上海工程技术大学

实　验　报　告

实验三　图的应用

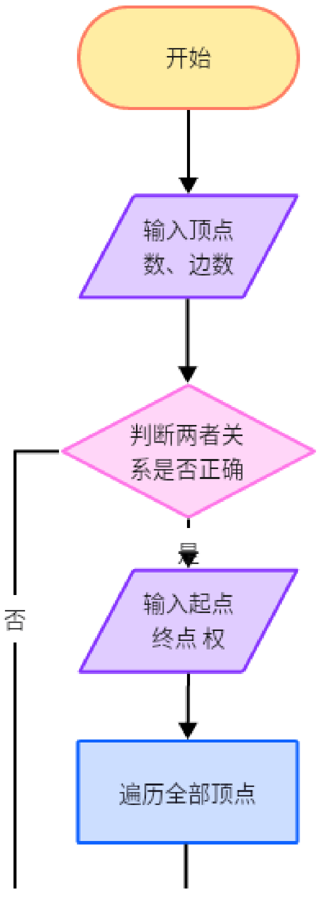
1.　题目 1

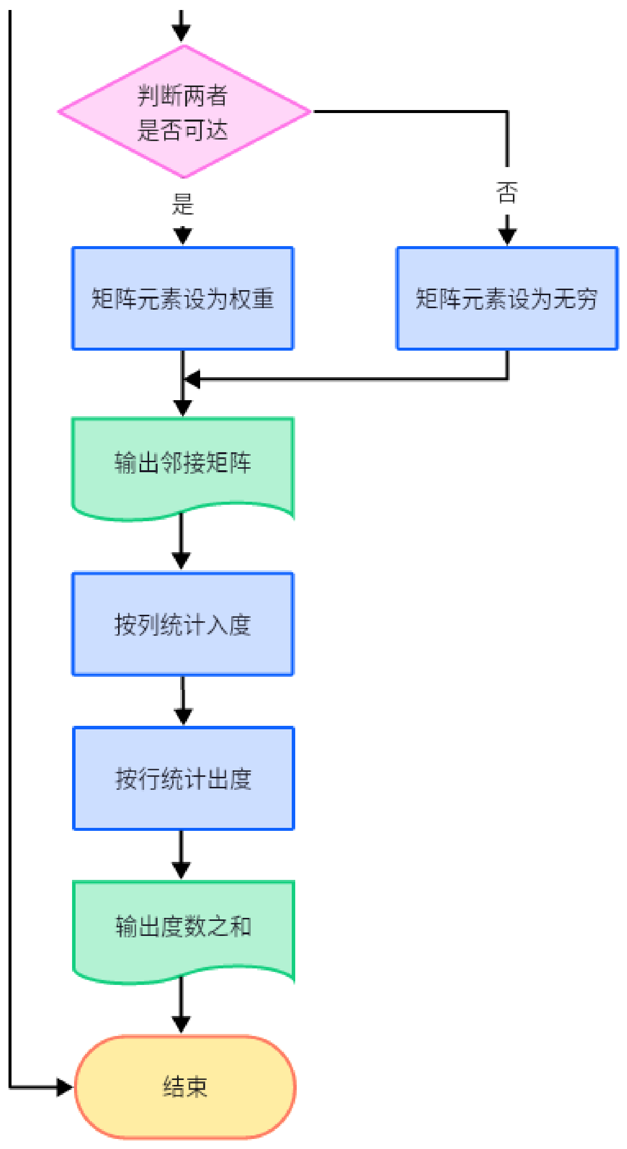
设计一个程序，采用交互方式建立一个网的邻接矩阵表示,并且：

（1）分行输出该邻接矩阵；

（2）求出各顶点的度并输出。

1.1.　流程图



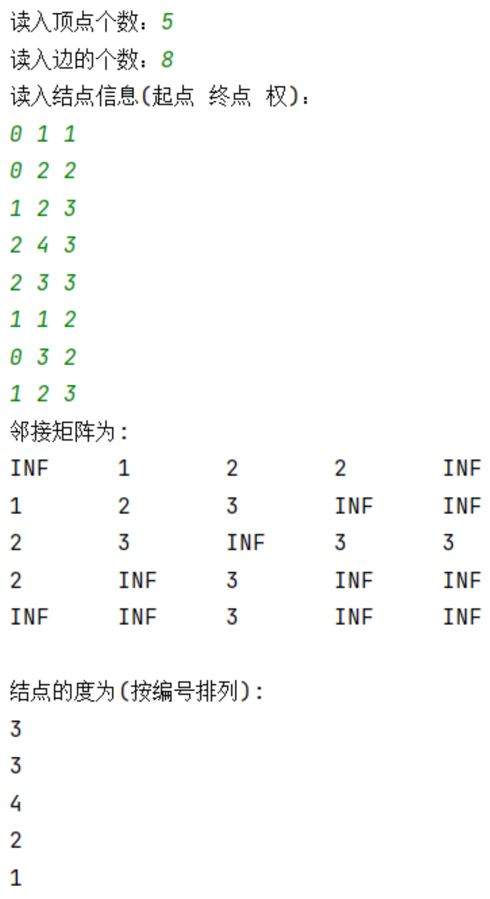


1.2. 代码

```
#include <cstdio>
#include <cstdlib>

#define MaxVertexNum 100
#define INFINITY 65535
typedef int Vertex;
typedef int WeightType;
typedef char DataType;

typedef struct ENode *PtrToENode;
struct ENode{
    Vertex V1, V2;
    WeightType Weight;
};
typedef PtrToENode Edge;
```

```c
typedef struct GNode *PtrToGNode;
struct GNode{
    int Nv;
    int Ne;
    WeightType G[MaxVertexNum][MaxVertexNum];
    DataType Data[MaxVertexNum];
};
typedef PtrToGNode MGraph;
MGraph CreateGraph(int VertexNum)
{
    Vertex V, W;
    MGraph Graph;

    Graph = (MGraph)malloc(sizeof(struct GNode));
    Graph->Nv = VertexNum;
    Graph->Ne = 0;
    for (V=0; V<Graph->Nv; V++)
        for (W=0; W<Graph->Nv; W++)
            Graph->G[V][W] = INFINITY;

    return Graph;
}

void InsertEdge(MGraph Graph, Edge E)
{
    Graph->G[E->V1][E->V2] = E->Weight;
    Graph->G[E->V2][E->V1] = E->Weight;
}

MGraph BuildGraph()
{
    MGraph Graph;
    Edge E;
    int Nv, i;
    printf("读入顶点个数：");
    scanf("%d", &Nv);
    Graph = CreateGraph(Nv);
    printf("读入边的个数：");
    scanf("%d", &(Graph->Ne));
    if ( Graph->Ne != 0 ) {
        E = (Edge)malloc(sizeof(struct ENode));
        printf("读入结点信息(起点 终点 权)：\n");
        for (i=0; i<Graph->Ne; i++) {
                scanf("%d %d %d", &E->V1, &E->V2, &E->Weight);
                InsertEdge(Graph, E);
            }
        }

        return Graph;
}

void ShowGraph(MGraph Graph){
    printf("邻接矩阵为:\n");
    int i,j;
    for(i=0;i<Graph->Nv;i++){
        for(j=0;j<Graph->Nv;j++){
            if(Graph->G[i][j]==INFINITY){
                printf("INF\t");
            }else{
                printf("%d\t",Graph->G[i][j]);
            }
        }
        printf("\n");
    }
}

void NodeDegree(MGraph Graph){
    MGraph temp=Graph;
    int i,j;
    int count;
    for(i=0;i<temp->Nv;i++){
        count=0;
        for(j=0;j<temp->Nv;j++){
            if(temp->G[i][j]!=INFINITY){
                count++;
            }
        }
        printf("%d\n",count);
    }

}
int main(){
    MGraph Graph=BuildGraph();
    ShowGraph(Graph);
    printf("\n");
    printf("结点的度为(按编号排列):\n");
    NodeDegree(Graph);
    return 0;
}
```

1.3. 结果

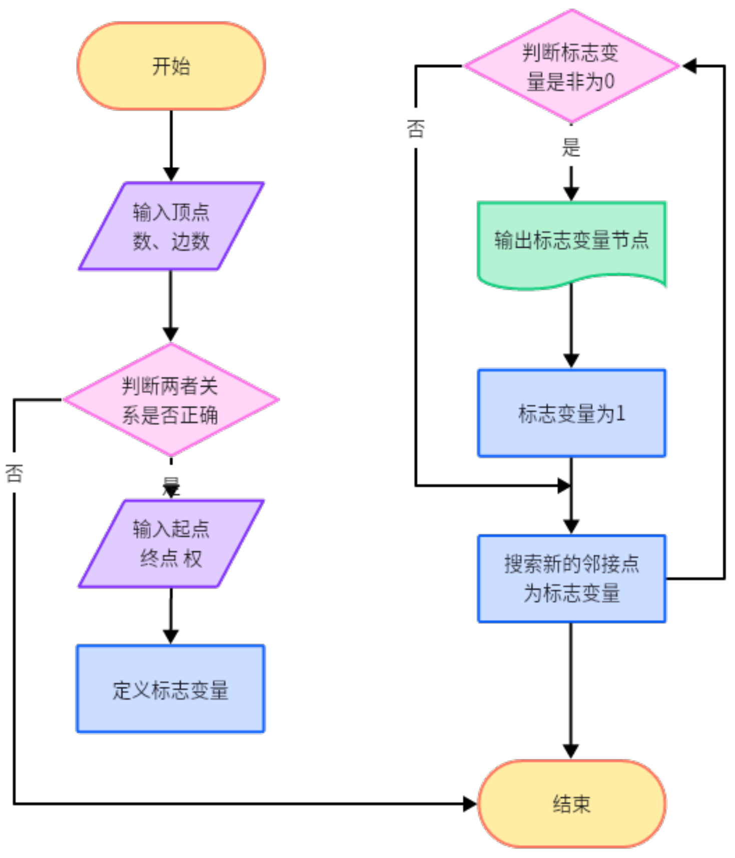


```
读入顶点个数：5
读入边的个数：8
读入结点信息(起点 终点 权)：
0 1 1
0 2 2
1 2 3
2 4 3
2 3 3
1 1 2
0 3 2
1 2 3
邻接矩阵为：
INF     1       2       2       INF
1       2       3       INF     INF
2       3       INF     3       3
2       INF     3       INF     INF
INF     INF     3       INF     INF

结点的度为(按编号排列)：
3
3
4
2
1
```

2. 题目 2

设计一个程序，采用交互方式建立一个无向图的邻接表表示，并输出该图的深度优先搜索遍历得到的顶点序列。

2.1. 流程图

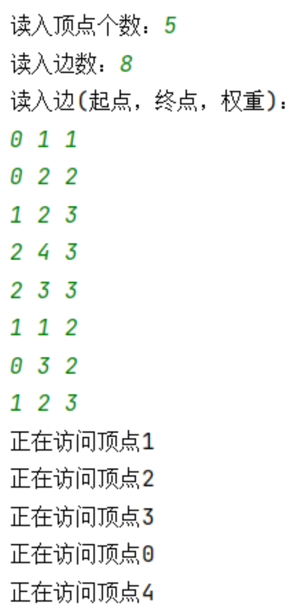


2.2. 代码

```
#include <cstdio>
#include <cstdlib>

#define MaxVertexNum 100
typedef int Vertex;
typedef int WeightType;
typedef char DataType;
int Visited[MaxVertexNum];
typedef struct ENode *PtrToENode;
struct ENode{
    Vertex V1, V2;
    WeightType Weight;
};
typedef PtrToENode Edge;

typedef struct AdjVNode *PtrToAdjVNode;
struct AdjVNode{
    Vertex AdjV;
    WeightType Weight;
    PtrToAdjVNode Next;
};
```

```
typedef struct Vnode{
    PtrToAdjVNode FirstEdge;
    DataType Data;
} AdjList[MaxVertexNum];

typedef struct GNode *PtrToGNode;
struct GNode{
    int Nv;
    int Ne;
    AdjList G;
};
typedef PtrToGNode LGraph;

LGraph CreateGraph( int VertexNum )
{
    Vertex V;
    LGraph Graph;

    Graph = (LGraph)malloc( sizeof(struct GNode) );
    Graph->Nv = VertexNum;
    Graph->Ne = 0;
    for (V=0; V<Graph->Nv; V++)
        Graph->G[V].FirstEdge = NULL;

    return Graph;
}

void InsertEdge( LGraph Graph, Edge E )
{
    PtrToAdjVNode NewNode;
    NewNode = (PtrToAdjVNode)malloc(sizeof(struct AdjVNode));
    NewNode->AdjV = E->V2;
    NewNode->Weight = E->Weight;
    NewNode->Next = Graph->G[E->V1].FirstEdge;
    Graph->G[E->V1].FirstEdge = NewNode;
    NewNode = (PtrToAdjVNode)malloc(sizeof(struct AdjVNode));
    NewNode->AdjV = E->V1;
    NewNode->Weight = E->Weight;
    NewNode->Next = Graph->G[E->V2].FirstEdge;
    Graph->G[E->V2].FirstEdge = NewNode;
}

LGraph BuildGraph()
{
    LGraph Graph;
    Edge E;
    int Nv, i;
    printf("读入顶点个数：");
    scanf("%d", &Nv);
    Graph = CreateGraph(Nv);
    printf("读入边数：");
    scanf("%d", &(Graph->Ne));
    if ( Graph->Ne != 0 ) {
        E = (Edge)malloc(sizeof(struct ENode));
        printf("读入边(起点，终点，权重)：\n");
        for (i=0; i<Graph->Ne; i++) {

            scanf("%d %d %d", &E->V1, &E->V2, &E->Weight);
            InsertEdge( Graph, E );
        }
    }
    return Graph;
}
void Visit( Vertex V )
{
    printf("正在访问顶点%d\n", V);
}

void DFS( LGraph Graph, Vertex V, void (*Visit)(Vertex) )
{
    PtrToAdjVNode W;

    Visit( V );
    Visited[V] = true;

    for( W=Graph->G[V].FirstEdge; W; W=W->Next )
        if ( !Visited[W->AdjV] )
            DFS( Graph, W->AdjV, Visit );
}
int main(){
    LGraph Graph=BuildGraph();
    DFS(Graph,1,Visit);
    return 0;
}
```

2.3. 结果



```
读入顶点个数：5
读入边数：8
读入边(起点，终点，权重)：
0 1 1
0 2 2
1 2 3
2 4 3
2 3 3
1 1 2
0 3 2
1 2 3
正在访问顶点1
正在访问顶点2
正在访问顶点3
正在访问顶点0
正在访问顶点4
```

3. 实验小结

我学习了邻接矩阵和邻接表的概念以及如何构建它们。邻接矩阵是通过将每个顶点与其他所有顶点之间的边关系表示为一个矩阵来描述一个无向图的数据结构。邻接表则是通过链式存储来表示一个无向图的数据结构。相比之下，邻接表在存储空间上更加节省。

在最后的深度优先搜索算法中，我采用了递归的方式来实现。这个算法可以用于解决许多实际问题，但还是存在大量的时间复杂度。希望在后续过程中可以通过记忆化搜索，再结合动态规划实现代码优化。

通过本次实验，我不仅掌握了邻接矩阵和邻接表的知识，也学会了如何利用深度优先搜索算法遍历无向图。这些知识对我的编程能力提升有着很大的帮助。我通过实践掌握了邻接矩阵和邻接表的数据结构以及如何编写一个深度优先搜索算法。

在实现过程中，我遇到了一些困难，例如如何输入无向图的边和顶点数、如何求出每个顶点的度数等。通过本次实验，自己的编程能力得到了提升，并且更加熟练地掌握了 C++语言。