or diagonal.
- The following function generates the queen graph Q_n:

```
Return the queen graph of an n by n chessboard
```

- To visualize the colouring of a queen graph we use functions that draw coloured queens on a chessboard. The visualizations make use of the drawing commands from the plots and plottools packages.

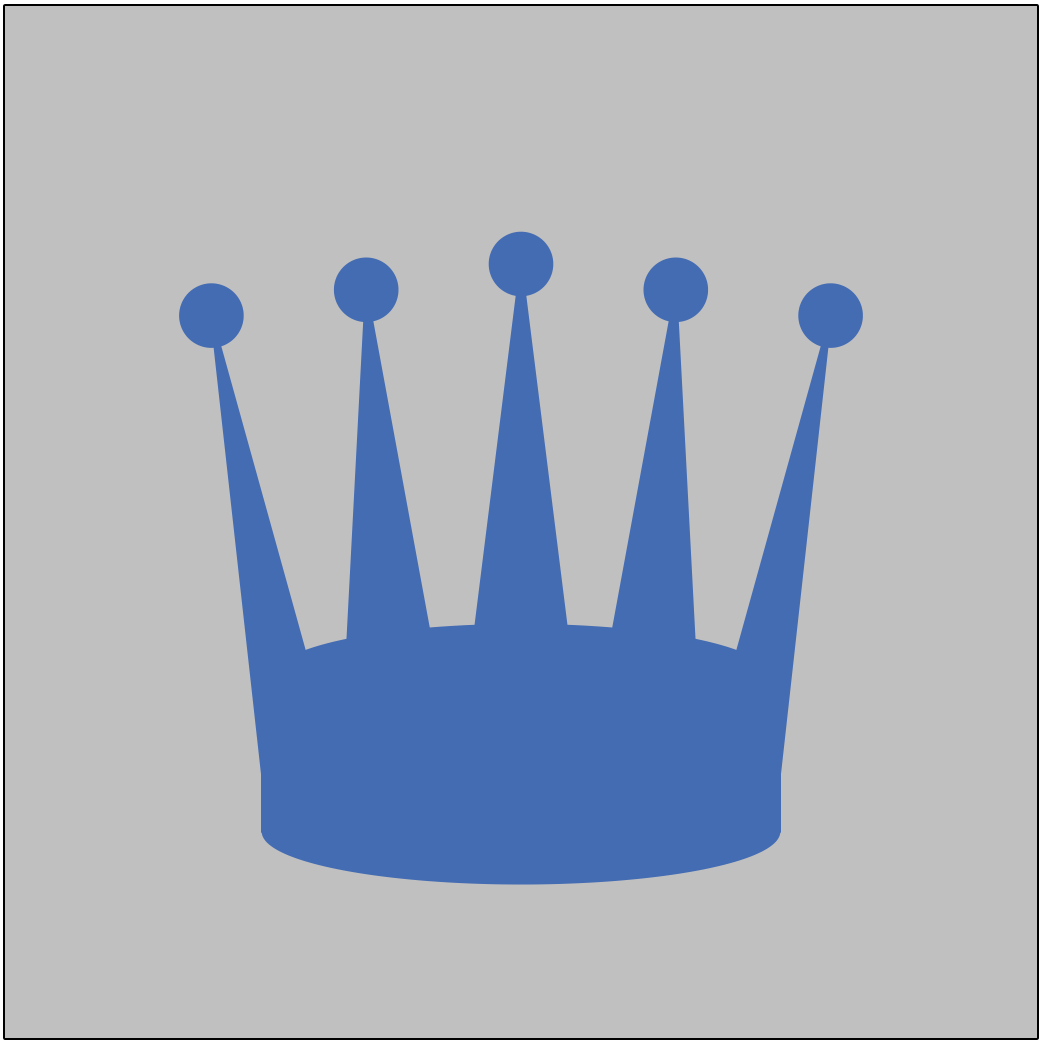
```
with(plots):
```

Finding colourings of queen graphs

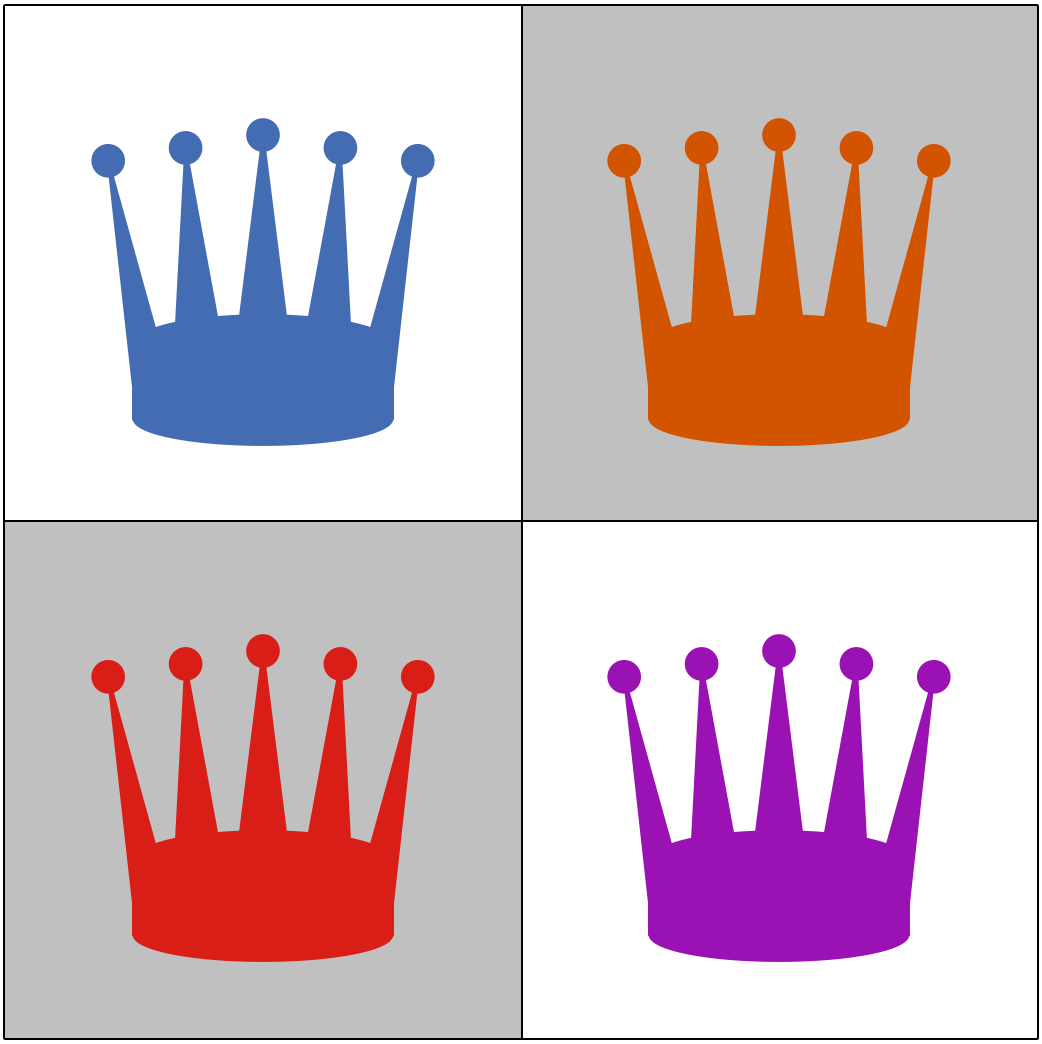
- We use the Satisfy command from the Logic package which finds a satisfying assignment of a logical formula if one exists.
- It is beneficial to initially assign colours to as many vertices as possible. In the queen graphs all squares in the first row are mutually connected, so each square in the first row must be coloured differently. Thus when colouring the queen graph Q_n we include the constraints $x_{i,i}$ for each $1 \le i \le n$ which say that vertex i receives colour i.
- The following code finds a colouring using the fewest number of colours for all queen graphs up to size 8×8; on Maple 2019 this takes less than 10 seconds. Conversely, the ChromaticNumber function required several hours to perform these computations prior to Maple 2019.
- The initial call to Satisfy tries to find a colouring of Q_n using n colours. If such a colouring cannot be found then the number of colours used is increased by 1 until such a colouring exists. The colourings are plotted using the previously defined visualization commands and by reading the satisfying assignment produced by Satisfy.

```
with(ColorTools):
```

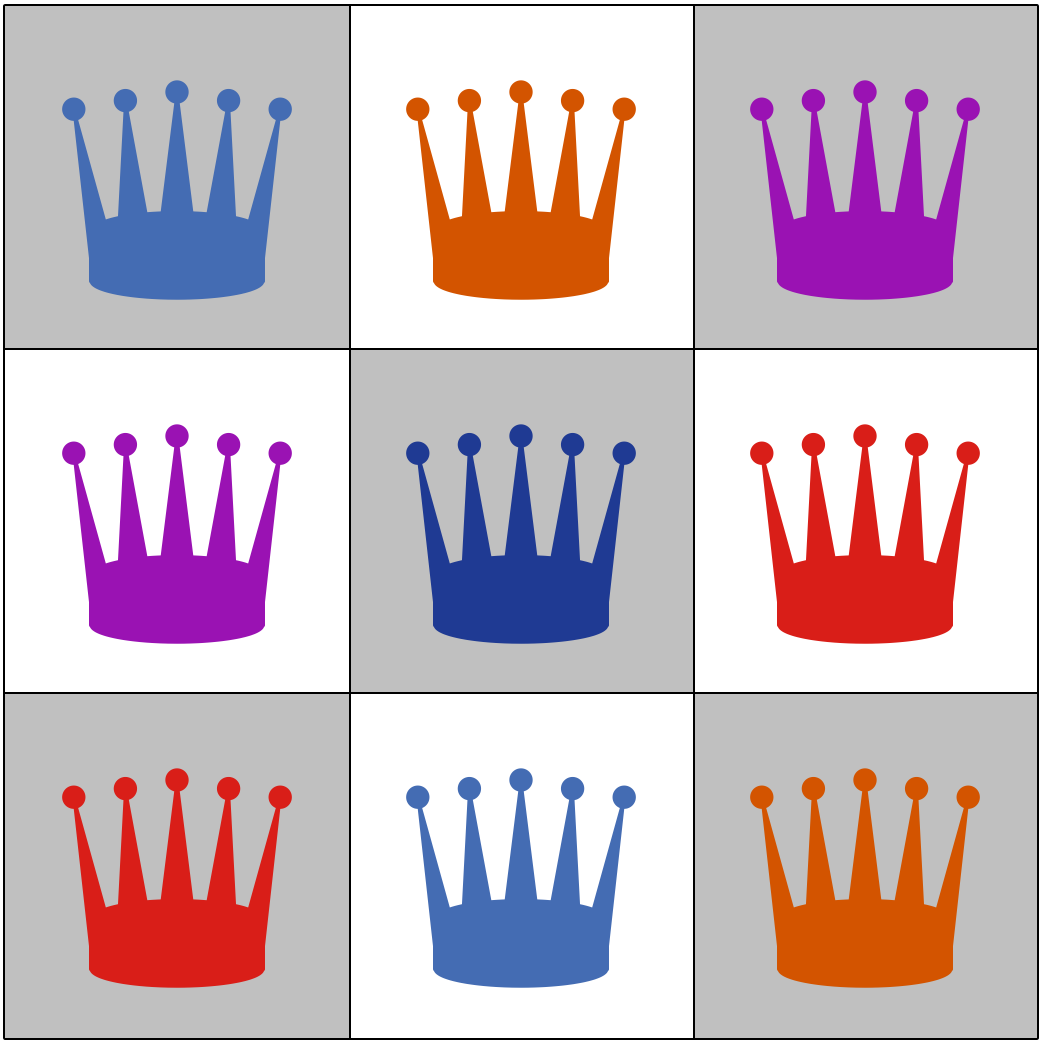
```
The 1 by 1 queen graph is colourable with 1 colours. Time used:
0.29 seconds
```



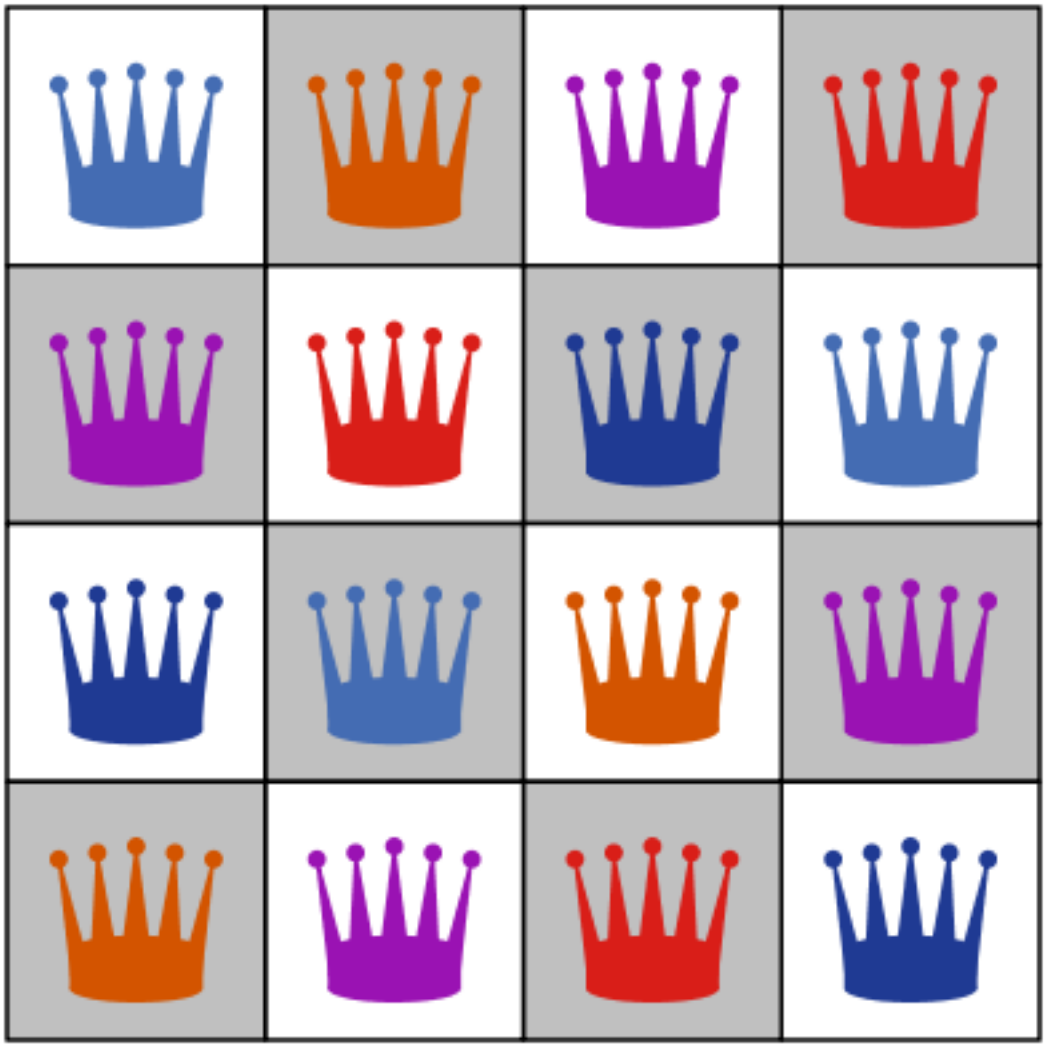
```
The 2 by 2 queen graph is not colourable with 2 colours. Time used:
0.00 seconds
The 2 by 2 queen graph is not colourable with 3 colours. Time used:
0.00 seconds
The 2 by 2 queen graph is colourable with 4 colours. Time used:
0.00 seconds
```



The 3 by 3 queen graph is not colourable with 3 colours. Time used: 0.00 seconds
The 3 by 3 queen graph is not colourable with 4 colours. Time used: 0.00 seconds
The 3 by 3 queen graph is colourable with 5 colours. Time used: 0.00 seconds



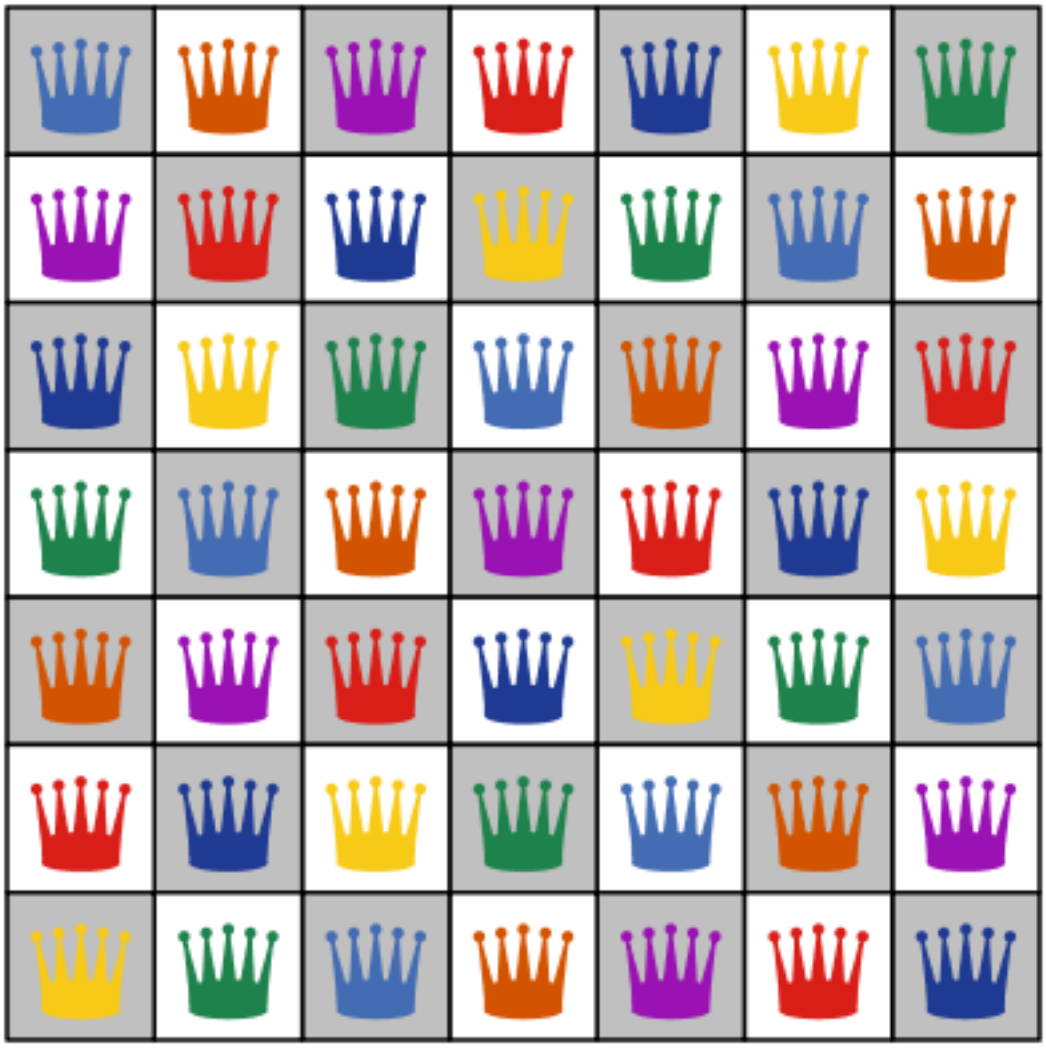
The 4 by 4 queen graph is not colourable with 4 colours. Time used: 0.01 seconds
The 4 by 4 queen graph is colourable with 5 colours. Time used: 0.01 seconds



The 5 by 5 queen graph is colourable with 5 colours. Time used: 0.01 seconds

The 6 by 6 queen graph is not colourable with 6 colours. Time used: 0.03 seconds
The 6 by 6 queen graph is colourable with 7 colours. Time used: 0.03 seconds

The 7 by 7 queen graph is colourable with 7 colours. Time used: 0.05 seconds



The 8 by 8 queen graph is not colourable with 8 colours. Time used: 9.90 seconds
The 8 by 8 queen graph is colourable with 9 colours. Time used: 1.48 seconds