Gabriel Shelton
CSCI48700 – Programming Assignment 1

To begin I am going to go ahead and list the platform on which my program is able to be run on. It at the moment to my knowledge is limited to a Linux environment and is compiled in C++11. It can be run on Tesla and is rather easy to compile and run. Simply clone the repository in, I will provide a link as a comment to the repository and you can also find it here: https://github.com/sheltongabe/cpp_ai. Then there are two shell scripts provided: compile.sh and run.sh. These will compile the project using Cmake and make, then run will run the program which is in the newly created bin/ directory.

The directories include/ and src/ hold the files for the cpp_ai library which is a more general set of classes designed to operate on any problem if provided the right operations. Meanwhile the test/ dir contains the files specific to the 8-Puzzle problem which is linked to the cpp_ai_lib during compilation.

With that out of the way, I will discuss my results. Initially when I was programming my breadth-first search I had problems with my algorithm not finishing. Initially I felt that algorithm was not detecting the goal case right even though I had thoroughly tested it before writing bread-first search. However after not being able to find the reason I began calculating the number of nodes expanded each second. I found that the number was decreasing exponentially as the number of nodes in explored increased to become rather large. And I found that I was looking up in my explored list in linear time so I fixed it to logarithmic time. $O(b^{nd}) \Rightarrow O(b^{\ln(n)d})$

This made it so the search finished in a reasonable amount of time and in order to ensure that I got at least a few solvable puzzles I have the program go and attempt to solve 20,000 puzzles with breadth-first and 20,000 with A*. Both reach solutions, however because I am running 20,000 of each even if my A* is drastically faster I am failing to solve so many (a failure requires exploration of all branches) that the speed is comparable.

There was an interesting pattern that I found, in that in each case (out of 40,000 randomly generated) that my algorithm was able to successfully solve, there were initially an odd number of misplaced tiles. So I would be interested to find out if this has anything to do with whether a puzzle is solvable or not.