

数据结构

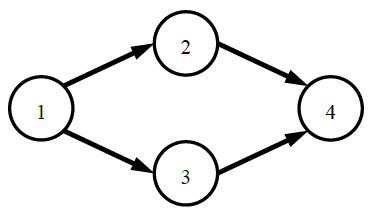
课程实验报告

|  |  |
| --- | --- |
| 题目： | 图的应用 |
| 学生姓名： | 刘大卫 |
| 学生学号： | 201826010215 |
| 专业班级： | 软件1802 |
| 完成时间： | 2019.11.27 |

1. **需求分析**
2. **问题描述**

某国的军队由N个部门组成，为了提高安全性，部门之间建立了M条通路，每条通路只能单向传递信息，即一条从部门a到部门b的通路只能由a向b传递信息。信息可以通过中转的方式进行传递，即如果a能将信息传递到b，b又能将信息传递到c，则a能将信息传递到c。一条信息可能通过多次中转最终到达目的地。

由于保密工作做得很好，并不是所有部门之间都互相知道彼此的存在。只有当两个部门之间可以直接或间接传递信息时，他们才彼此知道对方的存在。部门之间不会把自己知道哪些部门告诉其他部门。



上图中给了一个4个部门的例子，图中的单向边表示通路。部门1可以将消息发送给所有部门，部门4可以接收所有部门的消息，所以部门1和部门4知道所有其他部门的存在。部门2和部门3之间没有任何方式可以发送消息，所以部门2和部门3互相不知道彼此的存在。

　　现在请问，有多少个部门知道所有N个部门的存在。或者说，有多少个部门所知道的部门数量（包括自己）正好是N。

1. **问题分析**

首先确定本题是一个图论的习题。本题给出了起点与终点，可以看出是要求起点到终点的最短路径，然后输出最短路径中的最长的边。直接使用正反DFS可以解决问题，我们主要使用了实验5中定义的ADT完成图的存储、遍历等相关操作是很好的解决方式，只关心上层的应用，不需要关注底层的实现。

1. **输入数据**

输入的第一行包含两个整数n，m，用一个空格分隔，分别表示交通枢纽的数量和候选隧道的数量。

第2行到第m + 1行，每行包含两个整数a，b，表示枢纽a和枢纽b之间可以有一条通道。

1. **输出数据**

输出一个整数，有多少个部门所知道的部门数量（包括自己）正好是n。

1. **测试样例**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 |
| 输入 | 4 4  1 2  1 3  2 4  3 4 | 2 1  1 2 | 9 8  1 2  1 3  1 4  4 5  5 6  5 7  5 9  6 8 | 5 4  1 2  1 3  3 4  4 5 | 6 8  2 3  3 6  1 4  4 5  5 6  1 2  2 4  3 5 |
| 输出 | 2 | 2 | 1 | 1 | 4 |
| 设计目的 | 普通测试样例。（官方给出的测试样例） | 测试只有一条路的情况写能否正确解决问题。 | 测试多条路的情况，稍微加大数据的规模，看是否可以解决问题。 | 测试。 | 测试多条路的情况，同时打乱输入顺序，测试是否可以成功解决问题。 |

1. **概要设计**
2. **抽象数据类型**

为实现上述程序的功能，我们使用了实验5中完成的邻接矩阵的ADT。将输入的边存在邻接矩阵中，再使用DFS算法，深度优先搜索，当最后一个节点被访问时结束。抽象数据类型设计：

数据对象：n个节点，m条边；

数据关系：每个数据元素包含两个信息，为边的两端顶点，因为我们使用邻接矩阵进行存储，故不需要额外存储边的顶点，数组下标即为顶点。因为本题中的边是双向联通的，那么我们采取的是无向图的方法进行存储。是典型的图的数据结构。

基本操作：图的初始化，建立数组，并且建立标记数组mark（void Init(int n)）；返回图的节点数（int n()）与边数（int e()）；返回指定节点的第一条边（int first(int v)）与其下一条边（int next(int v, int w)）；边的建立（void setEdge(int v1, int v2, int wght)）；获取节点标记（int getMark(int v)）与设置节点标记（void setMark(int v, int val)）。

1. **算法的基本思想**

主要算法如下：

1. 创建sx数组变量存储节点是否被访问；
2. 进行初始化；
3. 将sx[0]初始化为0，并设置起始节点已访问；
4. 逆序图重复2）- 3）；
5. 判断是否满足条件；
6. 设置所有数组为未访问，重复1,2步；

进行上述算法则可以成功解决问题。

1. **程序的流程**

本题是简单的输入、处理与输出模型。按照“一、需求分析”中的输入输出格式进行即可。

1. **详细设计**
2. **物理数据类型**

具体的物理数据类型我们选择的是邻接矩阵表示图，我们的实现基于课本上实现的ADT，下面给出具体的物理结构：

class Graphm : public Graph {

int numVertex, numEdge; // 节点和边的数目

int \*\*matrix; // 存储边

int \*mark; // 对节点进行标记

public:

Graphm(int numVert = MAX\_SIZE) {}

~Graphm() {}

void Init(int n) {} // 初始化函数

int n() {} // 返回点数

int e() {} // 返回边数

int first(int v) {} // 返回指定节点的第一条边

int next(int v, int w) {} // 返回指定节点的下一条边

void setEdge(int v1, int v2, int wght) {} // 设置边

void delEdge(int v1, int v2) {} // 删除边

bool isEdge(int v1, int v2) {} // 判断是否是边

int weight(int v1, int v2) {} // 返回边权

int getMark(int v) {} // 获取标记

void setMark(int v, int val) {} // 设置标记

};

在此基础上进行图的定义，配合读入函数（void readIn(Graphm \*graphm, int option)）以及Prim算法（int findIt(Graphm \*graphm)）进行操作。

1. **输入输出的格式**

输入的第一行包含两个整数n，m，用一个空格分隔，分别表示交通枢纽的数量和候选隧道的数量。

第2行到第m + 1行，每行包含三个整数a，b，表示枢纽a和枢纽b之间可以有一条通道。

1. **算法的具体步骤**
2. 声明一个空图的指针Graphm \*graphm = new Graph；
3. 循环输入数据temp，对每一个数据使用graphm->setEdge(temp)，将数据插入到邻接矩阵中；
4. 接下来进行寻找最大修建天数的操作，调用findIt(Graphm \*)函数，在图的邻接矩阵中寻找最大建造天数；
5. 具体步骤参看“二、概要设计”中的“2。基本算法思想”部分，在这里不再赘述；
6. **算法的时空分析**
7. 进行邻接矩阵的建立操作时每次我们只需要直接在矩阵中赋值，所以时间复杂度为边的个数，是；
8. 进行寻找最大建造天数操作时，每次均需要对dis数组更新，并且寻找其中的最嚣张的操作，时间复杂度为；
9. 所以总时间复杂度为。
10. **调试分析**
11. **调试方案设计**

调试目的：

测试程序是否可运行，发现代码的语法错误、连接错误、逻辑错误和运算错误，是否有不严谨之处。

测试样例：

4 4

1 2

1 3

2 4

3 4

调试计划：

设定好断点，单步执行，查看graphm中元素的变化情况，在寻找最小建造天数时辅以演算，并监控sx,nx数组、mark数组的变化，确定当前的执行环节，想办法对算法进行优化。对各种情况进行完善，确定边界情况。寻找问题，检查遗漏。

设置：

在新建Graphm后设置断点，进行单步执行，观察邻接矩阵中每一个元素的变化情况，查看是否正确（主要采用输出邻接矩阵的方法进行调试工作）；针对寻找最小建造时间的处理，辅以演算，并监控sx,nx数组、mark数组的变化。及时进行修正，对代码的不完善之处进行修改。

1. **调试过程和结果以及分析**

调试时发现，输出结果不正确，发现自己输出的是，最短路径，出现了问题。之后成功解决问题，得到正确的答案。

1. **实验日志**

2019年11月26日：

基本完成BST中函数的编写工作，测试时发现了findIt中的没有输出最小建造天数的问题进行了修复；

2019年11月27日：

完成主程序的编写工作，完成文件读写的编写工作，设计完成测试样例，并且进行验证功能的完整性。

通过本题，我对于合理应用数据结构，提升算法有了更加深入的理解，也勾起了我对于数据结构学习的热情，相信我在这个学期中，一定可以学好数据结构，并且把他应用到以后的实践中去。更好的提升自己。