Panagiotis Photiadis 1031373

```
Give puzzle B = black, W = white and E = empty
bbbwwwe
Inital state size: 7

Solution path (from start to goal):
BBBWWWE Description: Initial State F: 27
BBWBWWE Description: Swap B and W at positions 2 and 3 F: 26
BWBBWWE Description: Swap B and W at positions 1 and 2 F: 25
WBBBWWE Description: Swap B and W at positions 0 and 1 F: 23
WBBWBWE Description: Swap B and W at positions 3 and 4 F: 21
WBWBBWE Description: Swap B and W at positions 2 and 3 F: 19
WWBBBWE Description: Swap B and W at positions 1 and 2 F: 17
WWBBWBE Description: Swap B and W at positions 4 and 5 F: 15
WWBWBBE Description: Swap B and W at positions 3 and 4 F: 13
WWWBBBE Description: Swap B and W at positions 2 and 3 F: 11

Nodes created: 44

Execution time: 40 ms

Used memory in bytes: 661848.0
Used memory in megabytes: 0.63118744
```

To solve “BBBWWWE” the Astar created 44 nodes and needed 40ms to find the solution “WWWBBBE” where all W are on the left site and all B are on the right independently of the position of E.

The heuristic (h) used was the distance to the solution.

```java
public int calculateHeuristic(char[] table) {
    int heuristic = 0;
    // Count how far each 'W' is from its target position on the left
    // and each 'B' from its target position on the right.
    for (int i = 0; i < table.length; i++) {
        char c = table[i];
        if (c == 'W') {
            // For 'W', the cost is its index, as we want it to move to the leftmost side.
            heuristic += i;
        } else if (c == 'B') {
            // For 'B', the cost is the distance from the rightmost side.
            heuristic += table.length - 1 - i;
        }
    }
    return heuristic;
}
```

And the cost (g) of the move of the current node plus the costs of all the moves that executed on the parent nodes.

```
int tentativeG = currentNode.getG() + successor.getCost();  // The cost to move to the successor
```

The heuristic of distance to solution is admissible because counting misplaced pieces without overestimating the moves required to place them correctly would satisfy admissibility. Also, the solution given is always the best because the successor nodes are added to a priority queue based on f = h+g so it always expands the node that is closer to the solution and has the cheaper cost on moves. By saving already visited nodes into a table it reassures the avoidance of looping around the same nodes.

The program can solve puzzles bigger than 7 e.x

```
Give puzzle B = black, W = white and E = empty
wbwbwbwewbwbwbw
Inital state size: 15

Solution path (from start to goal):
WBWBWBWEWBWBWBW Description: Initial State F: 98
WWBBWBWEWBWBWBW Description: Swap B and W at positions 1 and 2 F: 97
WWBWBBWEWBWBWBW Description: Swap B and W at positions 3 and 4 F: 96
WWWBBBWEWBWBWBW Description: Swap B and W at positions 2 and 3 F: 94
WWWBBWBEWBWBWBW Description: Swap B and W at positions 5 and 6 F: 92
WWWBWBBEWBWBWBW Description: Swap B and W at positions 4 and 5 F: 90
WWWWBBBEWBWBWBW Description: Swap B and W at positions 3 and 4 F: 88
WWWWBBBEWWBBWBW Description: Swap B and W at positions 9 and 10 F: 86
WWWWBBBEWWBWBBW Description: Swap B and W at positions 11 and 12 F: 84
WWWWBBBEWWWBBBW Description: Swap B and W at positions 10 and 11 F: 82
WWWWBBBEWWWBBWB Description: Swap B and W at positions 13 and 14 F: 80
WWWWBBBEWWWBWBB Description: Swap B and W at positions 12 and 13 F: 78
WWWWBBBEWWWWBBB Description: Swap B and W at positions 11 and 12 F: 76
WWWWBBEBWWWWBBB Description: Block at 6 moves to 7 F: 75
WWWWBBEWBWWWBBB Description: Swap B and W at positions 7 and 8 F: 73
WWWWBBEWWBWWBBB Description: Swap B and W at positions 8 and 9 F: 71
WWWWBBEWWWBWBBB Description: Swap B and W at positions 9 and 10 F: 69
WWWWBBEWWWWBBBB Description: Swap B and W at positions 10 and 11 F: 67
WWWWBEBWWWWBBBB Description: Block at 5 moves to 6 F: 66
WWWWBEWBWWWBBBB Description: Swap B and W at positions 6 and 7 F: 64
WWWWBEWWBWWBBBB Description: Swap B and W at positions 7 and 8 F: 62
WWWWBEWWWBWBBBB Description: Swap B and W at positions 8 and 9 F: 60
WWWWBEWWWWBBBBB Description: Swap B and W at positions 9 and 10 F: 58
WWWWEBWWWWBBBBB Description: Block at 4 moves to 5 F: 57
WWWWEWBWWWBBBBB Description: Swap B and W at positions 5 and 6 F: 55
WWWWEWWBWWBBBBB Description: Swap B and W at positions 6 and 7 F: 53
WWWWEWWWBWBBBBB Description: Swap B and W at positions 7 and 8 F: 51
WWWWEWWWWBBBBBB Description: Swap B and W at positions 8 and 9 F: 49

Nodes created: 242

Execution time: 39 ms

Used memory in bytes: 665912.0
Used memory in megabytes: 0.6350632
```

The program has been tested for puzzles as large as 100 characters (terminal output in Scripts.txt). The limit of the program is memory usage that is consumed by the tree created from the nodes and the time needed to create them and evaluate their heuristic. Even thought the program could be able to solve larger puzzles, it would be impractical due to the high usage of memory and increased computation times.