**第四次《数据结构》编程作业**

**姓名：叶文霆**

**学号：2014211519**

**班级：2014211314**

**分工情况：全部独立完成**

**编译环境：c++ (g++编译器)**

**完成日期：2015.12.7**

Problem 1

【问题描述】

给定一张图，用 Floyd算法求其任意两点间的最短路径。

输入格式：

第1行，输入两个整数 n,e 分别表示图的顶点数和边的个数。

第2行到第 e+1行，输入两个**小写字母** a,b 和一个整形w，表示从记号为a 的点到记号为 b 的点有一条边权为 w 的边.【**a 与 b 之间只能有一个空格**】

输出格式：

共 n(n-1) 行，输出任意两点间的最短距离。

【算法思想】

用 name[ch-‘a’] 存储字母 ch 对应的点在存储最短距离数组 dist中的下标是多少。 Data[i]表示下标为 i的点起对应的字母是什么。然后之行 Floyd 算法即可。

其中 path 数组我用了 c++里自带的 string 类。

【因为用的是 Floyd 算法，代码中我直接用 dist 表示了邻接矩阵】

【设计描述】

存储结构：邻接矩阵

处理流程及调用关系： 读取图→执行 Floyd 算法→输出

算法描述：略

【源程序】

#include <iostream>

#include <cstring>

#include <cstdio>

#define MaxN 27

#define BigNum 233333 //代指无穷

using namespace std;

int name[MaxN]={},dist[MaxN][MaxN],n,e; //name 存储字符到对应下标的转换， data 存储下标到对应字母的转换

char data[MaxN]={};

string path[MaxN][MaxN]={};

void Initialize()

{

memset(dist,63,sizeof(dist));

cin>>n>>e;

for (int i=1; i<=n; i++) dist[i][i]=0; //初始化dist 数组

for (int i=1, k=1; i<=e; i++)

{

//读入字符 a,b 还有一个整形 w

char a,b;

int w;

getchar();

cin>>a;

getchar();

cin>>b;

cin>>w;

//处理字符和数字的对应关系

if (0==name[a-'a'])

{

name[a-'a']=k;

data[k++]=a;

}

if (0==name[b-'a'])

{

name[b-'a']=k;

data[k++]=b;

}

//对 dist 数组赋初值

dist[name[a-'a']][name[b-'a']]=w;

}

}

void Floyd()

{

for (int k=1; k<=n; k++)

for (int i=1; i<=n; i++)

for (int j=1; j<=n; j++)

{

if (dist[i][j]>dist[i][k]+dist[k][j])

{

dist[i][j]=dist[i][k]+dist[k][j];

path[i][j]=path[i][k]+"->"+data[k]+path[k][j];

}

}

}

void Print()

{

for (int i=1; i<=n; i++)

for (int j=1; j<=n; j++)

{

if (i==j) continue;

if (dist[i][j]>BigNum) cout<<"There is no way from "<<data[i]<<" to "<<data[j]<<endl;

else if (!path[i][j].empty()) cout<<data[i]<<" to "<<data[j]<<" shortest way: "<<data[i]<<path[i][j]<<"->"<<data[j]<<", and length: "<<dist[i][j]<<endl;

else cout<<data[i]<<" to "<<data[j]<<" shortest way: "<<data[i]<<"->"<<data[j]<<", and length: "<<dist[i][j]<<endl;

}

}

int main(int argc, char const \*argv[])

{

Initialize();

Floyd();

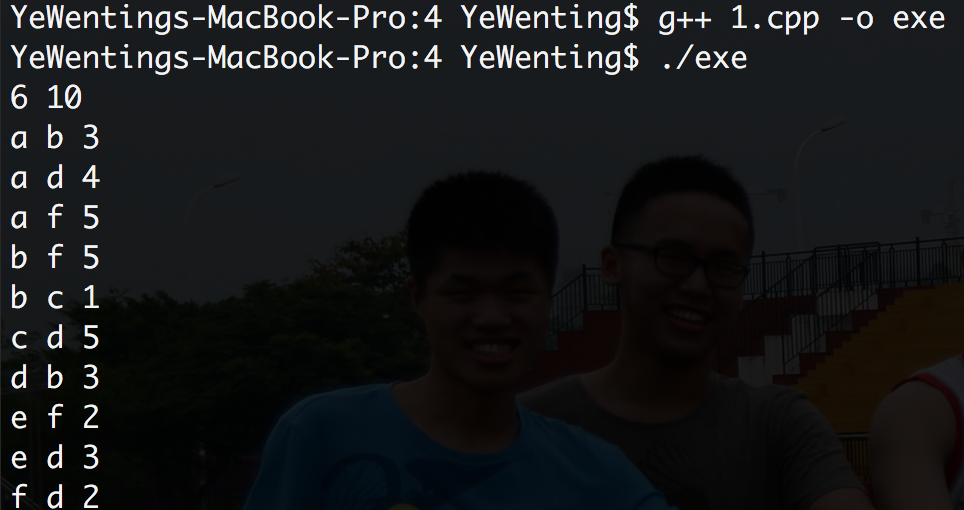
Print();

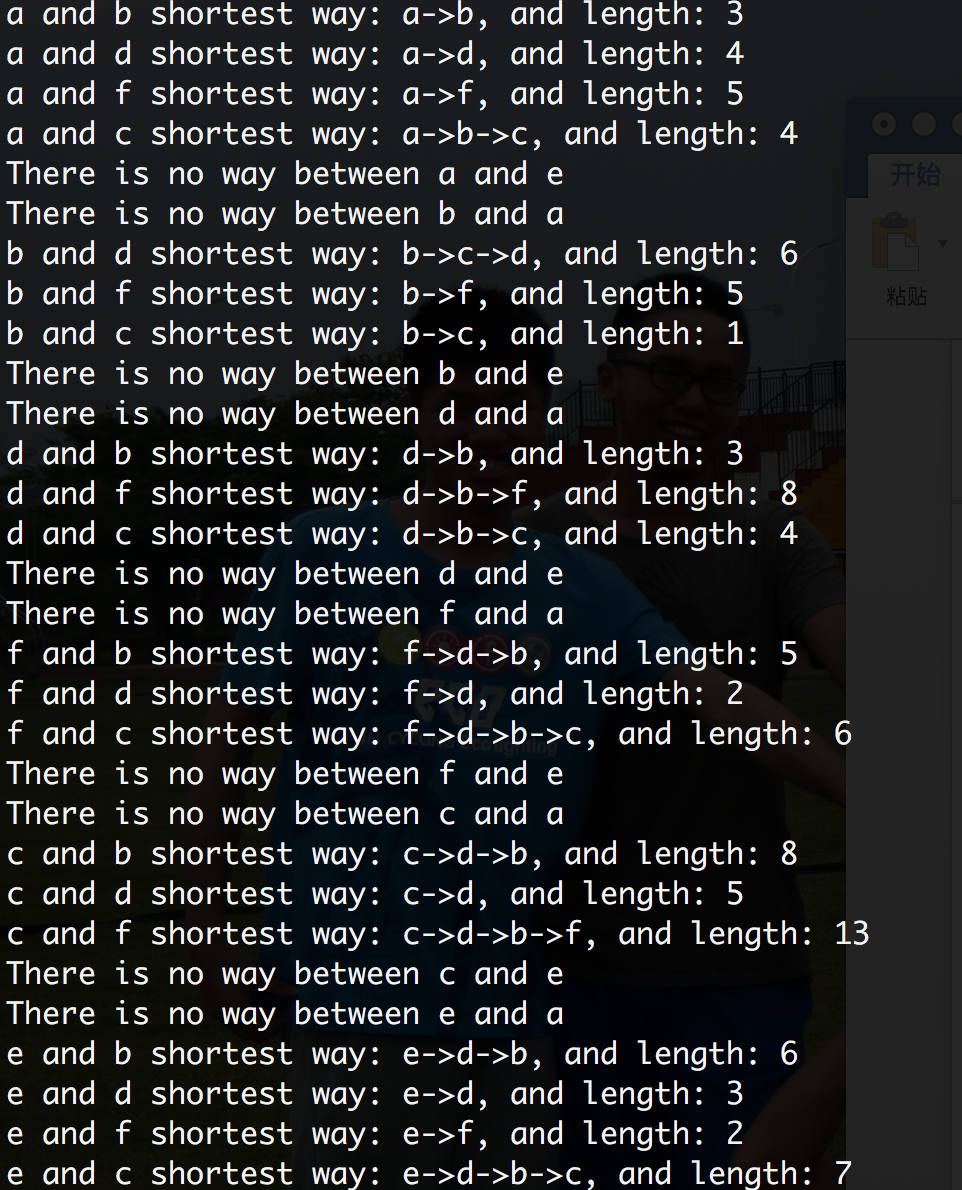
return 0;

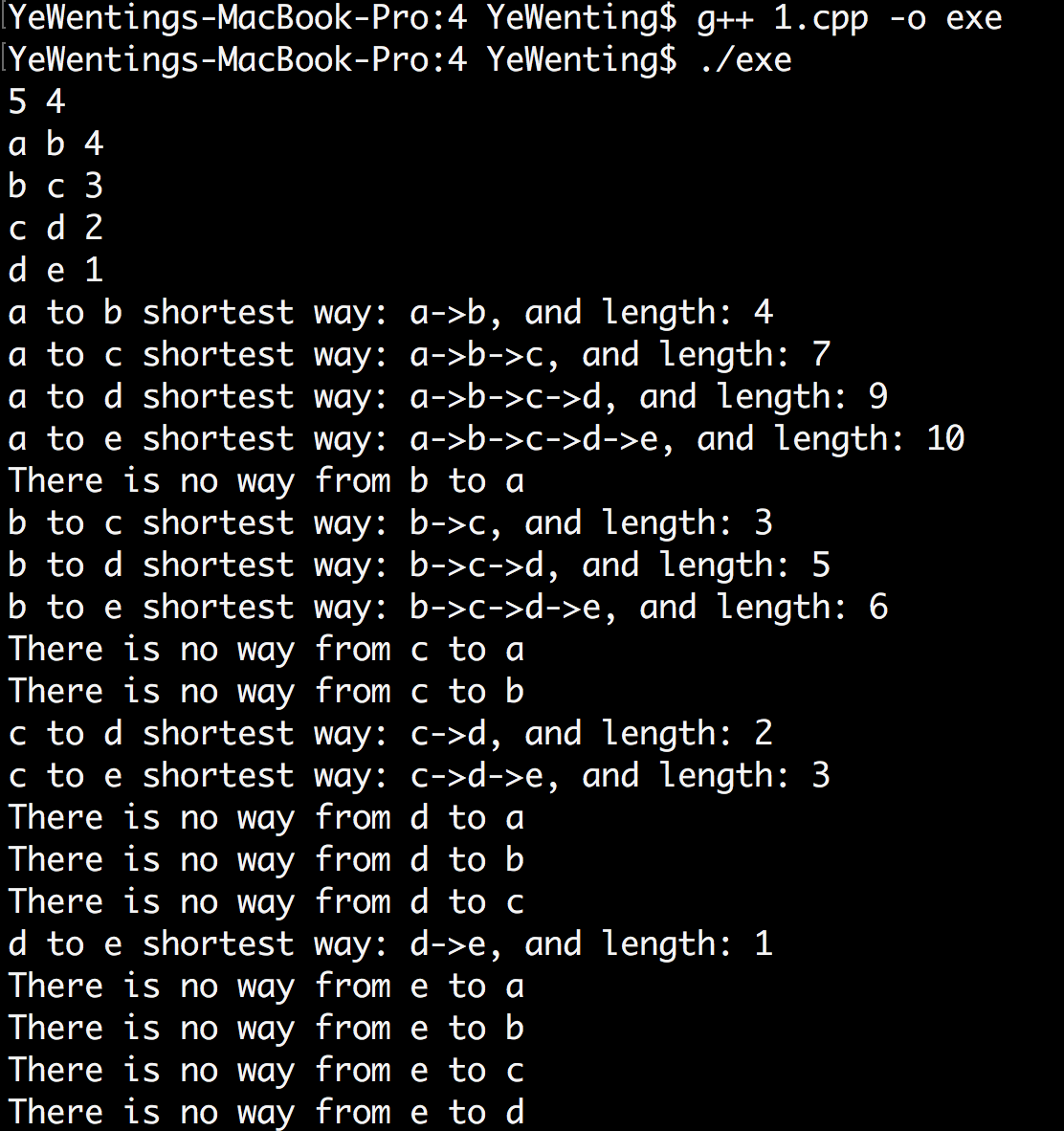
}

【测试说明】

此图为题目所求的图





****

此图为 a->b->c->d 的一个链

【用户使用说明】要求每个节点**用小写字母**表示

【心得体会】

做此题是一开始在读入读出处理的时候遇到了一些困难，后来就用 getchar 野蛮处理了..之后在 path 处理也遇到也一点小问题，出现了a->b->->c->d的问题，后面意识到是自己在Floyd 过程中多加了一个->，改完以后就AC了 心情愉悦~

Problem 2

【问题描述】

用邻接表存储一张无向图，求对于给定点对，是否存在一条长度为 k 的路径

输入：第1行，四个整数 n,e,k,start,end 表示无向图的顶点个数、边数、路径长度，以及起点和终点。

第2-e+1行，每行两个整数 a,b，表示 a与 b 之间有一条边。

输出：若存在，则输出路径，若不存在则输出“No Answer!”

【算法思想】

就是最基础的 dfs… 【其实 bfs 效率会更高一些但是在存储路径上面会多消耗一些空间】

【**设计描述**】

存储结构：邻接表

处理流程及调用关系： 读取图→建图→dfs→输出答案

算法描述：

【源程序】

#include <iostream>

#include <cstring>

#define MaxV 100

using namespace std;

bool used[MaxV]={}; //表示节点 i是否已经遍历过

int k,start,dest,tot=0,path[MaxV]={};

typedef struct node

{

int adjvex; //表示邻接点

struct node \*prev;

}ArcNode,\*FirstArc;

typedef struct gragh

{

FirstArc head[MaxV]; //head[i]表示节点 i 所邻接的第一条边

int VertNum,EdgeNum;

}ALGraph;

void CreateArc(ALGraph &g, const int a, int b) //新建一条 a->b 的边

{

ArcNode \*p=new ArcNode;

p->adjvex=b;

p->prev=g.head[a];

g.head[a]=p;

}

void GetGraph(ALGraph &g)

{

cin>>g.VertNum>>g.EdgeNum>>k>>start>>dest;

for (int i=1;i<=g.VertNum;i++)

g.head[i]=NULL;

for (int i=1;i<=g.EdgeNum;i++)

{

int x,y;

cin>>x>>y;

CreateArc(g,x,y);

CreateArc(g,y,x);

}

}

void Print()

{

tot++;

cout<<"Path "<<tot<<": "<<start<<"->";

for (int i=1;i<=k-1;i++)

cout<<path[i]<<"->";

cout<<dest<<endl;

}

void dfs(const ALGraph &g, const int now, const int length) //now 表示现在所在的点 length表示从 start 到 now 的路径长度

{

if (length==k&&now==dest) //如果k 步到达了目标点

{

Print();

return;

}

if (length<k)

{

for (FirstArc i=g.head[now]; i; i=i->prev)

if (used[i->adjvex]==0) //如果还有邻接的边

{

path[length+1]=i->adjvex;

used[i->adjvex]=true;

dfs(g,i->adjvex,length+1);

path[length+1]=0; //回溯

used[i->adjvex]=false;

}

}

}

void DFSTraverse(const ALGraph &g)

{

used[start]=true;

dfs(g,start,0);

}

int main(int argc, char const \*argv[])

{

ALGraph MyGraph;

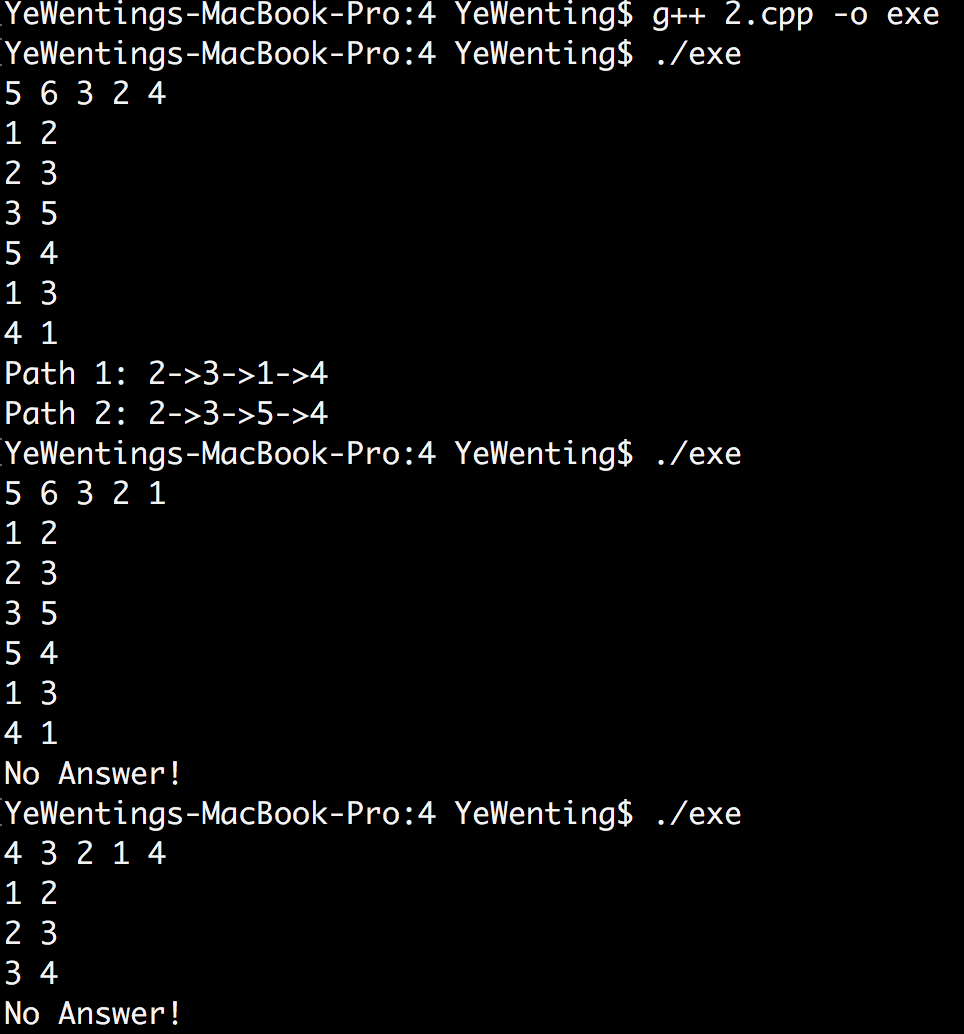
GetGraph(MyGraph);

DFSTraverse(MyGraph);

if (0==tot) cout<<"No Answer!"<<endl;

return 0;

}

【测试说明】

【用户使用说明】 用数字来表示一个节点

【心得体会】

这道题花了挺长时间的…

首先是构建图的一个数据结构，我把老师课件里面的Vnode类型的定义去掉了，原因是每个顶点不存在 info 可以记录，其下标就是唯一信息。这样使得 ALGraph 更为的精简和直观，head[i] 指向顶点 I 邻接的第一条边。此外我没有把 used[]和 path[]放进 ALGraph的结构体内，一是我认为其是并不是图的内在属性而是其的一个内推结果，二是这样在dfs 函数传参中可以把 ALGraph 设为 const，保证搜索过程中图的安全性。

其次我在 dfs 在回溯过程中一开始也写错了…导致搜索过程非常奇葩..后面发现是因为我把回溯的语句为了代码美观换了一个位置，结果就错了…事实证明代码还是原来的好用2333

最后函数传参的时候我也考虑了很多，先看：

void dfs(const ALGraph &g, const int now, const int length)

加 const 显而易见是为了增加代码的安全性，但是我在 const 和 length 并没有加引用来提高代码效率，是因为考虑到：



对于 i->adjvex和 length+1，系统会开一个临时变量来存储他们的值，然后自动返回一个引用给 dfs 调用。所以如果我“硬”是要提高效率，now 是无可奈何了，length 则可以先在调用 dfs 之前 length++，但是这样就与 const int length 矛盾了，相比较之下最终我选择了不加引用…

Plus,对于g.head[i]里面“.”“[]”是同一优先级运算符，所以这里可以省略掉 (g.head)[i]的括号