# 题目

用两个栈实现一个队列。队列的声明如下，请实现它的两个函数appendTail和deleteHead，分别完成在队列尾部插入整数和在队列头部删除整数的功能。(若队列中没有元素，deleteHead操作返回-1)

**示例 1：**

输入：

["CQueue","appendTail","deleteHead","deleteHead"]

[[],[3],[],[]]

输出：[null,null,3,-1]

**示例 2：**

输入：

["CQueue","deleteHead","appendTail","appendTail","deleteHead","deleteHead"]

[[],[],[5],[2],[],[]]

输出：[null,-1,null,null,5,2]

**提示：**

1 <= values <= 10000

最多会对 appendTail、deleteHead 进行 10000 次调用

# 分析

## 方法一：双栈

**思路：**

维护两个栈，第一个栈支持插入操作，第二个栈支持删除操作。

根据栈先进后出的特性，我们每次往第一个栈里插入元素后，第一个栈的底部元素是最后插入的元素，第一个栈的顶部元素是下一个待删除的元素。为了维护队列先进先出的特性，我们引入第二个栈，用第二个栈维护待删除的元素，在执行删除操作的时候我们首先看下第二个栈是否为空。如果为空，我们将第一个栈里的元素一个个弹出插入到第二个栈里，这样第二个栈里元素的顺序就是待删除的元素的顺序，要执行删除操作的时候我们直接弹出第二个栈的元素返回即可。

**插入元素**

stack1直接插入元素

**删除元素**

如果stack2为空，则将stack1里的所有元素弹出插入到stack2里

如果stack2仍为空，则返回-1，否则从stack2弹出一个元素并返回

**代码：**

class CQueue {

private:

stack<int> stack1,stack2;//stack1支持插入,stack2支持删除

public:

CQueue() {

}

void appendTail(int value) {

stack1.push(value);

}

int deleteHead() {

//插入数据全部存储在stack1中,所以这里需要判断stack1

if(stack1.empty()) return -1;

//将stack1的数据拷贝到stack2,等待删除

while(!stack1.empty())

{

int tmp = stack1.top();

stack2.push(tmp);

stack1.pop();

}

int ret = stack2.top();

stack2.pop();

//将stack2删除后的数据拷贝回stack1

while(!stack2.empty())

{

int tmp = stack2.top();

stack1.push(tmp);

stack2.pop();

}

return ret;

}

};

**复杂度分析**

时间复杂度：对于插入和删除操作，时间复杂度均为O(1)。插入不多说，对于删除操作，虽然看起来是O(n)的时间复杂度，但是仔细考虑下每个元素只会「至多被插入和弹出 stack2 一次」，因此均摊下来每个元素被删除的时间复杂度仍为O(1)。

空间复杂度：O(n)。需要使用两个栈存储已有的元素。