ARTIFICAL INTELLIGENCE

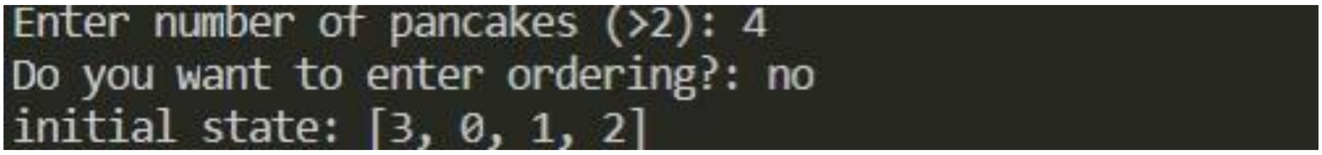
PANCAKE PROBLEM

PREPARED BY: OĞUZHAN ÇEVİK / EGE KUTAY YÜRÜŞEN
170316045/180316017

A- PROBLEM FORMULATION

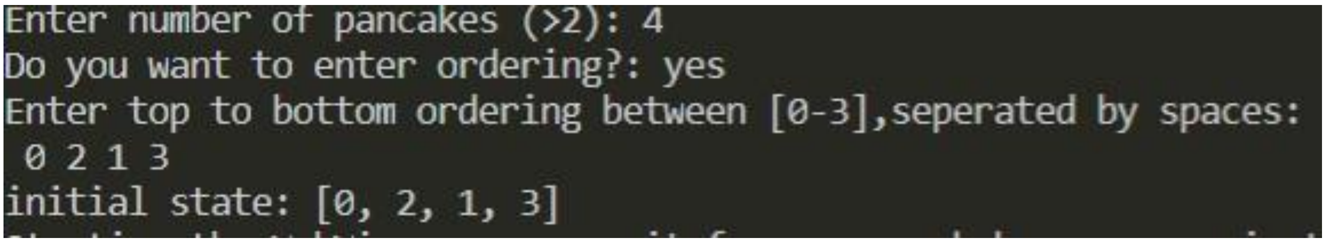
Initial state:

The initial state can be decided from the user by entering numbers separated by space or input can be generated randomly.



```
Enter number of pancakes (>2): 4
Do you want to enter ordering?: no
initial state: [3, 0, 1, 2]
```

OR



```
Enter number of pancakes (>2): 4
Do you want to enter ordering?: yes
Enter top to bottom ordering between [0-3],seperated by spaces:
 0 2 1 3
initial state: [0, 2, 1, 3]
```

Possible Actions:

Our possible actions are limited by count of pancakes. If the count of pancakes is four then there is 3 actions possible which are: flipping top two pancakes, flipping top three pancakes and flipping all the pancakes. However, flipping the first one will change nothing so, it’s not counts as an action.

Transition Model:

In graph model node can`t be repeated but in tree model it can be repeated.

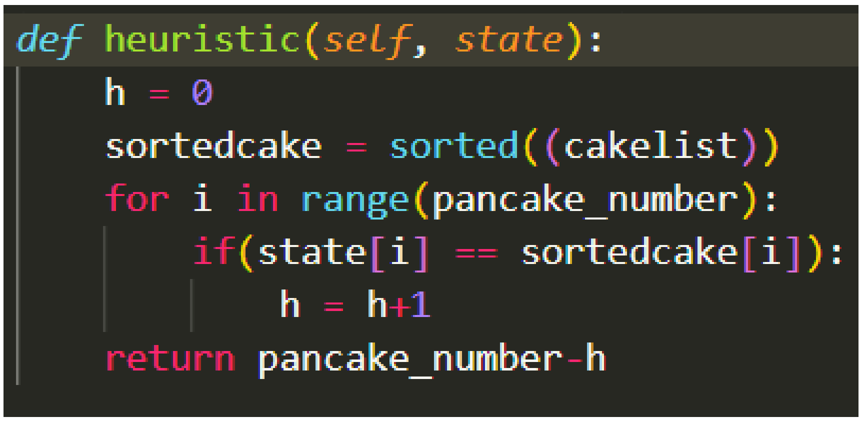
Goal Test:

When our initial state is viable the python code sorts it small to big numbers. Then set it as "goal node". When the search algorithm reaches to the goal node the program is executed.

Path Cost:

In cost part, we are defining our cost as a chosen index to flip. For example; if program decides to flip 3. pancake that means our cost is 3.

B- HEURISTIC FUNCTION



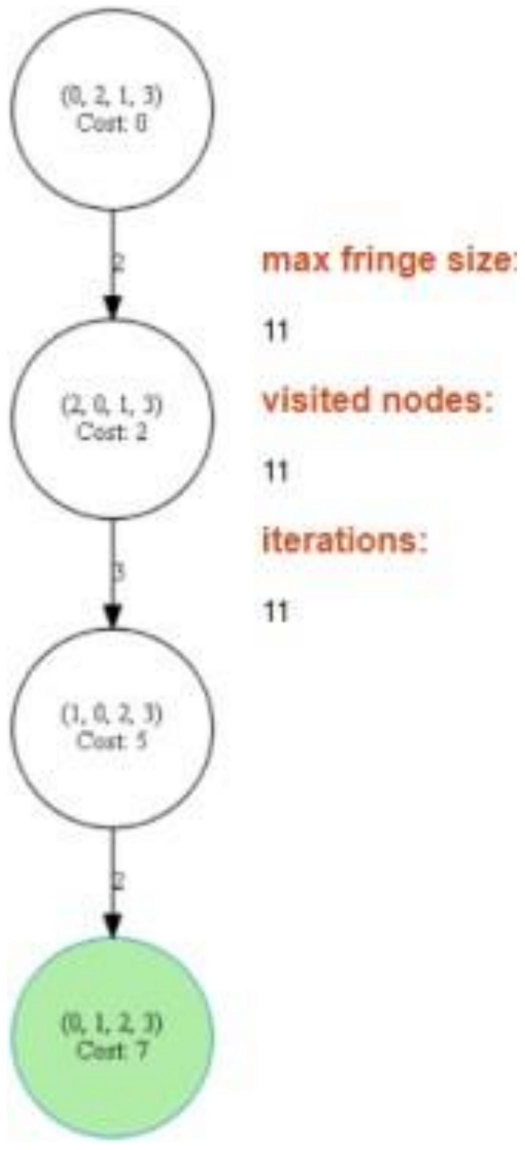
```
def heuristic(self, state):
    h = 0
    sortedcake = sorted((cakelist))
    for i in range(pancake_number):
        if(state[i] == sortedcake[i]):
            h = h+1
    return pancake_number-h
```

This function checks if our current state number spots are the same as goal state's number spots. Further explanation indicates that if each number spots in our current state equals the goal number spots, it increases h value by one and at the end returns pancake number minus h value as a heuristic value.

C- COMPARISON

GRAPH VS TREE SEARCH

GRAPH SEARCH



max fringe size:

11

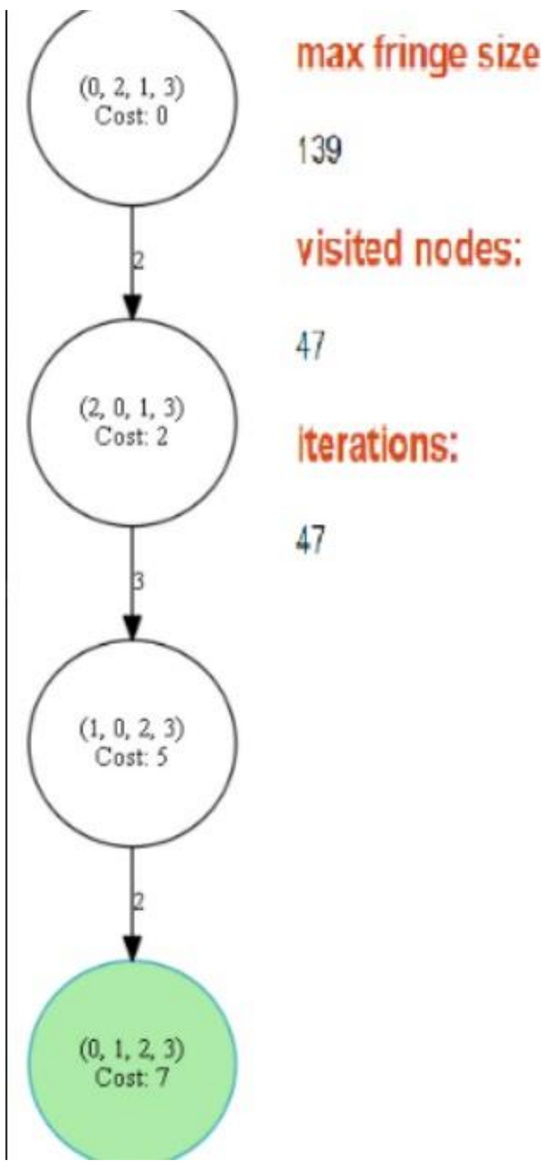
visited nodes:

11

iterations:

11

TREE SEARCH



max fringe size

139

visited nodes:

47

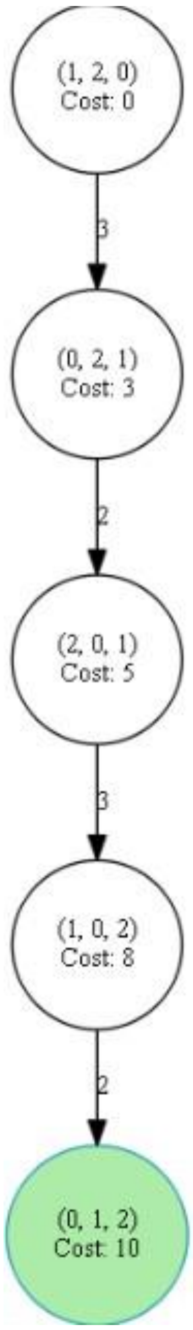
iterations:

47

Using breadth first search by given same inputs, tree search can expand the node that already discovered. Meanwhile in graph search, the program does not expand the node that already discovered. By thus, graph search is better than tree search in terms of complexity and time in this problem.

UNINFORMED SEARCH
VS
INFORMED SEARCH

DEPTH FIRST SEARCH



max fringe size

2

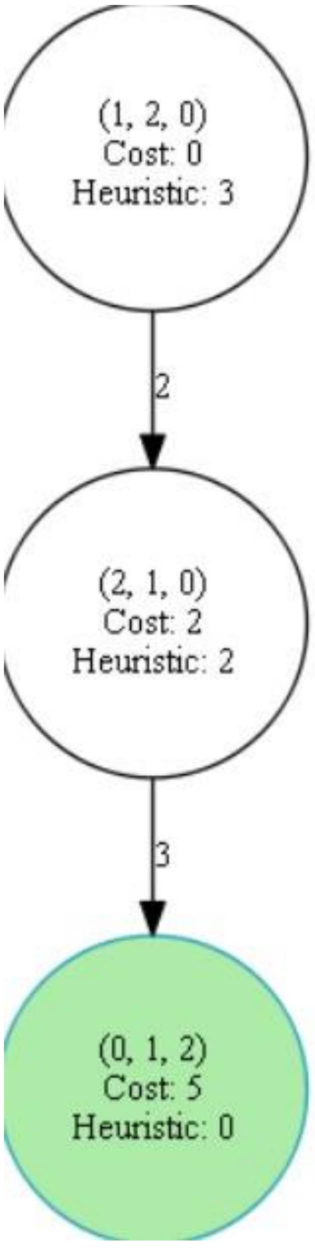
visited nodes:

5

iterations:

5

A*



max fringe size

9

visited nodes:

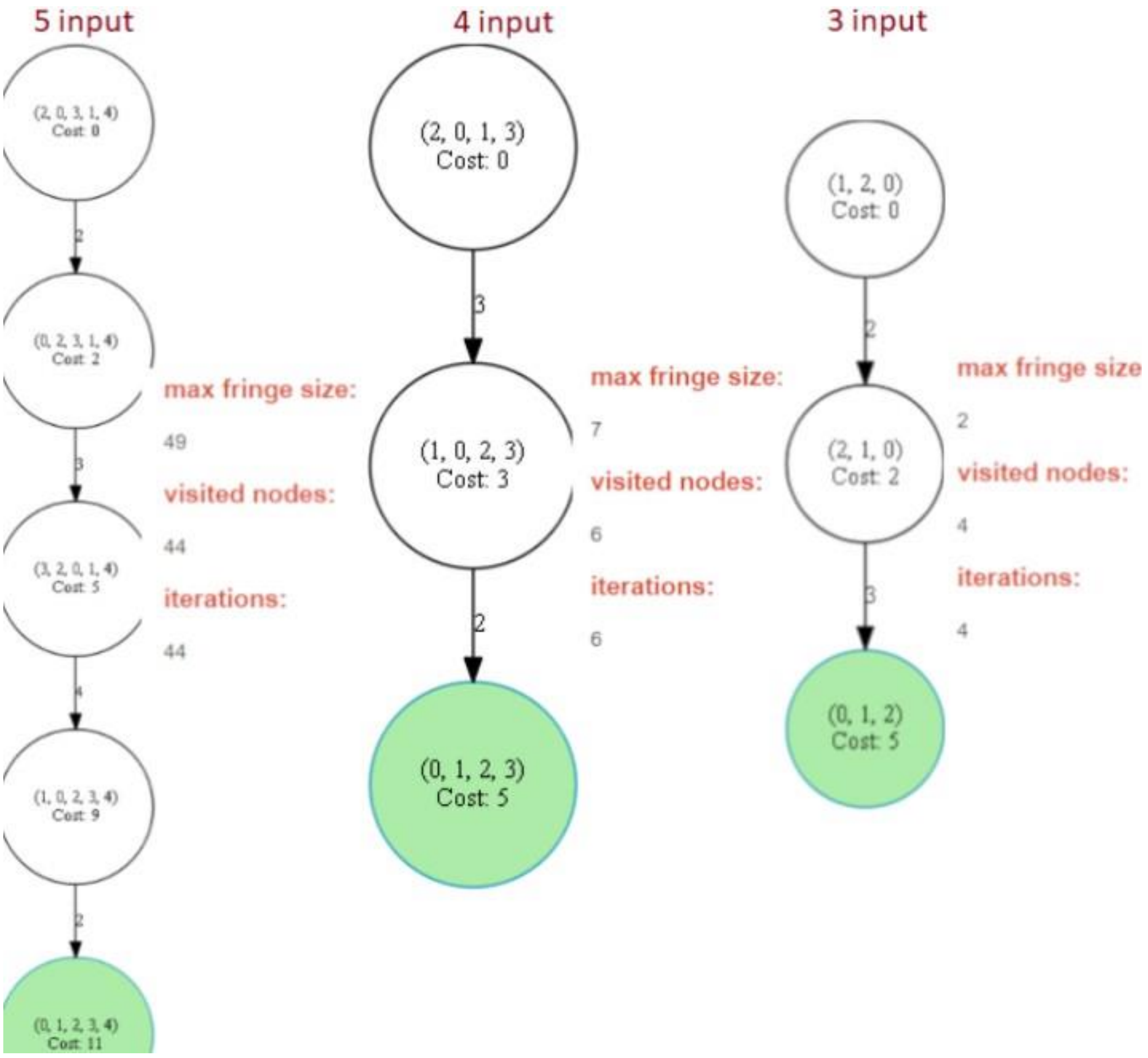
5

iterations:

5

In A* search algorithm cost is better than Depth first search algorithm because A* algorithm gets an idea about goal from heuristic and cost values of the state. Meanwhile DFS is expands as deep as it can expand from left to right until reach to the goal.

UCS



Uniform Cost Search searches for the least cost that reaching to goal. As you can see while number of pancakes increasing, cost, time and complexity also increases.

Sources:

https://github.com/simpleai-team/simpleai   :

AI library used for search algorithms and GUI.

https://stackoverflow.com/questions/17610096/reverse-indexing-in-python   :

Reversing array from index 0 to index k.