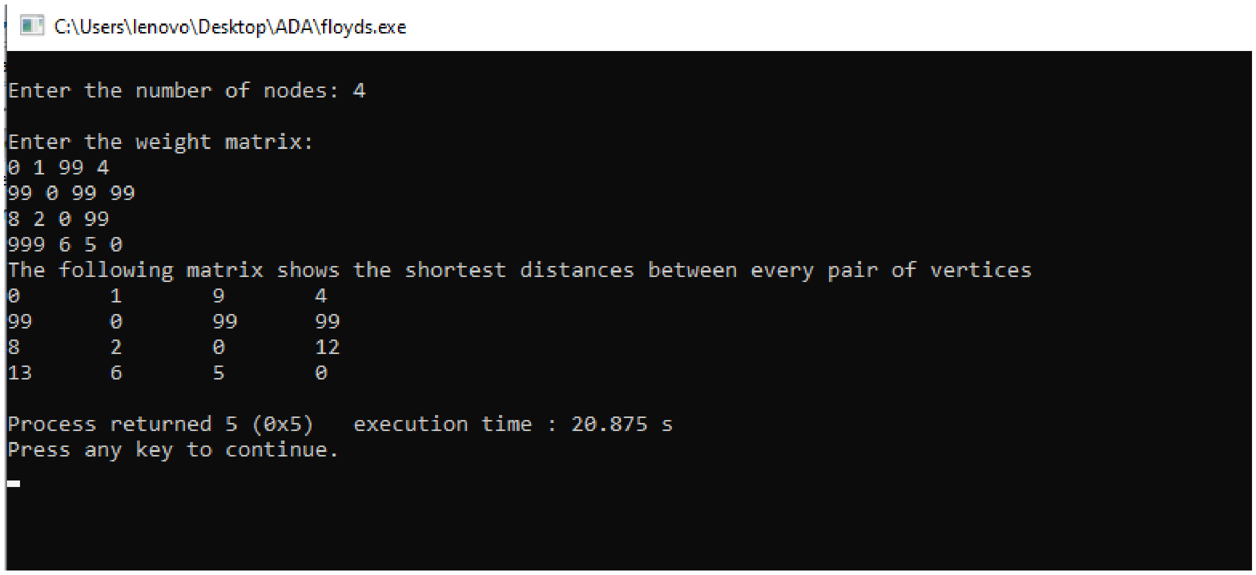
Implement all pair shortest path problem using Floyd's algorithm

```
#include<stdio.h>

#define MAX 10



void display(int n,int w[MAX][MAX])

{

   int i,j;

   printf("The following matrix shows the shortest distances between every pair of vertices \n");

   for (int i = 1; i <= n; i++)

   {

      for (int j = 1; j <= n; j++)

      {

            printf("%d\t", w[i][j]);

      }

      printf("\n");

   }


   //printf("\n The shortest paths are:\n");

      //for(i=1;i<=n;i++)

         //for(j=1;j<=n;j++)

         //{

           // if(i!=j)

               //printf("\n <%d,%d>=%d",i,j,w[i][j]);

         //}

}


void floyds(int n,int w[MAX][MAX])

{
    int i, j, k;

    for (k = 1; k <= n; k++)
    {
        for (i = 1; i <= n; i++)
        {
            for (j = 1; j <= n; j++)
            {
                if (w[i][k] + w[k][j] < w[i][j])
                    w[i][j] = w[i][k] + w[k][j];
            }
        }
    }
    display(n,w);
}

void main()
{
    int i,n,W,j;
    int w[MAX][MAX], dist[MAX][MAX] ;

    printf("\nEnter the number of nodes: ");
    scanf("%d",&n);
    printf("\nEnter the weight matrix:\n");
    for (i = 1; i <= n; i++)
    {
        for (j = 1; j <= n; j++)
        {
            scanf("%d",&w[i][j]);
        }
    }
    floyds(n,w);
}
```



C:\Users\lenovo\Desktop\ADA\floyds.exe

```

Enter the number of nodes: 4

Enter the weight matrix:
0 1 99 4
99 0 99 99
8 2 0 99
999 6 5 0
The following matrix shows the shortest distances between every pair of vertices
0       1       9       4
99      0       99      99
8       2       0       12
13      6       5       0

Process returned 5 (0x5)   execution time : 20.875 s
Press any key to continue.
```

Implement All pair shortest paths problem using Floyd's algorithm.

```
#include <stdio.h>
# define INF 999
void floyd (int graph [ ][100], int vertices)
{
int i, j, k;
for (k =0; k <vertices ; k++) {
for (i = 0; i < vertices; i++) {
for (j =0; j< vertices; j++) {
if (graph [i][k] + graph [k][j] < graph [i][j]) {
  graph [i][j] = graph [i][k] + graph [k][j];
 }
 }
}
}
}
int main( )
{
int vertices;
printf ("Enter the number of vertices:");
scanf ("%d", &vertices);
int graph [100] [100];
Printf ("Enter the adjacency matrix (%d x %d): \n",
vertices, vertices);
for (int i =0 ; i < vertices; i++) {
~~scanf~~ for (int j =0; j< vertices; j++) {
scanf ("%d", &graph [i][j]);
if (graph [i][j] = =0 && i! = j) {
    graph [i][j] = INF;
 }
 }
}
floyd (graph, vertices);
Print f ("\n shortest paths (Adjacency Matrin :)\n");
for (int i=0; i< vertices ; i++) {
for (int j=0; j< vertices ; j++) {
if (graph [i] [j] = INF) {
Printf ("INF \t");
}
else
{
Printf ("%d \t", graph[i][j]);
}
}
Printf ("\n");
}
return 0;
}
```

Output

```
Enter the number of vertices: 4
Enter the adjacency matrin (4x4):
0    1    999  4
999  0    999  999
8    2    0    999
999  6    5    0
shortest paths (Adjacency Matrin)
0    1    9    4
999  0    999  999
8    2    0    12
13   6    5    0
```

[illegible] 27/7/23