https://lh5.googleusercontent.com/dplghwJq6X4fhzS5H6mFhAFj9x6vI-Y8xCT8NFOTS1m1Xqxiq7nkadVUnCPhdF0ePu4loIUkqVjtvmt0NXfO2k9ohAj4vSqxuecZS-EBDoWiRGD-hgPkQa4QEs6nQaUoqsWtkTeeVLr0namIZbmEyQhttps://lh3.googleusercontent.com/BTBdPiSJjxGslQH3BeZD4BaoJZ39HCgQmAhUsT_pMmuCBkQpXF4Oufxkc29xElrbY7UOC_t-XYD8wCe8-xr0WMFCu3DhySoqaYXxkDd4zDvRd6uFglNfbvwNH7fYiWW7sNqHblYmu1wrAZV9wwFdXA

**GHARDA FOUNDATION**

**GHARDA INSTITUTE OF TECHNOLOGY, LAVEL**

Department of Computer Engineering

**Evaluation Sheet**

Class: TE-Computer Engineering Sem: V Subject: **Artificial Intelligence Lab(CSL604)**

Experiment No: 5

Title of Experiment: Study the implementation of Greedy Best First Search Algorithm.

Name of Student: Niraj Nitin Surve Roll No: 68

Date of Performance:

|  |  |  |  |
| --- | --- | --- | --- |
| Sr. No. | Evaluation Criteria | Max Marks | Marks Obtained |
| 1 | Practical Performance | 8 |  |
| 2 | Oral | 5 |  |
| 3 | Timely Submission | 2 |  |
|  | Total | 15 |  |

                   Signature of Subject Teacher

     (Prof. M. A. Khandke)

**Program Code –**

from queue import PriorityQueue

def greedy\_best\_first\_search(graph, start, goal, weights):

frontier = PriorityQueue()

frontier.put(start, 0)

came\_from = {}

came\_from[start] = None

while not frontier.empty():

current = frontier.get()

if current == goal:

break

for neighbor in graph[current]:

if neighbor not in came\_from:

priority = weights[neighbor]

frontier.put(neighbor, priority)

came\_from[neighbor] = current

return came\_from

graph = {}

n = int(input("Enter the number of nodes: "))

for i in range(n):

node = input("Enter the node: ")

neighbors = input("Enter the neighbors separated by spaces: ")

graph[node] = neighbors.split()

weights = {}

for node in graph:

weight = int(input("Enter the weight of node {}: ".format(node)))

weights[node] = weight

start = input("Enter the start node: ")

goal = input("Enter the goal node: ")

came\_from = greedy\_best\_first\_search(graph, start, goal, weights)

path = []

current = goal

while current != start:

path.append(current)

current = came\_from[current]

path.append(start)

path.reverse()

print("Shortest path:", path)

weight = int(input("Enter the weight of node {}: ".format(node)))

weights[node] = weight

start = input("Enter the start node: ")

goal = input("Enter the goal node: ")

came\_from = greedy\_best\_first\_search(graph, start, goal, weights)

path = []

current = goal

while current != start:

path.append(current)

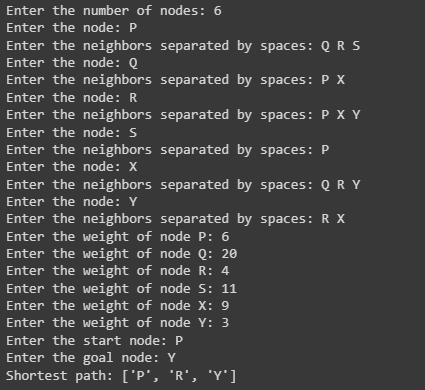
current = came\_from[current]

path.append(start)

path.reverse()

print("Shortest path:", path)

**Output –**

****