Koushik Sahu
118CS0597
Artificial Intelligence Lab-III
15th Sept 2021

Code:

```
import math, statistics, collections, copy

class EightPuzzle:
    def __init__(self, size):
        self.n = size
        self.open = []
        self.closed = []

    def f(self, start, goal):
        return self.h(start.data,goal) + start.level

    def h(self, start, goal):
        temp = 0
        for i in range(0, self.n):
            for j in range(0, self.n):
                if start[i][j] != goal[i][j] and start[i][j] != '*':
                    temp += 1
        return temp

    def accept(self):
        puzzle = []
        for i in range(0, self.n):
            temp = input().split(" ")
            puzzle.append(temp)
        return puzzle

    def run(self):
        print("Enter the start state of 8-puzzle:")
        start = self.accept()
        print("Enter the goal state of 8-puzzle:")
        goal = self.accept()
        start = Node(start, 0, 0)
        start.fscore = self.f(start, goal)
        self.open.append(start)
        print("Solving puzzle...")
        while True:
            cur = self.open[0]
            print("  |  ")
```

```
            print("  V  ")
            for i in cur.data:
                for j in i:
                    print(j,end=" ")
                print("")
            if(self.h(cur.data, goal) == 0):
                break
            for i in cur.get_next_configs():
                i.fscore = self.f(i,goal)
                self.open.append(i)
            self.closed.append(cur)
            del self.open[0]

            self.open.sort(key = lambda x: x.fscore)

class Node:
    def __init__(self, data, level, fscore):
        self.data = data
        self.level = level
        self.fscore = fscore

    def find_blank(self, puzzle, x):
        for i in range(0, len(self.data)):
            for j in range(0, len(self.data)):
                if puzzle[i][j] == x:
                    return i,j

    def move(self, puzzle, x1, y1, x2, y2):
        if x2 >= 0 and x2 < len(self.data) and y2 >= 0 and y2 < len(self.data):
            temp_puzzle = []
            temp_puzzle = copy.deepcopy(puzzle)
            temp = temp_puzzle[x2][y2]
            temp_puzzle[x2][y2] = temp_puzzle[x1][y1]
            temp_puzzle[x1][y1] = temp
            return temp_puzzle
        else:
            return None

    def get_next_configs(self):
        x, y = self.find_blank(self.data, '*')
        val_list = [[x,y-1], [x,y+1], [x-1,y], [x+1,y]]
        children = []
        for i in val_list:
            child = self.move(self.data, x, y, i[0], i[1])
```

```
            if child is not None:
                child_node = Node(child, self.level+1, 0)
                children.append(child_node)
        return children

if __name__ == '__main__':
    ep = EightPuzzle(3)
    ep.run()
```

Output: