PROGRAM-10

BEST FIRST SEARCH PROBLEM

AIM:-

To write and execute the python program for the Best first search program.

PROCEDURE:-

- **Import Required Module**:
  - Import the PriorityQueue class from the queue module to use priority queue data structure for Best-First Search.
- **Graph Class Definition**:
  - Define the Graph class to represent a graph.
  - The constructor initializes an empty dictionary graph to store the graph data.
- **Add Edge Function**:
  - Define the add_edge method to add edges between nodes in the graph along with their respective distances.
- **Best-First Search Algorithm**:
  - Define the best_first_search method to perform the Best-First Search traversal.
  - Add the start node to the priority queue with a priority of 0.
  - While the priority queue is not empty, get the node with the highest priority (lowest distance).
  - If the current node is the goal node, print "Goal reached!" and return.
  - If the current node is not visited, mark it as visited and print "Visiting" along with the current node.
- **Example Usage**:
  - Create an instance of the Graph class.
  - Add edges between nodes along with their distances using the add_edge method.
  - Call the best_first_search method to perform Best-First Search traversal from the start node to the goal node.

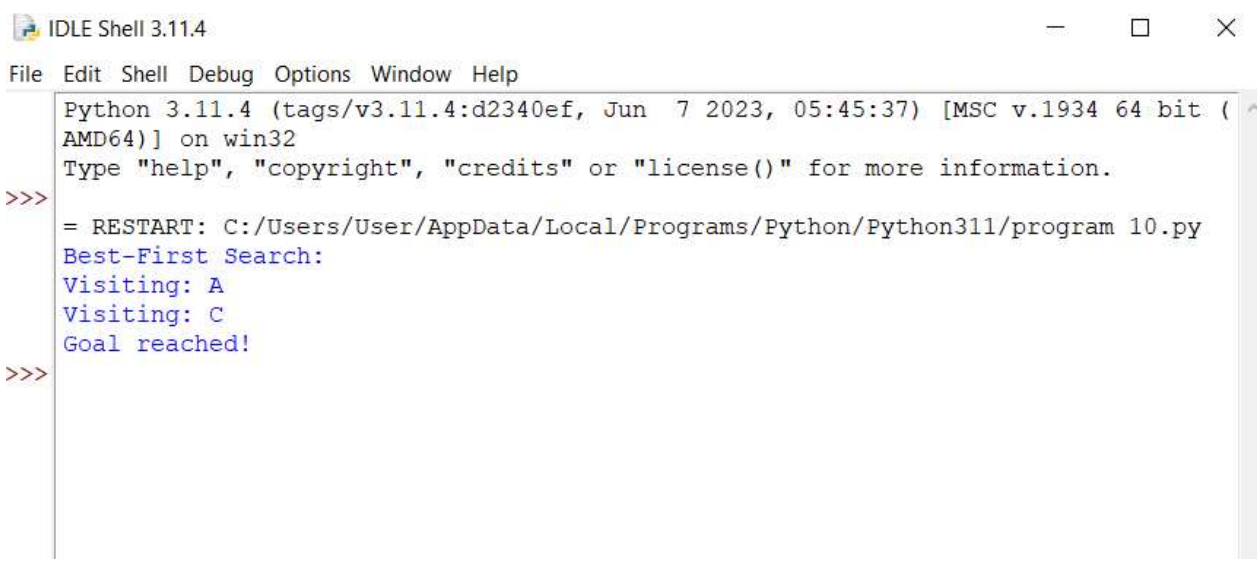
CODING:-

```
from queue import PriorityQueue
```

```
class Graph:
    def __init__(self):
        self.graph = {}
    def add_edge(self, node, neighbor, distance):
        if node not in self.graph:
            self.graph[node] = {}
        self.graph[node][neighbor] = distance
    def best_first_search(self, start, goal):
        visited = set()
        queue = PriorityQueue()
        queue.put((0, start))
        while not queue.empty():
            _, current_node = queue.get()
            if current_node == goal:
                print("Goal reached!")
                return
            if current_node not in visited:
                print("Visiting:", current_node)
                visited.add(current_node)
                for neighbor, distance in self.graph[current_node].items():
                    if neighbor not in visited:
```

```
                queue.put((distance, neighbor))


        print("Goal not reachable!")

graph = Graph()

graph.add_edge('A', 'B', 4)

graph.add_edge('A', 'C', 2)

graph.add_edge('B', 'D', 5)

graph.add_edge('C', 'E', 3)

graph.add_edge('C', 'F', 1)

print("Best-First Search:")

graph.best_first_search('A', 'F')
```

OUTPUT:-



RESULT:-

Hence the program has been successfully executed and verified.