Experiment 5 : Developing Best First Search and A\* Algorithm

Harsh Prasad

RA1811003010967

CSE-H2

Aim: To implement and execute a real-life Developing best first search and A\* Algorithm for real world problems.

Pseudocode:

Best-First-Search(Grah g, Node start)

1) Create an empty PriorityQueue

PriorityQueue pq;

2) Insert "start" in pq.

pq.insert(start)

3) Until PriorityQueue is empty

u = PriorityQueue.DeleteMin

If u is the goal

Exit

Else

Foreach neighbor v of u

If v "Unvisited"

Mark v "Visited"

pq.insert(v)

Mark u "Examined"

End procedure

Code:

from queue import PriorityQueue

v = 14

graph = [[] for i in range(v)]

# Function For Implementing Best First Search

# Gives output path having lowest cost

def best\_first\_search(source, target, n):

visited = [0] \* n

visited = True

pq = PriorityQueue()

pq.put((0, source))

while pq.empty() == False:

u = pq.get()[1]

# Displaying the path having lowest cost

print(u, end=" ")

if u == target:

break

for v, c in graph[u]:

if visited[v] == False:

visited[v] = True

pq.put((c, v))

print()

# Function for adding edges to graph

def addedge(x, y, cost):

graph[x].append((y, cost))

graph[y].append((x, cost))

# The nodes shown in above example(by alphabets) are

# implemented using integers addedge(x,y,cost);

addedge(0, 1, 3)

addedge(0, 2, 6)

addedge(0, 3, 5)

addedge(1, 4, 9)

addedge(1, 5, 8)

addedge(2, 6, 12)

addedge(2, 7, 14)

addedge(3, 8, 7)

addedge(8, 9, 5)

addedge(8, 10, 6)

addedge(9, 11, 1)

addedge(9, 12, 10)

addedge(9, 13, 2)

source = 0

target = 9

best\_first\_search(source, target, v)

Result:

The given implementation of BFS was implemented and solved successfully using Python and the results were verified successfully