HW4 图状结构 实验报告

姓名：徐恒 学号：2050035 日期：2022 年 6 月 12 日

1. 涉及数据结构和相关背景

图是由顶点集合(vertex)及顶点间的关系集合组成的一种数据结构：Graph = (V, R) ，其中：V= { x | x 某个数据对象 } 是顶点的有穷非空集合；R——边的有限集合 R = {(x, y) | x, y V && Path (x, y)}无向图 或 R = {<x, y> | x, y V && Path <x, y>}。
有向图是顶点之间关系的有穷集合，也叫做边(edge)集合。Path <x, y>表示从 x 到 y 的一条单向通路，它是有方向的。x 是弧尾，y 是弧头。
有向图中：边用<x, y>表示，且 x 与 y 是有序的
a. 有向图中的边称为“弧”
b. x——弧尾或初始点 y——弧头或终端点
无向图：边用(x, y) 表示，且顶 x 与 y 是无序的
完全图：在具有 n 个顶点的有向图中，最大弧数为 n(n-1) ；在具有 n 个顶点的无向图中，最大边数为 n(n-1)/2
顶点的度
无向图：与该顶点相关的边的数目
有向图：入度 ID(v) ：以该顶点为头的弧的数目
出度 OD(v) ：以该顶点为尾头的弧的数目
路径：在图 G = (V, E) 中，若从顶点 vi 出发，沿一些边经过一些顶点 vp1, vp2, ⋯, vpm，到达顶点 vj。则称顶点序列 (vi vp1 vp2 ... vpm vj) 为从顶点 vi 到顶点 vj 的路径。它经过的边(vi, vp1)、(vp1, vp2)、...、(vpm, vj)应是属于 E 的边。
路径长度：非带权图的路径长度是指此路径上边/弧的条数；带权图的路径长度是指路径上各边/弧的权之和。
简单路径：若路径上各顶点 v1,v2,...,vm 均不互相重复，则称这样的路径为简单路径。
回路：若路径上第一个顶点 v1 与最后一个顶点 vm 重合，则称这样的路径为回路或环。
连通图与连通分量：在无向图中，若从顶点 v1 到顶点 v2 有路径，则称顶点 v1 与 v2 是连通的。如果图中任意一对顶点都是连通的，则称此图是连通图。非连通图的极大连通子图叫做连通分量。
强连通图与强连通分量：在有向图中，若对于每一对顶点 vi 和 vj，都存在一条从 vi 到 vj 和从 vj 到 vi 的路径，则称此图是强连通图。非强连通图的极大强连通子图叫做强连通分量。
生成树：一个连通图的生成树是它的极小连通子图，在 n 个顶点的情形下，有 n-1 条边。
生成树是对连通图而言的，是连通图的极小连通子图，包含图中的所有顶点，有且仅有 n-1 条边。

2. 实验内容

2.1 图的存储结构

2.1.1 问题描述

图是一种描述多对多关系的数据结构，图中的数据元素称作顶点，具有关系的两个顶点形成的一个二元组称作边或弧，顶点的集合 V 和关系的集合 R 构成了图，记作 G=（V,R）。图又分成有向图，无向图，有向网，无向网。图的常用存储结构有邻接矩阵、邻接表、十字链表、邻接多重表。图的基本操作包括图的创建、销毁、添加顶点、删除顶点、插入边、删除边、图的遍历。

2.1.2 基本要求

第 1 行输入一个数字 1~4，1 为有向图,2 为有向网,3 为无向图,4 为无向网；第 2 行输入 2 个整数 n m，分别表示顶点数和边数，空格分割；第 3 行为 n 个字符的序列，一个字符表示一个顶点；后面 m 行，若前面选择的是图，则每行输入边的两个顶点字符，空格分割，若是网，则每行输入弧的两个顶点字符和弧的权值，空格分割

2.1.3 数据结构设计

判断进行的操作对象是有向图、有向网、无向图还是无向网，并选取对应的函数对其进行邻接表和邻接矩阵的创建与输出。

2.1.4 功能说明（函数、类）

```
typedef int AdjMatrix[50][50]; //定义邻接矩阵
typedef struct {
    char vexs[50]; //顶点表
    AdjMatrix arcs; //邻接矩阵
    int vexnum,arcnum; //图的顶点数和弧数
}MGraph;
typedef struct ArcNode{
    int adjvex; //该弧所指向的顶点的位置
    struct ArcNode *nextarc;//指向下一条弧的指针
    int info;
}ArcNode; //边结点类型
typedef struct VNode{
    char data; //顶点信息
    ArcNode *firstarc; //指向第一条依附该顶点的指针
}VNode,AdjList[50];
typedef struct{
    AdjList vertices; //邻接表
    int vexnum,arcnum;
}ALGraph;
int LocateVex1(MGraph G,char u) //对 u 判断其在邻接矩阵中的位置
int LocateVex2(ALGraph G,char u) //对 u 判断其在邻接表中的位置
void CreateGraph1(MGraph &G1,ALGraph &G2) //建立有向图的邻接矩阵和邻接表
int PrintGraph1(MGraph G1,ALGraph G2) //输出有向图的邻接矩阵和邻接表
void CreateNet2(MGraph &G1,ALGraph &G2) //建立有向网的邻接矩阵和邻接表
void PrintNet2(MGraph G1,ALGraph G2) //输出有向网的邻接矩阵和邻接表
void CreateGraph3(MGraph &G1,ALGraph &G2) //建立无向图的邻接矩阵和邻接表
void PrintGraph3(MGraph G1,ALGraph G2) //输出无向图的邻接矩阵和邻接表
```

void CreateNet4(MGraph &G1,ALGraph &G2) //建立无向网的邻接矩阵和邻接表
void PrintNet4(MGraph G1,ALGraph G2) //输出无向网的邻接矩阵和邻接表

2.1.5 调试分析（遇到的问题和解决方法）

oj 中换行符与 windows 系统不同，之前采用 scanf+getchar()的操作不行，后改用 cin,自动跳过了换行符读取。

2.1.6 总结和体会

初步熟悉图和网的创建、存储、输出等操作。

2.2 图的遍历

2.2.1 问题描述

本题给定一个无向图，用邻接表作存储结构，用 dfs 和 bfs 找出图的所有连通子集。所有顶点用 0 到 n-1 表示，搜索时总是从编号最小的顶点出发。使用邻接矩阵存储，或者邻接表（使用邻接表时需要使用尾插法）。

2.2.2 基本要求

第 1 行输出 dfs 的结果
第 2 行输出 bfs 的结果
连通子集输出格式为{v11 v12 ...}{v21 v22 ..}... 连通子集内元素之间用空格分割，子集之间无空格,'{'和子集内第一个数字之间、'}'和子集内最后一个元素之间、子集之间均无空格
对于 20%的数据，有 0<n<=15;
对于 40%的数据，有 0<n<=100;
对于 100%的数据，有 0<n<=10000;
对于所有数据，0.5n<=m<=1.5n，保证输入数据无错。

2.2.3 数据结构设计

建立邻接矩阵，分布对其广度和深度遍历并输出。

2.2.4 功能说明（函数、类）

int InitQueue(SqQueue &Q,int n) //对长度为 n 的队列 Q 初始化
void CreateALGraph_adjlist(ALGraph &G) //创建邻接表 G
void BFS(ALGraph G,int v) //G 从 v 出发广度遍历
void BFSTraverse(ALGraph G) //对邻接表 G 广度遍历
void DFS(ALGraph G,int v,int check) //从 v 出发对 G 深度遍历，check 用于帮助保证输出格式
void DFSTraverse(ALGraph G) //对邻接表 G 深度遍历

2.2.5 调试分析（遇到的问题和解决方法）

2.2.6 总结和体会

对 DFS 和 BFS 算法有了更深刻的认识和理解，算法中一些细节在具体实施时需要作更改，体会更深。

2.3 关键路径

2.3.1 问题描述

一个工程项目由一组子任务（或称活动）构成，子任务之间有的可以并行执行，有的必须在完成了其它一些子任务后才能执行，并且每个任务完成需要一定的时间。

对于一个工程，需要研究的问题是：

（1）由这样一组子任务描述的工程是否可行？

（2）若可行，计算完成整个工程需要的最短时间。

（3）这些任务中，哪些任务是关键活动（也就是必须按时完成的任务，否则整个项目就要延迟）。

现将这样一个工程项目用一个有向图表示，给定一组顶点，每个顶点表示任务之间的交接点（若任务 2 要在任务 1 完成后才可以开始，则这两任务之间必须有一个交接点，该点称作事件）。任务用有向边表示，边的起点是该任务可以开始执行的事件，终点是该任务已经完成的事件，边上的权值表示该任务完成需要执行的时间。

2.3.2 基本要求

输入第一行包含两个整数 n、m，其中 n 表示顶点数，m 表示任务数。顶点按 1~n 编号。接下来 m 行表示 m 个任务，每行包含三个正整数 ui、vi、wi，分别表示该任务开始和完成的顶点序号，及任务完成的时间。
整数之间用空格表示。
对于 20%的数据，有 0<n<=10
对于 40%的数据，有 0<n<=100
对于 100%的数据，有 0<n<=100000
对于所有数据，1.5n<=m<=2n，wi 是一个 1 到 100 的整数，保证输入的边按起点从小到大排序，起点相同的边按终点从小到大排序。
如果该有向图有环，则工程不可行，输出 0；否则
第 1 行输出完成整个工程项目需要的时间，
从第 2 行开始输出所有关键活动，每个关键活动占一行，按格式“ui->vi”输出，其中 u 和 v 为该任务开始和完成涉及的交接点编号。
注：关键活动输出的顺序规则是：按起点从小到大排序，起点相同的边按终点从小到达排序。

2.3.3 数据结构设计

建立邻接表 G，并初始化两个栈 T、S，其中 T 存放 G 拓扑排序后的顶点列，S 为零入度栈，判断该活动目前是否可以开始。通过拓扑排序，计算事件的最早最晚开始时间，活动的最早最迟开始时间，进而判断是否为关键路径。

2.3.4 功能说明（函数、类）

int InitStack(SqStack &S,int n) //将队列初始化长度为 n
int push(SqStack &S,int e) //将元素 e 弹入栈 S 中
int pop(SqStack &S,int &e) //弹出栈顶元素，并将值返回 e 中
int CreateGraph(ALGraph &G) //建立邻接表 G
int CriticalPath(ALGraph G,SqStack &T) //求 G 的关键路径，其中 T 存放 G 的拓扑序列的

顶点栈

2.3.5 调试分析（遇到的问题和解决方法）

未考虑到输入数据中会输入起终点相同但权值不同的路径，后在建立邻接表时采用 while 判断是否已存在来处理。若已存在该路径，则只需更新边的权值，不然则正常建立一条边。

2.3.6 总结和体会

2.4 最短路径

2.4.1 问题描述

本题给出一张交通网络图，列出了各个城市之间的距离。请计算出从某一点出发到所有点的最短路径长度。

2.4.2 基本要求

输入第一行包含三个整数 n、m、s，分别表示 n 个顶点、m 条无向边、出发点的编号。接下来 m 行，每行包含三个整数 ui、vi、wi，其中 1<=ui,vi<=n, 1<=wi<=1000，分别表示第 i 条无向边的出发点、目标点和长度。顶点编号从 1 开始。

输出一行，包含 n 个用空格分隔的整数，其中第 i 个整数表示从点 s 出发到点 i 的最短路径长度（若 s=i 则最短路径长度为 0，若从点 s 无法到达点 i，则最短路径长度为 2147483647，用 INT_MAX 表示）。

2.4.3 数据结构设计

根据给出图的结构，构建邻接表，运用迪杰斯特拉(Dijkstra)算法求出 v0 出发到各点的最短路径。

2.4.4 功能说明（函数、类）

int CreateGraph(ALGraph &G) //根据图的结构建立邻接表 G
int arcs(int v1,int v2) //返回 v1->v2 的距离
typedef int ShortPathDistance[10000]; //最短路径长度
typedef int ShortPathTable[10000]; //最短路径数组
void ShortestPath_DIJ(ALGraph G,int v0, ShortPathTable & pre,ShortPathDistance &D) //求从 v0 出发到各店的最短路径，D 中存放最短路径长度，pre 中存放最短路径数组

2.4.5 调试分析（遇到的问题和解决方法）

程序运行效率低，对 n=1000000 的数据运行超时，有待改进。

2.4.6 总结和体会

主要是对迪杰斯特拉(Dijkstra)算法的应用，但题中不能使用邻接矩阵，因为对 n=1000000 的数据需占用 int[1000000][1000000]的内存，过大。故采用了邻接表，但运行效率过低，究其原因在于不能直接得到边值，需要循环查找，有待改善。