IN THE NAME OF GOD

Each problem has 5 basic functions and 2 more functions if they could be run with bidirectional algorithm, also they have a structure for their states.

each state has 4 attributes.
Path: path from beginning up to now.
Val: value that shows the state.
Heuristic: the heuristic number of this state.
Paid: then paid cost up to now.

Also each state has a function called full_paid that is sum of paid and heuristic.

Pitcher problem:

what makes the states different from one problem to another problem is the value attribute.
In this problem I have stored the value of pitchers in an array. First is the 3 liter one and the second is the 4 liter.
The output is an array that shows the path from initial state to the goal state. The initial state is [0,0] and the goal states are [0,2] [1,2] [2,2] [3,2]. each of them become reached, the problem is solved.

- DFS(graph):
  in this algorithm , the path cost was 6 and seen nodes and expanded nodes were 7 and memory at the peak time was 9.
- Bidirectional:
  in this algorithm, the path cost was 6 and seen nodes and expanded nodes were 17 and memory at the peak time was 21.

puzzle problem:

value part of the states in this problem is a two dimensional array(3*3). Each state stores just one permutation of 0 to 8 number.
Then output is an array of actions that shows the path from initial state to the goal state. To edit the initial state you can easily edit “input_puzzle” file.
The default problem is solved in 8 moves. We consider this problem in below.

The heuristic function calculates the manhatan distance.

- BFS(tree):
  in this algorithm, the path cost was 8 and seen nodes and expanded nodes were 296 and memory at the peak time was 515.
- DFS(8 depth):
  in this algorithm, the path cost was 8 and seen nodes and expanded nodes were 2691 and memory at the peak time was 2699.
- A*:
  in this algorithm, the path cost was 8 and seen nodes and expanded nodes were 9 and memory at the peak time was 16.

I realized that if it requires 31 moves then, BFS and DFS are not good algorithms to solve this problem. Because they allocate too much space and also the time is also much worse than A*.

As a POC you can take the 31 moves puzzle from “inputs_puzzle” and replace it in what is in “input_puzzle” file. You will see A* finds the answer in a few seconds but BFS and DFS do not find it in short time!!!!!

robot problem:

value part of the states in this problem is a two dimensional array that the dimensions are got from user.
Then output is an array of actions that shows the path from initial state to the goal state. To edit the initial state you can easily edit “input_robot” file.
The default problem is solved in 19 moves. We consider this problem in below.

- Uniform cost:
  in this algorithm, the path cost was 19 and seen nodes and expanded nodes were 37 and memory at the peak time was 38.
- Bidirectional:
  in this algorithm, the path cost was 19 and seen nodes and expanded nodes were 33 and memory at the peak time was 35.
- A*:
  in this algorithm, the path cost was 19 and seen nodes and expanded nodes were 36 and memory at the peak time was 37.