```
#include <iostream>
#include <cstdio>
#define MAX 1000000
using namespace std;
int arcs[10][10];//邻接矩阵
int D[10];//保存最短路径长度
int p[10][10];//路径
int final[10];//若final[i] = 1则说明 顶点vi已在集合S中
int n = 0;//顶点个数
int v0 = 0;//源点
int v,w;
void ShortestPath_DIJ()
{
     for (v = 0; v < n; v++) //循环 初始化
     {
          final[v] = 0; D[v] = arcs[v0][v];
          for (w = 0; w < n; w++) p[v][w] = 0;//设空路径
          if (D[v] < MAX) {p[v][v0] = 1; p[v][v] = 1;}
     }
     D[v0] = 0; final[v0]=0; //初始化 v0顶点属于集合S
     //开始主循环 每次求得v0到某个顶点v的最短路径 并加v到集合S中
     for (int i = 1; i < n; i++)
     {
          int min = MAX;
          for (w = 0; w < n; w++)
          {
               //我认为的核心过程--选点
               if (!final[w]) //如果w顶点在V-S中
               {
                    //这个过程最终选出的点 应该是选出当前V-S中与S有关联边
                    //且权值最小的顶点 书上描述为 当前离V0最近的点
                    if (D[w] < min) {v = w; min = D[w];}
               }
          }
          final[v] = 1; //选出该点后加入到合集S中
          for (w = 0; w < n; w++)//更新当前最短路径和距离
          {
               if (!final[w] && (min+arcs[v][w]<D[w]))
               {
                    D[w] = min + arcs[v][w];
                   // p[w] = p[v];
                    p[w][w] = 1; //p[w] = p[v] +  [w]
               }
          }
     }
}
int main()
{
    cin >> n;
    for (int i = 0; i < n; i++)
    {
         for (int j = 0; j < n; j++)
         {
              cin >> arcs[i][j];
         }
    }
    ShortestPath_DIJ();
    for (int i = 0; i < n; i++) printf("D[%d] = %d\n",i,D[i]);
    return 0;
}
```