

# 8 Puzzle Problem

DFS method -

~~function~~

~~function~~

puzzle board = [ [1], [1], [1] ]

for i in range :

for j in range [3]:

puzzle board [j][i] = input()

goal = [0, 1, 2, 3, 4, 5, 6, 7, 8]

moves = { left: -1, right: 1, up: -3, down: 3 }

lis = []

dfs ( board, final\_move ):

if ( board == goal ):

return

else:

if ( board.move() not in lis ):

board.move()

lis.append ( board )

dfs ( board )

(loop, elegant) nalla baram

: 0 = false

(i) = 0

: 0 = ! (i) elegant

((i) elegant) nalla baram

(i) = 0

first return

: ( start, finish, loop, elegant ) nalla baram

: loop == elegant

start return

((elegant) elegant) nalla baram

(0) nalla baram



# Manhattan distance

Step -1 Initialise board

current-board = [2D list]

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

Step -2

Calculate dist b/w current and target position using sum of distances -  
dist = abs(current-row - goal-row) + (current-col - goal-col)

Step -3

Sum all distances and return distance.

Code-

```
def manhattan (puzzle, goal):  
    dist = 0:
```

```
    for i in range(9):
```

```
        if puzzle[i] != 0:
```

```
            goal_idx = goal.index (puzzle[i])
```

```
            dist += abs (i // 3 - goal_idx // 3) + abs (i % 3 - goal_idx % 3)
```

```
    return dist
```

```
def dfs - manhattan (puzzle, goal, visited, path):
```

```
    if puzzle == goal:
```

```
        return path
```

```
    visited.add (tuple (puzzle))
```

```
    idx = puzzle.index (0)
```



$$\text{moves} = [(1, 3), (-1, 3), (3, 1), (-3, 1)]$$
$$\text{rent} - \text{slates} = [ ]$$

for move, cond in moves:

$$\text{new\_idx} = \text{idx} + \text{move}$$

if  $0 \leq \text{new\_idre} < 9$  and  $(\text{new\_idre} \parallel 3 == \text{idre} \parallel 3 \text{ or } \text{new\_idre} \% 3 == \text{idre} \% 3)$ :

new\_puzzle = puzzle [:]

new - puzzle [idx], new - puzzle [new - idx] = new - puzzle [new - idx].  
if tuple (new - puzzle) not in visited

if tuple (new-huzzle) not in visited  $\times$   $\text{new} = \text{huzzle}[\text{new\_idx}]$   
 next\_state.append((new-huzzle, manhattan(new-huzzle, goal)))

for states, - in verb-states

res = dfs\_maharhan (state, goal, visited, path + (state))

if res:

Return Res

Return None

```
def prettyfy(res): // To display output
```

10

 ~~$[ ] = \text{instant} - \text{jump}$~~ 

17 = 1000000

Output -

1	0	2
3	7	5
4	6	8

0	1	2
3	7	5
4	6	8

$(111 \text{ Feb. } 2, 111 \text{ Feb. } 3) \quad 1 \quad 2 \quad 3$   
 $0 \text{ (4 Feb. } < 111 \text{ Feb. } 0 \quad 4 \text{ } ^S$   
 $6 \quad 7 \quad 8 \quad 6 \quad 7 \quad 8 \text{ } ^S$

0	1	2
3	4	5
6	7	8

Good State

$\therefore (\text{least} = \text{root}) \text{ file}$

leaf neuter

: [ ] nicht - 1/2 x 10^3 : 10^3

(i) belian bro (i) ~~nam~~ <sup>3710</sup> jho di

$$) = (i) \text{ beliebig}$$

$(1 + \text{Alpb} \text{ rel. Alpb. i}) \cdot \text{Alb}$  wieder