4.2. Graph Coloring

The m-Coloring problem concerns finding all ways to color an undirected graph using at most m different colors, so that no two adjacent vertices are the same color. We usually call the m-Coloring problem a unique problem for each value of m.

Example 1

Consider the graph in figure 1. There is no solution to the 2-Coloring problem for this graph because, if we can use at most two different colors, there is no way to color the vertices so that no adjacent vertices are the same color.

![Graph with 4 nodes](image)

**Figure 1: graph with 4 nodes**

There are six solutions to the 3-Coloring problem as in the table below:

<table>
<thead>
<tr>
<th>node</th>
<th>solutions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>color</td>
</tr>
<tr>
<td>V1</td>
<td>1</td>
</tr>
<tr>
<td>V2</td>
<td>2</td>
</tr>
<tr>
<td>V3</td>
<td>3</td>
</tr>
<tr>
<td>V4</td>
<td>2</td>
</tr>
</tbody>
</table>
Figure 2: state space tree for 3-coloring problem in example 1.
A straightforward state space tree for the $m$-Coloring problem is one in which each possible color is tried for vertex $v_1$ at level 1, each possible color is tried for vertex $v_2$ at level 2, and so on until each possible color has been tried for vertex $v_n$ at level $n$. Each path from the root to a leaf is a candidate solution. We check whether a candidate solution is a solution by determining whether any two adjacent vertices are the same color. To avoid confusion, remember in the following discussion that "node" refers to a node in the state space tree and "vertex" refers to a vertex in the graph being colored.

Figure 3 shows a portion of the pruned state space tree that results when this backtracking strategy is applied to a 3-coloring of the graph in Figure 1. The number in a node is the number of the color used on the vertex being colored at the node. The first solution is found at the shaded node. Nonpromising nodes are labeled with crosses. After $v_1$ is colored color 1, choosing color 1 for $v_2$ is nonpromising because $v_1$ is adjacent to $v_2$. Similarly, after $v_1$, $v_2$, and $v_3$ have been colored colors 1, 2, and 3, respectively, choosing color 1 for $v_4$ is nonpromising because $v_1$ is adjacent to $v_4$.

Figure 4: A portion of the pruned state space tree produced using backtracking to do a 3-coloring of the graph in Figure 1. The first solution is found at the shaded node. Each nonpromising node is marked with a cross.
An important application of graph coloring is the coloring of maps. A graph is called **planar** if it can be drawn in a plane in such a way that no two edges cross each other. The graph at the bottom of Figure 5 is planar. However, if we were to add the edges \((v_1, v_5)\) and \((v_2, v_4)\) it would no longer be planar. To every map there corresponds a planar graph. Each region in the map is represented by a vertex. If one region is adjacent to another region, we join their corresponding vertices by an edge. Figure 5 shows a map at the top and its planar graph representation at the bottom. **The \(m\)-Coloring problem for planar graphs is to determine how many ways the map can be colored, using at most \(m\) colors, so that no two adjacent regions are the same color.**

Figure 5: Map (top) and its planar graph representation (bottom)
Next we present an algorithm that solves the \( m \)-Coloring problem for all values of \( m \). In this algorithm the graph is represented by an adjacency matrix. However, each entry in the matrix is simply true or false depending on whether or not there is an edge between the two vertices.

**Algorithm: The Backtracking Algorithm for the \( m \)-Coloring Problem**

**Problem:** Determine all ways in which the vertices in an undirected graph can be colored, using only \( m \) colors, so that adjacent vertices are not the same color.

**Inputs:** positive integers \( n \) and \( m \), and an undirected graph containing \( n \) vertices. The graph is represented by a two-dimensional array \( W \), which has both its rows and columns indexed from 1 to \( n \), where \( W[i,j] \) is true if there is an edge between \( i \)th vertex and the \( j \)th vertex and false otherwise.

**Outputs:** all possible colorings of the graph, using at most \( m \) colors, so that no two adjacent vertices are the same color. The output for each coloring is an array \( vcolor \) indexed from 1 to \( n \), where \( vcolor[i] \) is the color (an integer between 1 and \( m \)) assigned to the \( i \)th vertex.

**Procedure** \( m\_coloring\) \( (i : \text{integer}) \)

```pascal
var color: integer;
begin
  if (promising\( (i) \)) then
    if \( (i = n) \) then
      Write \( vcolor[1] \) through \( vcolor[n] \);
    else
      for \( color = 1 \) to \( m \) \{ Try every \}
      begin
        \( vcolor[i + 1] = color \); \{ color for \}
        \( m\_coloring\ (i + 1) \); \{ next vertex.\}
      end;
  end;
end;
```
Function promising (i : integer):Boolean;
var j: integer; switch:Boolean;
begin
    switch = true;
    j = 1;
    while (j and switch) do { Check if an
        begin
            if (W[i,j] and vcolor[i] = vcolor[j]) then is already
                switch = false;
                this color.}
            j := j+1;
        end;
    promising := switch;
end;

The parameters n, m, W, and vcolor are not inputs to either routine. In
an implementation of the algorithm, the routines would be defined
locally in a simple procedure that had n, m, and W as inputs, and vcolor
defined locally. The top level call to m_coloring would be
m_coloring (0 )
The number of nodes in the state space tree for this algorithm is equal to

\[ 1 + m + m^2 + \cdots + m^n = \frac{m^{n+1} - 1}{m - 1}. \]

Homework:

1. Modify the Backtracking algorithm for the n-Queens problem so
    that, instead of generating all possible solutions, it finds only a
    single solution.
2. Find at least two instances of the n-Queens problem that have no
    solutions.
3. Apply the Backtracking algorithm for the n-Queens problem to
    the problem instance in which n = 5, and show the actions step by
    step. Draw the state space tree and the pruned state space tree
    produced by this algorithm up to the point where the first
    solution is found.
4. Use the Backtracking algorithm for the m-Coloring problem to find all possible colorings of the graph below using the three colors red, green, and white. Show the actions step by step.

![Graph Diagram]

5. Suppose that to color a graph properly we choose a starting vertex and a color to color as many vertices as possible. Then we select a new color and a new uncolored vertex to color as many more vertices as possible. We repeat this process until all the vertices of the graph are colored or all the colors are exhausted. Write an algorithm for this greedy approach to color a graph of $n$ vertices. Analyze this algorithm and show the results using order notation.

6. Suppose we are interested in minimizing the number of colors used in coloring a graph. Does the greedy approach of homework 5 guarantee an optimal solution? Justify your answer.

7. Compare the performance of the Backtracking algorithm for the m-Coloring problem and the greedy algorithm of homework 5. Considering the result(s) of the comparison and your answer to homework 6, why might one be interested in using an algorithm based on the greedy approach?

8. List some of the practical applications that are representable in terms of the m-Coloring problem.