Experiment 6

Objective

To be able solve 8-puzzle problem using AI.

Problem Statement

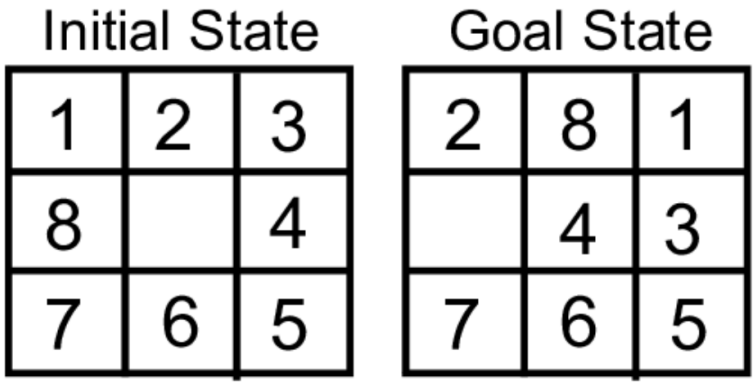
To apply A* search algorithm for finding out the solution to the 8-puzzle problem.

Lab Exercise

Write the program for finding out the path from a source node to a goal node in an 8-puzzle. Ask the user to provide the details of the source node.

Program Code

Example:




```
class Node:

    def __init__(self,state,level,fval):
        self.state = state
        self.level=level
        self.fval=fval

    def findpos(self,item):
        for i in range(3):
            for j in range(3):
                if item[i][j] == 0:
                    return i,j

    def swap(self,item,i,j):
        m,n = self.findpos(item)
        temp = item[m][n]
        item[m][n] = item[i][j]
        item[i][j] = temp
        return item

    def copy(self,item):
        temp = []
        for i in item:
            t = []
            for j in i:
                t.append(j)
            temp.append(t)
        return temp

    def generatechild(self):

        m,n = self.findpos(self.state)
```

```
33
34          possible_list = []
35          if(m>=1):#for up move
36              possible_list.append([m-1,n])
37          if(m<=1):#for down move
38              possible_list.append([m+1,n])
39          if(n>=1):#for left move
40              possible_list.append([m,n-1])
41          if(n<=1):#for right move
42              possible_list.append([m,n+1])
43
44          child=[]
45          for i in possible_list:
46              tempo = self.copy(self.state)
47              tmp = self.swap(tempo,i[0],i[1])
48              chi = Node(tmp,self.level+1,0)
49              child.append(chi)
50
51          return child
52
53
54  class enviroment:
55
56      def heuristic(self,item,goal):
57          temp = 0
58          for i in range(3):
59              for j in range(3):
60                  if item[i][j] != goal[i][j] and item[i][j]!=0:
61                      temp+=1
62          return temp
63
64      def function(self,item,goal):
65          return self.heuristic(item.state,goal)+item.level
66
67      def solv(self,item):
68          invcount = 0
69          sample=[]
70          for i in range(3):
71              for j in range(3):
72                  sample.append(item[i][j])
73
74          for i in range(0,9):
75              for j in range(j,i):
76                  if sample[j]!=0 and sample[i]!=0 and sample[i]>sample[j]:
77                      invcount+=1
78
79          return invcount
80
81      def astar(self):
82          print(" - - - Intial State - - - ")
83          print('\n')
84          sta = [[2,5,3],[1,0,6],[4,7,8]]
85          goal = [[1,2,3],[4,5,6],[7,8,0]]
86          open=[]
87          closed=[]
88          visited=[]
89
90          start = Node(sta,0,0)
91          start.fval = self.function(start,goal)
92          open.append(start)
93          visited.append(start)
94
95          for i in start.state:
96              for j in i:
97                  print(j,end=" ")
98              print("")
99
100         # count = self.solv(start.state)+1
101         # if (count % 2 == 0):
102         #     solv = True
103         # else:
104         #     solv =  False
105         #     print('---Unsolvalbe Problem---')
106
107         count = 1
108         while True:
```

```
            cur = open[0]
            print("")
            print(" \\\/ \n")
            count+=1
            for i in cur.state:
                for j in i:
                    print(j,end=" ")
                print("")

            if(self.heuristic(cur.state,goal)==0):
                break

            for i in cur.generatechild():
                if i not in visited:
                    visited.append(i)
                    i.fval=self.function(i,goal)
                    open.append(i)
            closed.append(cur)

            del open[0]
            open.sort(key = lambda x:x.fval,reverse=False)

        print('\n')
        print(" - - - Final State - - - ")

        print('-----------------------------------------------------------------')
        print('-----------------------------------------------------------------')
        print('\n')
        print(" - - - - -  Details - - - - - -")
        print('\n')

        print('=== Total Visited States ===  :',len(visited))
        print('\n')
        print('=== Required steps to solved problem ===  :',(count-1))
        print('\n')



# goal = [[1,2,3],[4,5,6],[7,8,0]]
# p = Node(goal,7,3)
# n = p.generatechild()
# print(n)
# # print(p.state)
# q = enviroment()
# print(q.function(p,p))

print('-------------------------------------------------------------------------')
print('-------------------------------------------------------------------------')
print(' ___     _____ _____ _____ _____ __    _____    _____ _____ _____ _____ __    _____
    _____ ')
print('| . |   |  _  |  |  |__   |__   |  |  |   __|  |  _  | __  |     | __  |  |  |   __|
     |')
print('| . |  |   __|  |  |   __|   __|  |__|   __|  |   __|    -|  |  | __ -|  |__|   __| | |
     |')
print('|___|  |__|  |_____|_____|_____|_____|  |__|  |__|__|_____|_____|_____|_____|_|_|
    _|')
print('
                                ')
print('-------------------------------------------------------------------------')
print('-------------------------------------------------------------------------')
print('\n')

# main code
p = enviroment()
p.astar()
```

Output

```
PS D:\SEM-5\AI> & C:/Users/ASUS/AppData/Local/Programs/Python/Python39/python.exe "d:/SEM-5/AI
    /8 puzzle advanced.py"
--------------------------------------------------------------------------------------------------

--------------------------------------------------------------------------------------------------

 ___     _____ _____ _____ _____ __    _____    _____ _____ _____ _____ __    _____ _____
| . |   |  _  |  |  |__   |__   |  |  |   __|  |  _  | __  |     | __  |  |  |   __|     |
| . |   |   __|  |  |   __|   __|  |__|   __|  |   __|    -|  |  | __ -|  |__|   __| | | |
|___|   |__|  |_____|_____|_____|_____|  |__|  |__|__|_____|_____|_____|_____|_|_|_|_|

--------------------------------------------------------------------------------------------------

--------------------------------------------------------------------------------------------------



 - - - Intial State - - -


2 5 3
1 0 6
4 7 8

 \\/

2 5 3
1 0 6
4 7 8

 \\/

2 0 3
1 5 6
4 7 8

 \\/

0 2 3
1 5 6
4 7 8

 \\/

1 2 3
0 5 6
4 7 8

 \\/

1 2 3
4 5 6
0 7 8

 \\/

1 2 3
4 5 6
7 0 8

 \\/

1 2 3
4 5 6
7 8 0


 - - - Final State - - -
--------------------------------------------------------------------------------------------------

--------------------------------------------------------------------------------------------------
```

```
 - - - - -  Details - - - - - -


=== Total Visited States ===  : 18


=== Required steps to solved problem ===  : 7


PS D:\SEM-5\AI>
```

Conclusion

The outcomes of an experiment solving the 8 puzzle problem using the A* algorithm would rely on a number of variables, including the heuristics employed, the search space explored, and the implementation specifics. However, in general, it is anticipated that the A* algorithm will work well in solving the 8 puzzle problem when combined with the proper heuristics. A* algorithm is able to find the optimal solution with a relatively low computational cost, particularly when compared to other search algorithms like BFS and DFS, one potential conclusion from an experiment using the algorithm to solve the 8 puzzle problem. However, the effectiveness of the A* algorithm may be influenced by the caliber of the employed heuristics and the scope of the search area. The experiment may also shown the effectiveness of various heuristics used in the A* algorithm, such as Manhattan distance or misplaced tiles, and their effects on the accuracy of the answer and computational cost. Overall, the A* algorithm continues to be one of the best methods for resolving the 8 puzzle issue in AI, and more research can be done to boost its efficacy and efficiency. In this experiment for this example we visited total 18 states and to solve the problem we required 7 steps.