Bryson Coons, CS480 Assignment 3 Analysis

In this assignment I implemented Greedy Best First Search (GBFS) algorithms to find solutions to the 24-puzzle problem. The program's main driver file, ***informed_search.py***, runs a program which can create new instances of the 24-puzzle problem. These may either be scrambled random positions, or pre-prepared positions in the form of a string. Position strings are formatted as a string of integers, 0 - 24, each appearing exactly once, delimited by spaces. These integers represent the values in each cell of the 5x5 board, starting with the top row from left to right, then the second row from left to right, and so on.

Positions may either be played manually, or a search algorithm may be implemented from the ***puzzle_search.py*** module.Both algorithms behave very similarly, valuing which node in the frontier to expand next based on the evaluation of its heuristic function, ***h(x)***. In the first algorithm, h(x) represents the number of blocks misplaced in its world state. In the second, h(x) represents the sum of distances between the actual positions of blocks in the world state and their goal positions. Both of these ended up acting identically in the cases where they could find solutions to problems, however the first algorithm ended up significantly more reliable and predictable than the second algorithm.

Below are a few hand-selected strings for the grader of this paper to paste into the program and load into the game, to observe my algorithms in real time:

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

A simple board state one move away from winning. The solution is very easy, just inputting right will complete the puzzle immediately. Using the gbfs1 or gbfs2 function in this position will return the correct solution extremely quickly, as the solution can be found as early as the first node expansion.

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

```
PS G:\CS480\Asst03_2> & C:/Users/Admin/AppData/Local/Programs/Python/Python313/python.exe g:/CS480/Asst03_2/src/informed_search.py
Welcome to Block Puzzle!
Would you like to start from a random position, or load from a string?
Type "new", or "load":
Selection:load
Please paste a valid string from a previous game:1 2 3 5 10 6 8 9 4 15 11 7 13 14 0 16 12 18 19 20 21 17 22 23 24
Type: up, down, left, right, string, gbfs1, gbfs2, or quit.

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

Move: gbfs1
Searching for solution using greedy best-first search algorithm...
h(x) = # of misplaced blocks...
Solution found!
Solution: ['U', 'U', 'L', 'D', 'L', 'L', 'D', 'D', 'D', 'R', 'R', 'R']
 1  2  3  5 10
 6  8  9  4 15
11  7 13 14
16 12 18 19 20
21 17 22 23 24

Move: gbfs2
Searching for solution using greedy best-first search algorithm...
h(x) = # sum of block displacements...
Solution found!
Solution: ['U', 'U', 'L', 'D', 'L', 'L', 'D', 'D', 'D', 'R', 'R', 'R']
 1  2  3  5 10
 6  8  9  4 15
11  7 13 14
16 12 18 19 20
21 17 22 23 24
```

This is a board which has been scrambled to be 12 moves away from the solution. While BFS and DFS were able to find solutions to this state, they took about 30 seconds to find the solution. The algorithms aided by heuristics, however, are much more adept at finding trivial solutions to puzzles close to being solved. However, both algorithms tend to fail after “loops” of moves lie between the world state and the goal state. In other words, the algorithms can fall into heuristic “valleys” in complex states.

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

This is the board state found in the original problem description for the assignment. I included this in case the person grading this submission wanted to observe how the program behaved under the original problem’s circumstances. For the record, neither of the algorithms provided will find the solution to this problem in a reasonable amount of time, as the solutions have paths too deep for an algorithm as primitive as this to find.