# 题目

写一个 RecentCounter 类来计算特定时间范围内最近的请求。

请你实现RecentCounter类：

RecentCounter()初始化计数器，请求数为0 。

int ping(int t)在时间t添加一个新请求，其中t表示以毫秒为单位的某个时间，并返回过去3000毫秒内发生的所有请求数（包括新请求）。确切地说，返回在[t-3000, t]内发生的请求数。

保证每次对ping的调用都使用比之前更大的t值。

示例：

输入：

["RecentCounter", "ping", "ping", "ping", "ping"]

[[], [1], [100], [3001], [3002]]

输出：

[null, 1, 2, 3, 3]

解释：

RecentCounter recentCounter = new RecentCounter();

recentCounter.ping(1); // requests = [1]，范围是[-2999,1]，返回1

recentCounter.ping(100); // requests = [1, 100]，范围是[-2900,100]，返回2

recentCounter.ping(3001); // requests = [1, 100, 3001]，范围是[1,3001]，返回3

recentCounter.ping(3002); // requests = [1, 100, 3001, 3002]，范围是 [2,3002]，返回 3

提示：

1 <= t <= 109

保证每次对 ping 调用所使用的 t 值都 严格递增

至多调用 ping 方法 104 次

# 分析

## 方法一：队列

我们只会考虑最近3000毫秒到现在的ping数，因此我们可以使用队列存储这些ping的记录。当收到一个时间t的ping时，我们将它加入队列，并且将所有在时间t - 3000之前的ping移出队列。

**代码：**

class RecentCounter

{

public:

queue<int> q;

int q\_len;

RecentCounter()

{

q\_len = 0;

}

int ping(int t)

{

q.push(t); // 存储ping的时间点

q\_len ++;

while (q.front() < t - 3000)

{

q.pop();

q\_len --;

}

return q\_len;

}

};

另一种写法：

class RecentCounter {

public:

RecentCounter() {}

int ping(int t) {

q.push(t);

while (q.front() < t - 3000) q.pop();

return q.size();

}

private:

queue<int>q;

};

复杂度分析：

时间复杂度：均摊O(1)，每个元素至多入队出队各一次。

空间复杂度：O(L)，其中L为队列的最大元素个数。