西南石油大学课程设计报告

课程	姓名	学号
算法分析与设计		
专业班级	**任课老师**	**成绩**
软件工程	王欣	

一、设计目的

在一个连通网的所有生成树中，各边的代价之和最小的那棵生成树称为该连通网的最小代价生成树，简称为最小生成树。

构造最小生成树有多种算法，其中多数算法利用了最小生成树的一种简称为MST的性质：假设$N=(V,E)$是一个连通网，U是顶点集V的一个非空子集，若(u,v)是一条具有最小权值（代价）的边，其中$u\in U$、$v\in V-U$，则必存在一棵包含边(u,v)的最小生成树。

$Kruskal$算法就是利用MST性质构造最小生成树的算法，本实验是为了深入掌握$Kruskal$算法的原理与实现。

二、设计内容

用邻接矩阵的形式表示图，借助辅助数据结构使用cpp语言实现$Kruskal$算法。

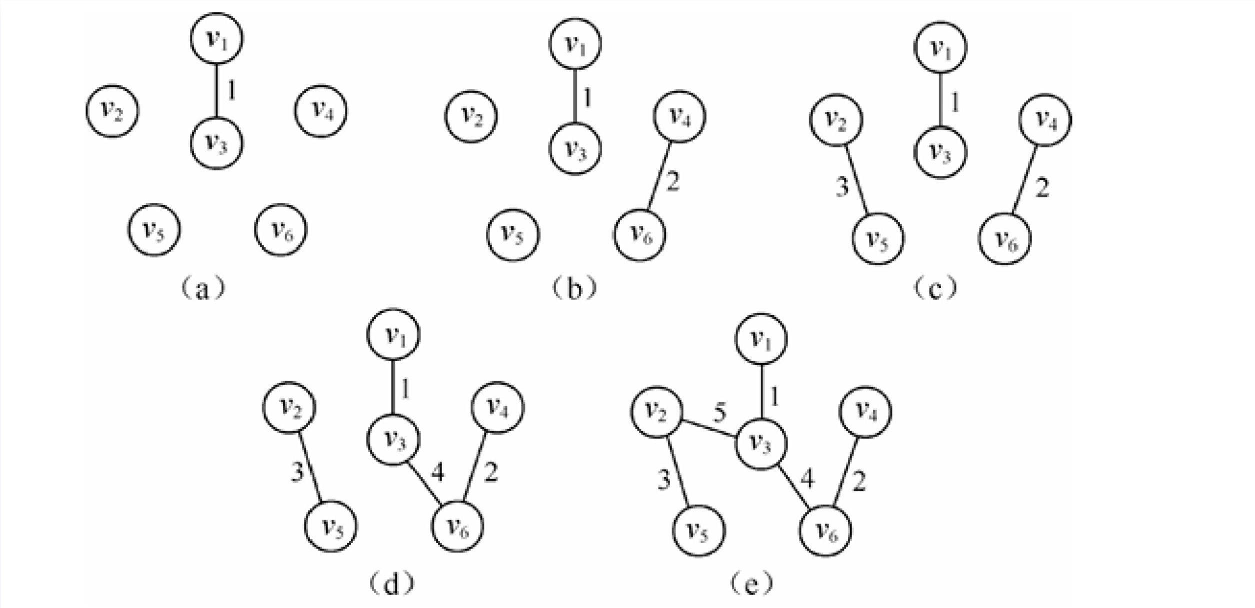
三、步骤

假设连通网$N=(V,E)$，将N中的边按权值**从小到大**的顺序排列。

1. 初始状态为只有n个顶点而无边的非连通图$T=(V,\{\})$，图中每个顶点自成一个连通分量。
2. 在E中选择权值最小的边，若该边依附的顶点落在T中不同的连通分量上（不形成回路），则将此边加入T中，否则舍去此边而选择下一条权值最小的边。
3. 重复2，直至T中所有顶点都在同一连通分量上为止。

$Kruskal$算法逐步增加生成树的边，称为"加边法"。每次选择最小边时，可能有多条**同样**权值的边可选，可以**任选其一**。

示例如下：



Kruskal算法构造最小生成树的过程

四、代码

引入辅助数据结构：

1. 结构体数组$Edge$：存储边的信息，包括边的两个顶点信息和权值。

```
// 辅助数组Edges的定义
// 存储边的信息，包括边的两个顶点信息和权值
struct {
    VerTexType Head;                    // 边的始点
    VerTexType Tail;                    // 边的终点
    ArcType lowcost;                    // 边上的权值
} Edge[(MVNum * (MVNum - 1)) / 2];
```

2. $Vexset[i]$：标识各个顶点所属的连通分量。对每个顶点$vi \in V$，在辅助数组中存在一个 相应元素$Vexset[i]$表示该顶点所在的连通分量。初始时$Vexset[i] = i$，表示各顶点自成一个连通分量。

```
// 辅助数组Vexset的定义
// 标识各个顶点所属的连通分量
int Vexset[MVNum];
```

$Kruskal$算法如下：

```
// 无向网G以邻接矩阵形式存储，构造G的最小生成树T，输出T的各条边
// 适合于求稀疏网的最小生成树
```

```cpp
void MiniSpanTree_Kruskal(AMGraph G) {
    int v1, v2, vs1, vs2;

    Sort(G);                                // 将数组Edge中的元素按权值从小到大排序
    for (int i = 0; i < G.vexnum; ++i)  // 辅助数组，表示各顶点自成一个连通分量
        Vexset[i] = i;

    // 依次查看排好序的数组Edge中的边是否在同一连通分量上
    for (int i = 0; i < G.arcnum; ++i) {
        v1 = LocateVex(G, Edge[i].Head);  // v1为边的始点Head的下标
        v2 = LocateVex(G, Edge[i].Tail);  // v2为边的终点Tail的下标
        vs1 = Vexset[v1];                 // 获取边Edge[i]的始点所在的连通分量vs1
        vs2 = Vexset[v2];                 // 获取边Edge[i]的终点所在的连通分量vs2

        // 边的两个顶点分属不同的连通分量
        if (vs1 != vs2) {
            std::cout << Edge[i].Head << "-->" << Edge[i].Tail << std::endl;  // 输出此边
            for (int j = 0; j < G.vexnum; ++j)  // 合并vs1和vs2两个分量，即两个集合统一编号
                if (Vexset[j] == vs2) Vexset[j] = vs1;  // 集合编号为vs2的都改为vs1
        }
    }
}
```

其中 `Sort` 为一个基本的排序函数。

五、体会

通过实现$Kruskal$算法，可以深入理解贪心算法的设计思想，以及并查集这一重要数据结构的应用。同时，这也是一个很好的练习，有助于提升解决图论问题的能力。