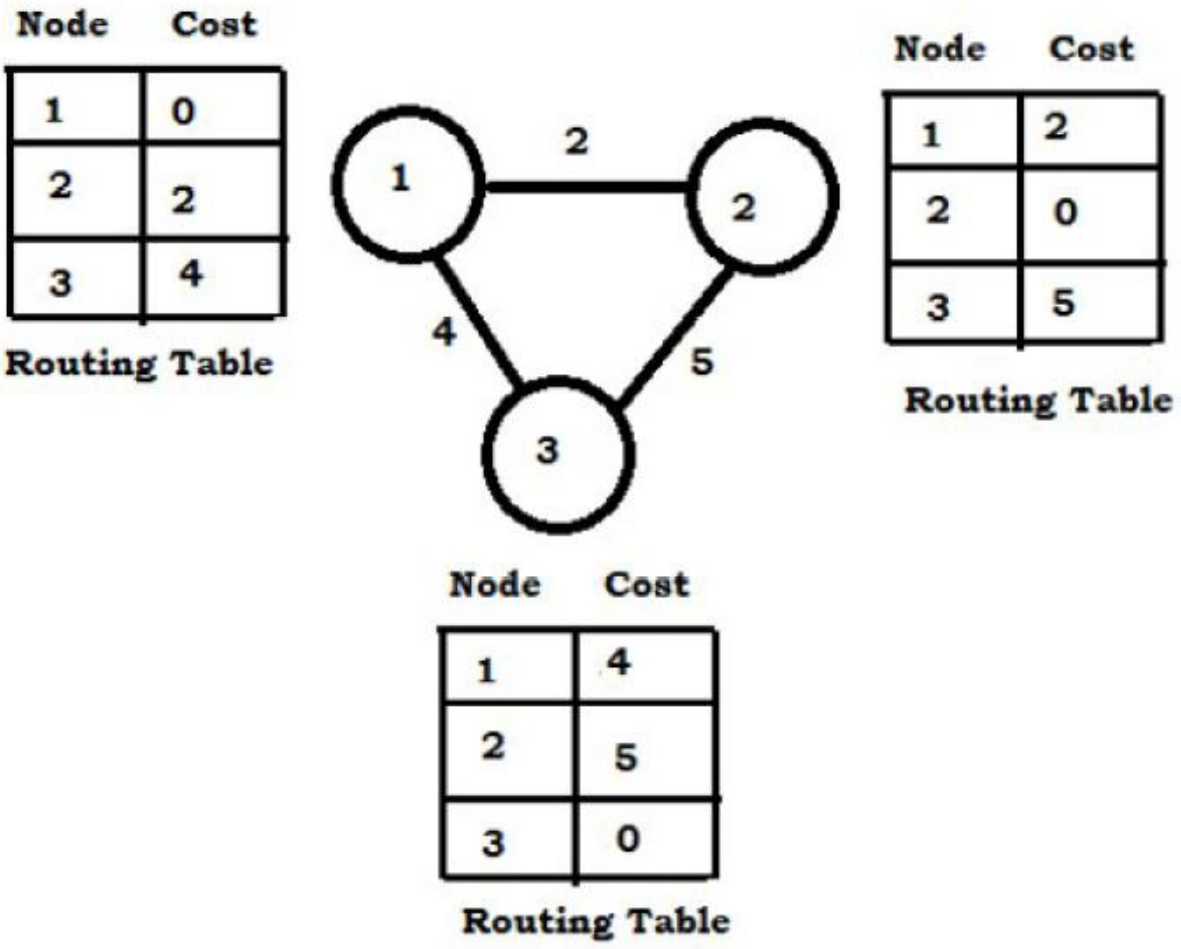
NAME OF THE EXPERIMENT: Distance Vector routing.
AIM: Obtain Routing table at each node using distance vector routing algorithm for a given subnet.



THEORY:
Distance Vector Routing Algorithms calculate a best route to reach a destination based solely on distance. E.g. RIP. RIP calculates the reach ability based on hop count. It's different from link state algorithms which consider some other factors like bandwidth and other metrics to reach a destination. Distance vector routing algorithms are not preferable for complex networks and take longer to converge.

ALGORITHM:
Begin
Step1: Create struct node unsigned dist[20],unsigned from[20],rt[10]
Step2: initialize int dmat[20][20], n,i,j,k,count=0,
Step3: write "the number of nodes "
Step4: read the number of nodes "n"
Step5: write" the cost matrix :"

Step6: intialize i=0
Step7: repeat until i<n
Step8: increment i
Step9: initialize j=0
Step10: repeat Step(10-16)until j<n
Step11: increment j
Step12: read dmat[i][j]
Step13: intialize dmat[i][j]=0
Step14: intialize rt[i].dist[j]=dmat[i][j]
Step15: intialize rt[i].from[j]=j
Step16: end
Step17: start do loop Step (17-33)until
Step18: intilialize count =0
Step19: initialize i=0
Step20: repeat until i<n
Step21: increment i
Step22: initialize j=0
Step23: repeat until j<n
Step24: increment j
Step25: initialize k=0
Step26: repeat until k<n
Step27: increment k
Step28: if repeat Step(28-32) until rt[i].dist[j]>dmat[i][k]+rt[k].dist[j]
Step29: intialize rt[i].dist[j]=rt[i].dist[k]+rt[k].dist[j]
Step30: intialize rt[i].from[j]=k;
Step31: increment count
Step32: end if
Step33: end do stmt
Step34: while (count!=0)
Step35: initialize i=0
Step36: repeat Steps(36-44)until i<n
Step37: increment i
Step38: write ' state values for router',i+1
Step39: initialize j=0
Step40: repeat Steps (40-43)until j<n
Step41: increment j
Step42: write 'node %d via %d distance % ',j+1,rt[i].from[j]+1,rt[i].dist[j]
Step43: end
Step44: end
end