《实验4: 有向图的最短路径 》

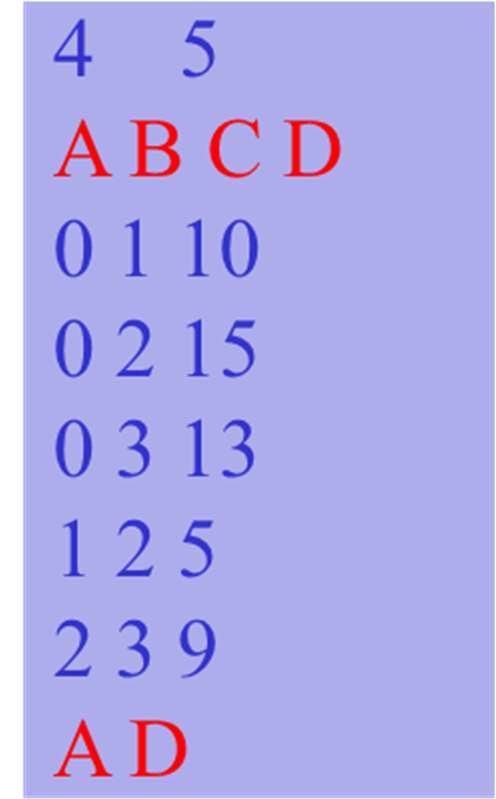
实验报告

|  |  |  |
| --- | --- | --- |
| 工作分工 | 姓名 | 学号 |
| 编码实现 | 胡少卿 | 2020211825 |
| 测试 | 喻成卓 | 2020212457 |
| 报告编写 | 赵航 | 2020212797 |

2020年 12 月 12日

# 题目：实验题目

## 问题描述：

编写一个程序，实现如下功能:  
1) 从文件或界面输入有向图，有向图采用如下形式输入: 

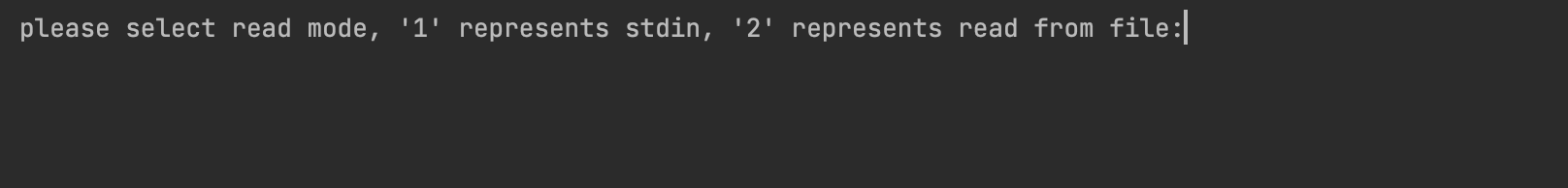
上述数据中第一行输入两个整数，按顺序分别是顶点数和弧数。 第二行输入所有的顶点 data;上面例子中使用 char 类型。 第三行开始，每行输入一条弧，三个整数值分别是弧尾、弧头和权值。 最后一行为需要计算最短路径的起点和终点的顶点。 注意:上图只是作为输入格式说明用途，无需当成一个测试用例。

2) 根据输入的两个顶点，输出两个顶点间的最短路径。

## 设计描述

1. 输入提示：

首先在运行该程序之前，程序会提示选择输入源，确认该次执行此程序读入的方式如下图：



按照该次实验要求的内容，该程序提供了两种输入的方式，分别是通过文件输入和通过标准键盘输入在界面输入。

1. 读入并存储有向图的流程：
2. 寻找并输出最短路径的流程：
3. 寻找并输出关键路径的流程：
4. 该程序中所定义的部分函数以及变量简述：

bool dijkstra(Graph &G); //求最短路  
bool AOE(Graph &G); //求关键路径  
void dfs(Graph &G, bool \*vis, int x);//求与x连通的点  
void AOE\_init(Graph &G, bool \*pre, int s, int e);//重构图删除无关节点, 以求局部AOE路径

void print\_path(int path[], const Graph &G, int s); //输出最短路  
bool read\_mode(char \*filename); //选择输入模式

此外在本次编写中，我小组采用了类来进行实现，现将类说明如下

**class Graph  
{  
private:  
 int n, m; //图的顶点数和边数, 最多有MaxNodeSize个点和MaxEdgeSize条边  
 int s, e; //起点和终点的编号  
 char data[MaxNodeSize][MaxNodeNameSize]; //每个顶点的data, 名字长度最长为MaxNodeNameSize  
 Edge edges[MaxEdgeSize]; //保存图中所有的边  
 int head[MaxNodeSize]; //保存图中以该顶点为起点的最后一条边的编号  
 int tail[MaxNodeSize]; //保存图中以该顶点为终点的最后一条边的编号  
  
public:  
 void init(); //初始化图  
 void addEdge(int x, int y, int z, int tot); //链式前向星加边  
 int nsize(){ return n;} //返回图的顶点数  
 Graph reverse(); //反转边的起点和终点  
 friend bool dijkstra(Graph &G); //求最短路  
 friend bool AOE(Graph &G); //求关键路径  
 friend void dfs(Graph &G, bool \*vis, int x); //求与x连通的点  
 friend void print\_path(int path[], const Graph &G, int s); //打印路径  
};**

**class Edge  
{  
private:  
 int from, to; //边的起点和终点  
 int length; //边的长度  
 int next, nextf; //同一顶点的边的下一条边的编号  
 int etime, ltime; //边的最早和最晚开始时间  
public:  
 Edge(int f = 0, int t = 0, int l = 0, int n = 0, int et = 0, int lt = 0) : from(f), to(t), length(t), next(n), etime(et), ltime(lt) {}  
 int \_next() { return next; }  
 int \_nextf() { return nextf; }  
 int \_from() { return from; }  
 int \_to() { return to; }  
 int \_length() { return length; }  
 friend class Graph;  
};**

1. 讨论要求：
2. 多文件设计:

在该次的试验中，我小组采用了多文件编写，在过去的几次试验中，我小组均采用了cmake来辅助我们完成多文件的编写和调试。

在这次试验中，我小组将该程序头文件分为以下三类，如图，从上到下依次为图的存储相关函数，算法相关函数以及输入输出相关函数。

图形用户界面, 文本

描述已自动生成

而针对代码部分，我小组将此次的程序代码分为四个模块，除主程序外，与头文件分类相同，将该程序代码分为三部分，分别对应三个头文件内容

图形用户界面, 文本, 应用程序

描述已自动生成

## 测试设计

1. 存在最短路径和关键路径的样例

日历

中度可信度描述已自动生成

运行程序，可得到如下结果

手机屏幕的截图

中度可信度描述已自动生成

该结果正确

1. 两点相同的样例

图片包含 日历

描述已自动生成

蓝色的标志

描述已自动生成

特殊样例不会报错

1. 不存在最短路径的样例

日历

低可信度描述已自动生成

该样例中1、5之间不存在最短路径



得到无最短路径的输出，结果正确

1. 教材中的样例

按照实验报告的要求，我们选取教材中p191中的样例作为测试输入

如下：

图形用户界面, 应用程序

描述已自动生成

教材样例

图形用户界面, 文本

描述已自动生成

得到结果如下

经过对比，发现与教材中所给出的解答结果相同，故认为正确

1. 其他样例

在经过上述几种样例的测试后，基本可以肯定该程序的正确性，但是之前实验的经验告诉我们小组，不能武断的下结论，于是我们利用网络资源继续寻找样例进行测试，在测试的过程中，我们成功的发现了两种极端情况下的错误，于是我们进行了代码的修改，在修改完成后，我们小组认为我们已经完成了这次实验的要求，给出了正确的程序。

## 心得体会

在这次最后的试验中，由于我们小组各成员之间在之前几次试验中的相互讨论和合作，我们已经有了互相合作的默契，所以完成这次实验并没有遇到很多困难，在完成这次实验后，我们对本学期的几次实验进行了反思。

值得高兴的是我们小组在一次次合作中排除困难，互相帮助，最终完成了所有布置的实验，更重要的是我们收获了大量与他人合作的经验，我相信这对于我们的帮助是前所未有的，相信在之后的生活、学习和工作中，这门课程所带给我们的帮助将会潜移默化改变我们之后的行为方法。

除了收获以上所述的经验外，回顾几次试验，我们小组自然而然地提出了团队相互交流，互相同步代码等需求，通过互联网，我们小组学习了git、onedrive、以及clion等现代化工具的使用。本学期这门课程的学习，不仅学习了有关数据结构的只是，还让我对之后大规模的编写代码有了一个初步的认识，掌握了更多工具，以及对遇到困难时如何利用手边的工具解决问题的能力有了巨大的提升。本学期的数据结构课程虽然已经告一段落，但该门课带给我们小组的收获却没有止步于此。希望在以后的学习生活中能够继续再接再厉

## 提交材料说明

所提交的材料内容已经在报告中说明。