



# 南昌大学实验报告

学生姓名： 丁俊      学 号： 8003119100      专业班级： 信安 193 班

实验类型： ☒ 验证 ☐ 综合 ☐ 设计 ☐ 创新      实验日期： 3.11      实验成绩：       

## 一、实验项目名称

图的运用

## 二、实验目的

掌握对图的建立方法遍历和最小生成树的 prim 算法

## 三、实验任务

- 1、编写程序输出以邻接表为存储结构的无向图的各项点的度。
- 2、图采用邻接表存储结构，编程对图进行深度优先遍历。
- 3、无向图采用邻接矩阵存储结构，编程实现 Prim 求解最小生成树的算法。

## 四、主要仪器设备及耗材

**Dec++5.15      windows10**

## 五、实验步骤

### 1、输出以邻接表为存储结构的无向图的各项点的度

代码

```
1. #include "ljb.h"
2.
3. void degree(LinkedGraph g) {
4.     EdgeNode *p;
5.     int i;
6.     for (i = 0; i < g.n; i++) {
7.         printf("%c's degree is ", g.adjlist[i].vertex);
8.         p = g.adjlist[i].FirstEdge;
9.         int cnt = 0;
10.        while (p) {
11.            cnt++; // 遍历邻接的边记录数量
12.            p = p->next;
13.        }
14.        printf("%d\n", cnt);
15.    }
16.}
17.
18.int main() {
19.    LinkedGraph g;
20.    creat(&g, "g11.txt", 0); // g11 中存储了图的信息
21.    printf("\n The graph is:\n");
22.    print(g);
23.    degree(g);
24.    return 0;
25.}
```

当“g11.txt”中的内容如下：



```
g11.txt - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
4 5
0 1 2 3
0 1
0 2
0 3
1 2
2 3
}
```

表明有 4 个顶点和 5 条边,0~3 为四个顶点的编号,后面 5 行表示一条边的信息,表示起点、终点有一条边。

## 2、图采用邻接表存储结构,编程对图进行深度优先遍历

### 代码

```
1. // 采用邻接表存储结构,对图进行深度优先遍历
2. #include "ljb.h"
3. int visited[M];
4.
5. void dfs(LinkedGraph g, int i) {
6.     EdgeNode *p;
7.     printf("visit vertex: %c \n", g.adjlist[i].vertex);
8.     visited[i] = 1;
9.     p = g.adjlist[i].FirstEdge;
10.    while (p) { // 遍历该点的邻接表,一直把能够和该点连通的点一次性遍
        历
11.        if (!visited[p->adjvex])
12.            dfs(g, p->adjvex);
13.        p = p->next;
14.    }
15.}
16.
17.void DfsTraverse(LinkedGraph g) {
18.    int i;
19.    for (i = 0; i < g.n; i++)
20.        visited[i] = 0;    /*初始化标志数组*/
21.    for (i = 0; i < g.n; i++)
22.        if (!visited[i]) /*vi 未访问过*/
23.            dfs(g, i);
24.}
25.
26.int main() {
27.    LinkedGraph g;
28.    creat(&g, "g11.txt", 0);
29.    printf("\n The graph is:\n");
30.    print(g);
31.    printf("深度优先遍历序列为: \n");
32.    DfsTraverse(g);
33.    return 0;
34.}
```

**过程：**把图中每个顶点作为出发点(如果没有被访问过)，使用 dfs 遍历它所能到达的顶点，先输出到达的点的信息并标记为访问过，然后继续以该点为起点向下搜索递归直到没有相邻的点，也即邻接表的结点为空。

### 3、最小生成树

代码

```
1. #include "ljjz.h"
2.
3. typedef struct edgedata {
4.     int beg, en;
5.     int length;
6. } edge;
7.
8. // n 个顶点 n-1 条边(0 到 n-2)
9. void prim(Mgraph g, edge tree[M - 1]) { // tree 数组表示的动态(表示
    两点之间的 d)的表格
10.     edge x;
11.     int d, min, j, k, s, v;
12.
13.     for (v = 1; v <= g.n - 1; v++) {
14.         tree[v - 1].beg = 0;
15.         tree[v - 1].en = v;
16.         tree[v - 1].length = g.edges[0][v];
17.     }
18.
19.     for (k = 0; k <= g.n - 3; k++) {
20.         min = tree[k].length;
21.         s = k;
22.
23.         for (j = k + 1; j <= g.n - 2; j++)
24.             if (tree[j].length < min) {
25.                 min = tree[j].length;
26.                 s = j; // 找到最短的两西边
27.             }
28.         v = tree[s].en;
29.         x = tree[s];
30.         tree[s] = tree[k];
31.         tree[k] = x; // x 只是一个中间变量用于交换,把最小的两西边
32.         // 放到还未选的边最前面
```

```

33.         for (j = k + 1; j <= g.n - 2; j++) { // 由于新顶点的加入修
           改两西边的信息
34.             d = g.edges[v][tree[j].en]; // v 是新加入的顶点, 从 v 出发
           到其他顶点的距离
35.             // 遍历 tree 数组的终点 en, 看从 v 出发到达 en 的距离有没有更
           小的
36.             if (d < tree[j].length) {
37.                 tree[j].length = d;
38.                 tree[j].beg = v; // 如果有则修改其起点为 v
39.             }
40.         }
41.     }
42.     printf("\n The minimum cost spanning tree is:\n"); // 输出最小
           生成树
43.     for (j = 0; j <= g.n - 2; j++) {
44.         printf("\n%c---%c %d\n", g.vexs[tree[j].beg], g.vexs[tree
           [j].en], tree[j].length);
45.         printf("\nThe root of it is %c\n", g.vexs[0]);
46.     }
47. }
48.
49. int main() {
50.     Mgraph g;
51.     edge tree[M - 1];
52.     char filename[20];
53.     printf("please input filename of Graph:\n");
54.     gets(filename);
55.     creat(&g, filename, 0); // 无向图邻接矩阵
56.     print(g);
57.     prim(g, tree);
58.     return 0;
59. }

```

**过程:** 建立一个 tree 数组用来存储最小生成树的边, 程序开始先把 0 顶点到其他顶点的距离信息存储到 tree 中, 找出最短的两栖边, 每次循环即是找出最短的两栖边, 此时下标+1 就表示接着处理下一条生成树的边。每次加入一条新的边即一个新的顶点 v, 都要判断从 v 顶点到 tree 数组中边的 en(终点)的距离是否可以更小, 如果可以缩短, 就更新 tree 数组该边的 beg(起点)为 v 结点, 注意每次找到最短的两栖边时, 要把该边提前做出交换到还未做出选择的边集的最前面。

## 六、实验数据以及处理结果

## 1、各顶点的度

```
D:\gitworkspace\fresh\数据结构\实验8\lab8_01.exe

The graph is:
0-->3-->2-->1
1-->2-->0
2-->3-->1-->0
3-->2-->0
0's degree is 3
1's degree is 2
2's degree is 3
3's degree is 2

-----
Process exited after 0.09425 seconds with return value 0
请按任意键继续. . .
```

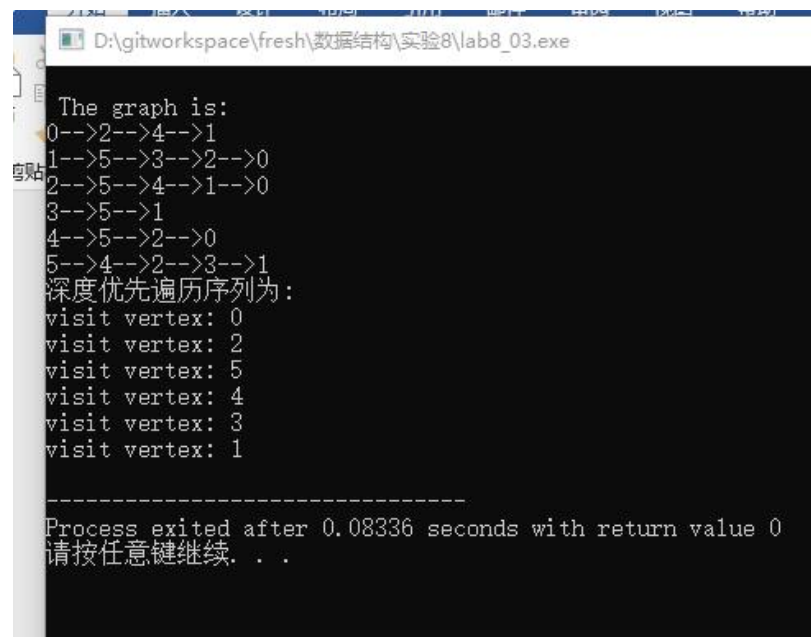
从图的邻接链表可以看出，0 顶点连接了 3 条边，所以度数为 3；其他顶点同理。

## 2、对图进行深度优先遍历

当我们把文件中的数据换成如下：

```
g11.txt - 记事本
文件(F) 编辑(E) 格式(O)
5 10
0 1 2 3 4 5
0 1
0 4
0 2
1 2
1 3
1 5
3 5
2 4
2 5
4 5
```

遍历结果为



```
D:\gitworkspace\fresh\数据结构\实验8\lab8_03.exe

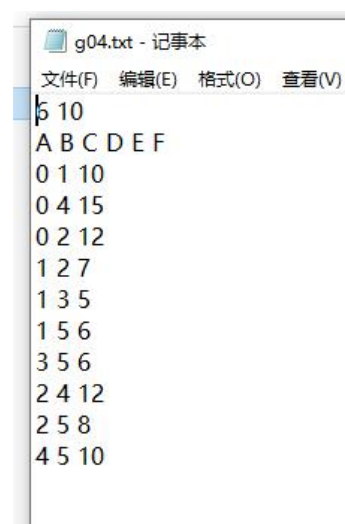
The graph is:
0-->2-->4-->1
1-->5-->3-->2-->0
2-->5-->4-->1-->0
3-->5-->1
4-->5-->2-->0
5-->4-->2-->3-->1
深度优先遍历序列为:
visit vertex: 0
visit vertex: 2
visit vertex: 5
visit vertex: 4
visit vertex: 3
visit vertex: 1

-----
Process exited after 0.08336 seconds with return value 0
请按任意键继续. . .
```

过程：先从 **0** 顶点为起点开始遍历，下一个顶点是 **2**，从 2 开始为起点的下一个顶点是 **5**，以 5 为起点的下一个顶点是 **4**，从 4 出发下一个顶点是 5(已访问),2(已访问),0(已访问)。回溯到以 5 为起点的下一个邻接点此时为 2(已访问),然后是 **3** 和 **1**，至此所有顶点访问完毕。

### 3、最小生成树

g04.txt 文件中的信息为：



```
g04.txt - 记事本
文件(F) 编辑(E) 格式(O) 查看(V)
6 10
A B C D E F
0 1 10
0 4 15
0 2 12
1 2 7
1 3 5
1 5 6
3 5 6
2 4 12
2 5 8
4 5 10
```

6 个顶点 10 条边,分别为 ABCDEF.

结果如下

```
please input filename of Graph:
g04.txt
A---->B : 10
A---->C : 12
A---->E : 15
B---->C : 7
B---->D : 5
B---->F : 6
C---->E : 12
C---->F : 8
D---->F : 6
E---->F : 10
```

The minimum cost spanning tree is:

A---B 10

The root of it is A

B---D 5

The root of it is A

B---F 6

The root of it is A

B---C 7

The root of it is A

F---E 10

The root of it is A

七、思考讨论题或体会或对改进实验的建议



通过这次实验我更加地体会到图数据结构的巧妙和灵活,也学会了怎么自主利用数据结构来解决一些实际问题。

## 八、参考资料