7/12/22

8 puzzle - Manhattan distance.

Aim: To implement 8 puzzle problem using manhattan distance.

Algorithm:

$h_1(n)$ = number of misplaced tile

$h_2(n)$ = total manhattan distance.

(i.e. no of squares from desired location of each tile)

7	2	4
5		6
8	3	1

start state

	1	2
3	4	5
6	7	8

goal state

$h_1(s) = 8$

$h_2(s) = 1(3) + 2(1) + 3(2) + 4(1) + 5(2) + 6(3) + 7(1) + 8(2)$

$= 3 + 1 + 2 + 2 + 2 + 3 + 3 + 2$

$h_2(s) = 18.$

$h_2(n) \geq h_1(n)$ for all n.

code:

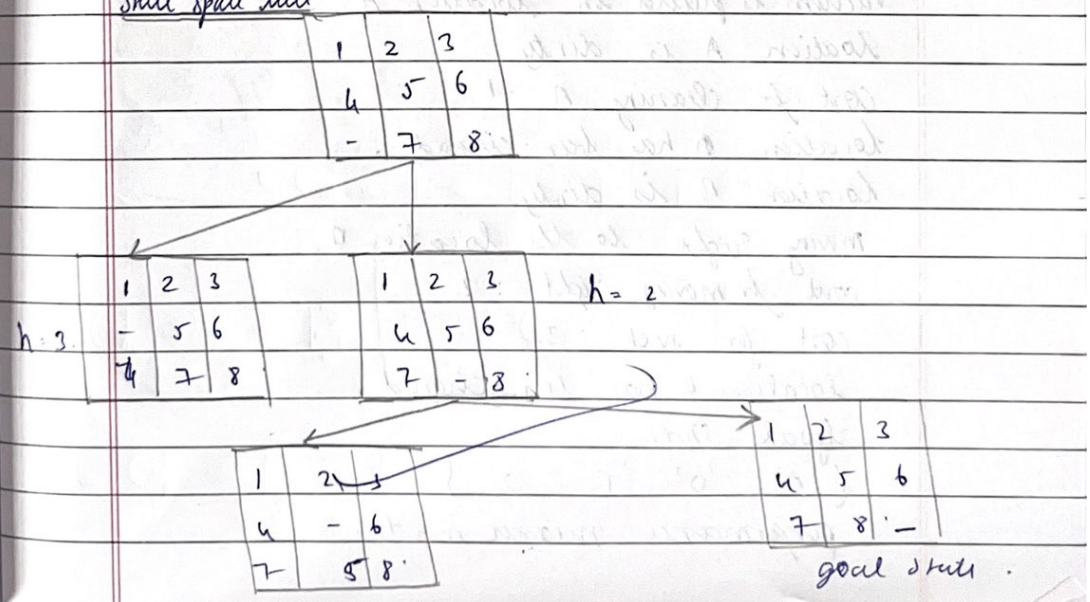
```
imp
import numpy as np
from queue import PriorityQueue
goal_state = np.array([[1,2,3],[4,5,6],[7,8,0]])

start_state = np.array([[7,1,4],[5,0,6],[8,3,1]])

def manhattan_distance(puzzle):
    distance = 0
    for i in range(3):
        for j in range(3):
            x,y = np.where(goal_state == puzzle[i][j])
            distance += abs(i-x) + abs(j-y)
    return distance

print("Puzzle:")
print(start_state)
print("goal state")
print(goal_state)
print("Manhattan distance:", manhattan_distance(start_state))
```

State space tree:



Manhatan distance.

O/P:

Start state:

1	2	3
4	5	6
-	7	8

goal state:

1	2	3
4	5	6
7	8	-

Manhattan distance : 16.