CSci 384: Artificial Intelligence

Home Assignment 1 (200 points.)

The sliding-title puzzle consists of five black titles, four white titles, and an empty space in the the configuration shown in the Figure.

The puzzle has four legal moves (i.e. actions) with associated costs:

1. A title may move into an adjacent empty location. – This has a cost of 1.
2. A title can hop over one, two or three other tiles into the empty position.
   – This has a cost equal to the number of tiles jumped over + 1: i.e. a cost of 2, 3 or 4.

The goal is to have all the white tiles to the right of all the black tiles.
The position of a blank is not important.

1. [30] **Problem Formulation**
   Clearly **formulate** the problem in terms of 6 factors below. – NOT a verbal description.
   (a) [5] States: how do you define and ***represent*** a state?
   e.g.) (*a, b, c, d, e, f, g, h, i*) where *a* is …, b is …, etc.
   (b) [5] Initial State
   (c) [5] Goal State(s)
   (d) [10] Actions: **Formulate** each action with the current state and its successor states.
   (e) [5] Action cost and Path cost

   A state is a specific layout of tiles in the puzzle, such as $(W, W, W, W, W, \ , B, B, B, B, B)$, where W is a white tile, B is a black tile, and the space is a blank tile. The initial state, as shown in the image, is all five white tiles on the left, the blank tile, and then the five black tiles. There are a few goal states. They are any state in which the white tiles are all to the right of the black tiles, such as $(B, B, B, B, B, \ , W, W, W, W, W)$. However, swapping the blank with any other position will still result in a goal state. There are 8 possible actions: Move left, jump left 1, jump left 2, jump left 3, move right, jump right 1, jump right 2, and jump right 3. The preconditions for all these actions are the openness of the position being moved into (i.e. not too close to an edge of the puzzle). The action cost is simply how many spaces are moved, i.e. moves are c(1), jump 1 is c(2), jump 2 is c(3), and jump 3 is c(4). Path cost is the sum of the costs of the individual actions taken to reach the goal state.

2. [170] **Solve** the problem to find an optimal solution using an **A* algorithm** with **BEST-FIRST-SEARCH** (fig.3.7 of AIMA-4th)
   (a) [10] Define your admissible and consistent heuristic function, h_1.
   The heuristic that I will use is the number of misplaced tiles (i.e. white tile is left of black tile)

(b) [*Optional,* 10] Prove that your heuristic in (a) is both admissible and consistent.

It is admissible because it will never overestimate, as each misplaced tile must be moved at least once to reach the goal, which takes a minimum cost of 1. Additionally, it's consistent because moving a tile should always reduce the number of misplaced tiles by at most 1 with at least a cost of 1.

(c) [100] Implement the problem to solve it. Use any programming language of your preference.

See Python file `csci384-hw1-astar`.

(d) [25] Your outputs must print the following result:

(d1) [7] the ***optimal*** solution? Represent your optimal solution as a *sequence of states*.

(d2) [7] the ***optimal cost*** of the solution (i.e. $f(Goal)$)?

(d3) [6] (a) the number of the nodes that expanded successors and (b) the number of the nodes remaining in the ***frontier*** list when a goal is reached, respectively.

(d4) [5] the total number of the nodes that were ever added in the frontier, regardless they're removed from or still remain in the frontier after the goal state is returned.

See Python file `csci384-hw1-astar`.

```
=== Solution Found ===
Optimal solution sequence:
    WWWWW_BBBBB (Cost: 0)
    WWWW_WBBBBB (Cost: 1)
    WWWWBW_BBBB (Cost: 3)
    WWW_BWWBBBB (Cost: 6)
    WW_WBWWBBBB (Cost: 7)
    WWBW_WWBBBB (Cost: 9)
    WWBWBWW_BBB (Cost: 12)
    WWBWBWWBB_B (Cost: 14)
    WWBWBW_BBWB (Cost: 17)
    WWBWB_WBBWB (Cost: 18)
    WWBWBBW_BWB (Cost: 20)
    WWBWBB_WBWB (Cost: 21)
    WWB_BBWWBWB (Cost: 24)
    W_BWBBWWBWB (Cost: 26)
    WB_WBBWWBWB (Cost: 27)
    WBBWB_WWBWB (Cost: 30)
    WBBWBBWW_WB (Cost: 33)
    WBBWBBW_WWB (Cost: 34)
    WBBWBBWBWW_ (Cost: 37)
    WBBWBBWBW_W (Cost: 38)
    WBBWBB_BWWW (Cost: 41)
    WBB_BBWBWWW (Cost: 44)
    _BBWBBWBWWW (Cost: 47)
    B_BWBBWBWWW (Cost: 48)
    BBBW_BWBWWW (Cost: 51)
    BBBWBBW_WWW (Cost: 54)
    BBBWBB_WWWW (Cost: 55)
    BBB_BBWWWWW (Cost: 58)
Optimal cost: 58
Expanded nodes: 2762
Frontier size at goal: 95
Total nodes added to frontier: 13555
```

(e) [25] Solve the same problem with the 2nd heuristic function h_2 by A* with BEST-FIRST-SEARCH (and print the following result).

(e1) [10] Define your 2nd heuristic function, h_2, which is admissible and consistent.

(e2) [5] What is your ***optimal solution***? Give it as a sequence of states.

(e3) [5] What is the ***optimal cost*** of the solution?

(e4= d4) [5] T he total number of the nodes that were ever added in the frontier, regardless they're removed from or still remain in the frontier after the goal state is returned.

(f) [10] Compare the performance of both heuristics h_1 and h_2, in terms of the total number of the expanded nodes in the frontier at (d4) & (e4).

See Python file `csci384-hw1-astar`. The second heuristic would be the distance between a tile in the wrong place & its intended position. The performance between the two, in terms of expanded nodes in the frontier, is non-existent. This could definitely be due to an error in my implementation, but I haven't been able to determine what is causing this. Another explanation would be that both heuristics are equally negligible in terms of affecting the number of nodes that must be expanded, i.e. the heuristics are weak.

```
Running A* with h2 (Manhattan distance)...

=== Solution Found ===
Optimal solution sequence:
    WWWWW_BBBBB (Cost: 0)
    WWWWWB_BBBB (Cost: 1)
    WWWWWBB_BBB (Cost: 2)
    WWWW_BBWBBB (Cost: 5)
    WW_WWBBWBBB (Cost: 7)
    WWBWW_BWBBB (Cost: 10)
    WWBWWB_WBBB (Cost: 11)
    WWBW_BWWBBB (Cost: 13)
    W_BWWBWWBBB (Cost: 16)
    WB_WWBWWBBB (Cost: 17)
    WBBWW_WWBBB (Cost: 20)
    WBBWWBWW_BB (Cost: 23)
    WBBWWBWWBB_ (Cost: 25)
    WBBWWBW_BBW (Cost: 28)
    WBBWWBWB_BW (Cost: 29)
    WBBWWBWBB_W (Cost: 30)
    WBBWWB_BBWW (Cost: 33)
    WBBW_BWBBWW (Cost: 35)
    WBB_WBWBBWW (Cost: 36)
    _BBWWBWBBWW (Cost: 39)
    BB_WWBWBBWW (Cost: 41)
    BBBWW_WBBWW (Cost: 44)
    BBBWWBWB_WW (Cost: 47)
    BBBWWB_BWWW (Cost: 49)
    BBBWWBB_WWW (Cost: 50)
    BBBW_BBWWWW (Cost: 53)
    BBBWBB_WWWW (Cost: 55)
    BBB_BBWWWWW (Cost: 58)
Optimal cost: 58
Expanded nodes: 2762
Frontier size at goal: 23
Total nodes added to frontier: 13555
```