# 题目

设计一个敲击计数器，使它可以统计在过去5分钟内被敲击次数。

每个函数会接收一个时间戳参数（以秒为单位），你可以假设最早的时间戳从1开始，且都是按照时间顺序对系统进行调用（即时间戳是单调递增）。

在同一时刻有可能会有多次敲击。

示例:

HitCounter counter = new HitCounter();

// 在时刻 1 敲击一次。

counter.hit(1);

// 在时刻 2 敲击一次。

counter.hit(2);

// 在时刻 3 敲击一次。

counter.hit(3);

// 在时刻 4 统计过去 5 分钟内的敲击次数, 函数返回 3 。

counter.getHits(4);

// 在时刻 300 敲击一次。

counter.hit(300);

// 在时刻 300 统计过去 5 分钟内的敲击次数，函数返回 4 。

counter.getHits(300);

// 在时刻 301 统计过去 5 分钟内的敲击次数，函数返回 3 。

counter.getHits(301);

进阶:

如果每秒的敲击次数是一个很大的数字，你的计数器可以应对吗？

# 分析

## 方法一：哈希字典统计

c++还可以边统计，边删除

注意

map.erase(iter ++)

iter = map.erase(iter)

map.erase(iter)

iter++

是不对的。

iter已经被删除了，已经失效了。再++，找不到map

代码：

class HitCounter

{

public:

unordered\_map<int,int> time\_freq;

/\*\* Initialize your data structure here. \*/

HitCounter()

{

}

/\*\* Record a hit.

@param timestamp - The current timestamp (in seconds granularity). \*/

void hit(int timestamp)

{

time\_freq[timestamp] ++;

}

/\*\* Return the number of hits in the past 5 minutes.

@param timestamp - The current timestamp (in seconds granularity). \*/

int getHits(int timestamp)

{

int res = 0;

auto it = time\_freq.begin();

while (it != time\_freq.end())

{

if (timestamp - (it->first) < 300)

{

res += (it->second);

it ++;

}

else

{

//time\_freq.erase(it ++); //如果删除了，再it++。不行。因为删除了，it就是一个非法指针了

it = time\_freq.erase(it); //返回下一个有效指针

}

}

return res;

}

};

## 方法二：数组+二分查找

空间复杂度会高一点

class HitCounter

{

public:

vector<int> nums;

/\*\* Initialize your data structure here. \*/

HitCounter()

{

}

/\*\* Record a hit.

@param timestamp - The current timestamp (in seconds granularity). \*/

void hit(int timestamp)

{

nums.push\_back(timestamp);

}

/\*\* Return the number of hits in the past 5 minutes.

@param timestamp - The current timestamp (in seconds granularity). \*/

int getHits(int timestamp)

{

int window\_L\_val = timestamp - 299;

int window\_L = lower\_bound(nums.begin(), nums.end(), window\_L\_val) - nums.begin();

//借助二分查找

return nums.size() - window\_L;

}

};

## 方法三：双端队列

解题思路：

hash表及普通队列适用于每秒只有一次敲击的情况

双端队列，适用于每秒有很多次敲击的情况。

class HitCounter1 {

public:

/\*\* Initialize your data structure here. \*/

HitCounter1() {

}

/\*\* Record a hit.

@param timestamp - The current timestamp (in seconds granularity). \*/

void hit(int timestamp) {

time2Cnt[timestamp]++;

}

/\*\* Return the number of hits in the past 5 minutes.

@param timestamp - The current timestamp (in seconds granularity). \*/

int getHits(int timestamp) {

int count = 0;

for (int i = timestamp - 300 + 1; i <= timestamp; i++) {

count += time2Cnt[i];

}

return count;

}

private:

unordered\_map<int, int> time2Cnt;

};

class HitCounter2 {

public:

/\*\* Initialize your data structure here. \*/

HitCounter2() {

}

/\*\* Record a hit.

@param timestamp - The current timestamp (in seconds granularity). \*/

void hit(int timestamp) {

q.push(timestamp);

}

/\*\* Return the number of hits in the past 5 minutes.

@param timestamp - The current timestamp (in seconds granularity). \*/

int getHits(int timestamp) {

while(!q.empty() && timestamp - q.front() >= 300) {

q.pop();

}

return q.size();

}

private:

queue<int> q;

};

class HitCounter {

public:

/\*\* Initialize your data structure here. \*/

HitCounter() {

}

/\*\* Record a hit.

@param timestamp - The current timestamp (in seconds granularity). \*/

void hit(int timestamp) {

if (!dq.empty() && dq.back().first == timestamp) {

dq.back().second++; // could have many hits in one second

} else {

dq.push\_back({timestamp, 1});

}

cnt++;

}

/\*\* Return the number of hits in the past 5 minutes.

@param timestamp - The current timestamp (in seconds granularity). \*/

int getHits(int timestamp) {

while(!dq.empty() && timestamp - dq.front().first >= 300) {

cnt -= dq.front().second;

dq.pop\_front();

}

return cnt;

}

private:

int cnt = 0;

deque<pair<int, int>> dq;

};