# 《数据结构》课程实践报告

| 院、系 | 计算机学院 | 年级专业 | 21 计算机科学与技术 | 姓名 | 赵鹏 | 学号 | 2127405037 |
|---|---|---|---|---|---|---|---|
| 实验布置日期 | 2022.11.28 | | 提交日期 | 2022.12.6 | | 成绩 | |

# 课程实践实验 11：Graph Connectivity

## 一、问题描述及要求

### Graph Connectivity

**Time Limit:** 8000MS **Memory Limit:** 131072K

**Case Time Limit:** 3000MS

### Description

Let us consider an **undirected** graph G = < V, E >. At first there is no edge in the graph. You are to write a program to calculate the connectivity of two different vertices. Your program should maintain the functions inserting or deleting an edge.

### Input

The first line of the input contains an integer numbers N (2 <= N <= 1000) -- the number of vertices in G. The second line contains the number of commands Q (1 <= Q <= 20000). Then the following Q lines describe each command, and there are three kinds of commands:

I u v: Insert an edge (u, v). And we guarantee that there is no edge between nodes u and v, when you face this command.
D u v: Delete an existed edge (u, v). And we guarantee that there is an edge between nodes u and v, when you face this command.
Q u v: A querying command to ask the connectivity between nodes u and v.

You should notice that the nodes are numbered from 1 to N.

### Output

Output one line for each querying command. Print "Y" if two vertices are connected or print "N" otherwise.

## Sample Input

```
3
7
Q 1 2
I 1 2
I 2 3
Q 1 3
```

```
D 1 2
Q 1 3
Q 1 1
```

## Sample Output

```
N
Y
N
Y
```

**问题分析**

本次实验的内容是一个关于图论的实际问题。给定含有 n 个结点的无向不带权图。可以对图进行插入边(u,v)，删除边(u,v)，以及查询 u,v 的连通性，如果连通则输出 Y 否则输出 N。显然，可以分别考虑三种操作的实现，对于执行操作调用对应方法即可。

输入：第一行输入无向图的结点数 N 和询问数 Q。接下来 Q 行每行三个数据，第一个字母表示操作，接下来两个数 u,v 代表两个结点。

输出：对于每次询问 Q，若 u,v 连通输出 Y，反之输出 N。

# 二、问题解决

对于图相关的问题，首先需要考虑存图。由于需要频繁进行加边和删边的操作，故可以使用邻接矩阵进行存图，便可以 O（1）的时间复杂度进行加边和删边。随后只需要以此实现三种操作即可。

**一、 加边**

在使用邻接矩阵存图时，edge[i][j]即代表量结点 i,j 之间是否存在，若 edge[i][j]=1 则代表 i,j 结点之间存在边，反之若 edge[i][j]等于 0，则代表 i,j 之间不存在边。由于此题是无向图，因此加边操作在修改邻接矩阵时只需要令 edge[i][j]=edge[j][i]=1 即可。

## 二、 删边

删边操作与加边操作相对称。同样修改邻接矩阵即可，若需要删除 i,j 之间的边，只需要令 edge[i][j]=edge[j][i]=0 即可。

## 三、 判断两点间的连通性

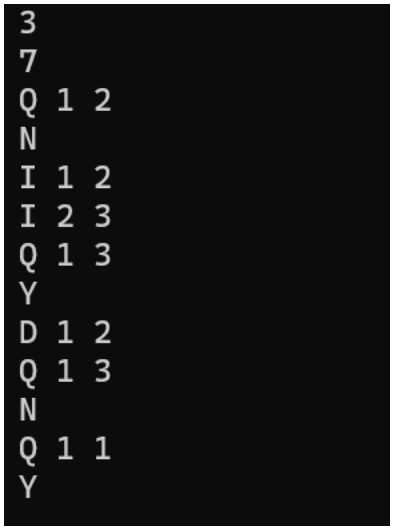
对于图中两点之间连通性的判断，只需要从一点开始进行宽度优先遍历或深度优先遍历即可。在此题中我选择使用宽度优先遍历，在实现过程中使用 C++模板库中的队列来进行实现宽度优先遍历的过程，首先将起点 u 入队并标记已经访问过，随后从起点向外进行扩展，弹出队头并遍历相邻起点，若相邻的结点没有访问过则进行入队操作，并标记此结点访问过。若某一次将终点 v 入队则说明两点之间连通，返回 true，循环这个过程直至队列为空。若队列为空循环结束，则说明从 u 无法遍历到 v，即两点之间不连通，返回 false 即可。

代码实现如下：

```
1. bool Graph<DataType>::JudgeConnectivity(int u, int v)
2. {
3.  if (u == v)return true;
4.  memset(st, 0, sizeof st);
5.  std::queue<int>Q;
6.  Q.push(u); st[u] = 1;
7.  while (Q.size())
8.  {
9.   auto head = Q.front(); Q.pop();
10.  for (int i = 1; i <= vertexNum; i++)
11.  {
12.   if (edge[head][i] && !st[i])
13.    if (i == v)return true;
14.    else
15.    {
16.     Q.push(i);
17.     st[i] = 1;
18.    }
19.  }
20. }
21. return false;
22.}
```

# 三、实验结果测试

使用题目中给定的数据进行测试，运行结果为：




结论：测试通过

# 四、小结

通过本次实验，我加深对了图相关的知识的理解，包括图的存储、图的遍历等，并且运用图相关的知识较好的解决了一道算法题目。