**数据结构课程设计**

**班级：1615403**

**学号：161540205**

**姓名：张佳钰**

**指导老师：孙涵**

4、最优灌溉(必做) （图）

[问题描述]

某村很多片麦田，为了灌溉这些麦田，村里在第一个麦田挖了一口水井，所有的麦田都从这口井来引水灌溉。还要建立一些水渠，以连接水井和麦田，可以利用部分麦田作为“中转站”，利用水渠连接不同的麦田，只要一片麦田能被灌溉，则与其连接的麦田也能被灌溉。已知哪些麦田之间可以建设水渠和建设每个水渠所需要的费用（注意不是所有麦田之间都可以建立水渠）。编程求解：灌溉所有麦田最少需要多少费用来修建水渠，以及修建水渠的连接说明。

[基本要求]

输入格式 输入的第一行包含两个正整数n, m，分别表示麦田的片数和可以建立的水渠的数量。麦田使用1, 2, 3, ……依次标号。 接下来m行，每行包含三个整数ai, bi, ci，表示第ai片麦田与第bi片麦田之间可以建立一条水渠，所需要的费用为ci。

输出格式。 输出一个整数，表示灌溉所有麦田所需要的最小费用。及水渠连接说明。

输入样例

4 4

1 2 1

2 3 4

2 4 2

3 4 3

输出样例 6 说明：建立以下三条水渠：麦田1与麦田2、麦田2与麦田4、麦田4与麦田3。

采用的数据结构：

链表

算法思想：

迪杰斯特拉算法

源程序：

#include<iostream>

#include<queue>

#include<stack>

#define TEST\_FILE "all\_test.txt"

#include<fstream>

using namespace std;

typedef struct country

{

int weight;

int next\_country;

country \*next\_sibling;

country(){weight=0;next\_country=0;next\_sibling=NULL;}

}country;

struct Save\_node{

int from;

int to;

int weight;

Save\_node(){from=0;to=0;weight=0;}

Save\_node(int a,int b,int c){from=a;to=b;weight=c;}

};

typedef struct all\_country

{

country all\_c[100];

int country\_num;

int L\_num;

}all\_country;

int visit[100];

int read\_table(char \* test\_file,all\_country &C)

{

fstream f1;

// f1.open("best\_watering.txt",ios::in);

f1.open(test\_file,ios::in);

if(f1.fail())

{

return 0;

}

f1>>C.country\_num>>C.L\_num;

country \*p;

int from,to,weight;

for(int i=0;i<C.L\_num;i++)

{

f1>>from>>to>>weight;

// cout<<from<<" "<<to<<" "<<weight<<endl;

p=new country;

p->next\_country=to;

p->weight=weight;

p->next\_sibling=C.all\_c[from].next\_sibling;

C.all\_c[from].next\_sibling=p;

p=new country;

p->next\_country=from;

p->weight=weight;

p->next\_sibling=C.all\_c[to].next\_sibling;

C.all\_c[to].next\_sibling=p;

}

f1.close();

return 1;

}

int get\_mini(all\_country C,int node,int &min\_node,int &min\_weight)

{

country \*p;

min\_node=-1;

min\_weight=-1;

int flag=0;

p=C.all\_c[node].next\_sibling;

while(p)

{

// cout<<endl<<p->next\_country<<" "<<visit[p->next\_country]<<endl;

if(visit[p->next\_country]==0)

{

if(min\_weight==-1||(p->weight<min\_weight))

{

min\_weight=p->weight;

min\_node=p->next\_country;

}

// cout<<endl<<"node: "<<node<<" min\_node: "<<min\_node<<" "<<min\_weight<<endl;

flag=1;

}

p=p->next\_sibling;

}

return flag;

}

void mini\_road(all\_country C)

{

memset(visit,0,sizeof(visit));

int v=1;

int n=C.country\_num;

queue<Save\_node> q;

stack<int> p;

int min\_node=-1;

int min\_weight=-1;

p.push(v);

visit[v]=1;

n--;

while(n)

{

if(min\_node==-1)

{

v=p.top();

p.pop();

}

else

{

q.push(Save\_node(v,min\_node,min\_weight));

visit[min\_node]=1;

p.push(v);

v=min\_node;

n--;

}

get\_mini(C,v,min\_node,min\_weight);

// cout<<v<<" "<<min\_node<<" "<<min\_weight<<" "<<n<<endl;

// system("pause");

}

Save\_node l;

int sum=0;

cout<<"The mini road: "<<endl;

while(!q.empty())

{

l=q.front();

q.pop();

sum+=l.weight;

cout<<l.from<<" "<<l.to<<" "<<l.weight<<endl;

}

cout<<"The mini cost :"<<sum<<endl;

}

void display\_table(all\_country C)

{

int i=1;

while(i<=C.country\_num)

{

country \*p;

p=C.all\_c[i].next\_sibling;

while(p)

{

cout<<i<<" "<<p->next\_country<<" "<<p->weight<<endl;

p=p->next\_sibling;

}

i++;

}

}

int main()

{

fstream f\_test;

int i=1;

char test\_file\_name[30];

f\_test.open("all\_test.txt",ios::in);

if(f\_test.fail())

{

cout<<"can't open file: "<<TEST\_FILE<<endl;

exit(1);

}

while(!f\_test.eof())

{

f\_test>>test\_file\_name;

cout<<"test\_file "<<i++<<": "<<test\_file\_name<<endl<<endl;

all\_country C;

if(read\_table(test\_file\_name,C))

{

// display\_table(C);

mini\_road(C);

cout<<endl;

}

else

{

cout<<"can't open file: "<<test\_file\_name<<endl;

}

}

return 0;

}

测试数据：

village.ttx

21 24

1 3 60

2 3 52

3 7 19

4 5 62

5 10 80

10 9 66

7 9 22

6 7 88

8 9 54

9 17 23

17 18 35

17 13 70

13 11 56

11 12 31

17 16 55

19 16 32

16 15 92

15 14 51

16 20 96

20 21 70

3 5 88

1 2 70

5 9 3

10 11 5

best\_watering.txt

4 4

1 2 1

2 3 4

2 4 2

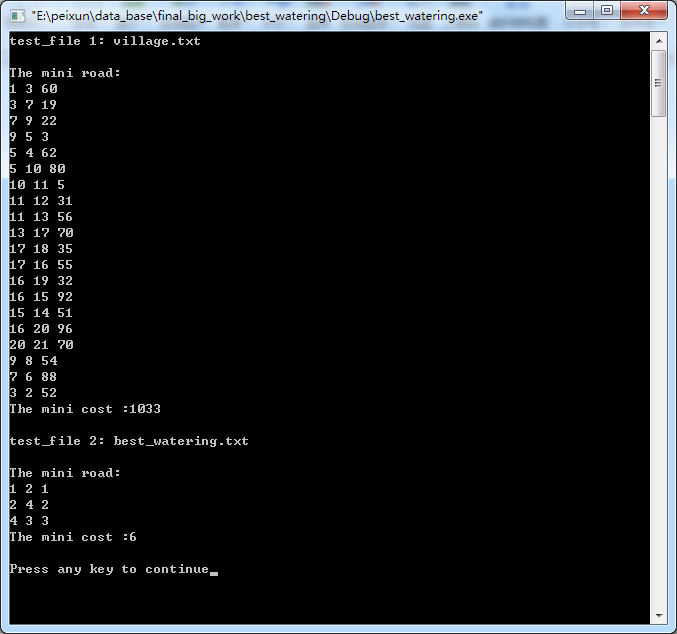
3 4 3

all\_test.txt

village.txt

best\_watering.txt

测试结果：



代码行数：

174行