NOPT042 Constraint programming: Tutorial 3 – Search strategies

In [ ]:

Example: Map coloring

Create a model to color the map of Australian states and territories 7 with four colors (cf. The 4-color Theorem). (We exclude the Australian Capital Territory, the Jervis Bay Territory, and the external territories. Map coloring is a special case of graph coloring, see this map.


Map of Australia

Let's use the following decision variables:

```
Territories = [WA, NT, SA, Q, NSW, V, T]
```

In [1]:
```
!picat map-coloring/map-coloring.pi
```

```
Western Australia is red.
Northern Territory is green.
South Australia is blue.
Queensland is red.
New South Wales is green.
Victoria is red.
Tasmania is red.
```

In [2]:
```
!cat map-coloring/map-coloring.pi
```

```
import cp.

color_map(Territories) =>
    % variables
    Territories = [WA, NT, SA, Q, NSW, V, T],
    Territories :: 1..4,

    % constraints
    WA #!= NT,
    WA #!= SA,
    NT #!= SA,
    NT #!= Q,
    SA #!= Q,
    SA #!= NSW,
    SA #!= V,
    Q  #!= NSW,
    V  #!= NSW.

% optional: symmetry breaking
precolor(Territories) =>
    WA #= 1,
    NT #= 2,
    SA #= 3.

% optional: better output than `println(Territories)` (we could also use a map, i.e. a d
ictionary)
output(Territories) =>
    Color_names = ["red", "green", "blue", "yellow"],
    Territory_names = ["Western Australia", "Northern Territory", "South Australia", "Qu
eensland", "New South Wales", "Victoria", "Tasmania"],
    foreach(I in 1..Territories.length)
        writef("%s is %s.\n", Territory_names[I], Color_names[Territories[I]])
    end.


main =>
    color_map(Territories),
    precolor(Territories), % optional
    solve(Territories),

    % println(Territories)
    output(Territories).
```

What is wrong with this model? We always want to separate the model from the data. (See the exercises below.)

Exercises

Exercise: Graph-coloring

1. Write a program that solves the (directed) graph 3-coloring problem with a given number of colors and a given graph. The graph is given by a list of edges, each edge is a 2-element list. We assume that vertices of the graph are $1, \ldots, n$ where n is the maximum number appearing in the list.
2. Generalize your program to graph k-coloring where k is a positive integer given on the input.
3. Modify your program to accept the incidence matrix (a 2D array) instead of the list of edges.
4. Add the flag `-n` to output the minimum number of colors (the chromatic number) of a given graph.

For example:

```
picat graph-coloring.pi [[1,2],[2,3],[3,4],[4,1]]
picat graph-coloring.pi [[1,2],[2,3],[3,1]] 4
picat graph-coloring.pi "{{0,1,1},{1,0,1},{1,1,0}}" 4
picat graph-coloring.pi -n [[1,2],[2,3],[3,4],[4,1]]
```