Luis Garcia
PID#: 730129287
Andrew Palay
February 22, 2020

HW1:
How I Implemented Backtracking and Local Search

To begin with, I would like to clarify that I did this assignment on my own, so I take full credit of the work on this repository. That being said, I approached backtracking by creating a java program which accepts a valid .txt file and that searches the file in order to get the information it needs. Any file works given that it is written in the format that you explained in your announcement. Moreover, after getting said file, I created variables to store the information in the file. Mostly, I used ArrayLists because I was not sure of the amount of lines that were on the file. Having constraint sati faction in mind, I ordered the list of states in a way that the one with the most “neighboring” states would be on top in order to quickly reduce the options for the other states. Afterwards I called a recursive function called solve which accept an index to iterate over all the states and assign them valid colors. It does this by getting the possible colors of each state and iterating through them until it finds one that not only works but that doesn’t limit the colors of its neighbors to 0. If those constraints were satisfied I then accepted that color as possible and incremented the amount of changes I made. Suprisingly, each time I ran the files that were given I realized that the number of nodes traversed was equal to the amount of states.

That being said, I tried to implement the Local Search through hill climing. Moreover, I had to go through a semi-recursive route because each time I attempted to run it, it would prematurely stop before the 1 minute timer ran out. With that in mind, I implemented it using a timer using executor and future given a method I found in "https://stackoverflow.com/questions/5715235/java-set-timeout-on-a-certain-block-of-code". What the code does is to first genera "n" number of random States with colors prefilled and attempt to fix the best one in an neverending while loop which ends if the solution is found. the way I chose the best one is to find the one that satisfies most of constraints given through the states that are neighboring each other. If it produces no progress after 100 iterations of the while loop it restarts the process by calling itself recursively. Moreover, I set the limit to 1000 recursive calls so that it would not reach the recursion limit. That being said it seems to usually produce an answer, but since the method used is more random than the backtracking one, the number of nodes it traverses is also random.