Homework 1 - Problem Solving Agents using Search Techniques (BFS & DFS)

Initial State

9	24	3	5	17
6	0	13	19	10
11	21	12	1	20
16	4	14	12	15
8	18	23	2	7

Goal State

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	0

Implementing a agent to solve 24 puzzle problem using Search Techniques like below

1. Breadth First Search

2. Depth First Search

Files

The homework report contains two files, one is the code file in python and otjer the other is the README file that explains the execution of the coding.

cs580-hw1.py - solves 5x5 puzzles(24 puzzle) using the search techniques.

Solution

To tackle the problem, we implemented blind search techniques - breadth first search(bfs) and depth first search(dfs).

In the code, we define a state for every problem statement, each state has below attributes.

1. state - has location of all the tiles, including blank one

2. parent - parent of the current state

3. move - from which move did we obtain the current state

4. depth - depth of current state

5. cost - cost of current state

6. key - decided by the heuristic function

The state given above is defined for every valid state in statespace.

Steps to execute

1. execute cs580-hw1.py

2. prompts user to select any choice of algorithm as shown below.

3. enter the algorithm choice

   Algorithm Prompt

   ---> Please select the algorithm

      1. bfs : Breadth First Search

      2. dfs : Depth First Search

   4. Prompts user to enter the puzzle elements.

   Puzzle Prompt

   --> Enter Puzzle elements:

   2, 11, 0, 4, 5, 6, 1, 3, 9, 12, 8, 19, 13, 7, 10, 18, 17, 14, 15, 20, 16, 21, 22, 23, 24

5. Press enter,

       ---> code gets executed, and an output file is generated.

```
257
258   if __name__ == '__main__':
259        main()
```

```
--> Please select the algorithm
1. bfs : Breadth First Search
2. dfs : Depth First Search
3. ast : A Star Search
4. greedy : Greedy Search
 enter the selection : bfs
--> Enter puzzle elements : 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, (
```

Sample Output Files

🗋 output_solvable - bfs - 1.txt	adding initial code files
🗋 output_solvable - bfs.txt	adding initial code files
🗋 output_solvable - dfs - 1.txt	adding initial code files
🗋 output_solvable - dfs.txt	adding initial code files
🗋 output_solvable.txt	organizing files
🗋 output_unsolvable - bfs.txt	adding initial code files
🗋 output_unsolvable - dfs.txt	adding initial code files

Belown is one of the sample output file.

```
Problem State : [6, 1, 2, 3, 4, 11, 7, 8, 9, 5, 16, 12, 13, 14, 10, 21, 17, 18, 19, 15, 0, 22, 23, 24, 20]
path_to_goal : ['U', 'U', 'U', 'U', 'R', 'R', 'R', 'R', 'D', 'D', 'D', 'D']
cost_of_path(Moves): 12
nodes_expanded: 12
fringe_size: 10
max_fringe_size: 11
search_depth: 12
max_search_depth: 12
running_time: 0.00098896
```

output_solvable - bfs - 1

<-- # SOLVABLE # -->

path_to_goal: ['Down', 'Down', 'Right', 'Right', 'Up', 'Up', 'Left', 'Left', 'Down', 'Down', 'Right', 'Right', 'Up', 'Up', 'Left', 'Left', 'Down', 'Down', 'Right', 'Right', 'Up', 'Up', 'Left', 'Left', 'Down', 'Down', 'Right', 'Right']

cost_of_path: 28

nodes_expanded: 178223

fringe_size: 2675

max_fringe_size: 24048

search_depth: 28

max_search_depth: 29

running_time: 9.63447360

Conclusion

From the initiaal analysis done on the 3x3 we were able to solve all the valid statespaces to reach the goal state using bfs, dfs. But when we implement the same code for 5x5, the following are the observations :

- using bfs we were not able to solve all the problems because in worst case, we need to traverse all the combinations in each level and creating many such levels leads to utilize more memory (space complexity $O(b^d)$)

- using dfs it runs forever as it has to reach the depth first(time complexity).

Observations and manipulations

Since the given initial state is costing a lot of time and we understood that the algorithm is working, the input has been manipulated with several inputs to check the functionalities.

Initial Test : 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,0,24

Solvable : 2,11,0,4,5,6,1,3,9,12,8,19,13,7,10,18,17,14,15,20,16,21,22,23,24

Final Test : 9,24,3,5,17,6,0,13,19,10,11,21,12,1,20,16,4,14,12,15,8,18,23,2,7

#----#-- BFS - Breadth First Search --#----#

Sno	board	number of moves	Time Taken (Seconds)	solution (s)
1	1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,0,24	1	0	R
2	1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,0,23,24	2	0.0009706	RR

	board			
3	1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,0,21,22,23,24	4	0	RRRR
4	1,2,3,4,0,6,7,8,9,5,11,12,13,14,10,16,17,18,19,15,21,22,23,24,20	4	0.00099707	DDDD
5	1,2,3,0,4,6,7,8,9,5,11,12,13,14,10,16,17,18,19,15,21,22,23,24,20	5	0.00498605	RDDDD
6	0,1,2,3,4,6,7,8,9,5,11,12,13,14,10,16,17,18,19,15,21,22,23,24,20	8	0.05382967	RRRRDDDD
7	6,1,2,3,4,11,7,8,9,5,16,12,13,14,10,21,17,18,19,15,0,22,23,24,20	12	1.33746266	UUUURRRRDDDD
8	6,1,2,3,4,11,7,8,9,5,16,12,13,14,10,21,17,18,19,15,22,23,24,20,0	16	16.48596382	LLLLUUUURRRRDDDD
9	6,1,2,3,4,11,7,13,8,9,16,12,0,14,5,21,17,18,19,10,22,23,24,20,15	Programs Crashes as the memory is full		

The BFS has been executed for several input values and it is costing a lot of time after a certain stage.

#----#-- DFS - Depth First Search --#----#				
Sno	board	number of moves	Time Taken (Seconds)	solution (s)
1	1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,0,24	1	0	R
2	1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,0,23,24	Program executes forever as there are many branches because of the depth.		

The DFS algorithm was executed for one possible move in no time but if the "0" is event 2 places away it went on running and stopped working.