Yujia Li A98064697

Ze Li A11628864

Wei Wang A97031723#

[yul200@ucsd.edu](mailto:yul200@ucsd.edu)

[zel014@ucsd.edu](mailto:zel014@ucsd.edu)

[wew026@ucsd.edu](mailto:wew026@ucsd.edu)

CSE150 Assignment 1

Problem 6

**Problem 2**

Description of the problem and the algorithms used

The problem is to implement a algorithm to solve random input puzzles, see if it can be solved by the 8-puzzle rules, no matter how many steps it takes. The algorithm used is BFS, which is one of the graph-search algorithm.

Data structure used

The data structure used to implement BFS graph search is Queue(FIFO). In this way, the positions at level n would be inserted before any positions at level n+1, and de-queue before any positions at level n+1 as well, which is exactly by the rules of BFS.

**Problem 3**

Description of the problem and the algorithms used

The problem is to implement a algorithm to solve random input puzzles, see if it can be solved by the 8-puzzle rules, and it can take no more than 5 steps. The algorithm used is DFS.

Data structure used

The data structure used to implement DFS graph search is Stack(FILO). In this way, the positions at level n would be inserted before any positions at level n+1, and de-queue after any positions at level n+1, which is exactly by the rules of DFS, search down to a path till dead end, come back and start another path.

**Problem 4**

Description of the problem and the algorithms used

The problem is to implement a algorithm to solve random input puzzles, see if it can be solved by the 8-puzzle rules, and it can take no more than 12 steps. The algorithm used is Deeping depth-first algorithm.

Data structure used

The data structure used to implement Deeping depth-first search is Stack(FILO). In this way, the positions at level n would be inserted before any positions at level n+1, and de-queue after any positions at level n+1, which is exactly by the rules of DFS, search down to a path till dead end, come back and start another path. By limit the depths it can take each iteration and call it 12 times. If the results is found before 12 times, return the result.

**Problem 5**

Description of the problem and the algorithms used

The problem is to implement a algorithm to solve random input puzzles, see if it can be solved by the 8-puzzle rules. The algorithm used is A\* algorithm, with Manhattan distance as the heuristic.

Data structure used

The data structure used to implement A\* graph search is Priority Queue. In this way, each position would be sorted every time there is a new position inserted into the Priority Queue based on their priority. Their priority , in this situation, is the addition of steps it visited and the steps to the destination.

The below table shows how many insertions into the data structure each method need to go over to find the correct solution:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Method 2 | Method 3 | Method 4 | Method 5 | Actual steps |
| 6,2,0,3,4  1,5,7,8,9  10,11,12,13,14  15,16,17,18,19  20,21,22,23,24 | 114  0m0.053s | 91  0m0.038s | 91  0m0.052s | 37  0m0.038s | LDRUL |
| 5,4,2,3  1,9,6,0  8,10,11,7 | 1433  0m0.210s | 83  0m0.032s | 873  0m0.265s | 938  0m0.133s | DLLUULDRUL |
| 1,4,2  3,0,5 | 4  0m0.024s | 2  0m0.027s | 2  0m0.026s | 3  0m0.029s | UL |
| Time Complexity |  |  |  |  |  |
| Space Complexity |  |  |  |  |  |

Space complexity of Different Method

Results of the analysis and a short discussion

A paragraph from Yujia Li stating what their contribution was and what they learned

I helped implement algorithms into Python languages by using different data structures, and debugged till they work. Physically using a language to implement these algorithms helps me understand how they are different from each other, and specific data structure to hold and why.

A paragraph from Ze Li stating what their contribution was and what they learned

A paragraph from Wei Wang stating what their contribution was and what they learned