

```

import copy
class Node:
    def __init__(self, data, level, fval):
        self.data = data
        self.level = level
        self.fval = fval
    def generate_child(self):
        x, y = self.find(self.data, '_')
        val_list = [[x, y - 1], [x, y + 1], [x - 1, y], [x + 1, y]]
        children = []
        for i in val_list:
            child = self.shuffle(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
    def shuffle(self, data, x1, y1, x2, y2):
        if x2 >= 0 and x2 < len(data) and y2 >= 0 and y2 < len(data):
            temp_data = copy.deepcopy(data)
            temp = temp_data[x2][y2]
            temp_data[x2][y2] = temp_data[x1][y1]
            temp_data[x1][y1] = temp
            return temp_data
        else:
            return None
    def find(self, data, x):
        for i in range(len(self.data)):
            for j in range(len(self.data)):
                if data[i][j] == x:
                    return i, j
class Puzzle:
    def __init__(self, size):
        self.size = size
        self.open = []
        self.closed = []

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

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

    def process(self):
        print("enter the start state matrix \n")
        start = self.accept()
        print("enter the goal state matrix \n")
        goal = self.accept()
        start = Node(start, 0, 0)
        start.fval = self.f(start, goal)
        self.open.append(start)
        print("\n\n")
        while True:

```

```

        cur = self.open[0]
        print("\n")
        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.generate_child():
            i.fval = self.f(i, goal)
            self.open.append(i)
        self.closed.append(cur)
        del self.open[0]
        self.open.sort(key=lambda x: x.fval, reverse=False)
puz = Puzzle(3)
puz.process()

```

```

/*
output :-
C:\Users\anike\Scripts\python.exe C:/Users/anike/PycharmProjects/pushup/codes.py

```

enter the start state matrix

```

1 2 3
4 5 _
7 8 6

```

enter the goal state matrix

```

1 2 3
4 5 6
7 8 _

```

```

1 2 3
4 5 _
7 8 6

```

```

1 2 3
4 5 6
7 8 _

```

Process finished with exit code 0 \*/