08/10/2024

8-Puzzle Problem.

manhattan

Algorithm. (~~DFS~~)

Puzzle is represented as 3×3 grid.

Goal State will be

1	2	3
4	5	6
7	8	0

→ Start with initial State
→ Create Compare the current Position of the tile to the goal State
→ make moves according to match the Position in the goal State
→ Finally add all the values of each tile
→ add the Final values
→ create a Priority Queue and add minimum the 'M' values which we got by Solving using manhattan technique to Queue and make it current State and keep repeating the Process until Puzzle completes
→ add the current State to closed list to mark it as visited.
→

2	3	4
5	-	8
1	7	6

2+0=2

DFS Search :-

→ ~~first~~
→ use 3 x 3 matrix
→ goal state [[1,2,3],[4,5,6],[7,8,-]]
→ If current state is goal state.
Return the
→ Create the list of possible moves
(up, down, left, right)

{ left = f(i, j-1)
right = f(i, j+1)
up = f(i-1, j)
down = f(i+1, j)
}
}

i = column
j = row

→ Push initial state to the stack.
→ if the state is already visited, continue
→ visited_set = add(current_state)
If (current state == goal-state)
return moves.
if (not in visited set)
print moves

1	4	5
6	2	7
8	3	.

visited node.

→ for each move check for visited
if visited, POP the stack until
you reach the current state &
make a different move
Else, push it into stack & move it as
visited.

make change

	.	7

→ If the current state is Equal to
final state then exit.

String to store in set)

```
neighbours = sorted (get_neighbours (initial_state),
    key = lambda x:
    manhattan_distance (x, goal_state))
Path = d.f.s - with manhattan (neighbour,
    Goal_state, visited)
Print ("Enter the initial state of
    8-Puzzle (row-wise, space separated
    for i in range (3)
```

```
Solution = dfs_with_manhattan
    (initial_state, Goal_state).
if Solution:
    Print ("Solution found")
    for state in solution:
        for row in state:
            Print (row)
        Print()
    else:
    Print ("No Solution found").
```

OP: Enter the initial state

0	1	2
5	6	3
4	7	8

Solution found:

0	1	2
5	6	3
4	7	8

$$\begin{bmatrix} 1 & 0 & 2 \\ 5 & 6 & 3 \\ 4 & 7 & 8 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 2 & 3 \\ 5 & 6 & 0 \\ 4 & 7 & 8 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 2 & 3 \\ 5 & 0 & 6 \\ 4 & 7 & 8 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 2 & 3 \\ 0 & 5 & 6 \\ 4 & 7 & 8 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 0 & 7 & 8 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 0 & 8 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 0 \end{bmatrix}$$