# 北京邮电大学软件学院

# 2023-2024学年第1学期实验报告

**课程名称： 数据结构**

**实验名称： 注意：这里填写每个实验的名称**

**实验完成人：**

**姓名：**\_\_\_\_禄禄鱼\_\_\_\_**学号：**\_\_\_\_\_\_\_\_**成绩：**\_\_\_\_\_\_\_\_

**指导教师：**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**（封面 5%）**

**日 期： 2023 年 11 月 27 日**

1. **实验目的**

（说明通过本实验希望达到的目的5%）

理解图的基本概念和邻接表存储结构。

掌握无向图的深度优先搜索（DFS）和广度优先搜索（BFS）算法。

使用C语言编程实现图的遍历算法。

学习递归算法在图遍历中的应用。

1. **实验内容**

（说明本实验的内容5%）

**[问题描述]**

　　很多涉及图上操作的算法都是以图的遍历操作为基础的。试写一个程序，演示无向图的遍历操作。

**[基本要求]**

　　以邻接表为存储结构，实现连通无向图的深度优先和广度优先遍历。以用户指定的结点为起点，分别输出每种遍历下的结点访问序列和相应生成树的边集。

**[测试数据]**

　　由学生依据软件工程的测试技术自己确定。注意测试边界数据，如单个结点。

**[实现提示]**

　　设图的结点不超过30个，每个结点用一个编号表示（如果一个图有n个结点，则它们的编号分别为1,2,…,n）。通过输入图的全部边输入一个图，每条边为一个数对，可以对边的输入顺序做出某种限制。注意，生成树的边是有向边，端点顺序不能颠倒。

1. **实验环境**

（说明本实验需要的环境5%）

Clion Windows10

1. **实验过程和实验结果**

（问题分析，设计方案、算法、设计图、程序、调试过程截图、运行结果截图等70%）

实验代码

#include <stdio.h>  
#include <stdlib.h>  
  
#define MAX\_NODES 30  
  
// 结构体表示邻接表中的一个节点  
struct Node {  
 int data**;** struct Node\* next**;**}**;**// 结构体表示每个节点的邻接表  
struct AdjList {  
 struct Node\* head**;**}**;**// 结构体表示图  
struct Graph {  
 int numNodes**;** struct AdjList\* array**;**}**;**// 函数创建一个新节点  
struct Node\* createNode(int data) {  
 struct Node\* newNode = (struct Node\*)malloc(sizeof(struct Node))**;** newNode->data = data**;** newNode->next = NULL**;** return newNode**;**}  
  
// 函数创建一个带有给定节点数量的图  
struct Graph\* createGraph(int numNodes) {  
 struct Graph\* graph = (struct Graph\*)malloc(sizeof(struct Graph))**;** graph->numNodes = numNodes**;** // 为邻接表数组分配内存  
 graph->array = (struct AdjList\*)malloc(numNodes \* sizeof(struct AdjList))**;** // 将每个邻接表初始化为空  
 for (int i = 0**;** i < numNodes**;** ++i)  
 graph->array[i].head = NULL**;** return graph**;**}  
  
// 函数向无向图中添加一条边  
void addEdge(struct Graph\* graph**,** int src**,** int dest) {  
 // 从src到dest添加一条边  
 struct Node\* newNode = createNode(dest)**;** newNode->next = graph->array[src].head**;** graph->array[src].head = newNode**;** // 由于图是无向的，也要从dest到src添加一条边  
 newNode = createNode(src)**;** newNode->next = graph->array[dest].head**;** graph->array[dest].head = newNode**;**}  
  
// 深度优先搜索的递归实现  
void DFSUtil(struct Graph\* graph**,** int node**,** int visited[]) {  
 visited[node] = 1**;** printf("%d "**,** node)**;** struct Node\* temp = graph->array[node].head**;** while (temp != NULL) {  
 if (!visited[temp->data])  
 DFSUtil(graph**,** temp->data**,** visited)**;** temp = temp->next**;** }  
}  
  
// 执行深度优先搜索  
void DFS(struct Graph\* graph**,** int startNode) {  
 int visited[MAX\_NODES] = {0}**;** // 初始化访问数组  
 printf("深度优先遍历: ")**;** DFSUtil(graph**,** startNode**,** visited)**;** printf("\n")**;**}  
  
// 广度优先搜索的递归实现  
void BFSUtil(struct Graph\* graph**,** int queue[]**,** int front**,** int rear**,** int visited[]) {  
 if (front >= rear)  
 return**;** int currentNode = queue[front++]**;** printf("%d "**,** currentNode)**;** struct Node\* temp = graph->array[currentNode].head**;** while (temp != NULL) {  
 if (!visited[temp->data]) {  
 visited[temp->data] = 1**;** queue[rear++] = temp->data**;** }  
 temp = temp->next**;** }  
  
 BFSUtil(graph**,** queue**,** front**,** rear**,** visited)**;**}  
  
// 执行广度优先搜索  
void BFS(struct Graph\* graph**,** int startNode) {  
 int visited[MAX\_NODES] = {0}**;** // 初始化访问数组  
 int queue[MAX\_NODES]**;** int front = 0**,** rear = 0**;** visited[startNode] = 1**;** printf("广度优先遍历: ")**;** // printf("%d ", startNode);  
 queue[rear++] = startNode**;** BFSUtil(graph**,** queue**,** front**,** rear**,** visited)**;** printf("\n")**;**}  
  
int main() {  
 setbuf(stdout**,** 0)**;** int numNodes**,** numEdges**,** startNode**;** printf("输入节点数量: ")**;** scanf("%d"**,** &numNodes)**;** struct Graph\* graph = createGraph(numNodes)**;** printf("输入边的数量: ")**;** scanf("%d"**,** &numEdges)**;** printf("输入边（作为节点对）:\n")**;** for (int i = 0**;** i < numEdges**;** ++i) {  
 int src**,** dest**;** scanf("%d %d"**,** &src**,** &dest)**;** addEdge(graph**,** src**,** dest)**;** }  
  
 printf("输入遍历的起始节点: ")**;** scanf("%d"**,** &startNode)**;** DFS(graph**,** startNode)**;** BFS(graph**,** startNode)**;** return 0**;**}

实验结果

测试数据为该图

1

0

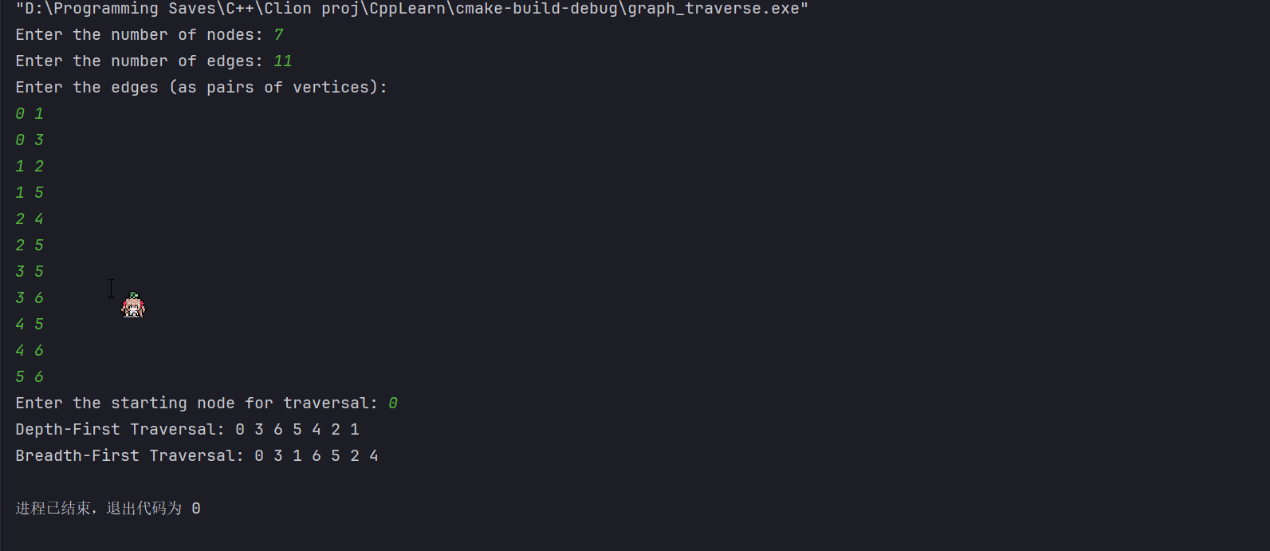
3

6

5

4

2



1. **实验心得**

（10%）

通过本次实验，我深刻理解了图的基本概念和遍历算法。编写C程序实现深度优先搜索和广度优先搜索，加深了对递归在图算法中的应用的理解。通过手动输入图的边和节点，我成功验证了算法的正确性。这次实验提高了我的编程能力和对数据结构的掌握，为今后解决更复杂问题打下了基础。