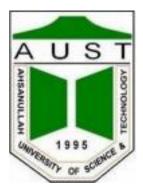
Ahsanullah University of Science and Technology



Department of Computer Science and Engineering

Program: Bachelor of Science in Computer Science and Engineering

Course No: CSE 4108

Course Title: Artificial Intelligence Lab

Assignment No: 03

Date of Submission: 29/01/2022

Submitted to:

Mr. Faisal Muhammad Shah

Associate Professor, Department of CSE, AUST.

Mr. Md. Siam Ansary

Lecturer, Department of CSE, AUST.

Submitted By:

Name: Mahin Opu

Student ID: 17.02.04.006

Question_02: Implement A* search algorithm using Python.

Code:

```
tree = {'S': [['A', 1], ['B', 5], ['C', 8]],
     'A': [['S', 1], ['D', 3], ['E', 7], ['G', 9]],
     'B': [['S', 5], ['G', 4]],
     'C': [['S', 8], ['G', 5]],
     'D': [['A', 3]],
     'E': [['A', 7]]}
heuristic = {'S': 8, 'A': 8, 'B': 4, 'C': 3, 'D': 5000, 'E': 5000, 'G': 0}
cost = \{'S': 0\}
def AStarSearch():
  global tree, heuristic
  closed=[]
  opened=[['S',8]]
  "finding the visited nodes"
  while True:
     fn = [i[1] \text{ for } i \text{ in opened}]
     chosen\_index = fn.index(min(fn))
     node = opened[chosen_index][0]
     closed.append(opened[chosen index])
     del opened[chosen_index]
     if closed[-1][0] == 'G':
        break
     for item in tree[node]:
        if item[0] in [closed_item[0] for closed_item in closed]:
          continue
        cost.update({item[0]: cost[node] + item[1]})
        fn_node = cost[node] + heuristic[item[0]] + item[1]
        temp = [item[0], fn\_node]
        opened.append(temp)
```

```
"finding optimal path"
  trace\_node = 'G'
  optimal sequence = ['G']
  for i in range(len(closed) - 2, -1, -1):
    check_node = closed[i][0]
    if trace_node in [children[0] for children in tree[check_node]]:
       children_costs = [temp[1] for temp in tree[check_node]]
       children_nodes = [temp[0] for temp in tree[check_node]]
       if cost[check_node] + children_costs[children_nodes.index(trace_node)] ==
cost[trace_node]:
         optimal_sequence.append(check_node)
         trace_node = check_node
         optimal_sequence.reverse()
         return closed, optimal_sequence
if __name__ == '__main__':
  visited_nodes, optimal_nodes = AStarSearch()
  print('visited nodes: ' + str(visited_nodes))
  print('optimal nodes sequence: ' + str(optimal_nodes))
```

```
tree = {'S': [['A', 1], ['B', 5], ['C', 8]],
        'A': [['S', 1], ['D', 3], ['E', 7], ['G', 9]],
'B': [['S', 5], ['G', 4]],
'C': [['S', 8], ['G', 5]],
'D': [['A', 3]],
'E': [['A', 7]]}
heuristic = {'S': 8, 'A': 8, 'B': 4, 'C': 3, 'D': 5000, 'E': 5000, 'G': 0}
cost = {'S': 0}
def AStarSearch():
    global tree, heuristic
    closed=[]
    opened=[['S',8]]
    '''finding the visited nodes'''
    while True:
         fn = [i[1] for i in opened]
         chosen index = fn.index(min(fn))
         node = opened[chosen index][0]
         closed.append(opened[chosen_index])
         del opened[chosen index]
         if closed[-1][0] == 'G':
             break
         for item in tree[node]:
             if item[0] in [closed item[0] for closed item in closed]:
                 continue
             cost.update({item[0]: cost[node] + item[1]})
             fn node = cost[node] + heuristic[item[0]] + item[1]
             temp = [item[0], fn node]
             opened.append(temp)
    '''finding optimal path'''
    trace_node = 'G'
    optimal_sequence = ['G']
```

```
II Item[U] in [Closed Item[U] for Closed Item in Closed]:
               continue
            cost.update({item[0]: cost[node] + item[1]})
            fn node = cost[node] + heuristic[item[0]] + item[1]
            temp = [item[0], fn node]
            opened.append(temp)
    '''finding optimal path'''
   trace node = 'G'
   optimal sequence = ['G']
    for i in range(len(closed) - 2, -1, -1):
        check node = closed[i][0]
        if trace node in [children[0] for children in tree[check node]]:
            children_costs = [temp[1] for temp in tree[check_node]]
            children nodes = [temp[0] for temp in tree[check node]]
            if cost[check node] + children costs[children nodes.index(trace node
               optimal sequence.append(check node)
               trace node = check node
               optimal sequence.reverse()
               return closed, optimal sequence
if name == ' main ':
   visited_nodes, optimal_nodes = AStarSearch()
   print('visited nodes: ' + str(visited nodes))
   print('optimal nodes sequence: ' + str(optimal nodes))
```

```
File Edit Shell Debug Options Window Help

Python 3.8.8 (default, Apr 13 2021, 15:08:03) [MSC v.1916 64 bit (AMD64)] on win 32

Type "help", "copyright", "credits" or "license()" for more information.

>>>

RESTART: E:\4.1 Lab\AI Assignments\Assignment 03\170204006_Assignment03_Task02

.py

visited nodes: [['S', 8], ['A', 9], ['B', 9], ['G', 9]]

optimal nodes sequence: ['B', 'G']

>>> |
```