# 362.Design a Hit Counter

Design a hit counter which counts the number of hits received in the past 5 minutes.

“Design hit counter” problem has recently been asked by many companies including Dropbox and the question is harder than it seems to be. This week, we’ll uncover all the mysteries of the problem. A couple of topics are discussed including basic data structures design, various optimization, concurrency and distributed counter.

Each function accepts a timestamp parameter (in seconds granularity) and you may assume that calls are being made to the system in chronological order (i.e. the timestamp is monotonically increasing). You may assume that the earliest timestamp starts at 1.

Example:
HitCounter counter = new HitCounter();

// hit at timestamp 1.
counter.hit(1);

// hit at timestamp 2.
counter.hit(2);

// hit at timestamp 3.
counter.hit(3);

// get hits at timestamp 4, should return 3.
counter.getHits(4);

// hit at timestamp 300.
counter.hit(300);

// get hits at timestamp 300, should return 4.
counter.getHits(300);

// get hits at timestamp 301, should return 3.
counter.getHits(301);

**1. Simple Solution:**

We can use a vector to store all the hits. These two functions are self explainatory.

```
private vector<int> v;
```

```
  /** Record a hit.
            @param timestamp - The current timestamp (in seconds
granularity). */
        void hit(int timestamp) {
            v.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 i, j;
            for (i = 0; i < v.size(); ++i) {
                if (v[i] > timestamp - 300) {
                    break;
                }
            }
            return v.size() - i;
        }
```

**2. Space Optimized Solution:**

We can use a queue to store the hits and delete the entries in queue which are of no use. It will save our space.

```
private queue<int> q;

/** 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();
    }
```

**3. Most optimized solution** :

What if the data comes in unordered and several hits carry the same timestamp.

Since the queue approach wouldn't work without ordered data, this time go with arrays to store the hit count in each unit of time.

If we are tracking hits in the past 5 mins in seconds granularity which is 300 seconds, create 2 arrays of size 300.
int[] hits = new int[300];

TimeStamp[] times = new TimeStamp[300]; // timestamp of the last counted hit
Given an incoming , mod its timestamp by 300 to see where it locates in the hits array.

int idx = timestamp % 300; => hits[idx] keeps the hit count took place in this second

But before we increase the hit count at idx by 1, the timestamp really belongs to the second that hits[idx] is tracking.
timestamp[i] stores the timestamp of the last counted hit.
If timestamp[i] > timestamp, this hit should be discarded since it did not happened in the past 5 minute.
If timestamp[i] == timestamp, then hits[i] increase by 1.
If timestamp[i] < timestamp, reset hits[i] to 1 since what it stored was the hit count before the past 5 minutes.

Upon the call of counter.getHits, traverse hits to get a sum of all hit count with a timestamp[i] > currentTime - 300.

```
vector<int> times, hits;

        times.resize(300);
        hits.resize(300);

    /** Record a hit.
        @param timestamp - The current timestamp (in seconds granularity).
*/
    void hit(int timestamp) {
        int idx = timestamp % 300;
        if (times[idx] != timestamp) {
            times[idx] = timestamp;
            hits[idx] = 1;
        } else {
            ++hits[idx];
        }
    }
```

```
    /** 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;
        for (int i = 0; i < 300; ++i) {
            if (timestamp - times[i] < 300) {
                res += hits[i];
            }
        }
        return res;
    }
```