栈和队列

一.解释：

1.栈

栈是一种特殊的线性表。其特殊性在于限定插入和删除数据元素的操作只能在线性表的一端进行。如下所示：

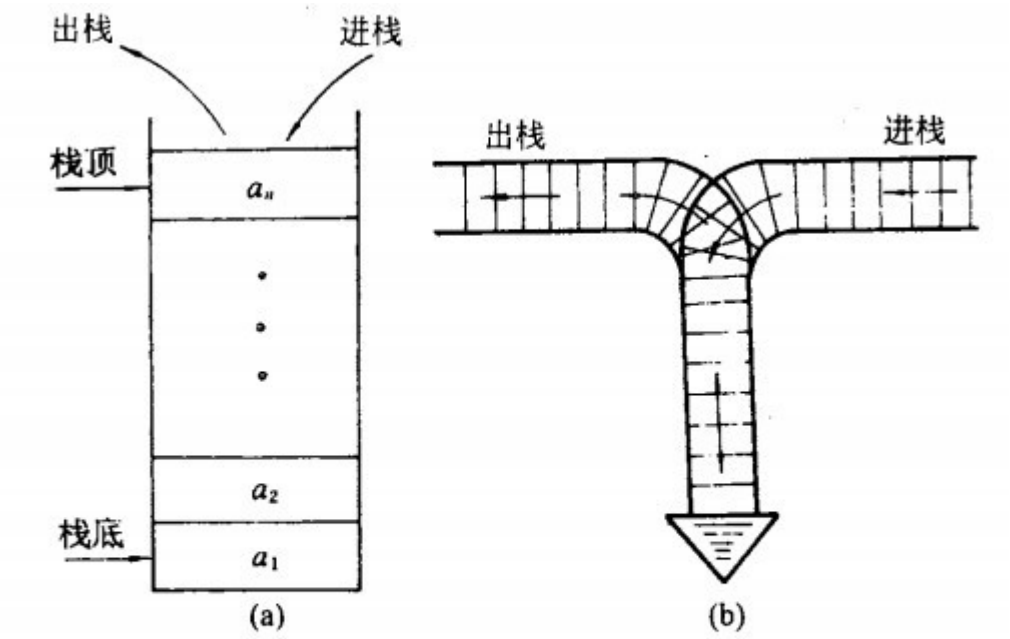

图 3.1 栈

(a) 栈的示意图；(b) 用铁路调度站表示栈

结论：后进先出（Last In First Out），简称为LIFO线性表。

举个例子：你在洗碗把洗好的碗编号为1、2、、、n依次摞起来，1号在最下面，向上编号依次增加，然后再从上到下把碗放好，这样的话，先被洗的碗，就后被放好。

2.队列

队列(Queue)也是一种运算受限的线性表，它的运算限制与栈不同，是两头都有限制，插入只能在表的一端进行(只进不出)，而删除只能在表的另一端进行(只出不进)，允许删除的一端称为队尾(rear)，允许插入的一端称为队头(Front),如图所示：

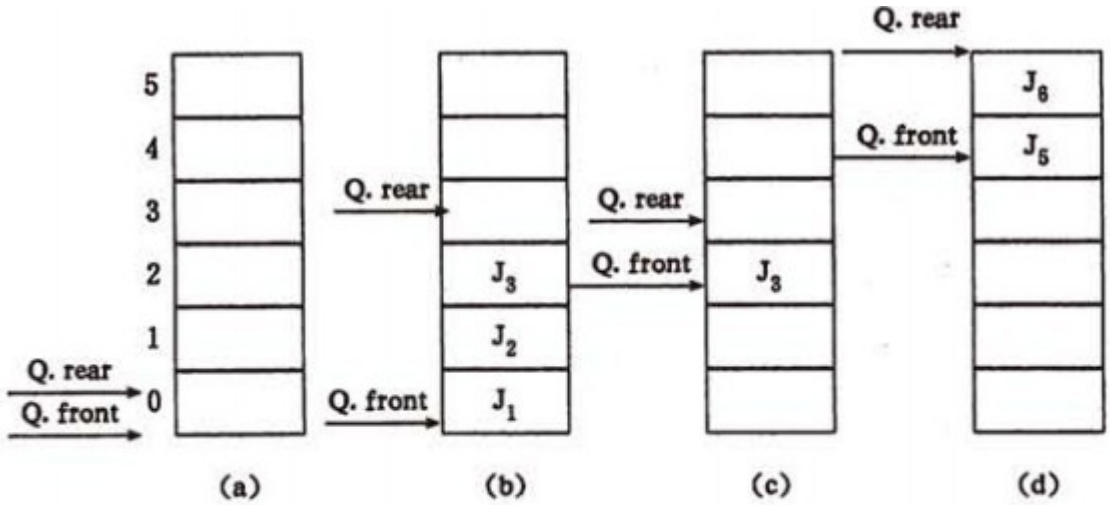

图 3.12　头、尾指针和队列中元素之间的关系

(a) 空队列；(b) J_1、J_2 和 J_3 相继入队列；(c) J_1 和 J_2 相继被删除；

(d) J_4、J_5 和 J_6 相继插入队列之后 J_3 及 J_4 被删除

结论：队列的操作原则是先进先出的，所以队列又称作FIFO表(First In First Out)

举个例子：超市付账的时候排队，先来的先排在最前面，后来的后排在队伍最后面，付账时排在队伍前面的先付账，排在后面的后付账。

二.用法

1.头文件

```
#include<queue>// 队列
#include<stack>//栈
```

2.定义方式

```
stack<int>  s;//参数也是数据类型，这是栈的定义方式
queue<int>  q; //参数是数据类型，这是队列的定义方式
```

3.常用操作

栈：

```
s.empty()//如果栈为空返回true，否则返回false
s.size()//返回栈中元素的个数
s.pop()//删除栈顶元素但不返回其值
s.top()//返回栈顶的元素，但不删除该元素
s.push(x)//在栈顶压入新元素 ，参数x为要压入的元素
```

队列：

```
q.empty()// 如果队列为空返回true，否则返回false
q.size() // 返回队列中元素的个数
q.pop()  //删除队列首元素但不返回其值
q.front()  // 返回队首元素的值，但不删除该元素
q.push(x) //在队尾压入新元素 ，x为要压入的元素
q.back() //返回队列尾元素的值，但不删除该元素
```

三、例子

```cpp
#include <iostream>
#include <algorithm>
#include <cstdio>
#include <cstring>
#include <queue>
#include <stack>
#include <vector>
using namespace std;
int main()
{
    queue<int> q;
    stack<char> s;
    q.push(1);
    cout << q.enpty() << endl;
    q.push(2);
    cout << q.front() << endl;
    q.pop();
    cout << q.front() << endl;
    q.pop();
    cout << q.empty() <<endl;
    s.push(a);
    cout << s.top() <<endl;
    s.push(b);
    cout << s.top();
    s,pop();
    cout << s.top();
}
```

单调队列

基本性质

简单应用

单调栈

基本性质

- 和单调队列类似，保证id和ｖａｌ的单调性即可，但入栈和出站是同一边
- 用于解决：可以找到从左/右遍历第一个比它小/大的位置

简单应用

栈的练习题

1.洛谷（简单题）P1739表达式括号匹配

题目分析

通过读题我们可知题目需求只是对一段字符串的括号进行匹配，那么根据括号的匹配规则来看一个右括号的左侧必须有一个左括号才算匹配，而对于左括号比右括号多的情况和右括号比左括号多的情况都是不匹配的，对于匹配串来讲唯一影响我们的是嵌套的括号，那么应该怎么去掉嵌套的括号对我们的影响呢？让嵌套的括号在匹配之前”消失“就可以了，通过栈的数据结构，当字符串是括号左括号字符时入栈，当遇到右括号时出栈，由此可得当左括号多余而造成不匹配时，栈应该是无标记且非空状态，如果右括号多余则应该是标记状态，反之如果无标记且空则匹配成功。

```
#include <iostream>
#include <algorithm>
#include <string>
#include <vector>
#include <queue>
#include <stack>

using namespace std;
int main() {
    string str;
    cin >> str;
    int i = 0, flag = 0;
    stack<char> stack1;
    while (str[i] != '@') {
        if (str[i] == '(') {
            stack1.push(str[i]);
        }
        if (str[i] == ')') {
            if (stack1.empty()) {
                flag = 1;
                break;
            }
            stack1.pop();
        }
        i++;
    }
    if (flag) cout << "NO" << endl;
    else if (stack1.empty()) cout << "YES" << endl;
    else cout << "NO" << endl;
    return 0;
}
```

洛谷P1449(普及)后缀表达式

由题意分析可知，题目中的输入数据已经将表达式的运算顺序后置在运算变量之后故采取栈的数据结构辅助运算即可

```
#include <iostream>
#include <algorithm>
#include <string>
#include <vector>
#include <queue>
#include <stack>

using namespace std;
int main() {
    stack<int> stack1;
    string s;
    cin >> s;
    int i = 0;
    while (s[i] != '@') {
```

```
        if (s[i] >= '0' && s[i] <= '9') {
            int num = 0;
            while (s[i] != '.')  {
                num = num * 10 + (s[i] - '0');
                i++;
            }
            stack1.push(num);
        } else {
            int a = stack1.top();
            stack1.pop();
            int b = stack1.top();
            stack1.pop();
            int temp;
            switch(s[i]) {
                case '+': {
                    temp = b + a;
                }break;
                case '-': {
                    temp = b - a;
                }break;
                case '*': {
                    temp = b * a;
                }break;
                case '/': {
                    temp = b / a;
                }break;
            }
            stack1.push(temp);
        }
        i++;
    }
    cout << stack1.top() << endl;
    return 0;
}
```

作业:杭电1022

由题意可知，我们模拟栈的入栈出栈操作即可

当火车没有全部经过前，每次入栈一个，当栈顶和入栈顺序一致时出栈，直至栈空。

最后判断如果经历了n次出栈则完成输出答案数组，反之，按题输出。

```
#include <iostream>
#include <cstdio>
#include <stack>

using namespace std;

int main() {
    int n;
    char str1[15] = {0}, str2[15] = {0};
    while (scanf("%d %s %s", &n, str1, str2) != EOF) {
        stack<int> s1;
        int num1[15] = {0}, num2[15] = {0}, ans[30];
        for (int i = 0; str1[i]; i++) num1[i] = str1[i] - '0';
        for (int i = 0; str2[i]; i++) num2[i] = str2[i] - '0';
        int cnt = 0;
        int flag = 0;
```

```
        int ind = 0;
        for (int i = 0; i < n; i++) {
            s1.push(num1[i]);
            ans[ind++] = 1;
            while (!s1.empty() && s1.top() == num2[cnt]) {
                s1.pop();
                ans[ind++] = 0;
                cnt += 1;
            }
        }
        if (cnt != n) {
            cout << "No." << endl << "FINISH" << endl;
            continue;
        }
        cout << "Yes." << endl;
        for (int i = 0; i < ind; i++) {
            if (ans[i]) cout << "in" << endl;
            else cout << "out" << endl;
        }
        cout << "FINISH" << endl;
    }
    return 0;
}
```

队列洛谷P1996约瑟夫问题

```
#include <iostream>
#include <algorithm>
#include <string>
#include <vector>
#include <queue>
#include <stack>

using namespace std;


int main() {kkkkkkkkkkkkkk
    queue<int> q;
    int n, m;
    cin >> n >> m;
    int a[n];
    for (int i = 0; i < n; i++) {
        a[i] = i + 1;
        q.push(a[i]);
    }
    int cnt = 1, ind = 0;
    while (n--) {
        while (cnt != m) {
            int temp = q.front();
            q.pop();
            q.push(temp);
            cnt++;
        }
        cnt = 1;
        int x = q.front();
        cout << x;
        if (n >= 1) cout << " ";
        q.pop();
    }
```

```
    return 0;
}
```

洛谷P1115最大子段和

```
#include <iostream>
#include <algorithm>
#include <string>
#include <vector>
#include <queue>
#include <stack>

using namespace std;
int main() {
    int n;
    cin >> n;
    int a[n];
    for (int i = 0 ; i < n; i++) {
        cin >> a[i];
    }
    int ans = a[0], sum = a[0];
    for (int i = 1; i < n; i++) {
        if (sum > 0) {
            sum += a[i];
        } else {
            sum = a[i];
        }
        ans = (ans > sum ? ans : sum);
    }
    cout << ans << endl;
    return 0;
}
```