Muhummad Yunus Patel (PTLMUH006)
Assignment3 – Functional Programming: Fixing the World
Answers to theoretical questions

2.2. Mathematical function defining how many states can be generated in n moves:

$$S(n) = 6^n$$

Reasoning: At any given state, there are 6 possible states reachable from that state (a single move in any of the axes). From each of those states, there are again 6 new states reachable from those states, and so on. This forms a tree structure where each node (including the root node) has 6 child nodes. Each depth level of the tree corresponds to that many moves (e.g. at depth 1 will be all the states reachable by one move, etc.). The total number of moves is, therefore, given by 6^n.

3.2. The search was attempted on shuffles of 6 to 12 moves. It was observed that the search worked in a reasonable amount of time for all shuffles of 6 moves, most shuffles of 7 moves, and some shuffles of 8 moves. The search appeared to stop responding when run on shuffles of 9 or more moves. On further investigation, it was found that when running the search on cubes with 9 or more shuffles, the search was actually working, but the memory usage and CPU usage were getting to be ridiculously high. For example, the search was run on a cube that was shuffled 12 times and the memory usage of the program was observed. It climbed to over ~13 gigabytes! (I only have 8 gigs of RAM) The CPU was observed to be under similar strain with the process using just under 100% of the CPU.
The reason for the apparent non-responsiveness of the search when run on too many shuffles is then obviously because the program is still trying to work out the solution. The problem has, however grown too large and the solution that was coded becomes infeasible. To illustrate this, lets examine the case of 12 shuffles. In this case the program will have to search 6^{12} (~2 billion) possible states for the solution. This solution scales exponentially.

4. In it's current state, the cost of solving a cube with n shuffles is approximately 6^n. By reducing the number of successor states to 5, this becomes $(6 * 5^{(n-1)})$ Note the initial 6, because at the start, there is no previous move that we could undo.
Lets then examine how many possibilities will need to be searched for shuffles of 10 moves.
Initial solution (no pruning): 6^{10}, which is approx. 60.5 million
Proposed new solution (with pruning): $(6 * 5^9)$, which is approx. 11.5 million
This will reduce the computational cost significantly (reduced by roughly a factor of 6).