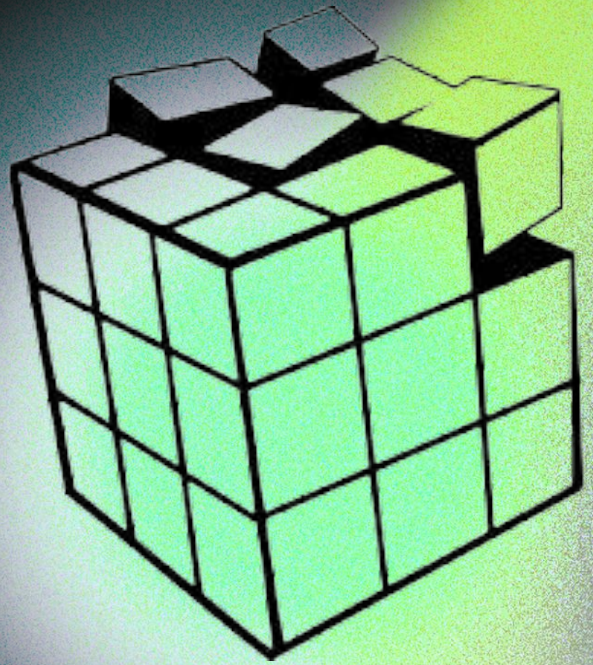
Computational Methods of Solving Rubik's Cubes

Dechawn Baker



Why Rubik's Cubes?

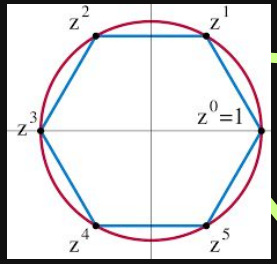

SUNDAY
MORNING


fZ
RUSH
fZ
calm

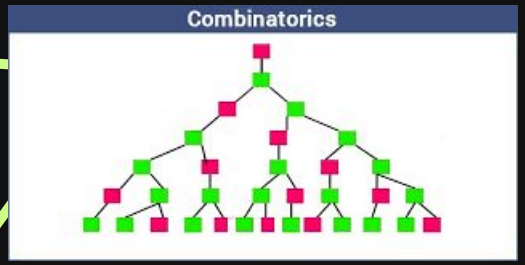
Rubik’s Cubes are more than silly little toys you might not know how to solve.

Rubik's Cubes as a subject of study

Group Theory



Combinatorics



Computer Science

Combinatorics

- The branch of math dealing with combinations of objects in a finite set with certain constraints
- Math of different ways of counting somethings.

Permutation: a specific way for something to be arranged.

How many different configurations of a 3x3 Rubik's Cube are there?

$$(8! * 3^8 * 12! * 2^{12})*½ *⅓ *½ = 4.3252 * 10^{19}$$

More than 350 million times the amount of people to ever exist

Group Theory

- Mathematical Study of the algebraic structure known as groups

  - A group G is basically a set with a couple of main properties
    - Any operation done, the result will still be in the group
    - Every operation has an inverse
    - There’s an identity element
    - It’s communicable

Rubik's Cube is Non- Abelian Group, meaning it’s not communicable

Computer Science

- The study of **computation,information,** and **automation**
- The application of theoretical techniques to make real world applications

- How can a Computer solve a Rubik's Cube?
  - My project basically

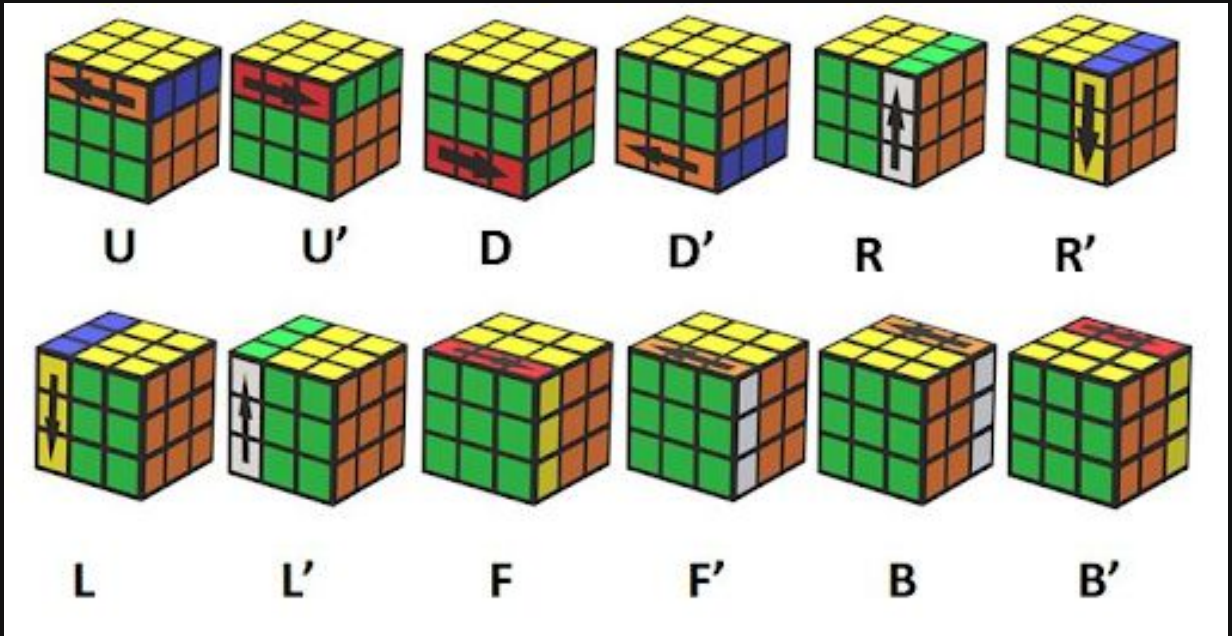
Algorithms: A set of instructions in order to solve a problem or do a task
Data Structure: A away of organizing, processing, retrieving, and storing data on a computer

Capacity Building

PART 2

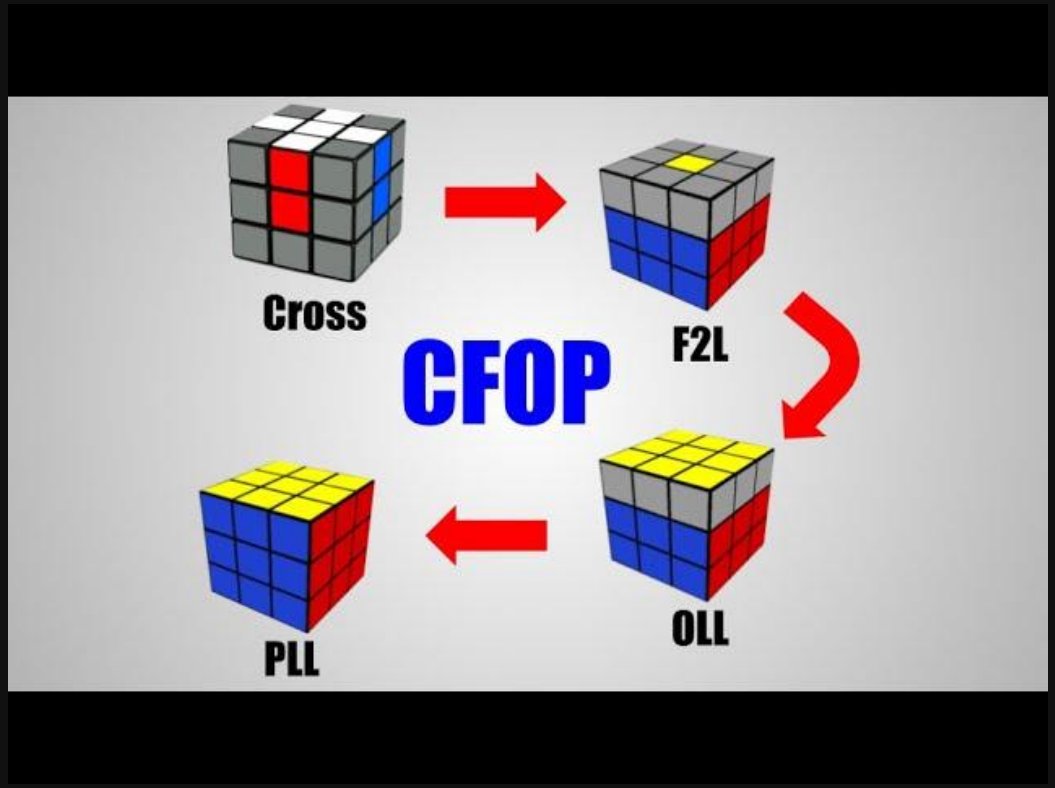
Learning to Solve a Rubik's Cube

Notation:



Learning to Solve a Rubik's Cube

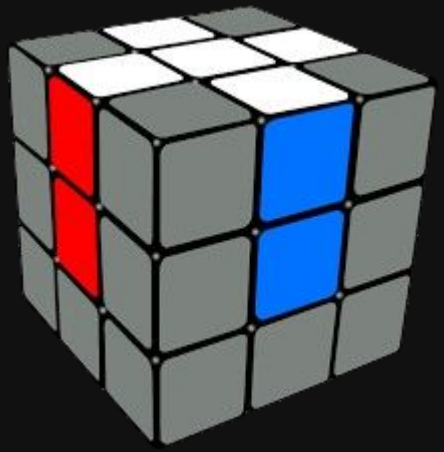
Actually solving the thing:



Learning to Solve a Rubik's Cube

Actually solving the thing:

Cross

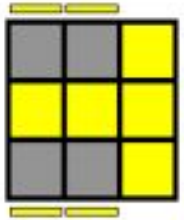


F2L

Learning to Solve a Rubik's Cube

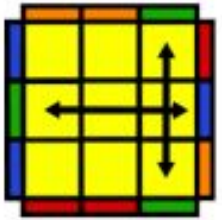
Actually solving the thing:

OLL

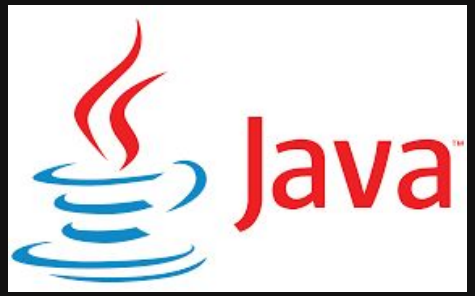


(R U R' U') (R' F R F')

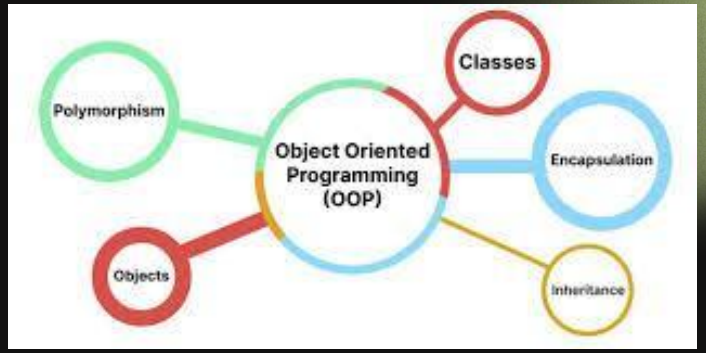
PLL



(R U R' U') (R' F R2 U') R' U' (R U R' F')



AP Computer Science A



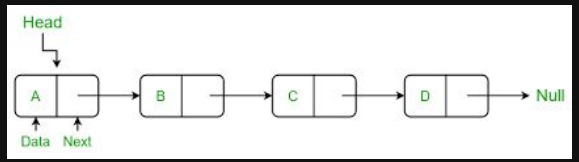
type	*set of values*	*common operators*	*sample literals*
`int`	*integers*	`+ - * // % **`	`99 12 2147483647`
`float`	*floating-point numbers*	`+ - * / **`	`3.14 2.5 6.022e23`
`bool`	*true-false values*	`and or not`	`True False`
`str`	*sequences of characters*	`+`	`'AB' 'Hello' '2.5'`

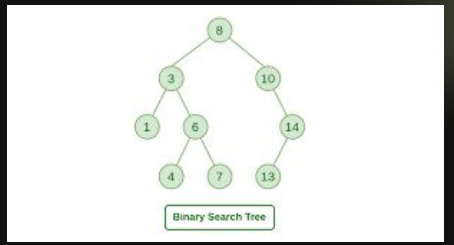
Basic built-in data types

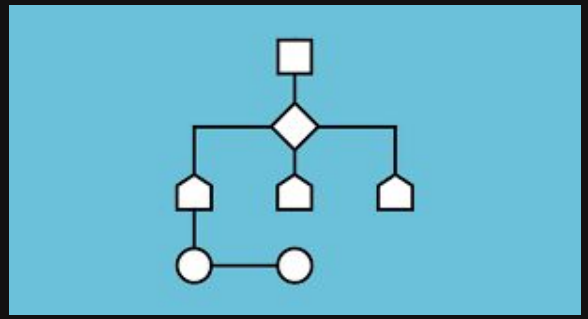
MIT OpenCourseWare

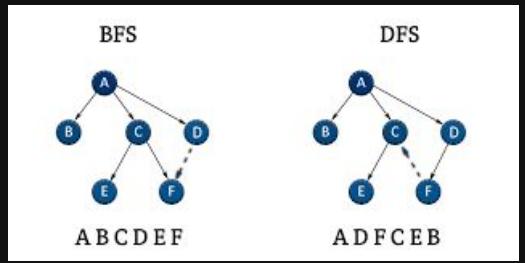
Data Structures and Algorithms

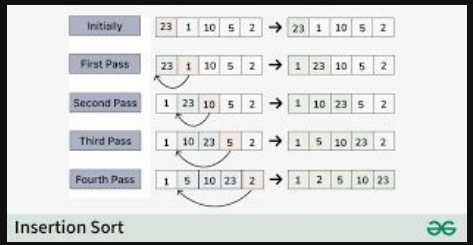
MIT OCW











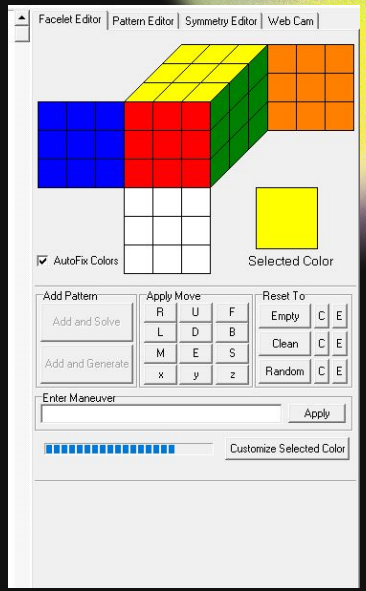
My Product

PART 3

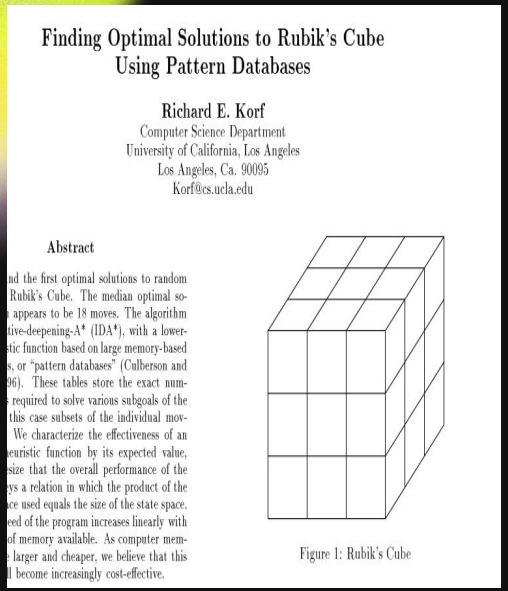
Methods

A 11 year old child

Kociemba



Korf's Alg



Finding Optimal Solutions to Rubik's Cube Using Pattern Databases

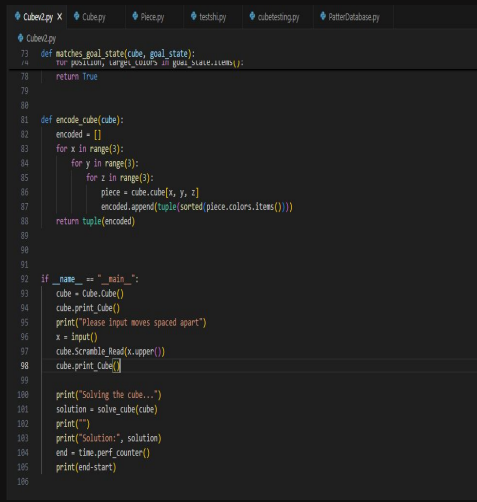
Richard E. Korf
Computer Science Department
University of California, Los Angeles
Los Angeles, Ca. 90095
Korf@cs.ucla.edu

Abstract

nd the first optimal solutions to random Rubik's Cube. The median optimal so- appears to be 18 moves. The algorithm tive-deepening-A* (IDA*), with a lower- tic function based on large memory-based s, or "pattern databases" (Culberson and 96). These tables store the exact num- required to solve various subgoals of the this case subsets of the individual mov- We characterize the effectiveness of an euristic function by its expected value, size that the overall performance of the ys a relation in which the product of the ce used equals the size of the state space. eed of the program increases linearly with of memory available. As computer mem- larger and cheaper, we believe that this ll become increasingly cost-effective.

Figure 1: Rubik's Cube

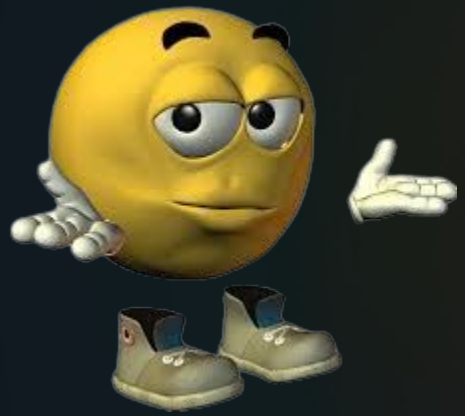
My program



```
Cube2.py X  Cube.py  Piece.py  testsh.py  cubetesting.py  PatterDatabase.py
Cube2.py
73  def matches_goal_state(cube, goal_state):
74      for position, target_colors in goal_state.items():
78      return True
79
80
81  def encode_cube(cube):
82      encoded = []
83      for x in range(3):
84          for y in range(3):
85              for z in range(3):
86                  piece = cube.cube[x, y, z]
87                  encoded.append(tuple(sorted(piece.colors.items())))
88      return tuple(encoded)
89
90
91
92  if __name__ == "__main__":
93      cube = Cube.Cube()
94      cube.print_Cube()
95      print("Please input moves spaced apart")
96      x = input()
97      cube.Scramble_Read(x.upper())
98      cube.print_Cube()
99
100     print("Solving the cube...")
101     solution = solve_cube(cube)
102     print("")
103     print("Solution:", solution)
104     end = time.perf_counter()
105     print(end-start)
106
```

Which Language?

Korf



A guess would be C or C++ though

Kociemba

Me

Cube Representation

Korf

Cube = [1,2,3,4,5,6,...,20]

Kociemba

```
36        U2 = 1
37        U3 = 2
38        U4 = 3
39        U5 = 4
40        U6 = 5
41        U7 = 6
42        U8 = 7
43        U9 = 8
44        R1 = 9
```

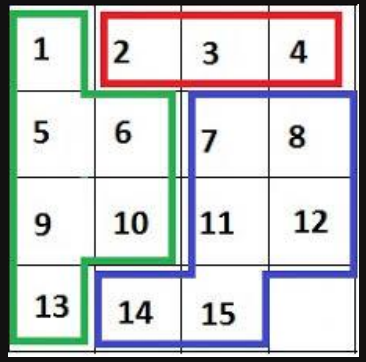
Me

```
[{'Left': 'R', 'Up': 'Y', 'Front': 'G'}, {'Up': 'Y', 'Front
[{'Left': 'R', 'Front': 'G'}, {'Front': 'G'}, {'Right': 'O'
[{'Left': 'R', 'Down': 'W', 'Front': 'G'}, {'Down': 'W', 'F

[{'Left': 'R', 'Up': 'Y'}, {'Up': 'Y'}, {'Right': 'O', 'Up'
[{'Left': 'R'}, {}, {'Right': 'O'}]
[{'Left': 'R', 'Down': 'W'}, {'Down': 'W'}, {'Right': 'O',

[{'Left': 'R', 'Up': 'Y', 'Back': 'B'}, {'Up': 'Y', 'Back':
[{'Left': 'R', 'Back': 'B'}, {'Back': 'B'}, {'Right': 'O',
[{'Left': 'R', 'Down': 'W', 'Back': 'B'}, {'Down': 'W', 'Ba
```

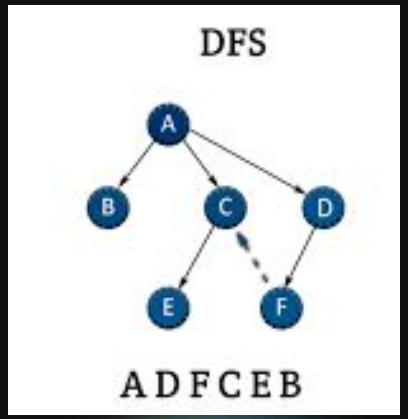
Solve Algorithm: Korf



Corner Positions: [0,3,5,1,4,2,6,7]
Moves away from solution: 8

f(n) = g(n) + h(n)

When f(n) <= threshold, continues going



Solve Algorithm: Kociemba

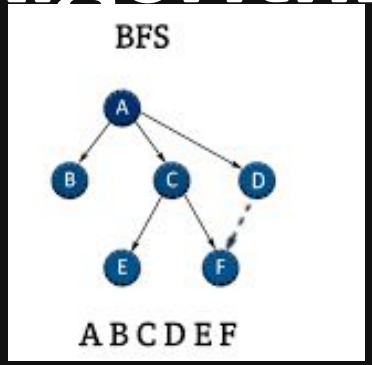
Two Phase Algorithm

G0: <U,R,D,F,B,L> Max Depth: 12 moves

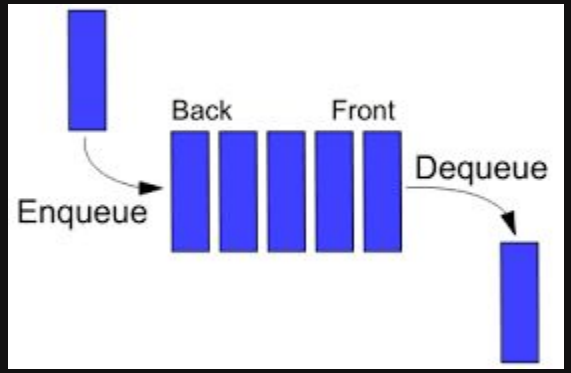
G1= <U, R2, F2, B2, L2, D> Max Depth: 18 moves

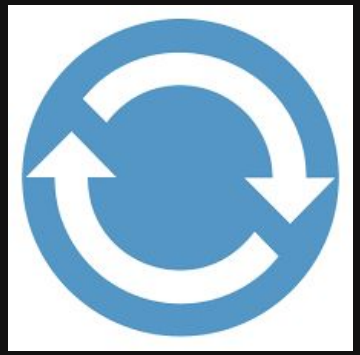
Similar to Korf, used lookup table for both phases in order to drastically reduce solving time.

Solve Algorithm: Me




```
goal_state = {
(0, 0, 0):{"Left":"R","Up":"Y","Front":"G"},
(0, 0, 1):{"Left":"R","Up":"Y"},
(0, 0, 2):{"Left": "R", "Up": "Y", "Back": "B"},
(0, 1, 0):{"Left": "R", "Front": "G"},
(0, 1, 1):{"Left": "R"},
(0, 1, 2):{"Left": "R", "Back": "B"},
(0, 2, 0):{"Left": "R", "Down": "W", "Front": "G"},
(0, 2, 1):{"Left": "R", "Down": "W"},
(0, 2, 2):{"Left": "R", "Down": "W", "Back": "B"},
```

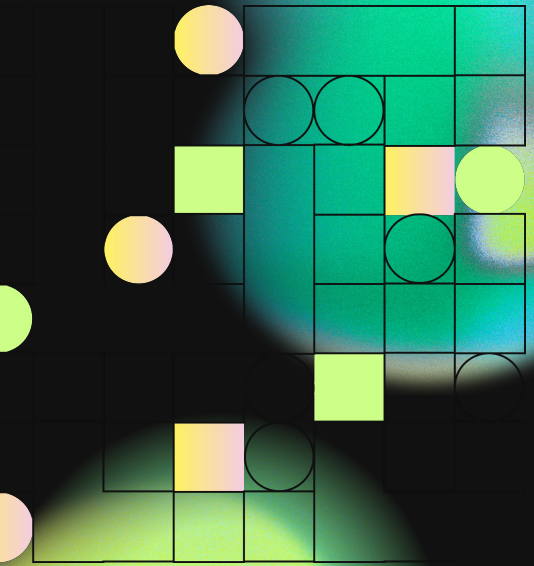




Reflections

Did I succeed?

PART 4



Yes & No

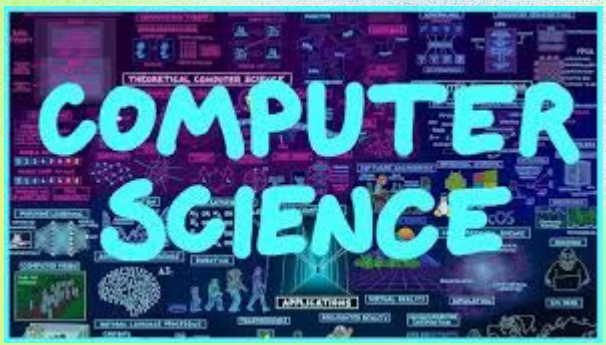
1. I mean.. it works
2. It’s inefficient to the next level
3. Basically only works as a proof of concept

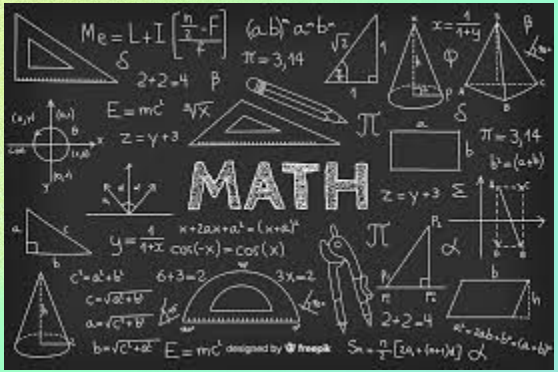
What I'd do differently

1. Use a different language
2. Use a different algorithm
3. Procrastinate less
4. Prepare more
5. Mahjong ai/sim

Next Steps:

GEORGE MASON UNIVERSITY





Acknowledgments

Ms

Mrs. Graves, Mr. Wheeler, My friends, Henry Yu, Saif Hasan

Thank You!

Any questions?