UNIFORM SEARCH ALGORITHMS

Programming Assignment #1

NOVEMBER 2, 2022

ISMAIL TOSUN / 180316030
YUSUF BURAK PEKER / 180316058

Contents

A-Development Environment

We use VisualStudio Code in a Docker container with WSL on Windows 11, our container include ubuntu 18 lts and Python 3.10.

We recommend you to use waterjug.ipynb file to checking Graphs and Outputs.

B-Problem Formulation

1. State Representations

We define states as tuple object and every jar represent as an integer in the state tuple;
(jar1 , jar2 , jar3) = (3 , 5 , 6)

2. Initial State:

We assume all Jars are empty at first (0 , 0 , 0) and we take user input for determine jar capacity

3. Possible Actions

- Fill jar 1-2-3
- Empty jar 1-2-3
- Pour
    - In pour process,pouring water from a source jug to a destination jug stops when either the source jug is empty or the destination jug is full.
    - 1 to 2
    - 1 to 3
    - 2 to 1
    - 2 to 3
    - 3 to 1
    - 3 to 2

4. Transition Model

We have 3 main action, example state is (3,5,2) and capacity is (4,8,6)

a. Fill: Selected jar fill with water as its capacity
    - Fill jar 1 → (4,5,2)

b. Empty: The water in the selected jar is emptied
    - Empty jar 1 → (0,5,2)

c. Pour: Pour water from a jar to another jar
    - Pour jar 2 to jar 3 → (3,1,6)
    - Jar 3 is full so 1 unit water still stay in jar 2

5. Goal Test

We take goal state from user and compare this state with current state in every iteration

6. Path Cost

We have two different cost function, they can switch using *is_cost_static* variable in **Waterjug** class

- If *is_cost_static* assigned **True** every process cost accept 1
- If *is_cost_static* assigned **False**, we calculate cost with water amount for fill and empty actions, every liter count as 1 cost, on the other hand pour actiton still 1 cost per process because there is no water waste

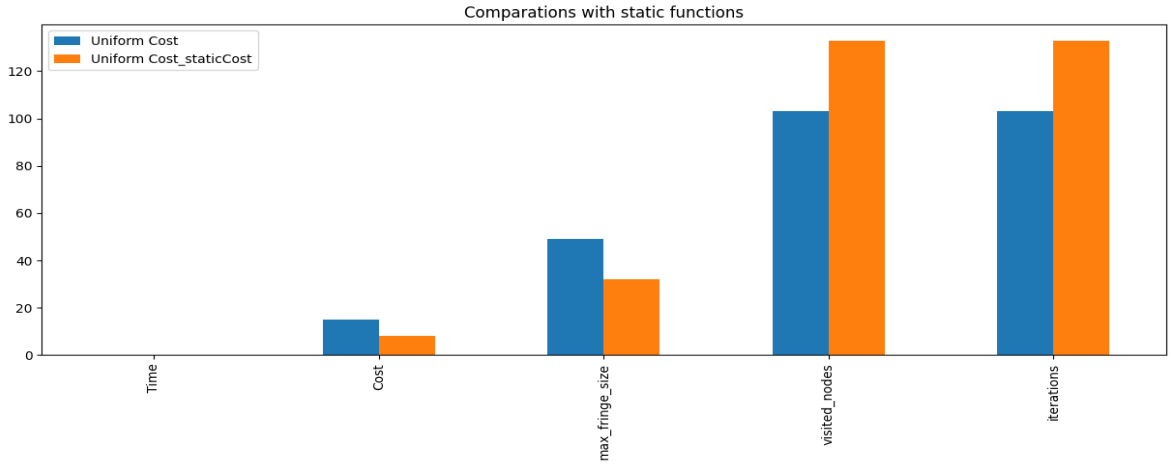
C-Results

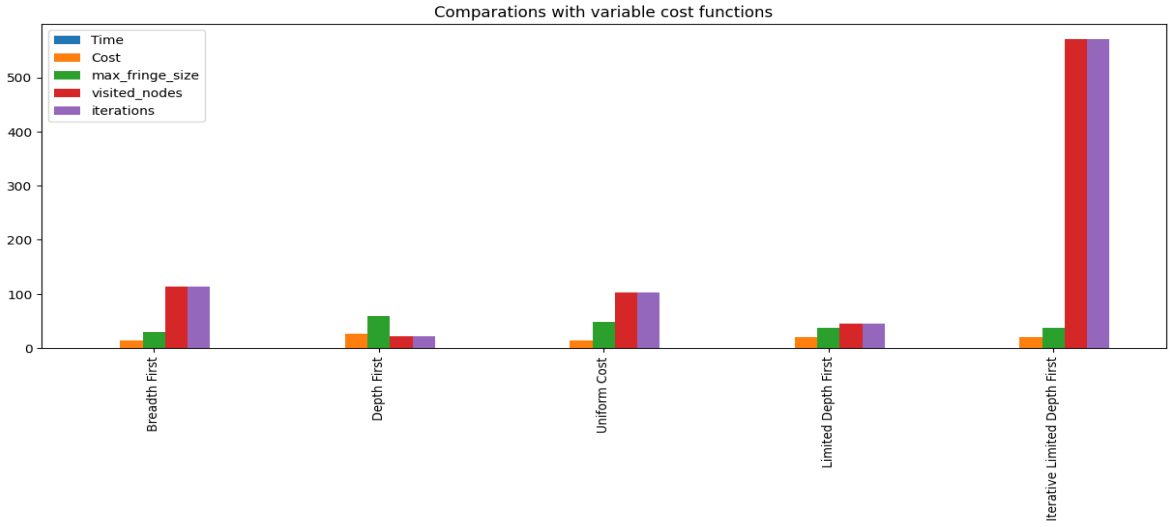
Output files come with this report in .txt format

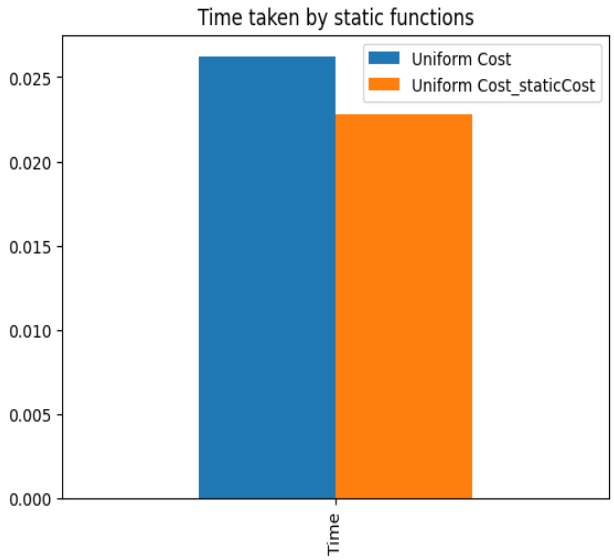
- Goal state → (4,4,0) , capacities → (8,5,3)
  - Static cost → here
  - Variable cost → here
- Goal state → (6,2,0) , capacities → (8,5,3)
  - Static cost → here
  - Variable cost → here
- Goal state → (6,2,0) , capacities → (12,8,2)
  - Static cost → here
  - Variable cost → here

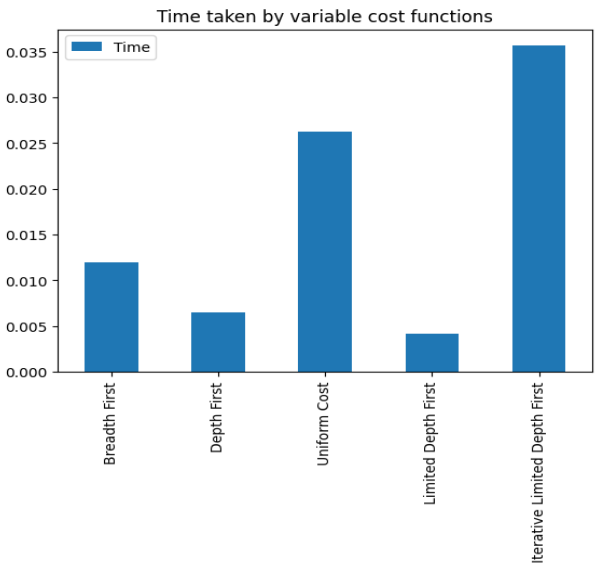
Also we plot some graph for see difference more obvious you can find all graph in here

1. First example Graphs:

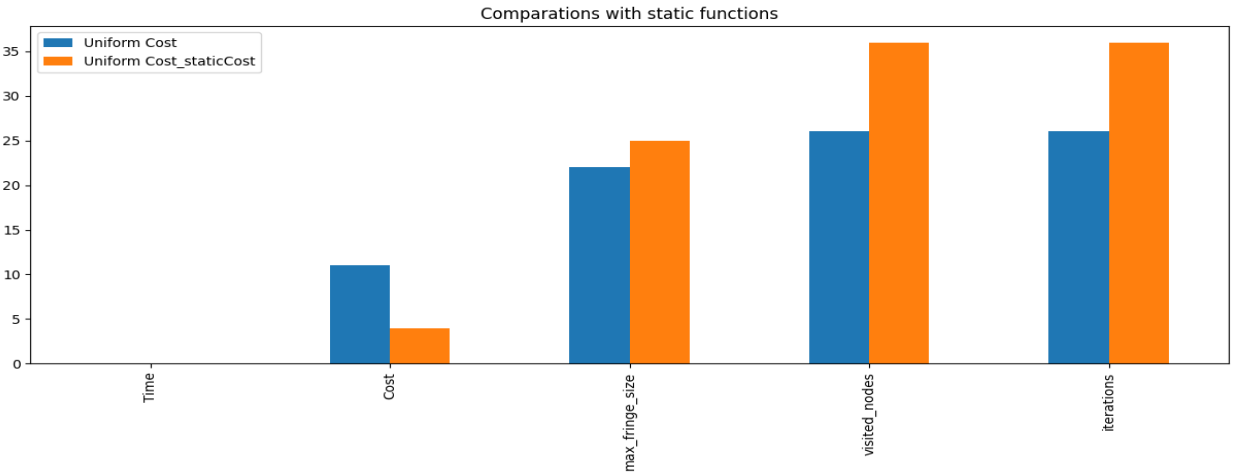


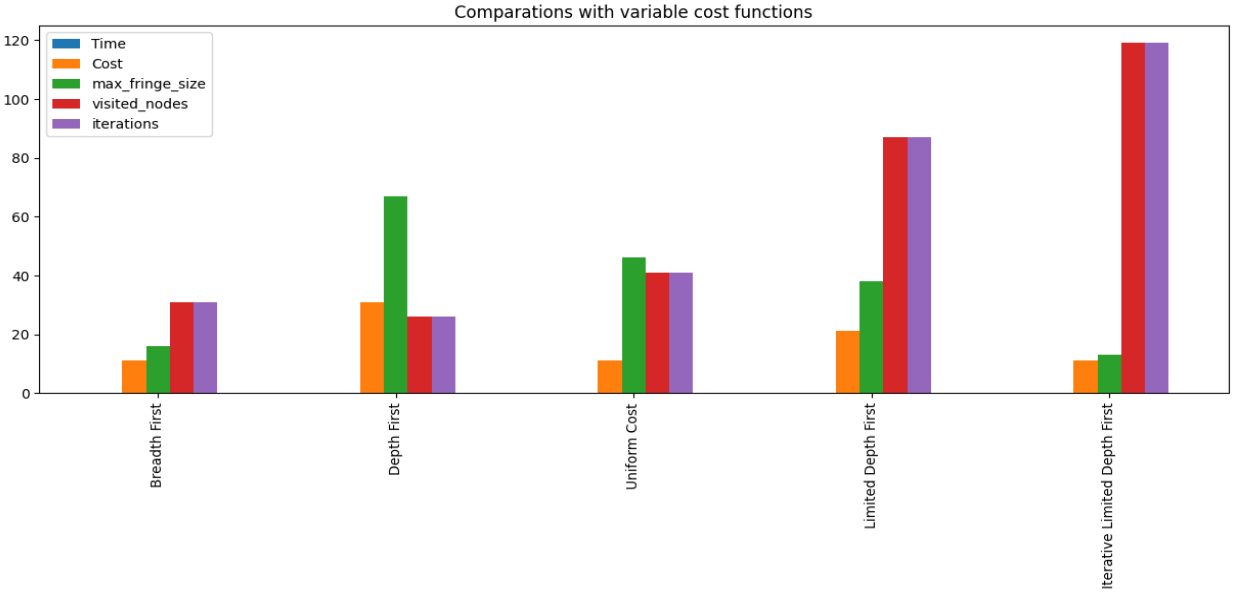


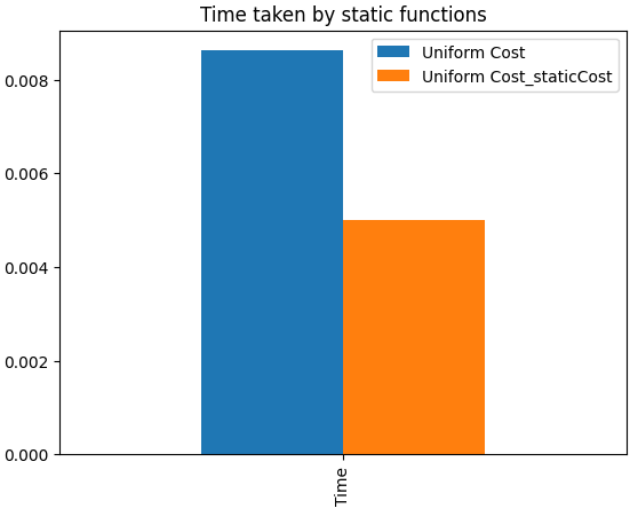


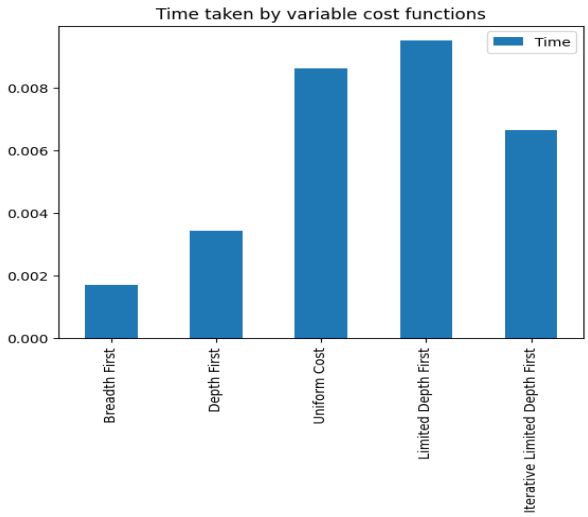


2. Second example Graphs:

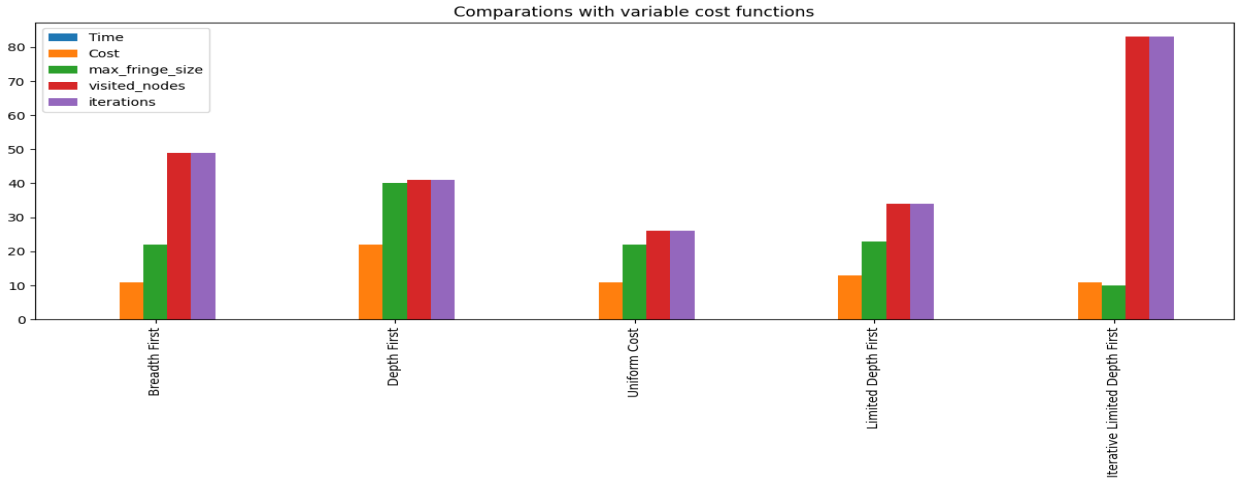


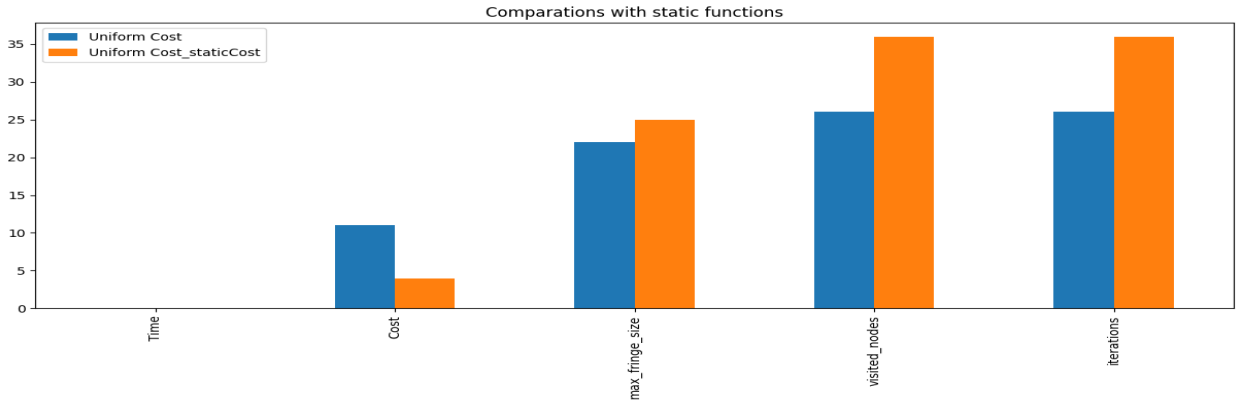


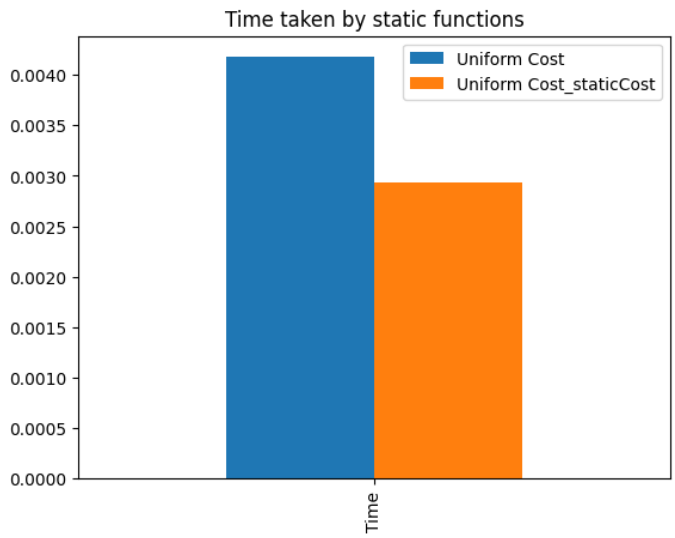


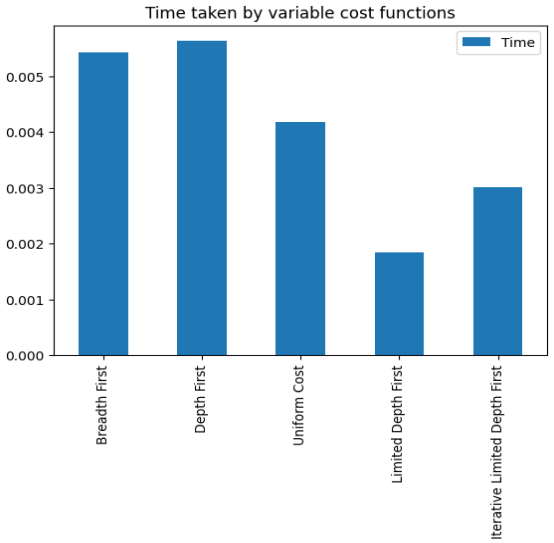


3. Third Example Graphs:









D-Discussion

	Time	Cost	max_fringe_size	visited_nodes	iterations
Breadth First	0.011983	15	29	114	114
Depth First	0.006476	27	60	22	22
Uniform Cost	0.026207	15	49	103	103
Limited Depth First	0.004213	21	38	45	45
Iterative Limited Depth First	0.035656	21	38	570	570

Goal State → (4 , 4 , 0) , Capacity → (8 , 5 , 3) , Cost → Variable, Graph Search → True

In this example we can see Limited Depth Search is perform well, we try different limit values then find optimal for this example, with larger limit value algorithm perform like Depth first but with smaller limit value algorithm can't find a solution.

Iterative limited search try the find optimal limit value by itself so it visit more nodes than others, its find the same path (same cost) with limited Depth search but in much longer timer.

Breadth First and Uniform Cost find the optimal cost, uniform cost take more time because it also consider every step costs, It visit less node than Bread First but use more memory and need more calculation.

Depth first search second fastest algorithm in this example, it doesn't care about cost or memory usage, it has most cost and most memory usage.

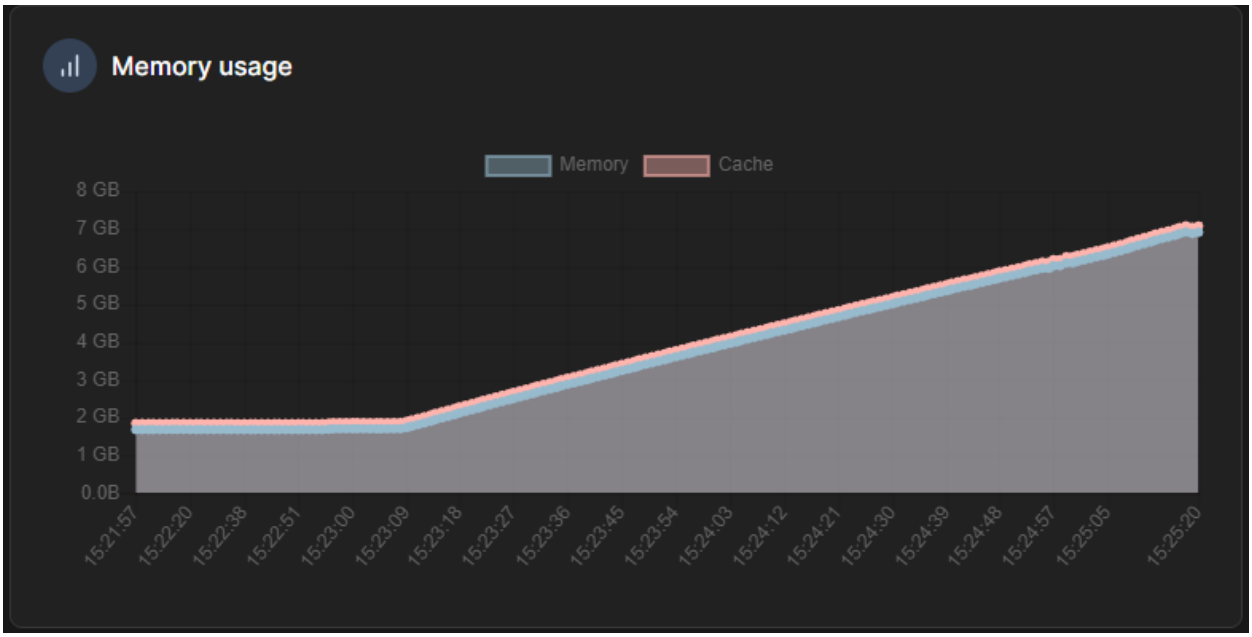
	Time	Cost	max_fringe_size	visited_nodes	iterations
Breadth First	0.020142	8	29	114	114
Depth First	0.002765	21	60	22	22
Uniform Cost_staticCost	0.018719	8	32	133	133
Limited Depth First	0.003561	11	38	45	45
Iterative Limited Depth First	0.040506	11	38	570	570

Goal State → (4 , 4 , 0) , Capacity → (8 , 5 , 3) , Cost → Static, Graph Search → True

Same example with static cost value (1),

Only remarkable change has been happening in Uniform Cost search because as we said it's consider cost values but in this example every iteration have same cost so its confusing for this algorithm.

Time also change a little bit but order of algorithms didn't change so it's probably cause of our computer.



Goal State → (4 , 4 , 0) , Capacity → (8 , 5 , 3) , Cost → Variable , Graph Search → False

We also try Tree search for all algorithms with different input values, in most of the examples we can't get result due to lack of memory, some algorithms runs and give output but there are not enough data for comparison. You can find this part end of the waterjug.ipynb file.