Cycle 2

Routing algorithms → Adaptive
→ Non adaptive

Distance vector Routing Algorithm

→ To find suitable path.

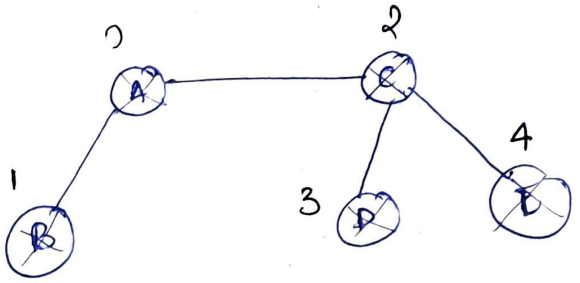
S1 → given a Topology.

I/P • create a Matrix [AM] from topology

O/P • print Routing table entries at all routers along with hop count

Router A B C D E

A → B
A → C
C → D
C → E.



Code:

```
vector<vector<int>>  floydWarshall(vector<vector<int>>
                                   cost, int n)
{
    for(int k=0;k<n;k++)
        for(int i=0;i<n;i++)
            for(int j=0;j<n;j++)
                if(cost[i][k]+cost[k][j]<
                                   cost[i][j])
                cost[i][j] = cost[i][k]
                             +cost[k][j]

    return cost;
}
```

```
void addConn(vector<vector<int>>& topo, int u, int v)
{
    topo[u][v] = 1;
    topo[v][u] = 1;
}

int main()
{
    cout << "Enter no of routers" << endl;
    int n;
    cin >> n;
    vector<vector<int>> topo(n, vector<int>(n, 9999));
    for(int i=0; i<n; i++) topo[i][i] = 0;

    addConn(topo, 0, 1);
    addConn(topo, 0, 2);
    addConn(topo, 2, 3);
    addConn(topo, 2, 4);

    cost = floydWarshall(topo, n);

    return 0;
}
```