AI ASSIGNMENT-1

Mushrif Shaikh Shahreyar
B170330CS
(https://github.com/mushrifshahreyar/AI-Assgn)

---

1. **Uninformed search: Breadth First Search**

BW WBWB => Cost to reach goal state = 6
WB WBWB => Cost to reach goal state = 5
BBB WWW => Cost to reach goal state = 14
BBBWWW_ => Cost to reach goal state = 14
WWW BBB => Already in goal state, hence cost = 0
BWBWB W => Cost to reach goal state = 10
WBB WWB => Cost to reach goal state = 6

2. **Informed Search: A***

Heuristic Function: Number of white tiles right to each Black tile. For example, if state is BBW_WWB
For first B: There are 3 white tiles,
For 2nd B: There are 3 white tiles
For 3rd B : There are 0 white tile
h(n) = 3 + 3 + 0 = 6

Proof of Consistency:

There are 3 actions possible without considering directions:

a. Swap space with adjacent tiles, Cost: 1
b. Swap space with one tile hop, Cost: 1
c. Swap space with two tile hop, Cost: 2

n: current node, n': successor node after performing any of the above action.

$$h(n) \le c(n, a, n') + h(n')$$

For nth state, is action is (a), then
After action(a) will be: $h(n) \le 1 + h(n'); c(n, a, n') = 1$

For action (a) state space won't change. Ex: BBWW_WB => BBW_WWB or BBWWW_B
For $h(n') = h(n)$
Hence $h(n) \le 1 + h(n')$ holds the condition for action (a)

After action(b) will be: $h(n) \le 1 + h(n');\ c(n, a, n') = 1$
For action (b) 2 cases may occur for state space:

i) Black may jump over white tile

$h(n') = h(n) - 1$ $(if\ jumped\ right)$ $h(n') = h(n) + 1$ $(if\ jumped\ left)$

=> $h(n) \leq 1 + h(n) - 1$ => $h(n) \leq h(n)$

Or

=> $h(n) \leq 1 + h(n) + 1$ => $h(n) \leq 2 + h(n)$

Condition hold **true** is either cases

ii) While tile may jump over black tile

$h(n') = h(n) + 1$ $(if\ jumped\ right)$ $h(n') = h(n) - 1$ $(if\ jumped\ left)$

Similar to case above the condition will remain **true**

iii) the state will remain same if black jumped over black, or white jumped over white,

=> $h(n') = h(n)$ $same\ for\ both\ left\ jump\ or\ right\ jump$

So $h(n) \leq 1 + h(n')$

Hence from above three cases condition holds **true** for action (b)

After action(c) will be: $h(n) \leq 2 + h(n')$; $c(n, a, n') = 2$

For action (c) these cases may occur for state space:

i) Black may jump over two white tile or 1white and 1black tile or 2 black tile

$h(n') = h(n) \pm 2$ $(2white)\ or$ $h(n') = h(n) \pm 1$ $(1B\ and\ 1W)$, $h(n') = h(n)$

=> $h(n) \leq 2 + h(n) \pm 2$ => $h(n) \leq h(n)\ or\ h(n) \leq h(n) + 4$

Or $h(n') = h(n) \pm 1$: As shown above

=> $h(n) \leq 1 + h(n) \pm 1$ => $h(n) \leq 2 + h(n)\ or\ h(n) \leq h(n)$,

Or $h(n') = h(n)$ also holds as shown in action(a)

Condition hold **true** is either cases

ii) While tile may jump **2 B** or **1B 1W** or **2 W**.

$h(n') = h(n) \pm 2$ $(2B)\ or$ $h(n') = h(n) \pm 1$ $(1B, 1W)\ or$ $h(n') = h(n)$

=> $h(n) \leq 2 + h(n) \pm 2$ => $h(n) \leq h(n)\ or\ h(n) \leq h(n) + 4$

Or $h(n') = h(n) \pm 1$:

=> $h(n) \leq 1 + h(n) \pm 1$ => $h(n) \leq 2 + h(n)\ or\ h(n) \leq h(n)$

Or $h(n') = h(n)$ also holds as shown above.

Condition holds true for all the cases shown above

Hence from above three cases condition holds **true** for action.

Hence h(n) is Consistent.

<u>Costs</u>

BBW WWB => Cost to reach goal state = 9

BWB WBW => Cost to reach goal state = 10

WWBBW B => Cost to reach goal state = 3

WWW BBB => Already in goal state, hence cost = 0

BBB WWW => Cost to reach goal state = 14

BBWW BW => Cost to reach goal state = 10
_BBWWBW => Cost to reach goal state = 10

3. **Comparison**

Input Configuration	Uninformed (BFS)	Informed (A*)
BBWWWB_	10	10
WWBBB W	5	5
BWBWB W	5	5
BWWW BB	6	5
BBB WWW	14	14
_BBBWWW	14	14
WWBBB W	5	5

4.

Input Conf	Uninformed	Informed
WBBBW W	['W', 'B', 'B', 'B', 'W', 'E', 'W'] 0 ['W', 'B', 'B', 'B', 'W', 'W', 'E'] 1 ['W', 'B', 'B', 'E', 'W', 'W', 'B'] 3 ['W', 'B', 'B', 'W', 'W', 'E', 'B'] 4 ['W', 'B', 'E', 'W', 'W', 'B', 'B'] 6 ['W', 'B', 'W', 'W', 'E', 'B', 'B'] 7 ['W', 'E', 'W', 'W', 'B', 'B', 'B'] 9	['W', 'B', 'B', 'B', 'W', 'E', 'W'] 0 ['W', 'B', 'B', 'E', 'W', 'B', 'W'] 1 ['W', 'B', 'E', 'B', 'W', 'B', 'W'] 2 ['W', 'B', 'W', 'B', 'E', 'B', 'W'] 3 ['W', 'B', 'W', 'E', 'B', 'B', 'W'] 4 ['W', 'B', 'W', 'W', 'B', 'B', 'E'] 6 ['W', 'B', 'W', 'W', 'E', 'B', 'B'] 7 ['W', 'E', 'W', 'W', 'B', 'B', 'B'] 9
BBBWWW_	['B', 'B', 'B', 'W', 'W', 'W', 'E'] 0 ['B', 'B', 'B', 'W', 'W', 'E', 'W'] 1 ['B', 'B', 'E', 'W', 'W', 'B', 'W'] 3 ['B', 'B', 'W', 'W', 'E', 'B', 'W'] 4 ['B', 'E', 'W', 'W', 'B', 'B', 'W'] 6 ['E', 'B', 'W', 'W', 'B', 'B', 'W'] 7 ['W', 'B', 'W', 'E', 'B', 'B', 'W'] 9 ['W', 'B', 'W', 'W', 'B', 'B', 'E'] 11 ['W', 'B', 'W', 'W', 'E', 'B', 'B'] 12 ['W', 'E', 'W', 'W', 'B', 'B', 'B'] 14	['B', 'B', 'B', 'W', 'W', 'W', 'E'] 0 ['B', 'B', 'B', 'W', 'E', 'W', 'W'] 1 ['B', 'B', 'E', 'W', 'B', 'W', 'W'] 2 ['B', 'E', 'B', 'W', 'B', 'W', 'W'] 3 ['B', 'W', 'B', 'E', 'B', 'W', 'W'] 4 ['B', 'W', 'B', 'W', 'B', 'E', 'W'] 5 ['B', 'W', 'B', 'W', 'B', 'W', 'E'] 6 ['B', 'W', 'B', 'W', 'E', 'W', 'B'] 7 ['B', 'W', 'E', 'W', 'B', 'W', 'B'] 8 ['E', 'W', 'B', 'W', 'B', 'W', 'B'] 9 ['W', 'W', 'B', 'E', 'B', 'W', 'B'] 11 ['W', 'W', 'B', 'W', 'B', 'E', 'B'] 12 ['W', 'W', 'E', 'W', 'B', 'B', 'B'] 14
BBWWW B	['B', 'B', 'W', 'W', 'W', 'E', 'B'] 0 ['B', 'B', 'W', 'W', 'E', 'W', 'B'] 1 ['B', 'E', 'W', 'W', 'B', 'W', 'B'] 3 ['B', 'W', 'W', 'E', 'B', 'W', 'B'] 4 ['E', 'W', 'W', 'B', 'B', 'W', 'B'] 6	['B', 'B', 'W', 'W', 'W', 'E', 'B'] 0 ['B', 'B', 'W', 'W', 'E', 'W', 'B'] 1 ['B', 'E', 'W', 'W', 'B', 'W', 'B'] 3 ['B', 'W', 'W', 'E', 'B', 'W', 'B'] 4 ['E', 'W', 'W', 'B', 'B', 'W', 'B'] 6

	['W', 'W', 'E', 'B', 'B', 'W', 'B'] 7 ['W', 'W', 'W', 'B', 'B', 'E', 'B'] 9	['W', 'W', 'E', 'B', 'B', 'W', 'B'] 7 ['W', 'W', 'W', 'B', 'B', 'E', 'B'] 9
BWWW BB	['B', 'W', 'W', 'W', 'E', 'B', 'B'] 0 ['B', 'W', 'W', 'E', 'W', 'B', 'B'] 1 ['E', 'W', 'W', 'B', 'W', 'B', 'B'] 3 ['W', 'E', 'W', 'B', 'W', 'B', 'B'] 4 ['W', 'W', 'W', 'B', 'E', 'B', 'B'] 6	['B', 'W', 'W', 'W', 'E', 'B', 'B'] 0 ['B', 'W', 'W', 'E', 'W', 'B', 'B'] 1 ['E', 'W', 'W', 'B', 'W', 'B', 'B'] 3 ['W', 'W', 'E', 'B', 'W', 'B', 'B'] 4 ['W', 'W', 'W', 'B', 'E', 'B', 'B'] 5

5. From the solutions generated by my program,

   The cost for the goal state with blank space in rightmost, leftmost or in the middle position will be ≥ the cost for all other goal states. It is because the goal state with blank space in the middle or rightmost or leftmost position (_WWWBBB) or (WWW_BBB) or (WWWBBB_) the jump will always lead to some other goal states. So the parent node of these goal states will also be a goal state. Hence it may have to perform an extra action to reach these particular states.

Input configuration	**Goal States**	**Uninformed (BFS)**	**Informed (A*)**
BBB WWW	WWWBBBE WWWBBEB WWWBEBB WWWEBBB WWEWBBB WEWWBBB EWWWBBB	15 14 14 15 14 14 15	15 14 14 15 14 14 15
WBBB WW	WWWBBBE WWWBBEB WWWBEBB WWWEBBB WWEWBBB WEWWBBB EWWWBBB	10 9 10 10 10 9 10	10 9 10 10 10 9 10
BWWW BB	WWWBBBE WWWBBEB WWWBEBB WWWEBBB WWEWBBB WEWWBBB EWWWBBB	6 6 5 6 6 7 9	6 6 5 6 6 7 9