**Artificial Intelligence Lab**

**LAB 5 – Developing Best first search and A\* Algorithm for real world**

**problems**

**Aditya Kumar**

**RA1911030010100**

**O2 Section**

**Date – 11/2/22**

**Problem Statement :**

Implementation of Best First Search algorithm for an application.

Algorithm:

1. Create two empty lists
2. Start from the initial node and add it to the ordered open list
3. Next the below steps are repeated until the final node or endpoint is reached
4. If the open list is empty exit the loop and return a False statement which says that the final node cannot be reached
5. Select the top node in the open list and move it to the closed list while keeping track of the parent node
6. If the node removed is the endpoint node return a True statement meaning a path has been found and moving the node to the closed list
7. However if it is not the endpoint node then list down all the neighboring nodes of it and add them to the open list
8. According to the evaluation function re order the nodes.

**Code:**

from queue import PriorityQueue

v = 14

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

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

visited = [0] \* n

visited[0] = True

pq = PriorityQueue()

pq.put((0, source))

while pq.empty() == False:

u = pq.get()[1]

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()

def addedge(x, y, cost):

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

graph[y].append((x, 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)

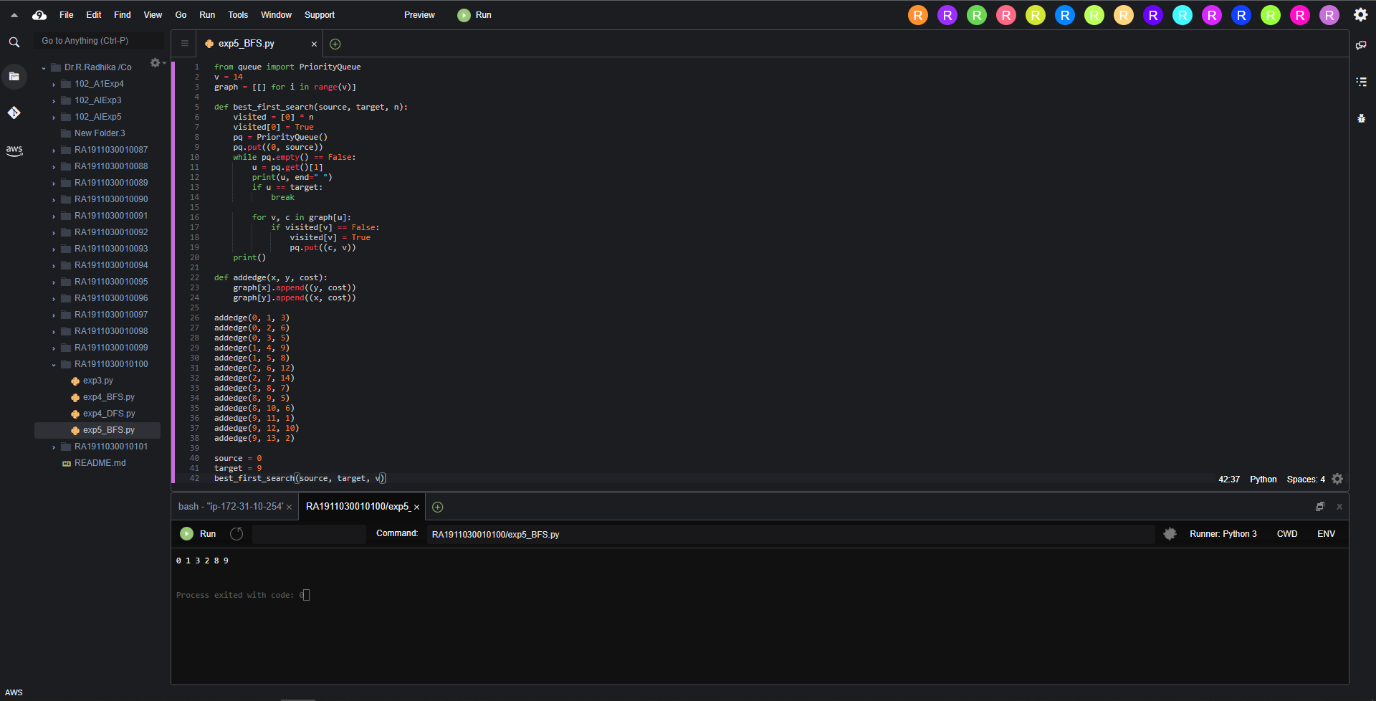
addedge(9, 13, 2)

source = 0

target = 9

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

**Output:**



**Result :**

Hence Best First Search was implemented and executed.