数据结构实验报告——实验八

学号：__20201060330__姓名：___胡诚皓___得分：_____

一、实验目的

1. **复习图的逻辑结构、存储结构及基本操作；**
2. **掌握邻接矩阵、邻接表及图的创建、遍历；**
3. **了解图的应用。**

二、实验内容

1. （必做题）图的基本操作

假设图中数据元素类型是字符型，请采用邻接矩阵实现图的以下基本操作：

(1) 构造图（包括有向图、有向网、无向图、无向网）；

(2) 根据深度优先遍历图；

(3) 根据广度优先遍历图。

三、数据结构及算法描述

1. （必做题）图的基本操作

数据结构

定义了枚举类型 GraphKind，可选值为 DG、DN、UDG、UDN，分别对应有向图、有向网、无向图、无向网。宏定义 INFINITY 为 INT_MAX 作为邻接矩阵中表示距离时的无穷大，同时定义 MAX_VERTEX_NUM 为 20，作为最大支持的顶点个数。

使用 int 作为顶点关系 VRType 的类型，此处不妨令顶点本身的类型 VertexType 也为 int。以 char*为表示边信息的类型，可以为顶点之间的关系附加一些信息，在代码中由于是抽象的关系，边信息都赋值为了 NULL。

AcrCell 作为邻接矩阵中每个元素的类型，包括了用于描述顶点关系的 VRType 类型变量 adj 与存储边或弧的相关附加信息 InfoType 类型的指针 info。

Mgraph 为图本身的类型，包括 VertexType 类型的顶点集 vexs，代表边集的邻接矩阵 arcs，顶点数、边或弧数 vexnum 和 arcnum，存储图具体类型的 kind。

另外，使用了和前几次相同的链队列，将队列相关的基本操作放在 queue.c 中，以 queue.h 为其头文件。

算法描述

int main()

定义 Mgraph 类型的变量 graph 用于存储要输入的图，code 为临时变量，用于临时存储图的类型。在读入用户要输入的图的类型后，调用对应的构建函数构建图，再分别调用

deepTraverseMap 和 breadthTraverseMap 输出图的深度优先遍历和广度优先遍历。

Boolean constructDG(Mgraph *mgraph)
此函数用于根据用户输入构建有向图。先读入图的顶点数和弧数，再按照“起点 终点”的格式读取有向图的各条弧，在输入弧的过程中，若起点或终点的顶点编号超过了之前输入的顶点数，将会要求用户重新输入。
在图构建完成后，会输出整个邻接矩阵供预览。（0 代表没有弧、1 代表有弧）

Boolean constructDN(Mgraph *mgraph)
此函数用于根据用户输入构建有向网。先读入网的顶点数和弧数，再按照“起点 终点 权值”的格式读取有向网的各条弧，在输入弧的过程中，若起点或终点的顶点编号超过了之前输入的顶点数，将会要求用户重新输入。
在图构建完成后，会输出整个邻接矩阵供预览。（INF 表示无法直接到达）

Boolean constructUDG(Mgraph *mgraph)
此函数用于根据用户输入构建无向图。先读入图的顶点数和边数，再按照“起点 终点”的格式读取无向图的各条边，在输入边的过程中，若起点或终点的顶点编号超过了之前输入的顶点数，将会要求用户重新输入。
在图构建完成后，会输出整个邻接矩阵供预览。（0 代表没有弧、1 代表有弧）

Boolean constructUDN(Mgraph *mgraph)
此函数用于根据用户输入构建无向网。先读入网的顶点数和边数，再按照“起点 终点 权值”的格式读取无向网的各条边，在输入边的过程中，若起点或终点的顶点编号超过了之前输入的顶点数，将会要求用户重新输入。
在图构建完成后，会输出整个邻接矩阵供预览。（INF 表示无法直接到达）

void dfs(Boolean visited[], Mgraph mgraph, int index)
作为深度优先遍历的递归函数，按以下步骤执行
①访问当前顶点 index，即在将当前顶点在 visited 中标记为 TRUE，并输出当前顶点的编号
②使用 for 循环，对于满足条件的未访问邻接点进行递归访问

void deepTraverseMap(Mgraph mgraph)
对 mgraph 进行深度优先遍历，先声明一个初始值均为 0（即 FALSE）的 visited 数组来记录各顶点是否被访问过，用 for 循环来保证每个顶点都被遍历到（在图非连通的情况下）。

void breadthTraverseMap(Mgraph mgraph)
对 mgraph 进行广度优先遍历，同样先声明一个初始值均为 0（即 FALSE）的 visited 数组来

记录各顶点是否被访问过。声明局部变量 cur 来存储当前访问的顶点编号以及 queue 来实现广度优先遍历，按以下步骤执行：

①将下标为 0 的顶点入队

②只要队列不为空，就出队一个顶点，保存在 cur 中。若 cur 已经访问过，就不再访问；若没有访问过，则访问之，即在 visited 中标注并输出

③将当前顶点满足条件的未访问邻接点一一入队

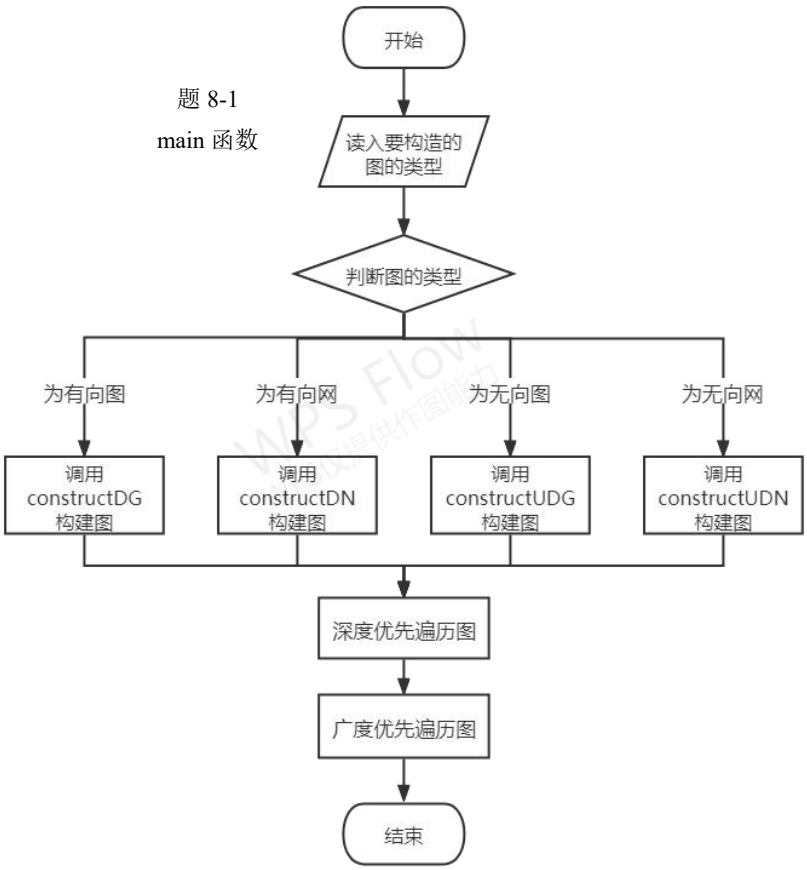
④若此时队列为空，用 for 循环找是否有未访问过的顶点，将未访问过的顶点入队（处理非连通图）。转到②

四、详细设计

1. （必做题）图的基本操作

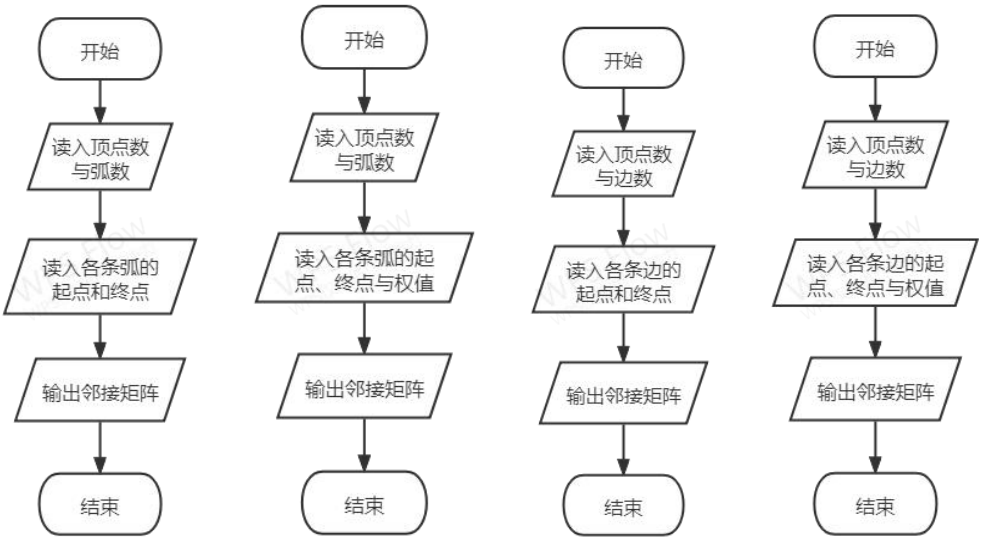
题 8-1

main 函数



题 8-1

constructDG、constructDN、constructUDG、constructUDN 函数

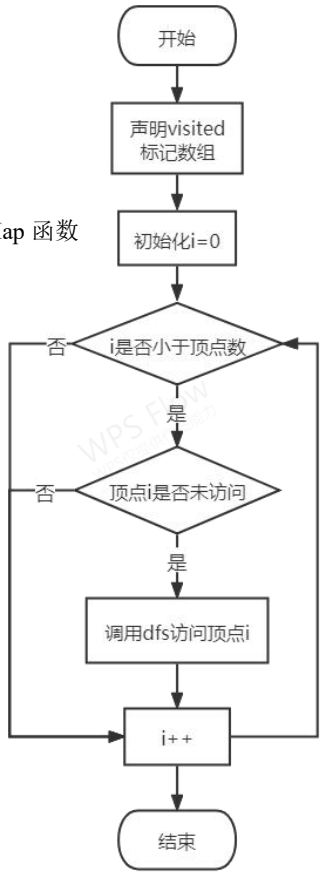


题 8-1

dfs 函数

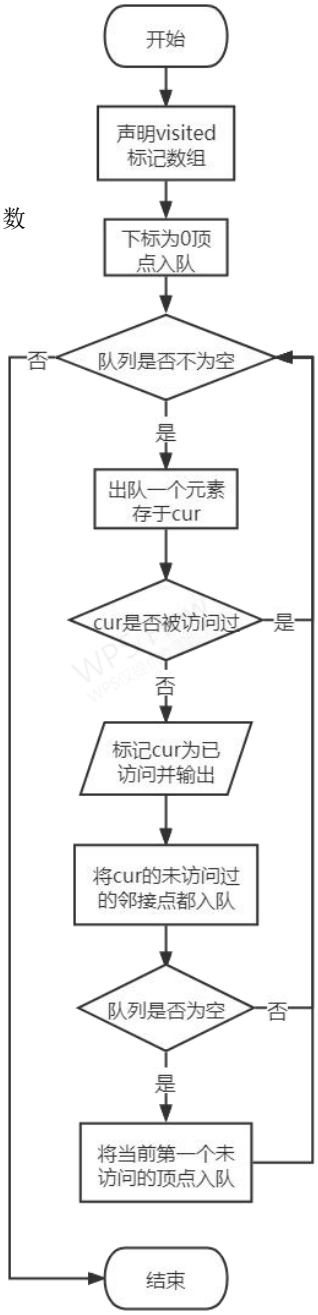
题 8-1

deepTraverseMap 函数



题 8-1

breadthTraverseMap 函数



五、程序代码

1. （必做题）图的基本操作

8-1.c

```
#include <stdio.h>
#include <stdlib.h>
#include "queue.h"

#define INFINITY INT_MAX
```

```
#define MAX_VERTEX_NUM 20
#define TRUE 1
#define FALSE 0

typedef int Boolean;
//分别为有向图、有向网、无向图、无向网
typedef enum {
    DG, DN, UDG, UDN
} GraphKind;
//表示顶点关系的类型
typedef int VRType;
//表示边信息的类型
typedef char *InfoType;

//图/网中的顶点类型
typedef int VertexType;

//邻接矩阵中元素的类型
typedef struct ArcCell {
    VRType adj;//顶点关系，为 1/0 表示是否相邻或表示权值
    InfoType *info;//边或弧的相关信息指针
} ArcCell, AdjMatrix[MAX_VERTEX_NUM][MAX_VERTEX_NUM];
/* 邻接矩阵中元素的 Setter */
void setArcCell(ArcCell *cell, VRType value, InfoType *infoPt) {
    cell->adj = value;
    cell->info = infoPt;
}
/* 把 n*n 的邻接矩阵 matrix 中的初始化，顶点关系为 value */
Boolean initAdjMat(AdjMatrix matrix, int n, int value) {
    if (n > MAX_VERTEX_NUM)
        return FALSE;
    for (int i = 0; i < n; i++)
        for (int j = 0; j < n; j++)
            setArcCell(&matrix[i][j], value, NULL);
    return TRUE;
}

//图
typedef struct {
    VertexType vexs[MAX_VERTEX_NUM];
    AdjMatrix arcs;//邻接矩阵
    int vexnum, arcnum;//顶点数、边/弧数
    GraphKind kind;
} Mgraph;
```

```
/* 构建有向图 */
Boolean constructDG(Mgraph *mgraph);
/* 构建有向网 */
Boolean constructDN(Mgraph *mgraph);
/* 构建无向图 */
Boolean constructUDG(Mgraph *mgraph);
/* 构建无向网 */
Boolean constructUDN(Mgraph *mgraph);
/* 深度优先遍历图 */
void deepTraverseMap(Mgraph mgraph);
/* 广度优先遍历图 */
void breadthTraverseMap(Mgraph mgraph);


int main() {
    Mgraph graph;
    int code;

    //读入要构造的图的类型
    while (1) {
        printf("-------------\n");
        printf("0:有向图\n1:有向网\n2:无向图\n3:无向网\n-1:退出\n 请选择要构造的类型\n");
        scanf("%d", &code);
        if (code == -1) {
            system("pause");
            return 0;
        } else if (code < -1 || code > 3) {
            printf("输入错误，请重新输入\n");
        } else {
            graph.kind = code;
            break;
        }
    }
    //根据不同图类型调用相应的构造函数
    if (graph.kind == DG) {
        constructDG(&graph);
    } else if (graph.kind == DN) {
        constructDN(&graph);
    } else if (graph.kind == UDG) {
        constructUDG(&graph);
    } else if (graph.kind == UDN) {
        constructUDN(&graph);
    }
```

```
    deepTraverseMap(graph);//深度优先遍历图 graph
    breadthTraverseMap(graph);//广度优先遍历图 graph

    printf("\n");

    system("pause");
    return 0;
}

Boolean constructDG(Mgraph *mgraph) {
    int start, end;
    printf("-----------------\n");
    printf("开始构造有向图\n");
    printf("请输入顶点数（不能超过 20）：\n");
    scanf("%d", &mgraph->vexnum);
    initAdjMat(mgraph->arcs, mgraph->vexnum, 0);
    printf("请输入弧数：\n");
    scanf("%d", &mgraph->arcnum);
    printf("请输入各条弧的起点和终点（起点终点之间以空格隔开）：\n");
    //读入各条弧
    for (int i = 0; i < mgraph->arcnum; i++) {
        scanf("%d %d", &start, &end);
        if (start >= mgraph->vexnum || start < 0 ||
            end >= mgraph->vexnum || end < 0) {
            printf("该条弧输入错误，请重新输入当前弧：\n");
            i--;
            continue;
        }
        setArcCell(&mgraph->arcs[start][end], 1, NULL);
    }
    printf("--------邻接矩阵预览--------\n");
    for (int i = 0; i < mgraph->vexnum; i++) {
        for (int j = 0; j < mgraph->vexnum; j++)
            printf("%6d", mgraph->arcs[i][j].adj);
        printf("\n");
    }

    return TRUE;
}

Boolean constructDN(Mgraph *mgraph) {
    int start, end, weight;
    printf("-----------------\n");
```

```
    printf("开始构造有向网\n");
    printf("请输入顶点数（不能超过 20）：\n");
    scanf("%d", &mgraph->vexnum);
    initAdjMat(mgraph->arcs, mgraph->vexnum, INFINITY);
    printf("请输入弧数：\n");
    scanf("%d", &mgraph->arcnum);
    printf("请输入各条弧的起点、终点与权值（以空格隔开）：\n");
    //读入各条弧
    for (int i = 0; i < mgraph->arcnum; i++) {
        scanf("%d %d %d", &start, &end, &weight);
        if (start >= mgraph->vexnum || start < 0 ||
            end >= mgraph->vexnum || end < 0) {
            printf("该条弧输入错误，请重新输入当前弧：\n");
            i--;
            continue;
        }
        setArcCell(&mgraph->arcs[start][end], weight, NULL);
    }
    printf("--------邻接矩阵预览--------\n");
    for (int i = 0; i < mgraph->vexnum; i++) {
        for (int j = 0; j < mgraph->vexnum; j++) {
            if (mgraph->arcs[i][j].adj != INFINITY)
                printf("%6d", mgraph->arcs[i][j].adj);
            else
                printf("   INF");
        }

        printf("\n");
    }

    return TRUE;
}

Boolean constructUDG(Mgraph *mgraph) {
    int start, end;
    printf("-----------------\n");
    printf("开始构造无向图\n");
    printf("请输入顶点数（不能超过 20）：\n");
    scanf("%d", &mgraph->vexnum);
    initAdjMat(mgraph->arcs, mgraph->vexnum, 0);
    printf("请输入边数：\n");
    scanf("%d", &mgraph->arcnum);
    printf("请输入各条边的起点和终点（起点终点之间以空格隔开）：\n");
    //读入各条边
```

```
    for (int i = 0; i < mgraph->arcnum; i++) {
        scanf("%d %d", &start, &end);
        if (start >= mgraph->vexnum || start < 0 ||
            end >= mgraph->vexnum || end < 0) {
            printf("该条边输入错误，请重新输入当前弧：\n");
            i--;
            continue;
        }
        //无向图，对称构建邻接矩阵
        setArcCell(&mgraph->arcs[start][end], 1, NULL);
        setArcCell(&mgraph->arcs[end][start], 1, NULL);
    }
    printf("--------邻接矩阵预览--------\n");
    for (int i = 0; i < mgraph->vexnum; i++) {
        for (int j = 0; j < mgraph->vexnum; j++)
            printf("%6d", mgraph->arcs[i][j].adj);
        printf("\n");
    }

    return TRUE;
}

Boolean constructUDN(Mgraph *mgraph) {
    int start, end, weight;
    printf("-----------------\n");
    printf("开始构造无向网\n");
    printf("请输入顶点数（不能超过 20）：\n");
    scanf("%d", &mgraph->vexnum);
    initAdjMat(mgraph->arcs, mgraph->vexnum, INFINITY);
    printf("请输入边数：\n");
    scanf("%d", &mgraph->arcnum);
    printf("请输入各条边的起点、终点与权值（以空格隔开）：\n");
    //读入各条边
    for (int i = 0; i < mgraph->arcnum; i++) {
        scanf("%d %d %d", &start, &end, &weight);
        if (start >= mgraph->vexnum || start < 0 ||
            end >= mgraph->vexnum || end < 0) {
            printf("该条边输入错误，请重新输入当前弧：\n");
            i--;
            continue;
        }
        //无向图，对称构建邻接矩阵
        setArcCell(&mgraph->arcs[start][end], weight, NULL);
        setArcCell(&mgraph->arcs[end][start], weight, NULL);
```

```
    }
    printf("--------邻接矩阵预览--------\n");
    for (int i = 0; i < mgraph->vexnum; i++) {
        for (int j = 0; j < mgraph->vexnum; j++) {
            if (mgraph->arcs[i][j].adj != INFINITY)
                printf("%6d", mgraph->arcs[i][j].adj);
            else
                printf("   INF");
        }

        printf("\n");
    }

    return TRUE;
}

void dfs(Boolean visited[], Mgraph mgraph, int index) {
    //访问当前顶点
    visited[index] = TRUE;
    printf("%d ", index);
    //访问邻接点
    for (int i = 0; i < mgraph.vexnum; i++) {
        //邻接点未访问则访问
        if (visited[i] == FALSE && (
                ((mgraph.kind    ==    DG    ||    mgraph.kind    ==    UDG)    &&
mgraph.arcs[index][i].adj != 0) ||
                ((mgraph.kind    ==    DN    ||    mgraph.kind    ==    UDN)    &&
mgraph.arcs[index][i].adj != INFINITY))) {
            dfs(visited, mgraph, i);
        }
    }
}

void deepTraverseMap(Mgraph mgraph) {
    Boolean visited[MAX_VERTEX_NUM]={0,};
    //使遍历同时适用于非连通图
    printf("-------------\n深度优先遍历：");
    for (int i = 0; i < mgraph.vexnum; i++) {
        if (visited[i] == FALSE)
            dfs(visited, mgraph, i);
    }
    printf("\n");
}
```

```
void breadthTraverseMap(Mgraph mgraph) {
    Boolean visited[MAX_VERTEX_NUM]={0,};
    QElemType cur;
    LinkQueue queue;
    Initqueue(&queue);
    printf("-------------\n广度优先遍历：");

    //从下标为0的顶点开始，将其入队
    Enqueue(&queue, 0);
    while (!Emptyqueue(queue)) {
        Dequeue(&queue, &cur);
        //访问过就不再访问，未访问过就访问
        if (!visited[cur]) {
            visited[cur] = TRUE;
            printf("%d ", cur);
            for (int i = 0; i < mgraph.vexnum; i++) {
                //将当前顶点的未访问邻接点入队
                if (visited[i] == FALSE && (
                        ((mgraph.kind    ==    DG    ||    mgraph.kind    ==    UDG)    &&
mgraph.arcs[cur][i].adj != 0) ||
                        ((mgraph.kind    ==    DN    ||    mgraph.kind    ==    UDN)    &&
mgraph.arcs[cur][i].adj != INFINITY))) {
                    Enqueue(&queue, i);
                }
            }
        }
        //处理非连通图
        if (Emptyqueue(queue)) {
            for (int i = 0; i < mgraph.vexnum; i++) {
                if (!visited[i]) {
                    Enqueue(&queue, i);
                    break;
                }
            }
        }
    }
    printf("\n");
}
```

queue.c

```
#include "queue.h"
```

```
#include <stdlib.h>


Status Emptyqueue(LinkQueue q) {
    if (q.head->next == NULL)
        return OK;
    return ERROR;
}

Status Enqueue(queuePtr q, QElemType elem) {
    QNode* tmp = (QNode *) malloc(sizeof(QNode));
    if (tmp == NULL)
        return ERROR;
    tmp->pt = elem;
    tmp->next = NULL;

    q->rear->next = tmp;
    q->rear = tmp;
    return OK;
}

Status Dequeue(queuePtr q, QElemType *out) {
    if (Emptyqueue(*q) == OK)
        return ERROR;
    QNode *tmp=q->head->next;
    *out = q->head->next->pt;
    q->head->next = tmp->next;
    //由于队列是有头结点的，对出队后变为空队列的情况做特殊处理
    if (q->head->next == NULL)
        q->rear = q->head;
    free(tmp);

    return OK;
}

Status Initqueue(queuePtr q) {
    if (q == NULL)
        return ERROR;
    q->rear = q->head = (QNode *) malloc(sizeof(QNode));
    if (q->rear == NULL)
        return ERROR;
    q->head->next = NULL;

    return OK;
```

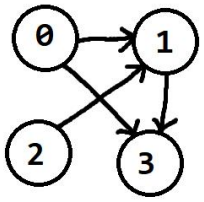
```
}
```

queue.h

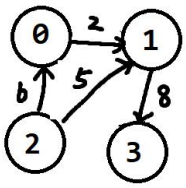
```
#ifndef UNTITLED3_QUEUE_H
#define UNTITLED3_QUEUE_H

#define Status int
#define OK 1
#define ERROR 0
typedef int QElemType;

typedef struct QNode{
    QElemType pt;
    struct QNode *next;
} QNode;

typedef struct LinkQueue {
    QNode *head, *rear;
} LinkQueue, *queuePtr;

/* 队列基本操作 */
Status Enqueue(queuePtr, QElemType);
Status Dequeue(queuePtr, QElemType *);
Status Emptyqueue(LinkQueue q);
Status Initqueue(queuePtr q);

#endif //UNTITLED3_QUEUE_H
```

六、测试和结果

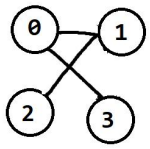
1. （必做题）图的基本操作



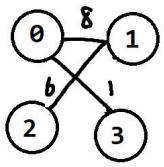
```
C:\Windows\System32\cmd.exe

D:\Documents\YNU文件及资料\大二上\课程相关\courses-of-2nd-year\data-struct
ure\experiment 8>8-1.exe
-------------
0：有向图
1：有向网
2：无向图
3：无向网
-1：退出
请选择要构造的类型
0
-----------------
开始构造有向图
请输入顶点数（不能超过20）：
4
请输入弧数：
4
请输入各条弧的起点和终点（起点终点之间以空格隔开）：
0 1
0 3
1 3
2 1
--------邻接矩阵预览--------
     0      1      0      1
     0      0      0      1
     0      1      0      0
     0      0      0      0
-------------
深度优先遍历：0 1 3 2
-------------
广度优先遍历：0 1 3 2

请按任意键继续. . .

D:\Documents\YNU文件及资料\大二上\课程相关\courses-of-2nd-year\data-struct
ure\experiment 8>
```




```
C:\Windows\System32\cmd.exe

D:\Documents\YNU文件及资料\大二上\课程相关\courses-of-2nd-year\data-struct
ure\experiment 8>8-1.exe
-------------
0：有向图
1：有向网
2：无向图
3：无向网
-1：退出
请选择要构造的类型
1
-----------------
开始构造有向网
请输入顶点数（不能超过20）：
4
请输入弧数：
4
请输入各条弧的起点、终点与权值（以空格隔开）：
0 1 2
2 0 6
2 1 5
1 3 8
--------邻接矩阵预览--------
   INF      2    INF    INF
   INF    INF    INF      8
     6      5    INF    INF
   INF    INF    INF    INF
-------------
深度优先遍历：0 1 3 2
-------------
广度优先遍历：0 1 3 2

请按任意键继续. . .

D:\Documents\YNU文件及资料\大二上\课程相关\courses-of-2nd-year\data-struct
ure\experiment 8>
```




```
C:\Windows\System32\cmd.exe

D:\Documents\YNU文件及资料\大二上\课程相关\courses-of-2nd-year\data-struct
ure\experiment 8>8-1.exe
------------
0：有向图
1：有向网
2：无向图
3：无向网
-1：退出
请选择要构造的类型
2
----------------
开始构造无向图
请输入顶点数（不能超过20）：
4
请输入边数：
3
请输入各条边的起点和终点（起点终点之间以空格隔开）：
0 1
1 2
3 0
--------邻接矩阵预览--------
     0     1     0     1
     1     0     1     0
     0     1     0     0
     1     0     0     0
------------
深度优先遍历：0 1 2 3
------------
广度优先遍历：0 1 3 2

请按任意键继续. . .

D:\Documents\YNU文件及资料\大二上\课程相关\courses-of-2nd-year\data-struct
ure\experiment 8>
```




```
C:\Windows\System32\cmd.exe

D:\Documents\YNU文件及资料\大二上\课程相关\courses-of-2nd-year\data-struct
ure\experiment 8>8-1.exe
------------
0：有向图
1：有向网
2：无向图
3：无向网
-1：退出
请选择要构造的类型
3
----------------
开始构造无向网
请输入顶点数（不能超过20）：
4
请输入边数：
3
请输入各条边的起点、终点与权值（以空格隔开）：
0 1 8
1 2 6
0 3 1
--------邻接矩阵预览--------
   INF     8   INF     1
     8   INF     6   INF
   INF     6   INF   INF
     1   INF   INF   INF
------------
深度优先遍历：0 1 2 3
------------
广度优先遍历：0 1 3 2

请按任意键继续. . .

D:\Documents\YNU文件及资料\大二上\课程相关\courses-of-2nd-year\data-struct
ure\experiment 8>
```

七、用户手册

1. （必做题）图的基本操作

图中边或弧的权值都以 int 存储，输入的值不能超过 int 的范围，调用的队列相关的函数都存在以 queue.h 为头文件的源代码 queue.c 中。