	Subject :		Software :	
			Hardware :	
	Branch :	Semester :	Page No.	Prog No. 5

LEM STATEMENT: Implement distance vector routing algorithm for obtaining routing tables at each node.

ORITHM & CODE:

The Distance Vector Routing Algorithm is one of the fundamental algorithms for routing in a network, where each node periodically sends its routing table to its neighbours, and each node updates its routing table based on the information it receives.

In this algorithm, each node maintains a routing table that contains the distance (or cost) to every other node in the network. The distance is usually represented in terms of hop count or delay, and the nodes exchange this information with each other to update their own routing tables.

Algorithm Overview:

1. Initialization:
   - Every node maintains a routing table where the distance to itself is 0, and the distance to all other nodes is infinity (∞), initially.

INPUT GIVEN	
OUTPUT OBTAINED	
REMARKS	
GRADE :	Signature of Faculty Date : Signature of Student Date :

Subject :		Software :	
		Hardware :	
Branch :	Semester :	Page No.	Prog No.

BLEM STATEMENT

ORITHM & CODE :

2. Distance vector Enchange:

• Each node sends its routing table to its direct neighbours.

• when a node receives the routing table from a neighbour, it updates its own table based on the information received.

3. Update Rule:

• If a node A has a direct route to node ~~B~~ B with distance d-AB, and node C has a direct route to node B with distance d-CB, node A will update its routes to B as d-AB + d-CB if it is smaller than its current value for node B.

4. convergence:

• The algorithm continues to run until no more changes happen in the routing table. i.e. the system has converaged and all nodes have found their best routes to all other nodes.

INPUT GIVEN

OUTPUT OBTAINED

REMARKS

GRADE :	Signature of Faculty Date :	Signature of Student Date :

	Subject :		Software :	
			Hardware :	
	Branch :	Semester :	Page No.	Prog No.

BLEM STATEMENT

ORITHM & CODE :

Pseudocode:

Here's a basic pseudocode outline for the Distance vector Routing Algorithm:

```
# initialize the network
nodes = ["A", "B", "C", "D"]
cost = {
    "A" : {"A" : 0, "B" : 1, "C" : 4, "D" : float ("inf")},
    "B" : {"A" : 1, "B" : 0, "C" : 2, "D" : 5},
    "C" : {"A" : 4, "B" : 2, "C" : 0, "D" : 1},
    "D" : {"A" : float ("inf"), "B" : 5, "C" : 1, "D" : 0}
}
# Function to update routing table.
def update_routing_table (node, neighbours):
        updated = false
        for neighbour in neighbours :
        for each neighbour's distance vector
            # for each neighbour's distance vector
            for dest , cost_to_dest in distance_vectors
[neighbor]. items ():
```

INPUT GIVEN

OUTPUT OBTAINED

REMARKS

GRADE :

Signature of Faculty

Date :

Signature of Student

Date :

	Subject :		Software :	
			Hardware :	
	Branch :	Semester :	Page No.	Prog No.
BLEM STATEMENT				

GORITHM & CODE :

```
        new_cost = cost [node][neighbor] + cost-to-dest
        if new-cost < distance_vectors [node][dest] :
            distance - vectors [node] [dest] = new-cost
            Updated = True
    return updated

#main distance vector routing Process.
def distance_vector_routing():
    Changed = True
    while changed:
        Changed = False
        # for each node, send its routing table to its neighbour.

        for node in nodes:
            neighbors = [neighbor for neighbor in nodes
if cost[node].get (neighbor, float ("inf")) < float ("inf")]
            if update_routing_table (node, neighbors):
                Changed = True
#Run the algorithm
distance_vector_routing()
#Print the final routing tables
for node in nodes:
    Print (f"Routing table for node {node}:{distance_vectors
```

INPUT GIVEN	
OUTPUT OBTAINED	
REMARKS	
GRADE :	Signature of Faculty Date :	Signature of Student Date :

	Subject :		Software :	
			Hardware :	
	Branch :	Semester :	Page No.	Prog No.

BLEM STATEMENT

GORITHM & CODE :

[node]{")

Example

for the given Setup:

• Node A: Direct connection to B with a cost of 1, C with a cost of 4, and no direct connection to D.

• Node B: Direct Connection to A with a cost of 1, C with a cost of 2, and D with a cost of 5.

• Node C: Direct connection to B with a cost of 2, D with a cost of 1, and A with a cost of 4.

• Node D: Only direct connection to C with a cost of 1 and B with a cost of 5.

The algorithm will update the routing tables by enchanging the tables and coverging to the Shortest Paths between all nodes.

INPUT GIVEN

OUTPUT OBTAINED

REMARKS

GRADE :	Signature of Faculty Date :	Signature of Student Date :

	Subject :		Software :	
			Hardware :	
	Branch :	Semester :	Page No.	Prog No.

OBLEM STATEMENT

GORITHM & CODE :

Example Output

```
Routing table for node A: {'A': 0, 'B': 1, 'C', 3, 'D': 8}
Routing table for node B: {'A': 1, 'B': 0, 'C': 2, 'D': 5}
Routing table for node C: {'A': 3, 'B': 2, 'C': 0, 'D': 1}
Routing table for node D: {'A': 8, 'B': 5, 'C': 1, 'D': 0}
```

INPUT GIVEN

OUTPUT OBTAINED

REMARKS

GRADE :	Signature of Faculty Date :	Signature of Student Date :