Name: Ha Viet Pham

CSCE 420

**User manual**

Programming language use: C++

IDE used: Eclipse

Purpose of program: to solve 8-puzzle by using bfs, dfs, ids, greedy and astar algorithms.

**Result**

**DFS**

Easy puzzle:

Input: (dfs '(1 3 4 8 6 2 7 0 5))

**BFS:**

Easy puzzle:

Input: (bfs '(1 3 4 8 6 2 7 0 5))

Output: (up right up left down)

Maximum length of the node list is 36

Number of nodes visited is 41

Medium puzzle:

Input: (bfs '(2 8 1 0 4 3 7 6 5))

Output: (up right right down left left up right down)

Maximum length of the node list is 278

Number of nodes visited is 385

**IDS:**

Easy puzzle:

Input: (ids '(1 3 4 8 6 2 7 0 5))

Output: (up right up left down)

Maximum length of the node list is 7

Number of nodes visited is 42

Medium puzzle:

Input: (ids '(2 8 1 0 4 3 7 6 5))

Output: (up right right down left left up right down)

Maximum length of the node list is 11

Number of nodes visited is 442

**Greedy:**

Easy puzzle:

Input (with h1): (greedy '(1 3 4 8 6 2 7 0 5) 'h1)

Output: (up right up left down)

Maximum length of the node list is 20

Number of nodes visited is 9

Easy puzzle:

Input (with h2): (greedy '(1 3 4 8 6 2 7 0 5) 'h2)

Output: (up right up left down)

Maximum length of the node list is 25

Number of nodes visited is 10

Running time: 0

Medium puzzle:

Input (with h1): (greedy '(2 8 1 0 4 3 7 6 5) ‘h1)

Output: (up right right down left left up right down)

Maximum length of the node list is 111

Number of nodes visited is 148

Medium puzzle:

Input (with h2): (greedy '(2 8 1 0 4 3 7 6 5) ‘h2)

Output: (up right right down left left up right down)

Maximum length of the node list is 45

Number of nodes visited is 34

Hard puzzle:

Input (with h2): (greedy '(5 6 7 4 0 8 3 2 1) 'h2)

Output: (down left up up right down down left up up right down down left up right right up left left down right right down left up right up left down left up right right down left down left up right right down left left up right down right up left)

Maximum length of the node list is 204

Number of nodes visited is 196

**A\*:**

Easy puzzle:

Input (with h1): (astar '(1 3 4 8 6 2 7 0 5) 'h1)

Output: (up right up left down)

Maximum length of the node list is 9

Number of nodes visited is 6

Running time: 0

Easy puzzle:

Input (with h2): (astar '(1 3 4 8 6 2 7 0 5) 'h2)

Output: (up right up left down)

Maximum length of the node list is 8

Number of nodes visited is 5

Medium puzzle:

Input (with h1): (astar '(2 8 1 0 4 3 7 6 5) ‘h1)

Output: (up right right down left left up right down)

Maximum length of the node list is 29

Number of nodes visited is 31

Medium puzzle:

Input (with h2): (astar '(2 8 1 0 4 3 7 6 5) ‘h2)

Output: (up right right down left left up right down)

Maximum length of the node list is 14

Number of nodes visited is 15

**Analysis:**

DFS fails to produce output because of its usage of too much space.

For easy puzzle:

-Maximum length of the node in list comparison: IDS(7) < A\*(9/8) < Greedy(20/25) < BFS(36) <DFS. This order is also the order of the comparison of space-complexity of the algorithms since it takes memory to store the nodes into list.

- Number of nodes visited comparison: A\*(6/5) <greedy(9/10) <BFS(41) <IDS(42)<DFS. This order is also the order of the comparison of time-complexity of the algorithms since it takes time to visit and expand each node.

For medium puzzle:

-Maximum length of the node in list comparison: IDS(11) < A\*(29/14) < Greedy(111/45) < BFS(278) <DFS. This order is also the order of the comparison of space-complexity of the algorithms since it takes memory to store the nodes into list.

- Number of nodes visited comparison: A\*(31/15) <greedy(148/34) <BFS(385) <IDS(442)<DFS. This order is also the order of the comparison of time-complexity of the algorithms since it takes time to visit and expand each node.

For hard puzzle:

-the order of the comparison of space-complexity is the same as for medium puzzle and easy puzzle.

-the order of the comparison of time-complexity is the same as for medium puzzle and easy puzzle.

=> Based on the orders above, we can tell that:

-the strength of IDS is space-saving, and the weakness of it is time-consuming.

-the strength of A\* is relatively both space-saving and timesaving. However, for A, we have to construct heuristic function (need more information: dept + h1/h2) to implement it, which is its weakness.

-greedy strengths and weaknesses are the same with A\*. Its strength is not as good as A\* but it also needs less information (only h1/h2) to implement it.

-BFS is both time-consuming and space-consuming (weakness), but we do not need additional information to implement it (do not need dept and h1/h2), which is the strength. BFS is also better for nodes that are close to the root, but not good for search nodes that are further away from the root.

- DFS is both time-consuming and space-consuming (weakness), but we do not need additional information to implement it (do not need dept and h1/h2), which is the strength. DFS is also better for nodes that lie more to the left of the tree, but not good for nodes that lie more to the right and further to root.