Hunter Finger

Scott Hofbauer

Comp 560 Assignment 1 Write-Up

Backtracking Search:

To begin with backtracking search, we first had to create the methodology to read in the input file, input.txt. We placed this code in the constructor of the main class so that it always executes first. The way we setup the problem was we created an array of Nodes (States) and each Node had an ArrayList of “neighbors” (other nodes), effectively creating a non-directed graph. Once all of the information was stored and the graph, we turned our attention to the backtracking search. The way this code was implemented was by starting at the first state in our list (Alabama), and did a depth first search on nodes where once one node is colored it pulls the next non-colored neighbor from the it’s list and recursively calls the same function and continues. The coloring methodology was we loop through the possible colors and check if each color would work for each node. The way we checked the colors was by pulling each node’s neighbors and see if say the color “Red” matched one of the node’s neighbor’s, if it did then it would return false and the coloring method would move to the next color, if it didn’t then the node would be set to the color red. Once we reach a node with no uncolored neighbors it traverses back up the tree to the parent node and looks to see if that node has any uncolored neighbors. Whenever we ran into a neighboring node that had been colored by a previous branch, we checked to make sure that node did not conflict with the current neighbor. If it did, we resolved the conflict and worked back down the tree from there. Throughout this, we were counting the nodes that we had visited during the search to display during the end result.

Local Search:

For local search, we began by randomly assigning colors to all nodes in the tree. We sorted through the nodes selecting the node with the highest number of conflicts as our starting point. We attempted to resolve that node's conflicts and checked to see if the map is now a solution before moving onto the next highest conflicting node. Sometimes the node is able to be colored where it reduces the conflicts of that node to zero, however, if there are no colors that will reduce the number of conflicts to zero then the node will be colored the same color of the neighbor with the least amount of neighbors. The idea behind this is that by setting the color to a node to the same color as the neighbor with the least amount of neighbors will allow that neighbor to be open up to more colors and the color will be changed and thus resolving both the original node's conflicts and the neighbor. This methodology where after each node is colored the whole solution is checked for correctness, if the solution is correct then the solution is returned if not the coloring will repeat at most 150 times (150 was just arbitrarily chosen, it seemed to keep the time to complete the solution faster) . After 150 colorings, if there is still no solution then the entire map is reset to random colors and the process repeats until a solution is found.

Contributions:

Backtracking Search Algorithm - Scott

Backtracking Count - Hunter

Local Search Algorithm - Hunter and Scott

Local Search Count - Hunter

Write-up - Hunter