Project: Rubik's Cube Solver

Aditi Taneja, Ayush Pandey, Karan Taneja

April 26, 2017

1 Overview

We implemented a C++ program to solve a Rubik's Cube in the **minimum** number of steps, using breadth first search algorithm.

The states of the cube are stored as nodes of a graph and a rotation operation represents an edge. Since the edges are not weighted, BFS gives the shortest path from initial state to the solved state.

2 Data Structures

2.1 Hash Map

As the states of cube (nodes) are explored, a efficient representation of that state using string is used as the key value in the hash map.

Hash map is used to eliminate the need of visiting the same nodes again and again, as the same state can be achieved from various rotations.

2.2 Queue

A queue is used to store the nodes to be explored. The top of the queue contains the frontier along which the algorithm is currently searching.

3 Algorithm: Breadth First Search

3.1 Pseudocode

```
create empty hash map H
insert initial state into H
create empty queue Q
insert initial state into Q

if initial state is already solved: return

while Q is not empty:
    top <- Q.front
    Q.pop
    generate next possible states
    iterate next possible state S:
        if S is solved:
            back trace to through the parent pointers
                to generate path and return it
        else:
            if S is not present in H:
                insert S into H
                insert S at the end of Q
```

4 Results

4.0.1 2x2 Cube

2x2 cube has approximately 3.7 million possible states. Maximum number of turns required to solve the cube is upto 11 half or quarter turns.

We have tested our algorithms on various inputs. We could solve for the minimum steps within 15 to 20 minutes of algorithm running time.

4.1 3x3 Cube

3x3 cube is has much greater number of possible states. Within 6 rotations, there are as many as 7.7 million possible states reachable. Hence it is much more difficult for the algorithm to solve it. Maximum number of minimum steps is claimed to be around 20. It has been calculated that, with 18 rotations, 246 billion billion (2.46×10^{20}) states.

Also, the memory required to represent and store a 3x3 cube is more than double of that of 2x2 cube. Because of this, a normal personal computer starts to slow down around 1.5 to 3.0 million nodes. We were not able to solve a cube state which takes more than 6 rotations to solve.

5 Code and Test Inputs

Please find code and test inputs in git repository by clicking here.

6 References

- Finding Optimal Solutions to Rubik's Cube Using Pattern Databases
- Pocket Cube - Wikipedia (2x2 Cube)