# 数据结构课程设计实验报告

姓名： 侯越 学号： 16051615 专业： 计科 周 三 班

## 题目说明：

小L居住的地方有很多城市，每个城市编号从1到n，城市之间共有m条道路相连，这些道路每一条都有一个固定的行驶速度V。

一天，小L要从城市S出发去另一个城市T游玩，连接这些城市的每条道路都有一个固定的行驶速度V，现在小L想知道从S出发到T所经过的道路中最大速度Vmax和最小速度Vmin最小差值是多少？输入数据保证S一定可以到达T。

两个城市可能有多条道路。

输入：一个整数Q，代表有多少组测试数据。

接下来对每组测试数据：

第1行:n、m，城市数目和道路数目。

第2~m+1行：3个整数Ui,Vi,Wi, (i=1,…..,M)，道路的两个城市编号和道路的行驶速度。

最后一行：两个整数S T ,代表起点和终点的城市编号。

约束条件：

2≤Q≤5 ，1<n≤500，0<m≤5000, 1≤ Ui, Vi , S , T ≤n, 0< Wi <30000，

输出：一个整数代表所求最小差值。

样例输入：

2

3 2

1 2 1

2 3 3

1 3

3 3

1 2 6

1 2 5

2 3 8

1 3

样例输出：

2

2

需求规格说明表

|  |  |
| --- | --- |
| 功能需求 | 详细说明 |
| 使用说明 | 用户按照提示规定输入相应信息即可得出答案 |

功能模块结构说明表

|  |  |  |  |
| --- | --- | --- | --- |
| 函数名 | 参数说明 | 返回值说明 | 操作行为说明 |
| init | 无参数 | 无返回值 | 初始化函数，为结果RES变量赋值为最大，为记录边数的tot变量赋值为1，为记录父亲节点的father数组赋值为自身，为要找的最大路径和最小路径分别赋值为最小值和最大值 |
| addedge | 参数1：传入一条弧的弧尾  参数2：传入一条弧的弧头  参数3：这条弧的权值 | 无返回值 | 添加边的操作，记录边的端点和权值。 |
| merge | 参数1：传入要查找的结点编号 | 返回值：返回该结点的父亲节点的编号 | 递归记录结点的父亲节点，若该结点的父亲节点还有父亲节点，则该结点为其父亲的父亲结点的孩子 |
| comp | 参数1：常无类型的指针  参数2：常无类型的指针 | 返回值：整数类型 | 比较两个整数的大小，用来排序边的权值 |

测试计划表

|  |  |  |  |
| --- | --- | --- | --- |
| 测试项目 | 测试步骤 | 期望结果 | 测试目的 |
| 整体代码测试 | 1. 按照要求输入相应信息 | 得出正确结果 | 正面测试，测试程序是否正确运行 |

缺陷记录表

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 程序缺陷说明 | 修复情况 | 备注 |
| 1 | 排序函数qsort使用不当导致不排序 | 已修复 | 严重bug必须修复 |

//代码

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

const int inf = (1 << 31) - 1;

struct Edge

{

int u;

int v;

int w;

};

struct Edge edge[5010];

int father[510];

int tot, RES, Min\_road, Max\_road, S, T, N\_city, M\_road;

void init()

{

int i;

tot = 1;

RES = inf;

Max\_road = -inf;

Min\_road = inf;

for (i = 1; i <= N\_city; i++)

father[i] = i;

}

void addedge(int u, int v, int w)

{

edge[tot].u = u;

edge[tot].v = v;

edge[tot++].w = w;

}

int merge(int n)

{

if (father[n] != n)

father[n] = merge(father[n]);

return father[n];

}

int comp(const void \*a, const void \*b)

{

return (((const struct Edge\*)a)->w - ((const struct Edge\*)b)->w);

}

int main()

{

int i, j, u, v, w, n, m;

printf("Please input number of test case:\n");

scanf("%d", &n);

for (m = 1; m <= n; m++)

{

printf("Please input the number of cities and roads:\n");

scanf("%d %d", &N\_city, &M\_road);

init();

printf("Please input the details:\n");

for (i = 1; i <= M\_road; i++)

{

scanf("%d %d %d", &u, &v, &w);

addedge(u, v, w);

}

qsort(edge + 1, M\_road, sizeof(struct Edge), comp);

printf("Please input the start city and end city:\n");

scanf("%d %d", &S, &T);

for (i = 1; i <= M\_road; i++)

{

for (j = 1; j <= N\_city; j++)

father[j] = j;

Max\_road = -inf;

Min\_road = inf;

for (j = i; j <= M\_road; j++)

{

u = edge[j].u;

v = edge[j].v;

u = merge(u);

v = merge(v);

if (u != v)

{

father[u] = v;

if (Max\_road < edge[j].w)

Max\_road = edge[j].w;

if (Min\_road > edge[j].w)

Min\_road = edge[j].w;

}

u = S;

v = T;

u = merge(u);

v = merge(v);

if (u == v && RES > Max\_road - Min\_road)

{

RES = Max\_road - Min\_road;

}

}

}

printf("Case #%d Result:%d\n", m, RES);

}

return 0;

}