COMPUTER NETWORK'S
ASSIGNMENT- 8

2023

Apoorv Gupta
AIML A1
21070126018

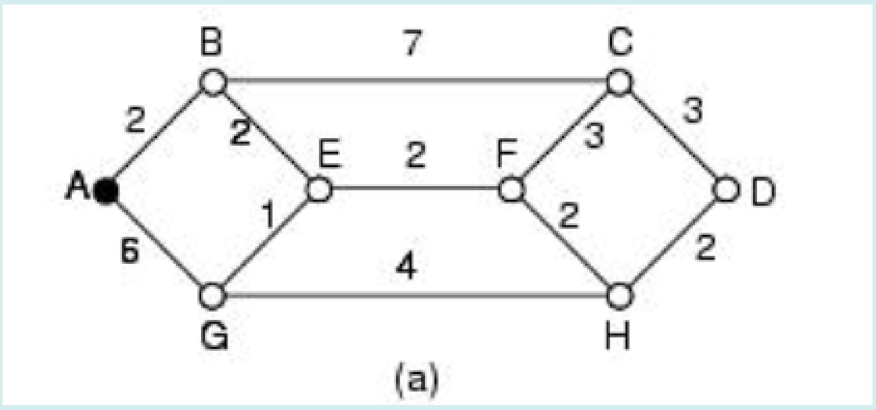
Aim -
Implementation of Shortest Path Algorithm (Dijkstra’s Algorithm)

Theory -:

Dijkstra's Shortest Path Algorithm is a fundamental and widely utilized graph theory algorithm designed to determine the shortest path between various nodes within a graph. It serves as an invaluable tool in numerous applications, including network routing, transportation optimization, and even robotics. At its core, this algorithm efficiently calculates the minimum distance from a designated source node to all other nodes in the graph.
The premise of Dijkstra's algorithm relies on certain critical conditions. It is specifically designed to address single-source shortest path problems, meaning it computes the shortest distances from a single source node to all other nodes within the graph. Moreover, the algorithm's effectiveness is contingent on the assumption that all edge weights within the graph are non-negative. This condition is essential for its reliable and correct operation.
In practice, Dijkstra's algorithm begins with the source node and gradually explores neighboring nodes, systematically evaluating and updating the minimum distances from the source to each node as it traverses the graph. It employs a data structure, often a priority queue, to keep track of the nodes to be explored in an order that prioritizes nodes with the shortest current distance from the source.
As the algorithm proceeds, it greedily selects the next node to visit and, based on its current distance from the source, calculates potential shorter paths to its neighbors. If a shorter path is found, the algorithm updates the recorded distance for that neighbor. This process continues until all nodes have been explored, and the algorithm produces the shortest path and its associated distance from the source to all other nodes.

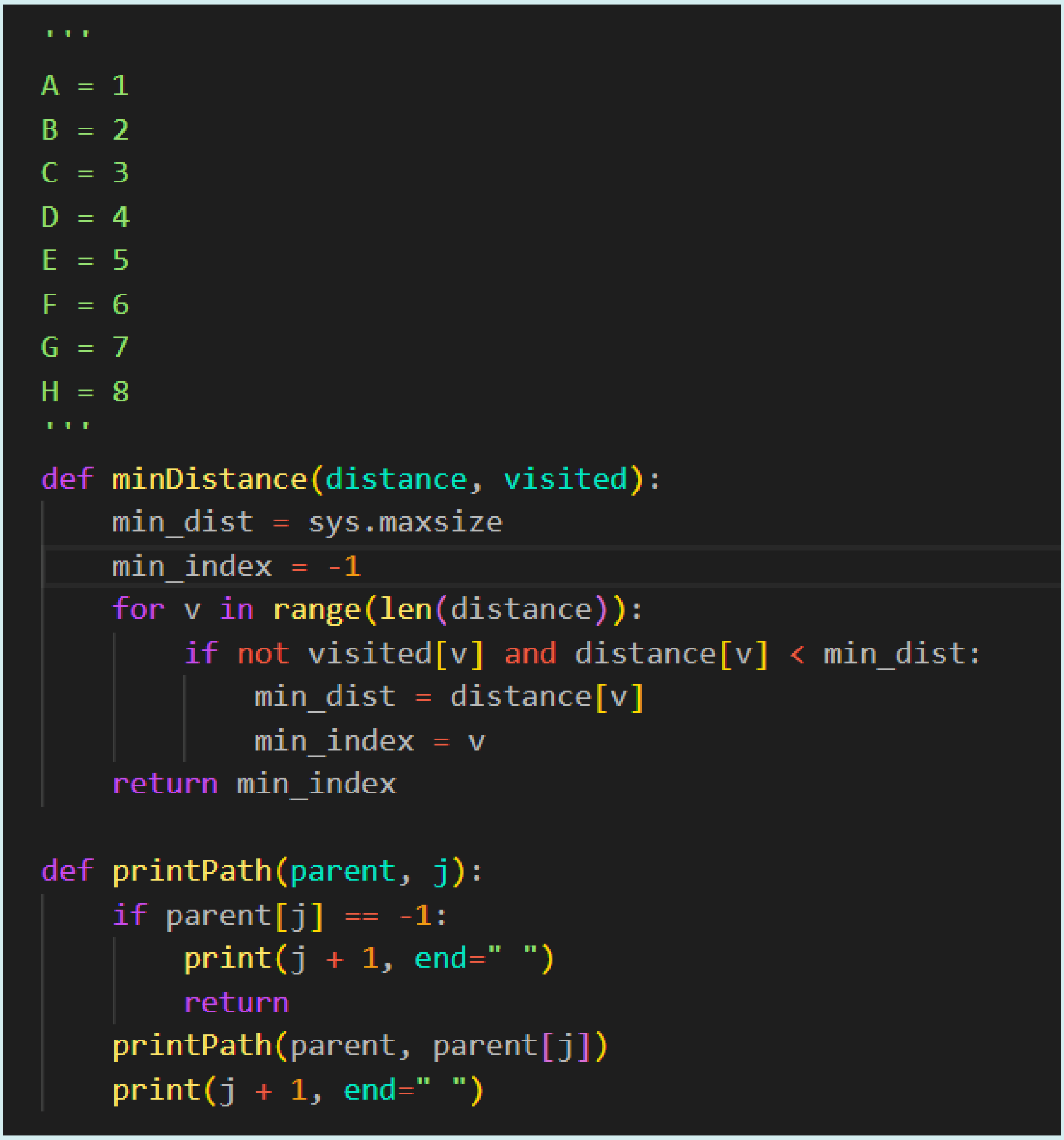
Observation:-

Question:



(a)

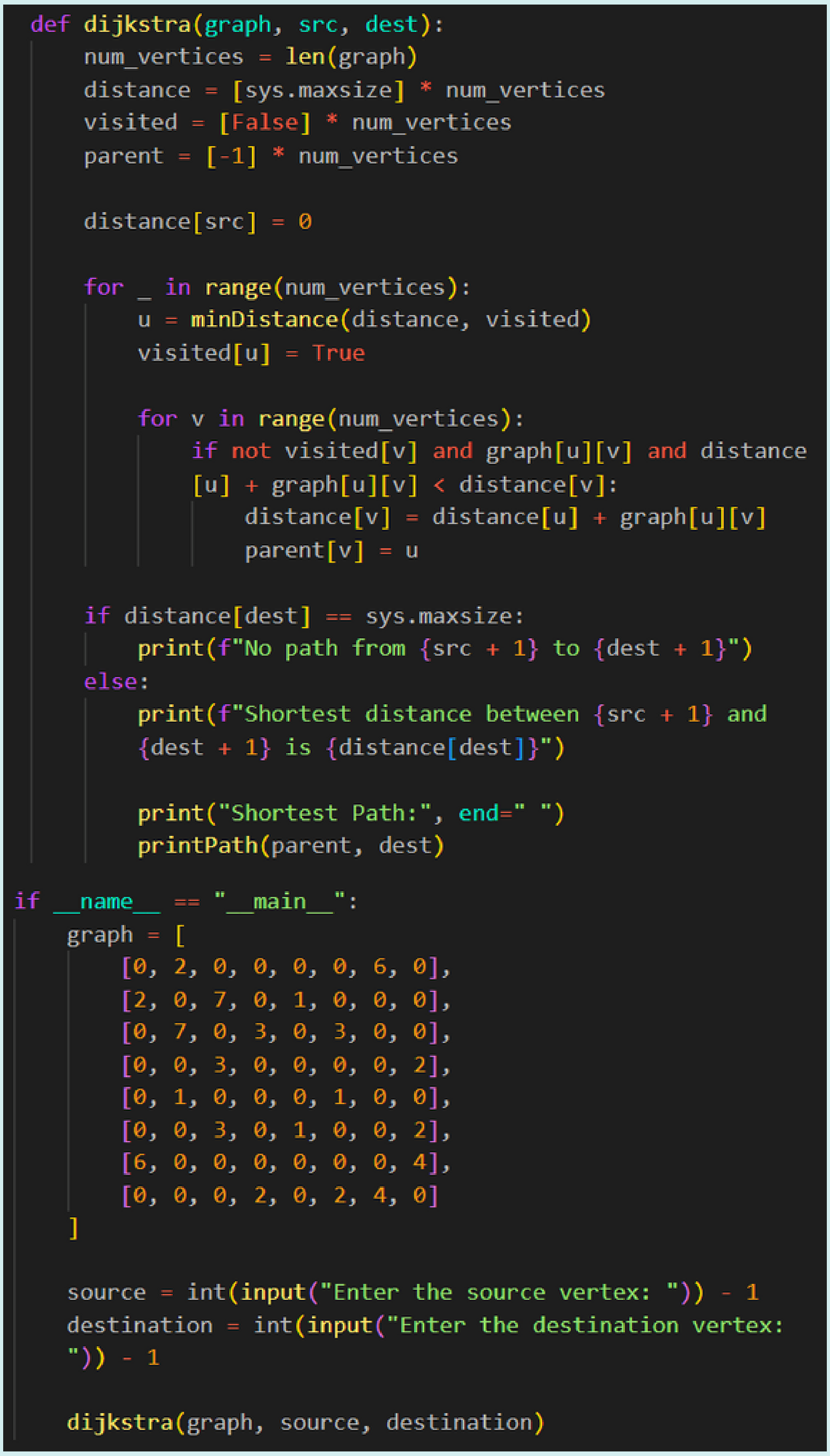
Code



```python
'''
A = 1
B = 2
C = 3
D = 4
E = 5
F = 6
G = 7
H = 8
'''
def minDistance(distance, visited):
    min_dist = sys.maxsize
    min_index = -1
    for v in range(len(distance)):
        if not visited[v] and distance[v] < min_dist:
            min_dist = distance[v]
            min_index = v
    return min_index

def printPath(parent, j):
    if parent[j] == -1:
        print(j + 1, end=" ")
        return
    printPath(parent, parent[j])
    print(j + 1, end=" ")
```

Observation:-

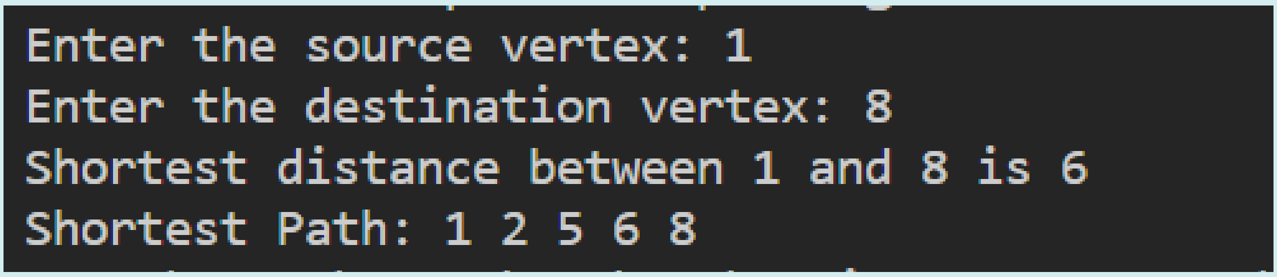
Code



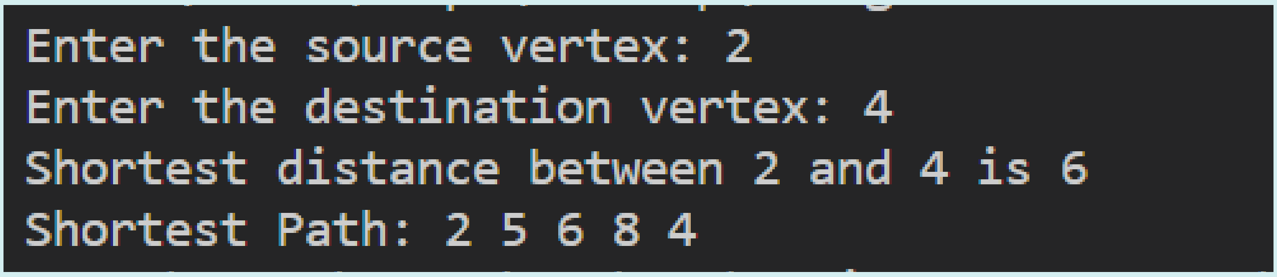
```python
def dijkstra(graph, src, dest):
    num_vertices = len(graph)
    distance = [sys.maxsize] * num_vertices
    visited = [False] * num_vertices
    parent = [-1] * num_vertices

    distance[src] = 0

    for _ in range(num_vertices):
        u = minDistance(distance, visited)
        visited[u] = True

        for v in range(num_vertices):
            if not visited[v] and graph[u][v] and distance
            [u] + graph[u][v] < distance[v]:
                distance[v] = distance[u] + graph[u][v]
                parent[v] = u

    if distance[dest] == sys.maxsize:
        print(f"No path from {src + 1} to {dest + 1}")
    else:
        print(f"Shortest distance between {src + 1} and
        {dest + 1} is {distance[dest]}")

        print("Shortest Path:", end=" ")
        printPath(parent, dest)

if __name__ == "__main__":
    graph = [
        [0, 2, 0, 0, 0, 0, 6, 0],
        [2, 0, 7, 0, 1, 0, 0, 0],
        [0, 7, 0, 3, 0, 3, 0, 0],
        [0, 0, 3, 0, 0, 0, 0, 2],
        [0, 1, 0, 0, 0, 1, 0, 0],
        [0, 0, 3, 0, 1, 0, 0, 2],
        [6, 0, 0, 0, 0, 0, 0, 4],
        [0, 0, 0, 2, 0, 2, 4, 0]
    ]

    source = int(input("Enter the source vertex: ")) - 1
    destination = int(input("Enter the destination vertex:
    ")) - 1

    dijkstra(graph, source, destination)
```

Observation:-

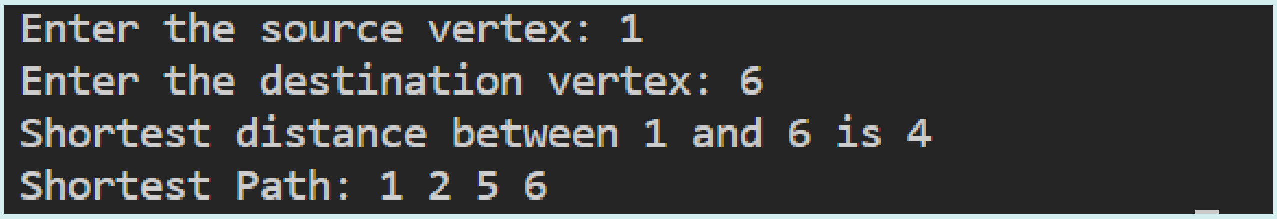
Output:



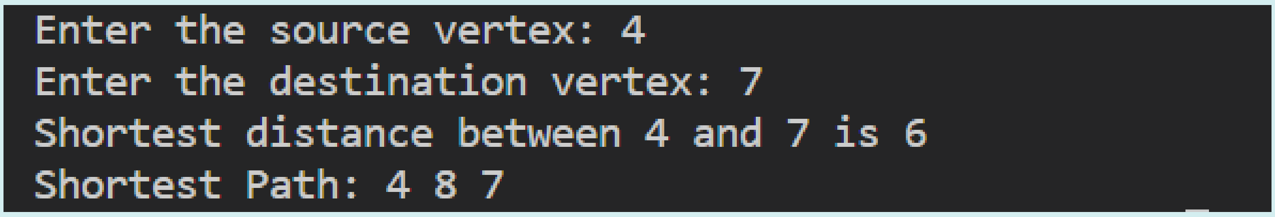
```
Enter the source vertex: 1
Enter the destination vertex: 8
Shortest distance between 1 and 8 is 6
Shortest Path: 1 2 5 6 8
```



```
Enter the source vertex: 2
Enter the destination vertex: 4
Shortest distance between 2 and 4 is 6
Shortest Path: 2 5 6 8 4
```



```
Enter the source vertex: 1
Enter the destination vertex: 6
Shortest distance between 1 and 6 is 4
Shortest Path: 1 2 5 6
```



```
Enter the source vertex: 4
Enter the destination vertex: 7
Shortest distance between 4 and 7 is 6
Shortest Path: 4 8 7
```

Self Assessment:

Q1) Enlist types of routing algorithms in Computer network?

Static Routing: Static routing offers administrators precise control over network paths, making it a straightforward choice for small networks, but it can be cumbersome to manage and lacks adaptability in large or dynamic environments.

Dynamic Routing: Dynamic routing protocols like RIP, OSPF, and EIGRP automatically adjust routing tables based on network changes, ensuring real-time adaptability and scalability in complex network infrastructures.

Distance Vector Routing: Distance vector routing, as seen in RIP, involves routers periodically sharing their routing tables with neighbors, potentially leading to slower convergence and susceptibility to routing loops in intricate topologies.

Link-State Routing: Link-state routing, exemplified by OSPF, maintains a comprehensive and up-to-date network view, resulting in faster convergence and enhanced scalability, albeit demanding more computational resources and memory.

Path Vector Routing: Path vector routing in BGP is a critical protocol on the global scale, primarily used in the Internet, emphasizing path attributes and policy-based route selection to navigate the complex and diverse inter-domain routing environment, facilitating route selection based on various considerations.

Self Assessment:

Q2) Explain shortest path routing in Computer network?

Shortest path routing is a fundamental concept in computer networks used to determine the most efficient path for data packets to traverse from a source node to a destination node within a network. The objective is to minimize the total cost or distance associated with the path, which can be based on various metrics like hop count, bandwidth, delay, or any other network-related criteria. Dijkstra's algorithm is an excellent example of a shortest path routing algorithm used in computer networks.

Dijkstra's algorithm operates as follows:

Initialization: It begins by initializing the distance to the source node as zero and all other distances to infinity. Additionally, it maintains a set of unvisited nodes.

Iteration: The algorithm repeatedly selects the unvisited node with the smallest distance from the source and marks it as visited. It then updates the distances to its neighbors by considering the cost of reaching them through the current node.

Selection of Next Node: The algorithm selects the next node to visit based on the minimum distance. This process continues until all nodes have been visited or the destination node is reached.

Shortest Path Reconstruction: Once the algorithm has visited all necessary nodes and computed the shortest path distance to the destination, it can reconstruct the shortest path by backtracking through the parent or predecessor nodes, starting from the destination node back to the source node.

Self Assessment:

Q3) What is static routing and dynamic routing?

Static routing and dynamic routing are two fundamental approaches to routing in computer networks:

Static Routing:

- Definition: Static routing involves the manual configuration of routing tables on routers within a network. Administrators manually specify the paths that data packets should take to reach their destinations.
- Characteristics: Routes are fixed and do not change unless explicitly reconfigured by an administrator. Simple to set up and understand. Suitable for small, stable networks with predictable traffic patterns.
- Advantages:
    - Control: Administrators have full control over routing decisions.
    - Security: It can be more secure since routes don't change without manual intervention.
- Disadvantages:
    - Scalability: Managing static routes becomes impractical in large or dynamic networks.
    - Inefficiency: Inefficient in adapting to network changes, leading to suboptimal routes.

Dynamic Routing:

- Definition: Dynamic routing algorithms automatically adjust routing tables based on network topology changes and traffic conditions. They adapt to real-time changes in the network.
- Characteristics: Routes are determined dynamically and can change as the network evolves. Complex and require more configuration but offer adaptability. Examples include RIP (Routing Information Protocol), OSPF (Open Shortest Path First), and EIGRP (Enhanced Interior Gateway Routing Protocol).
- Advantages:
    - Adaptability: Dynamic routers respond to network changes, ensuring optimal path selection.
    - Scalability: Suitable for large, complex networks with evolving topologies.
- Disadvantages:
    - Complexity: Requires more configuration and management, making it less suitable for small networks.
    - Potential for Misconfigurations: Complex configurations can introduce errors.

Self Assessment:

Q4) What is spanning tree algorithm?

A spanning tree algorithm is a fundamental concept in graph theory and computer networking, designed to create a subgraph that encompasses all the vertices of the original graph without forming any cycles. Spanning trees are crucial for ensuring network reliability, preventing loops in data transmission, and optimizing network performance. One of the most widely used algorithms for constructing a spanning tree in computer networks is the Spanning Tree Protocol (STP), which ensures that network bridges and switches can efficiently communicate and forward data without causing broadcast storms or loops. By creating a loop-free topology, spanning tree algorithms play a pivotal role in enhancing network stability and efficiency.

Q5) What is need of different routing algorithms in Computer network?

The need for different routing algorithms in computer networks stems from the wide-ranging and evolving requirements of diverse network environments. Various factors such as network size, topology, traffic patterns, security, scalability, and optimization objectives demand routing algorithms tailored to their specific demands. Smaller networks with straightforward topologies may benefit from simple static routing, while larger, dynamic networks rely on complex dynamic routing protocols. The need for routing diversity arises from the imperative to efficiently and securely navigate the intricate landscape of modern computer networks, ensuring optimal data transmission, fault tolerance, and adherence to network policies.

Conclusion:-

The implementation of the Shortest Path Algorithm, notably Dijkstra's Algorithm, in computer networks is indispensable for efficiently determining optimal paths for data transmission. By systematically analyzing network topology and link costs, it enables routers to make intelligent routing decisions, reducing latency, conserving bandwidth, and enhancing network performance. The adaptability of Dijkstra's Algorithm to various network configurations and its ability to handle dynamic changes make it a valuable tool in ensuring reliable communication and seamless data flow in modern computer networks. Its impact on network optimization and the avoidance of congestion is pivotal, facilitating the efficient exchange of information in a connected world.

End of Report