24/11/23 Lab-4 [illegible] 26-11-23

8-puzzle Problem

Algo

we will use ~~branch and bound~~ this technique in branch and bound

① 3 types of ~~no~~ nodes involved in branch and bound

a. Live node is a node whose children node are currently being explored

b. E-node is a node currently being expanded

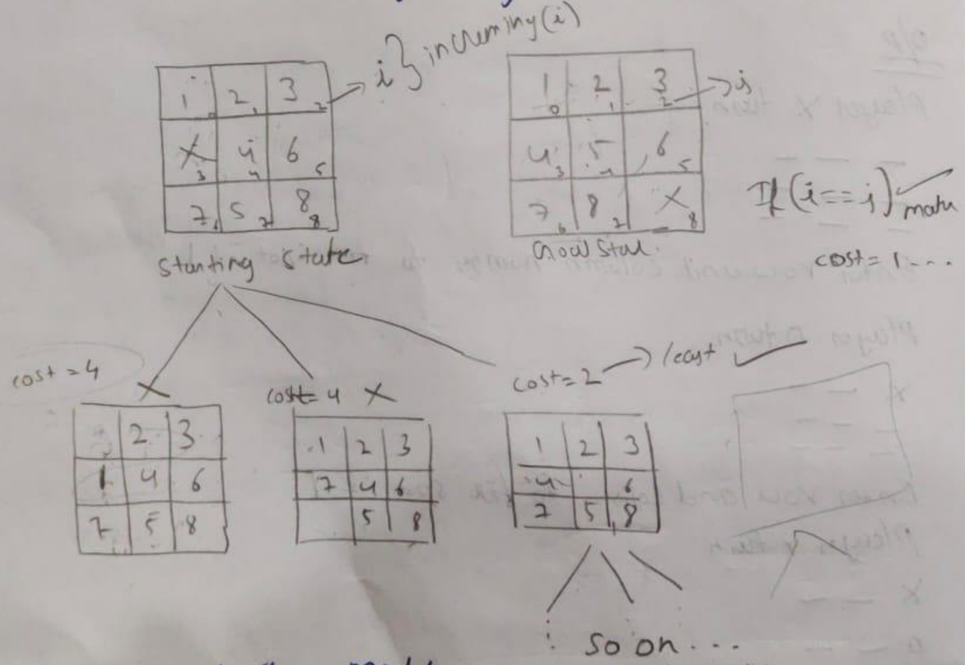
c. Dead node is a node that is not to be expanded

② cost function ; use to determine the next E-node ↳ have least cost

c(x) + g(x) + h(x)

where g(x) = cost of reaching the current node from the root.

h(x) = cost of reaching an answer node from x



③ once we get the match for All possible starting state to Goal state, we stop the function and Agent will solve the 8-puzzle problem

#code

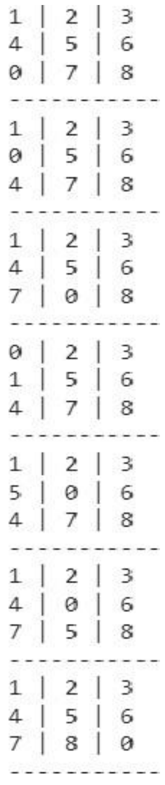
```
from collections import deque
def find_blank(board):
    for i in range (3):
        for j in range (3):
            if board [i][j] = = 0;
            return i,j

def generate_moves (board):
    moves = []
    blank_row, blank_col = find_blank(board)
    possible_moves = [
        (1,0), (-1,0), (0,1), (0,-1)
    ]
  for dr,dc in possible_moves:
    new_row, new_col = blank_row + dr, blank_col + dc
    if 0<=new_row < 3 and 0 <= new_col<3:
        new_board = [row[:] for row in board]
      new_board [blank row] [blank_col], new_board [new_
      row] [new_col] = new_board [new_row][new_col],
      new_board [blank _row] [blank_col]
      moves. append (new_board)
return moves
def print_steps (soln_path):
    if solution_path:
      print ("steps to reach the goal:")
        for step in solution_path:
          print ("------")
          for row in step:
              print ("|", end ="")
              for val in row:
                  if val == 0;
                      print (" ", end = "|")
```

```
                    else:
                        print (val, end=" ,")
            print()
        print ("- - - - . . -")
        print ()
else:
    print ("No solution exists.")

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

goal = [
    [0,1,2]
    [3,4,5],
    [6,7,8]
]
Solution_path = solve_puzzle (initial,goal)
print_steps (solution_path)
```

<u>O/P</u>

```
Steps to reach the goal :

1 2 3         2 3      2   3      2 3
  4 5       1 4 5      1 4 5      1 4 5
6 7 8       6 7 8      6 7 8      6 7 8
Initial state.

2 3 5       2 3 5      2   5        2 5
1 4         1   4      1 3 4      1 3 4
6 7 8       6 7 8      6 7 8      6 7 8

1 2 5       1 2 5      1 2        1   2
3   4       3 4        3 4 5      3 4 5
6 7 8       6 7 8      6 7 8      6 7 8

  1 2   Goal State
3 4 5
6 7 8
```

Success


```
1 | 2 | 3
4 | 5 | 6
0 | 7 | 8
-----------
1 | 2 | 3
0 | 5 | 6
4 | 7 | 8
-----------
1 | 2 | 3
4 | 5 | 6
7 | 0 | 8
-----------
0 | 2 | 3
1 | 5 | 6
4 | 7 | 8
-----------
1 | 2 | 3
5 | 0 | 6
4 | 7 | 8
-----------
1 | 2 | 3
4 | 0 | 6
7 | 5 | 8
-----------
1 | 2 | 3
4 | 5 | 6
7 | 8 | 0
-----------
```