Matthew Ragoza
mtr22@pitt.edu

Assignment 1

ISSP 2160
September 12, 2021

Problem 1. Map coloring problem

a. The map coloring problem can be formulated as a search problem as follows:

- The *initial state* is the map with every country assigned a color randomly or naively, i.e. without checking the colors of bordering countries.
- The *operators* are changing the color of any country to a different color.
- The *goal condition* is when no two countries sharing a border have the same color.

b. Let n be the number of colors and let k be the number of countries. Since each of the k countries is assigned one of n colors, the *search space size* is n^k. In the provided example, $n = 3$ and $k = 9$, so the search space size is $19,683$.

c. An alternative formulation of the search problem is as follows:

- The *initial state* is the map with no colors assigned to any country.
- The *operators* are assigning colors to the countries one-by-one in a predetermined order, e.g. alphabetically.
- The *goal condition* is when every country has been assigned a color and no two countries that share a border have the same color.

Let S_i be the number of map states where i countries have been assigned a color. Since we assign each country a color one by one, the total search space size is $\sum_{i=0}^{k} S_i$. There is only one state with no colors assigned, so $S_0 = 1$. At each step, we select from n colors and assign it to the next country in the predetermined order. So we can define $S_{i+1} = S_i n$. From this recursive definition, we can obtain $S_i = n^i$. Therefore, the *search space size* is $\sum_{i=0}^{k} n^i$. Given $n = 3$ and $k = 9$, this comes out to 29,524.

This is almost 50% larger than the search space size from Part b., so it is a less efficient way of setting up the problem at hand. There are ways to make it more efficient by incorporating knowledge about the problem. For instance, when choosing colors for a country, we could select only from the colors that are not already assigned to a bordering country, which prunes the search tree to avoid exploring bad states. However, this improvement could be applied to either of the two formulations, so it still appears that the first formulation is more advantageous.

d. Figure 1 shows one possible solution to the provided map coloring problem.

Assignment 1

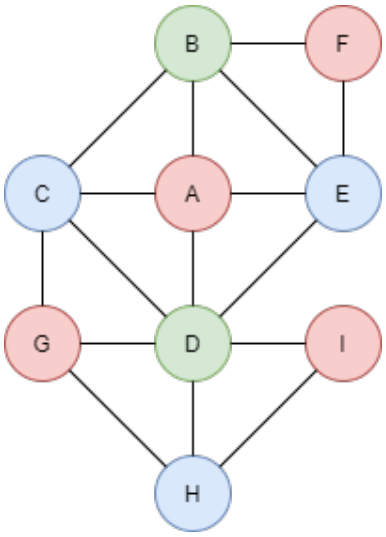

Figure 1: A solution to the map coloring problem.

Problem 2. Traveler problem

a. $S \to A \to B \to C \to D \to S \to C \to S \to A \to B$

b. $S \to A \to C \to B \to D \to F \to E \to G \to H \to H$

c. $S \to A \to B \to C \to D \to C \to B \to D \to E \to F$

d. $S \to A \to B \to C \to D \to E \to F \to G \to H$

Problem 3. A problem-solving agent for the 8-puzzle problem

a. `bfs.py`

b. `bfs_stats.py`

c. `bfs_cycles.py`

d. `bfs_repeats.py`

e. Analysis of Parts b., c., and d.

f. `dfs_limit.py` and analysis

	# expanded	# generated	Max. queue length	Solution length
`bfs_stats.py`	29	81	53	4
`bfs_cycles.py`	16	44	22	4
`bfs_repeats.py`	16	44	22	4
`dfs_limit.py`	164	467	17	4

Figure 2: Search statistics for Example 1.

	# expanded	# generated	Max. queue length	Solution length
`bfs_stats.py`	500	1,382	883	7
`bfs_cycles.py`	84	234	105	7
`bfs_repeats.py`	84	234	105	7
`dfs_limit.py`	106	274	16	7

Figure 3: Search statistics for Example 2.

	# expanded	# generated	Max. queue length	Solution length
`bfs_stats.py`	1,962	5,622	3,661	9
`bfs_cycles.py`	176	500	226	9
`bfs_repeats.py`	173	490	216	9
`dfs_limit.py`	692	1,863	17	9

Figure 4: Search statistics for Example 3.

	# expanded	# generated	Max. queue length	Solution length
`bfs_stats.py`	31,658	87,542	55,885	11
`bfs_cycles.py`	808	2,198	934	11
`bfs_repeats.py`	723	1,950	775	11
`dfs_limit.py`				

Figure 5: Search statistics for Example 4.

	# expanded	# generated	Max. queue length	Solution length
`bfs_stats.py`				
`bfs_cycles.py`	59,524	160,638	67,175	19
`bfs_repeats.py`	31,591	84,812	30,611	19
`dfs_limit.py`				

Figure 6: Search statistics for Example 5.