

## 数据结构实验报告

学号-姓名	桑龙龙 20030540015	实验时间	2020 年 12 月 19 日
诚信声明	本实验及实验报告所写内容为本人所作 		
实验题目	图的遍历 题目一 深度优先遍历 题目二 广度优先遍历		
实验过程中遇到的主要问题	无		
实验小结	本次试验进行了图的遍历的试验，在本次实验中分别用邻接矩阵和邻接链表的方法实现图的构造，并分别进行了图的深度优先遍历和广度优先遍历。		
数据结构 (自定义数据类型)	<pre> 1.  struct adjMatrics{ 2.      //邻接矩阵 3.      int** g; 4.      //矩阵 5.      int vertexNum; 6.      //节点数 7.      int edgeNum; 8.      //边数 9.      adjMatrics(int n){ 10.         vertexNum=n; 11.         g=new int*[n+1]; 12.         for(int i=1;i&lt;=n;i++){ 13.             g[i]=new int[n+1]; 14.             memset(g[i],0,sizeof(int)*(n+1)); 15.         } 16.     } 17.     ~adjMatrics(){ 18.         for(int i=0;i&lt;vertexNum;i++) delete []g[i]; 19.         delete []g; 20.     } 21.     void add(int from,int to,int weight){ 22.         //加入一条权重为 weight 由 from 到 to 的边 23.         g[from][to]=weight; 24.     } 25. }; </pre>		

```

1. struct adjList{
2.     //邻接链表
3.     int vertexNum;
4.     int edgeNum;
5.     int* head;
6.     struct edge{
7.         int to;
8.         int weight;
9.         int next;
10.    }e[N];
11.    int tot;
12.    //静态链表实现的邻接链表
13.    adjList(int n){
14.        vertexNum=n;
15.        head=new int[n+1];
16.        memset(head,0,(n+1)*sizeof(int));
17.        memset(e,0,sizeof(e));
18.        tot=0;
19.    }
20.    adjList(){
21.        delete []head;
22.    }
23.    void add(int from,int to,int weight){
24.        //加入一条权重为weight 由from 到 to 的边
25.        e[++tot].to=to;
26.        e[tot].weight=weight;
27.        e[tot].next=head[from];
28.        head[from]=tot;
29.    }
30. };

```

主要算法  
(或算法说明)

```
1. #include <stdlib.h>
2. #include <stdio.h>
3. #include <string.h>
4. #include <queue>
5. #define N 10000
6. struct adjMatrics{
7.     //邻接矩阵
8.     int** g;
9.     //矩阵
10.    int vertexNum;
11.    //节点数
12.    int edgeNum;
13.    //边数
14.    adjMatrics(int n){
15.        vertexNum=n;
16.        g=new int*[n+1];
17.        for(int i=1;i<=n;i++){
18.            g[i]=new int[n+1];
19.            memset(g[i],0,sizeof(int)*(n+1));
20.        }
21.    }
22.    ~adjMatrics(){
23.        for(int i=0;i<vertexNum;i++) delete []g[i];
24.        delete []g;
25.    }
26.    void add(int from,int to,int weight){
27.        //加入一条权重为 weight 由 from 到 to 的边
28.        g[from][to]=weight;
29.    }
30. };
31. struct adjList{
32.     //邻接链表
33.     int vertexNum;
34.     int edgeNum;
35.     int* head;
36.     struct edge{
37.         int to;
38.         int weight;
39.         int next;
40.     }e[N];
41.     int tot;
42.     //静态链表实现的邻接链表
43.     adjList(int n){
44.         vertexNum=n;
45.         head=new int[n+1];
```

```

46.     memset(head,0,(n+1)*sizeof(int));
47.     memset(e,0,sizeof(e));
48.     tot=0;
49. }
50. adjList(){
51.     delete []head;
52. }
53. void add(int from,int to,int weight){
54.     //加入一条权重为 weight 由 from 到 to 的边
55.     e[++tot].to=to;
56.     e[tot].weight=weight;
57.     e[tot].next=head[from];
58.     head[from]=tot;
59. }
60. };
61. bool vis[N];
62. void bfsAdjMatrics(int cur,adjMatrics& graph){
63.     //广度优先遍历（邻接矩阵），由点 cur 初始
64.     memset(vis+1,0,graph.vertexNum);
65.     std::queue<int> Q;
66.     Q.push(cur);
67.     vis[cur]=true;
68.     printf("sAdjMatrics BFS:\n");
69.     while(!Q.empty()){
70.         cur=Q.front();
71.         Q.pop();
72.         printf("%d ",cur);
73.         for(int i=1;i<=graph.vertexNum;i++){
74.             if(vis[i] || !graph.g[cur][i]) continue;
75.             Q.push(i);
76.             vis[i]=true;
77.         }
78.     }
79.     printf("\n");
80. }
81. void dfsAdjMatrics(int cur,adjMatrics& graph){
82.     //深度优先遍历（邻接矩阵）
83.     if(vis[cur]) return;
84.     printf("AdjMatrics dfs vis: %d\n",cur);
85.     vis[cur]=true;//标记为访问过
86.     for(int i=1;i<=graph.vertexNum;i++){
87.         if(!graph.g[cur][i]) continue;
88.         dfsAdjMatrics(i,graph);
89.     }
90.     vis[cur]=false;//取消访问标记

```

```

91. }
92. void bfsAdjList(int cur,adjList& graph){
93.     //广度优先遍历（邻接链表），由点 cur 初始
94.     memset(vis+1,0,graph.vertexNum);
95.     std::queue<int> Q;
96.     Q.push(cur);
97.     vis[cur]=true;
98.     printf("adjList BFS:\n");
99.     while(!Q.empty()){
100.         cur=Q.front();
101.         Q.pop();
102.         printf("%d ",cur);
103.         for(int i=graph.head[cur];i;i=graph.e[i].next){
104.             if(vis[graph.e[i].to]) continue;
105.             Q.push(graph.e[i].to);
106.             vis[graph.e[i].to]=true;
107.         }
108.     }
109.     printf("\n");
110. }
111. void dfsAdjList(int cur,adjList& graph){
112.     //深度优先遍历（邻接链表）
113.     if(vis[cur]) return;
114.     printf("AdjList dfs vis: %d\n",cur);
115.     vis[cur]=true;
116.     for(int i=graph.head[cur];i;i=graph.e[i].next){
117.         dfsAdjList(graph.e[i].to,graph);
118.     }
119.     vis[cur]=false;
120. }
121. adjMatrics* buildAdjMatrics(){
122.     //初始化一个邻接矩阵
123.     int n,m,from,to;
124.     scanf("%d %d\n",&n,&m);
125.     adjMatrics* graph=new adjMatrics(n);
126.     graph->edgeNum=m;
127.     for(int i=0;i<m;i++){
128.         scanf("%d %d\n",&from,&to);
129.         graph->add(from,to,1);
130.     }
131.     return graph;
132. }
133. adjList* buildAdjList(){
134.     //初始化一个邻接链表
135.     int n,m,from,to;

```

```

136.     scanf("%d %d\n",&n,&m);
137.     adjList* graph=new adjList(n);
138.     graph->edgeNum=m;
139.     for(int i=0;i<m;i++){
140.         scanf("%d %d\n",&from,&to);
141.         graph->add(from,to,1);
142.     }
143.     return graph;
144. }
145. /* 输入说明:
146.     第一行两个数 n, m
147.     之后 m 行, 每行 3 个数 from, to, weight
148.     解释:
149.     n 为节点个数 (节点编号由 1 到 n), m 为边数
150.     from, to, weight 表示一条由 from 到 to 的权重为 weight 的边
151.
152. */
153. int main(){
154.     if(0){
155.         //邻接链表测试
156.         adjList* g=buildAdjList();
157.         dfsAdjList(1,*g);
158.         bfsAdjList(1,*g);
159.     }else{
160.         //邻接矩阵测试
161.         adjMatrics* g=buildAdjMatrics();
162.         dfsAdjMatrics(1,*g);
163.         bfsAdjMatrics(1,*g);
164.     }
165.
166. }

```