CS480/580 Introduction to Artificial Intelligence
Herambeshwar Pendyala | 01130541 | Assignment 1

Total Points: 100
Due Date: 9/26/2019

The 24-puzzle problem

The 24-puzzle is a larger version of the 8-puzzle.
Goal:

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

Initial

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

For this programming assignment, you will create a set of search algorithms that find solutions to the 24-puzzle problem. You are requested to implement the programs to the 24-puzzle problem using

1. **breadth-first** search (BFS)
2. **depth-first** search (DFS)
3. **informed** search algorithms using
   h1(x) = number of misplaced tiles
   and
   h2(x) = sum of the distances of every tile to its goal position.

For each of the search routines, avoid returning to states that have already been visited on the current solution path i.e., there should be no repeated states in a solution.

What to Hand in

1. Well documented codes implementing breadth first search, depth first search, and informed search. A README file should provide instructions on how to compile and execute the code.
2. Provide the sample solutions generated by your programs using BFS, DFS, and informed search.
3. Analysis of your program. Compare the computational time and lengths of solutions of BFS, DFS, and informed search using different heuristic functions.

Please send the program and the analysis report to yaohang@cs.odu.edu before the assignment due date.

Hints:

1. It is much better to model the problem of moving the blank around than moving movable tiles.
2. Good data structure representation can make your life easy.
3. You may want to start from the 8-puzzle problem.

Solution

Executing the Code

1. execute sherlock.py
   Command - python sherlock.py
2. prompts user to select any choice of algorithm as shown below.

3. enter the algorithm choice

```
--> 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 : greedy
```

4. If choice is a greedy search or A star, program prompts for heuristic function as shown below, user can select one. else Code moves to step 5

```
-- Please select the Heuristic Function
 h1 : number of misplaced tiles
 h2 : sum of the distances of every tile to its goal position.
-- Enter your choice : h2
```

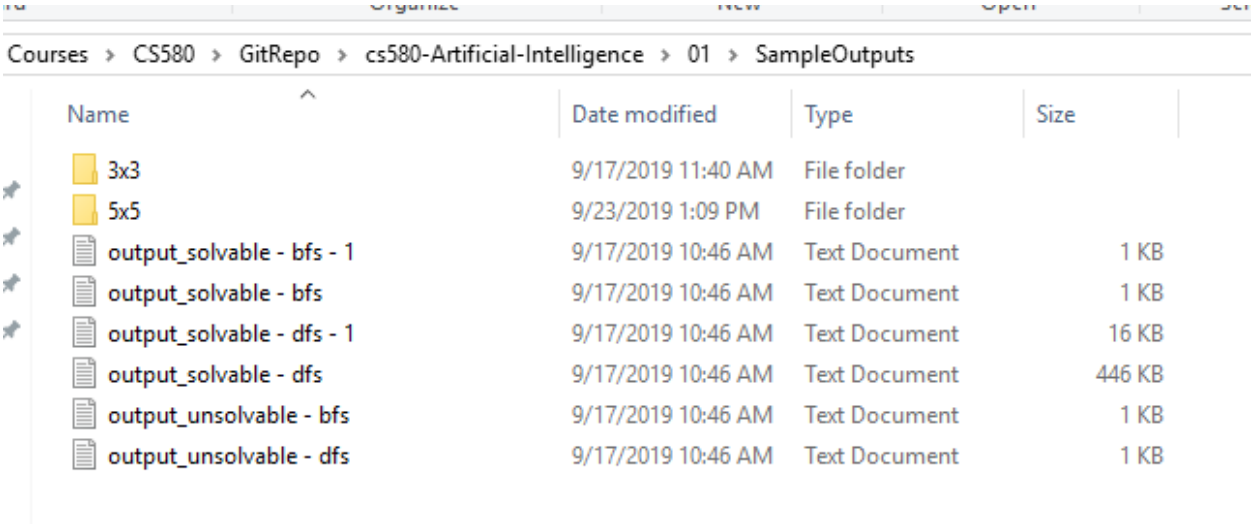
5. Prompts user to enter the puzzle elements.

```
--> 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
```

6. press enter code gets executed and an output file is generated.

Sample Outputs

All the sample outputs are in the directory /SampleOutputs



Below is one of the sample output

```
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
```

Analysis and Conclusion

From the initial analysis done on the 3x3 we were able to solve all the valid state spaces to reach the goal state using bfs, dfs, A star, Greedy search. but when we implement the same code for 5x5:

- 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 must reach the depth first (time complexity).

- using Informed searches, we were able to solve the problem as we defined good heuristic function. we can see that in the given heuristic functions, 2nd heuristic function outperforms 1st as we can get the best solution in less time.

Heuristics	**H1**	**H2**
Computational Time	11.33088827	1.56281567
Solution length	698	450

So we can see why blind search techniques cannot be used in real time and how defining a good heuristic function can help us solving the problem also saving time and space.