Name : Tejas Redkar
PRN : 1032210937
Panel - C
Roll No : PC-44

AIES Assignment - 1

* Aim : Solve 8 - puzzle problem using A* algorithm

* Objective : To study & implement A* algorithm for 8 - puzzle problem.

* Theory :

- Best first search is a graph search algorithm used in AI. It explores a graph by selecting the most promising node based on a heuristic function. The heuristic estimates the cost or value of reaching the goal from a particular node. Best-first search keeps a priority queue of nodes to expand & selects the one with the lowest heuristic value OR graphs are used in decision problems with multiple possible actions or choices at each decision point.

- The 8 puzzle problem is a classic problem in artificial intelligence. It consists of a 3x3 grid with eight numbered tiles & one empty space. The goal is to rearrange the tiles from a given initial state to a desired goal state using the minimum no of moves.

- Data Structure & other details about the A* algorithm excluding the algorithm are as follows
  - Nodes
  - Queues
  - Heuristic Function
  - Closed set
  - Open set

* Input : Initial state & goal state

* Output : Solution / goal state & with optimal path.

* A* Algorithm

OPEN = nodes on frontier CLOSED = expanded nodes
OPEN = { <s, nil> }
while OPEN is not empty
remove from OPEN the node <n,p> with minimum f(n)
place <n,p> on CLOSED
if n is a goal state
return success (path P)
Four each edge connecting n $ on with cost C.
if <m,q> is a CLOSED on d { p l e } is cheaper than q
→ then remove n from CLOSED

put <m, {P/e}> on OPEN
else if <m, q> is on OPEN & {P/e} is cheaper than q.
→ then replace q with {P/e}
else if m is not on OPEN
→ then put <m, {P/e}> on OPEN
return Failure.

* FAQ's

Q1) What is a heuristic function? What is the advantage of using heuristic function?

Ans A heuristic function, often denoted as "h(n)", is a crucial component in many search & optimization algorithms, including A*. It provides an estimate of the cost or distance from a given state or node in a search space to the goal state.

The key advantages of using a heuristic function are as follows:

- Guidance ~~for~~ search
- Efficiency
- Faster convergence
- Real world applications (route planning, robotics, scheduling).

Q2) Explain A* Algorithm with example.

Ans A* Algorithm is a searching algorithm that searches for the shortest path between the initial & the final state.

Ex

1	2	3
	4	5
7	8	6

1	2	3
4	5	6
7	8	

Consider, $g(x)$ = depth of the node

$h(x)$ = no. of misplaced files.

$$f(x) = g(x) + f(x)$$

A* algorithm proceeds to take the path when $F(x)$ has the least value

Initial State :

1	2	3
	4	5
7	8	6

$g(x)=0$
$h(x)=3$
$f(x)=3$

$g(x)=1$
$h(x)=4$
$f(x)=1$

	2	3
1	4	5
7	8	6

1	2	3
4		5
7	8	6

$g(x)=1$
$h(x)=2$
$f(x)=3$

1	2	3
7	4	5
	8	6

$g(x)=1$
$h(x)=4$
$f(x)=5$

$g(x)=2$
$h(x)=3$
$f(x)=5$

1		3
4	2	5
7	8	6

1	2	3
4	5	
7	8	6

$g(x)=2$
$h(x)=1$
$f(x)=3$

1	2	3
4	5	6
7	8	

$g(x)=3$
$h(x)=0$
$f(x)=3$

final state

classmate

Date :

Page :

Q.3) Explain different heuristic functions that can be used for the 8-puzzle problem.

Ans 1) Hamming Distance

- counts the no. of tiles that are not in their goal positions

2) Manhatten Distance

- Calculates the sum of the manhatten distances of each tile from ~~to~~ its current position to goal

3) Euclidean distance

- Computes the Euclidean distance of each tile.

4) Max Heuristic

- Considers both the Manhattan & misplaced tiles heuristic & chooses maximum of two values.

11/12/23

AIES 1

November 29, 2023

[1]:
```
class Node:
    def __init__(self,data,level,fval):
        """ Initialize the node with the data, level of the node and the␣
↪calculated fvalue """
        self.data = data
        self.level = level
        self.fval = fval

    def generate_child(self):
        """ Generate child nodes from the given node by moving the blank space
            either in the four directions {up,down,left,right} """
        x,y = self.find(self.data,'_')
        """ val_list contains position values for moving the blank space in␣
↪either of
            the 4 directions [up,down,left,right] respectively. """
        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,puz,x1,y1,x2,y2):
        """ Move the blank space in the given direction and if the position␣
↪value are out
            of limits the return None """
        if x2 >= 0 and x2 < len(self.data) and y2 >= 0 and y2 < len(self.data):
            temp_puz = []
            temp_puz = self.copy(puz)
            temp = temp_puz[x2][y2]
            temp_puz[x2][y2] = temp_puz[x1][y1]
            temp_puz[x1][y1] = temp
            return temp_puz
        else:
            return None
```

```python
    def copy(self,root):
        """ Copy function to create a similar matrix of the given node"""
        temp = []
        for i in root:
            t = []
            for j in i:
                t.append(j)
            temp.append(t)
        return temp

    def find(self,puz,x):
        """ Specifically used to find the position of the blank space """
        for i in range(0,len(self.data)):
            for j in range(0,len(self.data)):
                if puz[i][j] == x:
                    return i,j
```

[2]:
```python
class Puzzle:
    def __init__(self,size):
        """ Initialize the puzzle size by the specified size,open and closed␣
↪lists to empty """
        self.n = size
        self.open = []
        self.closed = []

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

    def f(self,start,goal):
        """ Heuristic Function to calculate hueristic value f(x) = h(x) + g(x)␣
↪"""
        return self.h(start.data,goal)+start.level

    def h(self,start,goal):
        """ Calculates the different between the given puzzles """
        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
```

```python
        return temp


    def process(self):
        """ Accept Start and Goal Puzzle state"""
        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)
        """ Put the start node in the open list"""
        self.open.append(start)
        print("\n\n")
        while True:
            cur = self.open[0]
            print("")
            print("  | ")
            print("  | ")
            print(" \\\'/ \n")
            for i in cur.data:
                for j in i:
                    print(j,end=" ")
                print("")
            """ If the difference between current and goal node is 0 we have␣
↪reached the goal node"""
            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]

            """ sort the opne list based on f value """
            self.open.sort(key = lambda x:x.fval,reverse=False)
```

[4]:
```python
puz = Puzzle(3)
puz.process()
```

```
Enter the start state matrix

2 8 3
1 6 4
7 _ 5
Enter the goal state matrix
```

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



  |
  |
 \'/

2 8 3
1 6 4
7 _ 5

  |
  |
 \'/

2 8 3
1 _ 4
7 6 5

  |
  |
 \'/

2 8 3
_ 1 4
7 6 5

  |
  |
 \'/

2 _ 3
1 8 4
7 6 5

  |
  |
 \'/

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

```
  |
  |
 \'/

1 2 3
_ 8 4
7 6 5

  |
  |
 \'/

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