Rubik’s Cube Solver for *Minimum* Steps

Aditi Taneja, Ayush Pandey, Karan Taneja
29th April, 2017

Introduction

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

States of the cube => nodes of a graph

A rotation operation => edge of a graph

Since the edges are not weighted, BFS gives the shortest path from initial state to the solved state.

Struct ‘cube’

```
struct cube {

    vector<vector<vector<short> > > state;

    string const lastStep;

    cube* const parent;

};
```

Data Structures

- Hash maps
- Queue

Hash Maps

As the states of cube (nodes) are explored, an *efficient representation (more on this later)* 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.

Queues

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.

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
```

Challenges

- Efficient representation of state:
  - The numbers: (2x2: 48 MB/M-nodes, 3x3: 108 MB/M-nodes)
  - Rotation (2 MB/M-nodes)
  - Parent pointer (8 MB/M-nodes)
- Too many nodes:
  - 2x2 solver: 3.7M nodes [2]
  - 3x3 solver: 4.26E20 nodes (6 rotations: 7.7M nodes) [1]

Depth	Nodes
1	18
2	243
3	3,240
4	43,254
5	577,368
6	7,706,988
7	102,876,480
8	1,373,243,544
9	18,330,699,168
10	244,686,773,808
11	3,266,193,870,720
12	43,598,688,377,184
13	581,975,750,199,168
14	7,768,485,393,179,328
15	103,697,388,221,736,960
16	1,384,201,395,738,071,424
17	18,476,969,736,848,122,368
18	246,639,261,965,462,754,048

Improvements

- Deleting the node ‘state’ once its children are computed.
- Efficient state representation for hash maps.
- Parallel computation.
- A* search algorithm.
- Compromising ‘minimum’ to achieve solution in reasonable time with ‘almost minimum’ solution: heuristics.

Thank you!

References

[1] Finding Optimal Solutions to Rubik's Cube Using Pattern Databases, Richard E. Corf

[2] Pocket Cube - Wikipedia (2x2 Cube)