2. **Write a C program for distance vector algorithm to find suitable path for transmission.**

THEORY :

- In Distance Vector routing scheme, each router periodically shares its knowledge about the entire network with its neighbours.
- Each router has a table with information about network. These tables are updated by exchanging information with the immediate neighbours.
- It is also known as **Belman-Ford** or Ford-Fulkerson Algorithm.
- It is used in the original ARPANET, and in the Internet as RIP.
- Neighboring nodes in the subnet exchange their tables periodically to update each other on the state of the subnet (which makes this a dynamic algorithm). If a neighbor claims to have a path to a node which is shorter than your path, you start using that neighbor as the route to that node.
- Distance vector protocols (a vector contains both distance and direction), such as RIP, determine the path to remote networks using hop count as the metric. A hop count is defined as the number of times a packet needs to pass through a router to reach a remote destination.

ALGORITHM :

1. Enter cost matrix.
2. Find minimum distance from one node to another and update.
3. Record route from every node to all nodes.

C CODE :

```
#include<stdio.h>
#include<stdlib.h>
int d[10][10],via[10][10];
int main()
{
        int i,j,k,n,g[10][10];
        printf("\n enter the no of nodes :");
        scanf("%d", &n);
        for(i=0;i<n;i++)
         {
                printf("enter the record for route %c \n", i+97);
                for(j=0;j<n;j++)
                 {
                        printf("(%c : %c) :: ", i+97 ,j+97);
                        scanf("%d", &g[i][j]);
                        if(g[i][j]!=99)
                          d[i][j]=1;
                        }
        }
```

```
        for(i=0;i<n;i++)
         for(j=0;j<n;j++)
            via[i][j]=i;
         for(i=0;i<n;i++)
          {
               for(j=0;j<n;j++)
                if(d[i][j]==1)
                  for(k=0;k<n;k++)
                   if((g[i][j]+g[j][k])<g[i][k])
                    {
                       g[i][k]=g[i][j]+g[j][k];
                       via[i][k]=j;
                                        }

                }

for(i=0;i<n;i++)
{ for(j=0;j<n;j++)
 { if(g[i][j]>g[j][i])
    { g[i][j]=g[j][i];
      via[i][j]=via[j][i];
    }
}
}
                for(i=0;i<n;i++)
                {
                       printf("cost of route %c :\n",i+97);
                       for(j=0;j<n;j++)
                        printf("%c : %d via %c \n",j+97,g[i][j],via[i][j]+97);
       }
       return 0;
}
```

NETWORK :