ID: 2017B3A70285G

Approach taken: **A\* Search with Additive Disjoint Pattern Database** heuristics

Patterns used (4-4-4-3 format):

\* 1 2 3 \* \* \* \* \* \* \* \* \* \* \* \*

\* \* \* 7 4 \* \* \* \* 5 6 \* \* \* \* \*

\* \* \* \* 8 \* \* \* \* 9 A \* \* \* \* B

\* \* \* \* C D \* \* \* \* \* \* \* \* E F

*(Patterns were chosen such that the number of linear conflicts in each pattern could be maximized within file size restrictions)*

All tiles are segregated into patterns such that each tile is part of only one pattern (called disjoint patterns). Using breadth-first search, pattern databases of: the minimum number of moves of the tiles in each group that are required to get those tiles to their goal positions are precomputed. Only those moves which involve one of the tiles in the group are counted making them truly disjoint.

A heuristic value for A\* search can be determined for each puzzle state by *adding* the corresponding costs obtained from the pattern databases. They can be added because pattern databases are independent and do not interfere with each other.

A pattern is a relaxed representation of the actual goal state of the puzzle where we are counting only the moves made by the tiles part of the pattern. When we add the values obtained for each pattern, the resulting estimate will never be greater than the actual cost which will be incurred on solving the problem as the interactions between tiles from two different pattern groups are not taken into account. This makes the heuristic admissible.

Since the heuristic is admissible, the tree-search variant of A\* is optimal. Thus, the approach I’ve used always gives the optimal solution for any valid input.

**All Approaches I tried**:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Search Strategy** | **Heuristic** | **initial\_state1.txt** | | **initial\_state2.txt** | | **initial\_state3.txt** | | **initial\_state4.txt** | |
| Nodes generated | Time taken (s) | Nodes generated | Time taken (s) | Nodes generated | Time taken (s) | Nodes generated | Time taken (s) |
| A\* search | Manhattan distance | 31 | 0.00 # | 2,55,001 | 3.53 # | 23,34,860 | 37.80 # | 13,02,476 | 19.85 # |
| A\* search$@ | Non-additive pattern databases | 32 # | 0.32 # | 2,06,400 # | 3.40 # | 12,53,500 # | 29.70 # | 24,19,916 # | 48.30 # |
| Bidirectional A\* search&@ | Forward: Disjoint pattern databases  Backward: Manhattan distance | 31 | 0.75 # | 60,637 | 2.23 # | 3,05,619 | 12.37 # | 2,86,726 | 12.62 # |
| A\* search$@ | Additive disjoint pattern databases | 31 | 0.51 # | 24,529 | 0.53 # | 1,93,639 | 4.77 # | 1,24,459 | 2.71 # |
| *#on average; $with randomness; &with appropriate check for optimality; @supplementary file used;* | | | | | | | | | |