Frank Lu 2019–9–10

Problem assignment 1

Due: Wednesday, September 11, 2019

Problem 1. Map coloring problem

Part a.

Initial state: A map with no color painted on.

Operators: Valid paint steps

Goal condition: A map colored with no countries on the map that share a border are assigned the same color.

Part b.

The search space can be defined as all valid configurations of paint to the map.

The upper estimate bound for the search space size is the number of countries times color choice; In this case, it's at most 9*3=27.

Part c.

Considering the abstraction of the spatial layout, the problem can be formulated to another search problem: Divide the set of nodes in to three independent subsets(green red blue). In each subsets, the nodes are nonadjacent.

In this way, the formulation can be defined as:

Initial state: A set with all nodes

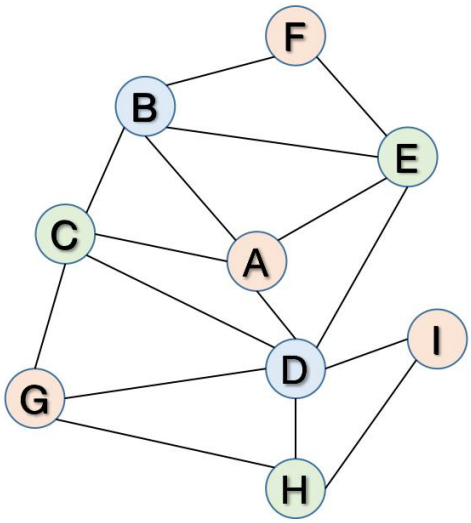
Operations: Divide each node into three subsets based on the nonadjacent rule,

Goal condition: three well divided independent subsets.

The search space size is the size of the original sets of nodes, 9. This is a more advantageous formulation.

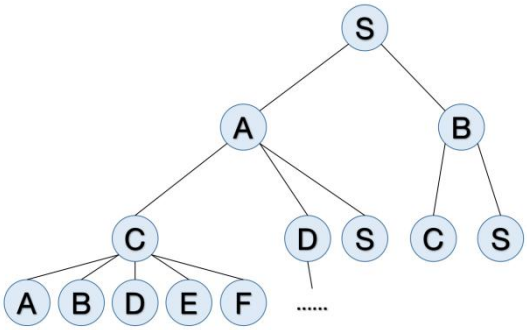
Part d.

One possible solution for the problem.



Problem 2. Traveler problem

Part a.

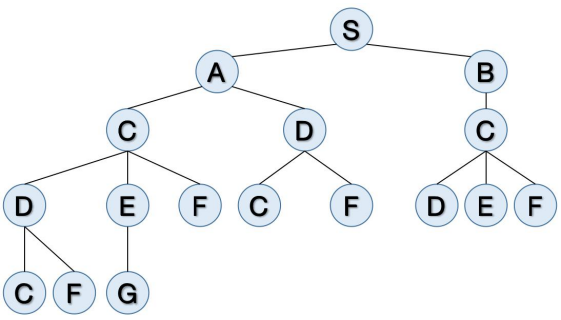


The expanded order would be S-A-B-C-D-S-C-S-A-B-D-E-F-...

Part b.

If using the depth-first search, with the elimination of cyclic, the route could be S-A-C-E-G.

Part c.

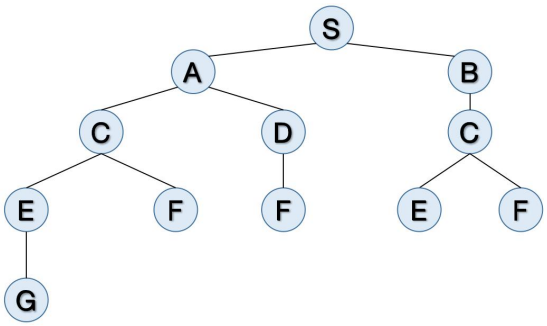


The expand step of the BFS that prevents cyclic would be:

S-A-B-C-D-C-D-E-F-C-F-D-E-F-C-F-G

And the searched route would be S-A-C-E-G.

Part d.



The expand step of BFS that checks all state repeats would be:

S-A-B-C-D-C-E-F-F-E-F-G

And the searched route would be S-A-C-E-G.

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

Part a . See bfs.py

Part b. See bfs_stats.py

Part c. See bfs_cycles.py

Part d. See bfs_repeats.py

Part e.

Vanilla BFS search:

	Nodes generated	Nodes expanded	Maximum length of queue	Length of solution path
Example 1	82	29	53	4
Example 2	1383	500	883	7
Example 3	5623	1962	3661	9
Example 4	87543	31658	55885	11

BFS search with the elimination of cyclic state repeats.

	Nodes generated	Nodes expanded	Maximum length of queue	Length of solution path
Example 1	45	16	22	4
Example 2	235	84	105	7
Example 3	501	176	226	9

Example 4	2199	808	934	11
Example 5	160639	59524	67175	19

BFS search with the elimination of all state repeats.

	Nodes generated	Nodes expanded	Maximum length of queue	Length of solution path
Example 1	45	16	22	4
Example 2	235	84	105	7
Example 3	491	173	216	9
Example 4	1951	723	775	11
Example 5	84813	31591	30611	19

As shown by the table above, the BFS search with the elimination of all state repeats has the best performance, then the BFS search with the elimination of cyclic state repeats. That's because it reduces all the redundancy when generating a search tree. Some useless nodes don't have to be generated several times.

Part f. See dfs_limit.py

Depth-limited DFS search (with depth limitation 10)

	Nodes generated	Nodes expanded	Maximum length of queue	Length of solution path
Example 1	264	96	15	4
Example 2	167	58	15	7
Example 3	1108	401	16	9

Generally, the DFS search doesn't preformed as well as the elimination of all state repeats. That's because it might have to go through more nodes before search out the optimal path rather than a sub-optimal. However, it performs very well at the queue length control, for it doesn't need to expanded as many nodes as BFS search do.