***Assignment 1***

**Using Informed and Uninformed Search Algorithms to Solve 8-Puzzle**

|  |  |  |
| --- | --- | --- |
| **Department** | **I.D** | **Names** |
| **Artificial intelligence**  **(A.i)** | **20221441784** | **Abdelrhamn Ahmed fathy** |
| **Artificial intelligence**  **(A.i)** | **20221372883** | **Ahmed Mostafa Abdelrahman** |
| **Artificial intelligence**  **(A.i)** | **20221372110** | **Mazen Gaber Mahmoud** |

**A picture containing furniture, cabinet, chest of drawers

Description automatically generatedA picture containing box, sign, building material, brick

Description automatically generated**

**A picture containing box

Description automatically generated**

**Description**

🡺🡺🡺An instance of the 8-puzzle game consists of a board holding 8 distinct movable tiles, plus an empty space. For any such board, the empty space may be legally swapped with any tile horizontally or vertically adjacent to it. In this assignment, the blank space is going to be represented with the number 0.

🡺🡺🡺Given an initial state of the board well be the input of the user , the search problem is to find a sequence of moves that transitions this state to the goal state, that is, the configuration with all tiles arranged in ascending order 0,1,2,3,4,5,6,7,8 .

🡺🡺🡺The search space is the set of all possible states reachable from the initial state. The blank space may be swapped with a component in one of the four directions `"up", "Dowen", "left", "Right", one move at a time.

🡺🡺🡺The cost of moving from one configuration of the board to another is the same and equal to one. Thus, the total cost of path is equal to the number of moves made from the initial state to the goal state.

🡺🡺🡺 It’s required to implement the game solution using different algorithms, **BFS**, **DFS**, **A\*** (Manhattan Distance & Euclidean Distance).

**Note: The output well be in a traceable format as every step will be printed.**

**Algorithms**

**DFS:**

**1)**The algorithm idea is to deeply explore the nodes within the tree until we reach the goal.

**2)**Same as BFS the main defect is the long running time

because DFS is an uninformed search so we have no information about the path that may help us reach the goal.

**3)**DFS is implemented using Stack data structure

**4)** start with pushing initial state in Stack.

**5)** check if Stack is not empty.

**6)** pop state from Stack.

**7)** add this state to set.

**8)** check if state equal goal state.

**9)** if not equal expand this state and get their children.

**10)** check for each child if not exist in Stack and Set,push this child in Stack.

**Table

Description automatically generatedTable

Description automatically generated**

**Initial state**

**Goal state**

**BFS:**

1)The algorithm idea is to horizontally explore the nodes within the tree until we reach the goal.

2) In general its search depth value is fine compared to other algorithms as DFS.

3)The main defect is the long running time because BFS is an uninformed search so we have no information about the path that may help us reach the goal.

4)BFS is implemented using Queue data structure.

5) start with enqueue initial state in Queue.

6) check if Queue is not empty.

7) dequeue state from Queue

8) add this state to set.

9) check if state equal goal state.

10) if not equal expand this state and get their children.

11) check for each child if not exist in Queue and Set,

enqueue this child in Queue.

Table

Description automatically generated with low confidenceTable

Description automatically generated

**Initial state**

**Goal state**

**A\*(star):**

1)The algorithm idea is to work on finding the goal with putting into consideration the Heuristics that would help reaching the goal in shortest path.

2)A\* is an informed search method, so we have some knowledge (Heuristics) that lead us to reach the goal in the shortest and fastest way so its running time is a lot better than any uninformed search method.

3)A\* is implemented using priority queue data structure.

4)For the Heuristics we compute it using any of the following 2 methods:

o Manhattan Distance. ,

o Euclidean Distance.

1) calculate cost of initial state.

2) insert initial state with its cost in Heap.

3) check if Heap is not empty.

4) get state with min cost from Heap

5) add this state to set.

6) check if state equal goal state.

7) if not equal expand this state and get their children.

8) check for each child if not exist in Set, calculate cost of this child.

90 check if child is not exist in Heap, insert it in the Heap.

10) if the child exist in the Heap, decrease its cost as possible

o Manhattan Distance :

**Text

Description automatically generated**

**Table

Description automatically generated**

**Goal state**

**Initial state**

o Euclidean Distance:

Text

Description automatically generated with medium confidence***Table

Description automatically generated***

**Goal state**

**Initial state**

***Initial state {1,2,5,3,4,0,6,7,8)***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ***Nodes expanded*** | ***maxcost*** | ***Cost of path*** | ***Search depth*** | ***Running time*** | ***Methods*** |
| ***261*** | ***1424*** | ***3*** | ***320*** | ***327*** | ***DFS*** |
| ***12*** | ***8*** | ***1*** | ***4*** | ***11*** | ***BFS*** |
| ***6*** | ***8*** | ***1*** | ***4*** | ***5*** | ***A\*(*Euclidean)** |
| ***4*** | ***8*** | ***1*** | ***4*** | ***4*** | ***A\*(Manhattan***) |

***A picture containing text

Description automatically generated***