**project setup**

My program has three parts.

First is the input part:

In this part I receive the data from config file. Then generate different graph for each problem. The structure in graph is

{current state: [(info which depends different), parent, [first child], [second child],…]}

For each child I record the info of:

[child state, cost, (action to get this child)]

Second part is the output part:

In this part, I mainly use two function to analyze the returned graph and parent dist. Then print the Path, Time, Space.

Third part is the search algorithm part:

In this part, I write DFS, BFS, deep\_limit\_dfs, IDDFS, unicast, greedy, A\* and IDA\*.

**Computer configurations**

Thinkpadx1 yoga 3rd.

OS Name

Microsoft Windows 10 Pro for Workstations

System Type

x64-based PC

Processor

Intel(R) Core(TM) i7-8650U CPU @ 1.90GHz, 2112 Mhz, 4 Core(s), 8 Logical Processor(s)

Installed Physical Memory (RAM)

16.0 GB

Python 3.6

**Reference**

Wiki

https://www.youtube.com/watch?v=ySN5Wnu88nE

https://algorithmsinsight.wordpress.com/graph-theory-2/ida-star-algorithm-in-general/

<http://trycode.blogspot.com/2016/03/implementation-of-iterative-deepening.html>

**For each puzzle type**

Water jugs

Action order: for i jug and j jug:

empty (i), fill (i), pour (i) to (j), pour (j) to (i).

heuristic: myheuj

Cities

heuristic: Manhattan , Euclidean

Pancakes

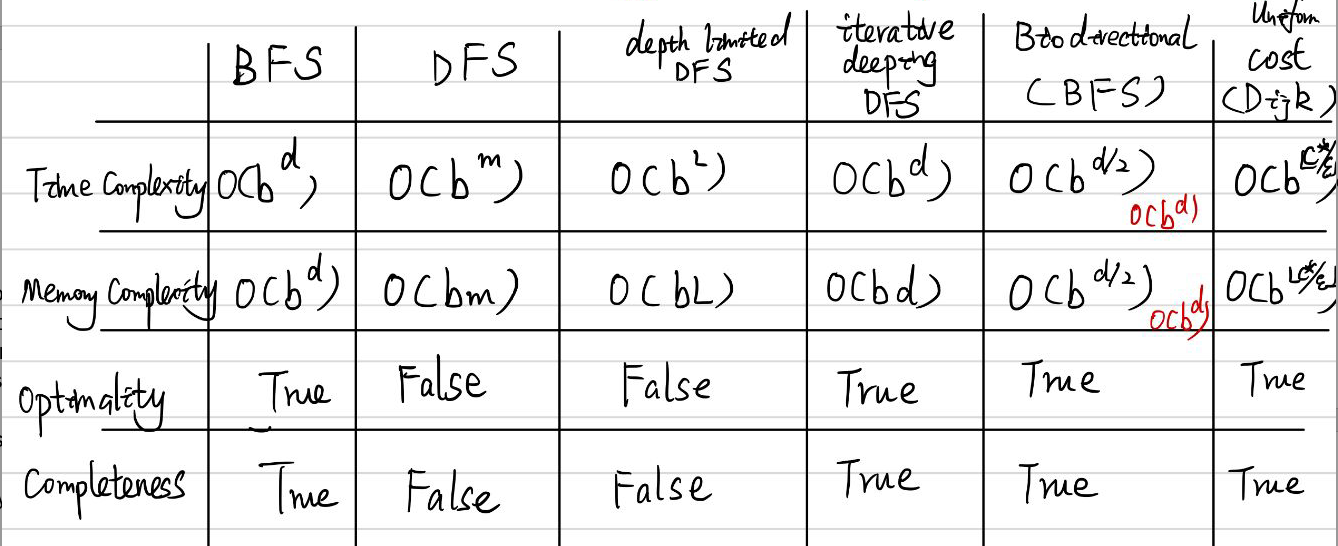
heuristic: Manhattan , Euclidean

for water jugs and pancakes, I used my own config.

Jugsthree.config include three jugs case. You can alter the number as you want

Pancakes.config include 5 pancakes. For most I can run on 7 pancakes.

**Additional discussion**



For water jugs, I think the best is IDDFS, in my out put it use less space and created relatively less nodes. BFS is also good on find the solution.

For cities, I think the best might be IDA\* for it reasonably used the information we have, and it takes less space than A\*.

For pancakes, I think IDDFS is better for me since I didn’t come up with good heuristic for this problem.

For water jugs, my heuristic is trying to move the state closer to the goal, since I take the water jugs which have smallest difference with the goal each time I choose the child. I think it’s admissible.

For pancakes, my heuristic couldn’t make thing better. So I won’t discuss it here.

**Program’s output**

**For water jugs problem: test on test\_jug.config**

**BFS**

path to the goal: [(0, 0), (4, 0), (0, 4), (4, 4), (0, 8), (4, 8), (1, 11), (1, 0)]

Time (how much nodes we have created): 67

Space (how big the frontier list grew to): 16

**DFS**

path to the goal: [(0, 0), (0, 11), (4, 7), (0, 7), (4, 3), (0, 3), (3, 0), (3, 11), (4, 10), (0, 10), (4, 6), (0, 6), (4, 2), (0, 2), (2, 0), (2, 11), (4, 9), (0, 9), (4, 5), (0, 5), (4, 1), (0, 1), (1, 0)]

Time (how much nodes we have created): 107

Space (how big the frontier list grew to): 25

**Greedy (I use the subtraction of target state and current state as heuristic)**

path to the goal: [(0, 0), (4, 0), (0, 4), (4, 4), (0, 8), (4, 8), (1, 11), (1, 0)]

Time (how much nodes we have created): 143

Space (how big the frontier list grew to): 12

**For cities problem: test on test\_cities.config**

**Greedy (I use the** **Euclidean distance as heuristic)**

path to the goal: ['C00', 'C11', 'C02', 'C13', 'C23', 'C33', 'C44']

Time (how much nodes we have created): 132

Space (how big the frontier list grew to): 25

**Unicost**

path to the goal: ['C00', 'C11', 'C02', 'C13', 'C24', 'C34', 'C44']

Time (how much nodes we have created): 218

Space (how big the frontier list grew to): 13

least cost to each node : {'C00': 0, 'C01': 5, 'C02': 6, 'C03': 11, 'C04': 12, 'C10': 7, 'C11': 4, 'C12': 9, 'C13': 10, 'C14': 15, 'C20': 6, 'C21': 9, 'C22': 12, 'C23': 15, 'C24': 18, 'C30': 9, 'C31': 12, 'C32': 15, 'C33': 18, 'C34': 21, 'C40': 10, 'C41': 13, 'C42': 16, 'C43': 19, 'C44': 22}

**A\*(I use the Euclidean distance as heuristic)**

path to the goal: ['C00', 'C11', 'C02', 'C13', 'C24', 'C34', 'C44']

Time (how much nodes we have created): 173

Space (how big the frontier list grew to): 25

**For pancakes problem: test on pancakes.config**

**I tried to give a heuristic to have less nodes how ever it didn’t work out. Here is my result with self-designed config. There are 5 pancakes, and the first one is the one heuristic.**

**Greedy(I use the absolute value of subtraction between target state and current state as heuristic)**

path to the goal: [(3, -1, 2, 5, 4), (-4, -5, -2, 1, -3), (4, -5, -2, 1, -3), (-1, 2, 5, -4, -3), (1, 2, 5, -4, -3), (3, 4, -5, -2, -1), (5, -4, -3, -2, -1), (-5, -4, -3, -2, -1), (1, 2, 3, 4, 5)]

Time (how much nodes we have created): 28086

Space (how big the frontier list grew to): 3831

**Greedy (only use the cost info)**

path to the goal: [(3, -1, 2, 5, 4), (-3, -1, 2, 5, 4), (-5, -2, 1, 3, 4), (-4, -3, -1, 2, 5), (-2, 1, 3, 4, 5), (-1, 2, 3, 4, 5), (1, 2, 3, 4, 5)]

Time (how much nodes we have created): 8806

Space (how big the frontier list grew to): 3138

**IDDFS**

path to the goal: [(3, -1, 2, 5, 4), (1, -3, 2, 5, 4), (-5, -2, 3, -1, 4), (-4, 1, -3, 2, 5), (-2, 3, -1, 4, 5), (-3, 2, -1, 4, 5), (1, -2, 3, 4, 5), (2, -1, 3, 4, 5), (-2, -1, 3, 4, 5), (1, 2, 3, 4, 5)]

Time (how much nodes we have created): 14320

Space (how big the frontier list grew to): 2864

**IDA\***

Can’t find the solution

**For test\_pancakes1 and test\_pancakes2 my program didn’t work out. I wait for more than 30 mins but nothing comes out.**