中山大学计算机学院本科生实验报告

（2021 学年第 1 学期）

课程名称：Data structures and algorithms　　任课教师：张子臻

年级	20 级	专业（方向）	软件工程
学号	20337270	姓名	钟海财
电话	13397996670	Email	2940599563@qq.com
开始日期	2021/9/16	完成日期	2021/9/22

1.实验题目

1）Boring

2）Breadth First Search

2.实验目的

1．实现队列的基本功能（入队，出队，获取队首元素，获取队列规模，判断队列是否为空）——1）Boring

2．通过对队列的使用，解决广度优先搜索问题，加深对队列特点“先进先出，后进后出”的理解——2）Breadth First Search

3.程序设计

1）Boring

设计思路：用一个数组 data[]来记录队列中的元素；通过两个索引 front 和 rear，front 记录队首的位置，rear 记录队尾的后一个位置，则队首元素为 data[front],队尾为 data[rear-1]。同时用 count 记录元素个数。max 为数组容量。

则入队时将入队元素赋给 data[rear],并++rear，++count；出队时++front，--count 。为实现循环队列，在 front==max 时令 front=0，在 rear==max 时令 rear=0.

代码：

```
//Boring
#include<iostream>
using namespace std ;
class Queue{
private:
    int data[1000];
    int front;//记录队首位置
    int rear; //记录队尾的后一个空位置
    int count;
public:
    Queue(){
        rear  = 0;
        front = 0;
        count = 0;
    }
    void In(int t){//队尾入队
        data[rear]=t;//rear 记录队尾的后一个空位置
        ++rear;
        if(rear==1000) rear=0;
        ++count;
    }
    void Count()const{//计数
        cout<< count << endl ;
    }
    void Out(){ //队首出队
        if( rear-front > 0){
        cout<<data[front]<<endl;
        ++front;
        if(front==1000) front=0;
        --count;
        }
        else cout<<-1<<endl;
    }
};
int main(){
    int times;
    cin>>times;//读入指令数目
    Queue b;
    for(int i = 0 ; i <times ; ++i ){
        string op ;
```

```
        cin >> op ;//读入指令，根据指令操作
        if(op=="In"){
            int num;
            cin>>num;
            b.In(num);
        }
        else if(op=="Out"){
            b.Out();
        }
        else if(op=="Count"){
            b.Count();
        }
    }
    return 0 ;
}
```

2）Breadth First Search

设计思路：可使用队列实现广度优先搜索，队列 Queue 中设置两个数组，一个数组 data[]为 private 类型，记录队列元素（未遍历过的新增可达点）；另一个数组 able[]为 public 类型，记录所有已经遍历过的点（able[]初始值全为 0；able[i]==0 时表示 i 不可达；able[i]==1 时表示 i 为新增的可达点；able[i]>1 时表示 i 为已遍历过的可达点）。

首先将起点 start 入队，同时通过一个 while 循环实现 start 的所有可达点的广度优先遍历（当队列中有新增可达点时循环继续），在 while 循环中：对队列中的未遍历过的可达点依次遍历和出队，同时将找到的新增可达点入队，直至本轮循环前队列中已有的新增可达点全部遍历完，开始下一次 while 循环。（循环中 able[end]>=1 即可跳出循环）

终点位置用 end 记录，若 able[end]>=1,意味着起点可以到终点；反之 able[end]==0,意味着起点不能到达终点。

代码：

```
#include<iostream>
using namespace std;
class Queue{
private:
    int data[100];//记录新增可达点
    int front;//记录队首位置
    int rear; //记录队尾的后一个空位置
    int count;
public:
    int able[20] = {0};   /*记录所有可达点，able[i]==0 时表示 i 不可达，
able[i]==1 时表示 i 为新增的可达点，able[i]>1 时表示 i 为已遍历过的可达点*/
      Queue(){
        rear = 0 ;
        front = 0;
        count = 0 ;
    }
    void push(int t){ //队尾入队
        data[rear]=t;//rear 记录队尾的后一个空位置
        ++rear;
        if(rear==100) rear = 0  ;
           ++count;
    }
    int size()const {
       return count ;
    }
    bool empty(){
        if(count==0)  return true ;
        else return false ;
    }
    int getfront(){
        return data[front];
    }
    void pop(){ //队首出队
        if( count > 0)  ++front ;
        if(front==100) front = 0  ;
        --count;
    }
};
int main(){
    int num, start, end ;    //点的个数，起点，终点
    cin >> num >> start >> end ;
    int a[num][num];
    for(int i = 0 ; i <num ; ++i){ //记录邻接矩阵
        for(int j = 0 ; j<num ; ++j){
```

```
            cin>>a[i][j] ;
        }
    }
    Queue  A; //用队列储存新增可达点
    A.push(start);//push 起点入队
    while(!A.empty()){//当有新增可达点时循环继续
        int k=A.size();
        for(int i = 0 ; i < k ; ++i){ //对可达点进行遍历
            int t=A.getfront();
            A.pop(); //遍历过的可达点出队
            A.able[t]+=1;//并记录为已遍历
            for(int j = 0 ; j < num ; ++j){
                if(a[t][j] ==1 && A.able[j]== 0){
                    A.able[j] =1 ;
                    A.push(j);//确认有新增可达点 j 并入队
                }
                if(A.able[end]>=1) break;
            }
            if(A.able[end]>=1) break;
        }
        if(A.able[end]>=1) break;
    }

    if(A.able[end] >= 1) cout<<"yes" ;
    else cout<<"no" ;
    return 0 ;
}
```

4.程序运行与测试

1）Boring ：

序号	测试输入	输出
1	20 Out In 1240 Out	-1 1240 5 1542

	In 1542	3348
	In 3348	5
	In 5579	5
	In 5048	5579
	In 8730	4
	Count	4
	Out	4
	In 1996	5048
	In 1223	
	Out	
	Count	
	Count	
	Out	
	Count	
	Count	
	Count	
	Out	
2	14	0
	Count	2681
	In 2681	-1
	Out	0
	Out	2221
	Count	2

	In 2221 In 1198 In 9253 Out Count Count Out In 7881 Count	2 1198 2
3	11 Out Count In 8074 Count In 3 Count Out Out Out Count Count	-1 0 1 2 8074 3 -1 0 0
4	20 Count	0 0

	Count	0
	Count	1
	In 6207	6207
	Count	5937
	In 5937	1
	Out	1
	Out	7501
	In 7501	0
	Count	2
	Count	8179
	Out	696
	Count	1
	In 8179	
	In 696	
	Count	
	Out	
	Out	
	In 6717	
	Count	

2）Breadth First Search：

序号	测试输入	输出
1	**5 0 2** **1 1 0 0 0** **1 1 0 0 1** **0 0 1 0 0** **0 0 0 1 1** **0 1 0 1 1**	**no**
2	**7 0 6** **1 1 0 0 0 0 0** **1 1 1 0 0 0 0** **0 1 1 0 0 0 0** **0 0 0 1 1 0 0** **0 0 0 1 1 1 0** **0 0 0 0 1 1 1** **0 0 0 0 0 1 1**	**no**

3	**7 0 6** **1 1 0 0 0 0 0** **1 1 1 0 0 0 0** **0 1 1 1 0 0 0** **0 0 1 1 1 0 0** **0 0 0 1 1 1 0** **0 0 0 0 1 1 1** **0 0 0 0 0 1 1**	**yes**

5.实验总结与心得

本次实验通过对队列的基本功能的实现以及使用队列实现广度优先搜索判断两点是否可达,加深了我对队列这种线性结构特点的理解，先进先出，后进后出，只能在队首删除元素，只能在队尾添加元素。

附录、提交文件清单