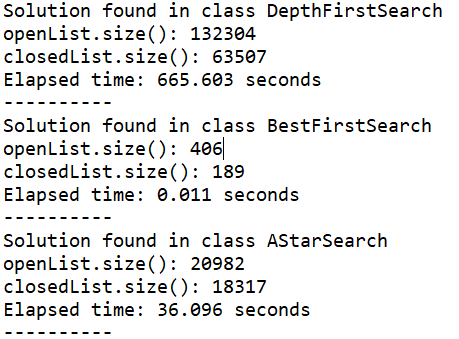
Notes (to be considered for report):

* ~~Currently solving using recursion – if finding a solution takes long enough, it will cause StackOverflowError even if the puzzle is known to be solvable~~
* ~~Could probably use some more diagnostic tools (e.g. total moves, time taken to find solution, etc.)~~
* ~~DFS consistently causes StackOverflowError~~
* A\* has worse performance than BFS due to plateaus
  + f(n) for A\* never decreases as it goes deeper in the search tree
  + BFS is consistently faster than A\*, but A\* has shorter solution paths
  + Choice of heuristic seems to have a more pronounced effect on A\* than on BFS
  + Can only be seen when stack overflow/no solution found
* How to know that h is admissible?
  + Run breadth-first search?
* To check the absolute shortest path, run breadth-first search, which is known to be optimal
  + HOWEVER: knowing that B = 8 (at most), and that some puzzles take dozens of steps to solve, breadth-first search quickly becomes unfeasible
* ~~Some puzzles are either unsolvable or take a very long time to solve (all three algorithms return StackOverflowError)~~
* Recursion good for testing, bad for accurate results (timeout before solution is found)
* Iterative implementation: may run for a very, very long time, but will \*probably\* find a solution
* Many proofs and formulas to check for solvability of 8-puzzle with standard 4 moves, but none with diagonal moves
  + Out of this mini-project’s scope
* Only way to show “no solution found” message: Exhaust every possible state (takes a long time) or stack overflow
* Changing puzzle dimensions to 3x3 is also good for testing (smaller state space)
  + For puzzle 3 5 7 4 6 2 8 0 1
  + 

|  |  |
| --- | --- |
| **NGUYEN, Réal** | **SOEN 472 SEC F** |
| **ID#27566263** | **DUE 15/10/2018** |
|  |  |

**MINI-PROJECT 1 REPORT**

**Introduction**

The purpose of this report is to analyse the results and efficiency of uninformed and informed search algorithms, the informed algorithms’ use of different heuristics on a variation of the classic sliding 8-puzzle, with a dimension of 3x4 and with legal diagonal moves.

I will first compare the use of the different heuristics (or lack thereof) in the search algorithms; second, I will discuss the difficulties I ran into while developing and testing the program; third, I will give details on various experiments I integrated into the program to get different results; and finally, I will conclude the report with short insights on what I have learned from this mini-project.

**Heuristics**

**Difficulties**

**Experiments**

**Conclusion**

**References**