Cerca i Anàlisi d'Informació (CAI)
Grau en Ciència i Enginyeria de Dades, UPC

# Session 7: Streams and sketches

**Exercise List, Fall 2019**

---

**Basic comprehension questions.**
**Check that you can answer them before proceeding**

1. Define in your words "sketch" and why it is useful.
2. Of the three V's that people use to characterize Big Data (Volume, Velocity, Variety), which ones do you think are addressed by the sketches seen in class?
3. Explain in one sentence what each of the following sketches computes, and how much memory each one uses: Reservoir sampling, Morris' counter, Hyperloglog, SpaceSaving, Exponential Histograms.

---

Note: We use Mb, Gb, to denote Megabytes, Gigabytes, etc. We use Mbit and Gbit for Megabits and Gigabits.

log denotes base 2 logarithm.

---

## Exercise 1

You need to keep 10 billion counters each of which can count up to approximately 900. Describe a data structure that does this in the smallest memory you can think of.

You can assume that no counter will be asked to count more than 900 items. Or that after a counter reaches 900, its result doesn't matter any more.

Each count you give is expected to be correct within 20% approximately.

### Answer of exercise 1

A basic Morris' counter to count up to $t = 900$ uses $\log\log t = \log\log 900 = 3.29$ bits (whatever that means), but has standard deviation about $0.7t$ according to the slides. This means its estimation will typically be off by about 70%.

To be correct within 20%, we need standard deviation $0.2t$. According to the slides, if we replace 2 with some $b < 2$ in the counter, the standard deviation becomes $\sqrt{(b-1)/2} \cdot t$. Solving $\sqrt{(b-1)/2} = 0.2$ gives $b = 1.08$. A counter to count up to $t$ using resolution $b$ uses memory $\log\log t - \log\log b$ Then the memory used by the counter is $\log\log 900 - \log\log 1.08 = 6.46 \simeq 7$ bits.

To have $N$ counters like this, use an array of $7N$ bits $= 7N/8$ bytes; for $N = 10$ billion we thus use 70 Gigabits. The counters are packed - we do not use one byte per counter, but rather the $i$-th counter uses the bits $7i \ldots 7i + 6$. To increment a counter, extract these bits in a byte, increment it, and add the 7 least significant bytes in place (or implement in-place bit addition).

Given the overhead that this bit-by-bit operation introduces, perhaps it is not worth to use 7 bit counters, and instead let us us full bytes, and gain in accuracy. What $b$ can we use that fits in a byte? The $b$ such that $\log\log 900 - \log\log b = 8$, which is $b = 1.027$. Then the standard deviation that we get is $\sqrt{(1.027-1)/2} \cdot t = 0.116$, therefore the count is typically accurate within 11.6%.

## Exercise 2

You are an Internet switch. You see about 1 trillion ($10^{12}$) packets per day, with each packet containing, besides other info, the IPv6 addresses that are the origin and the destination of the package; an IPv6 address has 128 bits.

You have very limited memory for statistics, say 1Mb. Can you do the following tasks? How?

- **Stat1:** Keep every pair of (origin,destination) addresses that have appeared in at least 0.1% of the packets you've seen.
- **Stat2:** Keep the number of distinct pairs (origin,destination) that you have seen in each of the 24 hours, with precision 10%.

The statistics are reset at the start of each day, and after 24 hours you send them somewhere, so you can forget them and start a new cycle.

**Futuristic scenario:** Move forward to the 24th century. You are the largest switch for the internet of the United Federation of Planets, located near the huge black hole at the center of the Milky Way - from which, by the way, you draw your energy. You see a sextillion ($10^{21}$) packets of IPv27 each day. IPv27 was adopted as the pangalactic standard in 2366; an IPv27 address has 1024 bits.

Your job is so trivial for your HyperAI that, out of boredom, you make a bet with your HyperAI buddy in Andromeda: that you can still do both Stat1 and Stat2 in 1Mb. That's the same memory used by the routers of the early 21st century - those primitive things with sub-biological intelligence and infraluminic speed. You once met one in an archaeology museum, and the poor thing couldn't even start the most basic conversation besides work, checksums, and protocol.

Can you win your bet?

## Answer of exercise 2

For Stat1, use a Space Saving sketch. There are at most $k$=1/0.001=1000 heavy hitters, and SpaceSaving uses $O(k)$ memory to keep $k$ heavy hitters, so you can do this with half Mb provided the constant in the $O$ is less than 500 bytes. Which sounds believable, given the description of SpaceSaving and the fact that storing a pair of IPv6 addresses is 128 bytes.

For Stat2, use Hyperloglog. The memory to keep an $\epsilon$-approximate distinct element count up to $n$ is $O((\log\log n)/\epsilon^2)$. For $n \leq 1$ trillion and $\epsilon = 0.1$, $(\log\log 10^{12})/0.01$ is 530. So if hidden constant in the $O$ is less than about 0.5Mb/500 = 1000 bytes for one Hyperloglog, which is believable, it should be doable in much less than 0.5Mb. The slides also mention that counting to 1 billion with precision 2% is doable with a HyperLogLog in a few Kb, which makes us believe that the $O$ constant shouldn't be that large.

So yes with decent implementations of SpaceSaving and HyperLogLog we should be able to do both tasks in 1Mb.

In the 24th century, for $n = 1$ sextillion, the figure for $(\log\log 10^{21})/0.01$ is 610. This is the strength of $\log\log n$ - such a small increase, after a blowup of $10^9$ in $n$.

So for Stat2, let us say that the $O()$ constant is 100 for Hyperloglog (which I think is generous). So one Hyperloglog uses $100 \cdot 670$ bytes = 67Kb, and you have more than 0.9Mb left for Stat1. For Stat1, we need to store one SpaceSaving sketch for 1/0.1% = 1000 heavy hitters, plus a pair of IPv27 addresses for each element in the skech. Assuming that each entry in the SpaceSaving sketch uses 100 bytes (which is generous) and that a pair of IPv27 addresses uses $2 \cdot 1024 = 2048 bits = 256$ bytes, we need $1000 \cdot (100 + 256) = 356,000$ bytes $< 0.9Mb$. So yes, it seems you can win your bet with the Andromeda HyperAI.

## Exercise 3

We are the IPv6 switch in the exercise above. Now, at the end of each day, we are asked to report the fraction of packets that we have seen circulating from each country to each other country. Assume that we can know the country of an IP address, and that there are $C$=200 countries, so $C^2 =$ 40,000 potential pairs (country of origin, country of destination).

For example, if in one day we see 1 trillion packets, and 1 billion of them went from Australia to the US, the report for the pair (Australia,US) should be 0.001. The approximation for each pair should be within 10% of correct.

Use an array of Exponential Histograms (EH) to solve this problem, and answer the following questions:

- What is the length of the sliding window you need to use, $n$?
- What parameter $k$ do you use for each EH?
- How many buckets will each EH have?
- How much memory does a bucket use?
- So, how much memory does one EH use?
- And so, how much memory does the total data structure use?

It is not guaranteed that you can give totally precise answers to all items. Make reasonable guesses. (Think of your boss asking you "but can I do this with 10Mb? Or 1Gb? Or do I need to buy a 100Gb RAM server?", what do you tell her?)

**Answer of exercise 3**

- $n$ is about 1 trillion (maximum number of packets per day).
- $k$ is $1/(2 \cdot 10\%) = 5$.
- The number of buckets is about $k \cdot \log(n/k) = 5 \log(10^{12}/5) \simeq 187$.
- From the description in the slides, EH stores in each bucket the timestamp of the oldest 1 in it, so an integer up to $n$ if we do mod $n$. This is $\log n = 40$ bits or 5 bytes. Add a few more integers for maintaining the data structure. Say 10 4-byte int's per bucket; this is probably a pessimistic figure, and with careful programming one can do with less.
- So one EH should use about $187 \cdot (10 \cdot 4 + 5) \leq 9$Kb.

- And so, the whole data structure uses less than $40,000 \cdot 9$ Kb $\leq 360$Mb.

## Exercise 4

Now you are again the 24th century hyperswitch at the center of the Milky Way. You are asked to compute Stat1 and Stat2 daily, but aggregating by traffic among planetary systems. There are a billion $C = 10^9$ inhabited planetary systems in the Federation, so $C^2 = 10^{18}$ possible counts to keep.

Yet, you know that the traffic among the vast majority of the $C^2$ pairs of planetary system is negligible. So the Federation is only interested in keeping the counts for each pair of systems that are using at least $\theta = 0.0001$ of the bandwidth.

Explain how you approach the problem. You can use one sketch cleverly, or perhaps the combination of several sketches.

## Answer of exercise 4

A difficulty is in the expression "that are using at least…", which admits several interpretations.

One can use a SpaceSaving sketch to remember the at most $1/\theta$ pairs that use at least $\theta$-fraction of the traffic. This sketch uses memory $O(1/\theta)$ times the size used by a pair of elements in $C$. The latter is $2\log(C) = 59$ bits. So the memory should be $(1/0.0001) \cdot (c + 59/8)$ bytes where $c$ is a reasonable constant depending on the implementation of SpaceSaving. This should be less than 1Mb.

If "are using at least…" is interpreted as "that yesterday used at least …", then you can transfer the SpaceSaving from yesterday to today to know which $1/\theta$ pairs you need to monitor today. And keep a table that tells you how many packets you saw today for each of the pairs.

If "are using at least …" means something stricter, like "in the last 24 hours (but not necessarily 24 hours of the same day)" or "in the last hour" or so, then you need to use sliding windows. One can use Exponential Histograms that are created and deleted dynamically. When some element $i$ is stored in the SpaceSaving, you create an EH for it and start monitoring its traffic. When $i$ is removed from the SpaceSaving, you remove its associated EH and make place for a new element in the SpaceSaving, and its EH.

The memory will be $O(1/\theta)$ for the SpaceSaving, which is small, and $O((1/\epsilon)\log n)$ for each EH, where $\epsilon$ is the desired accuracy for each count. Assume $\theta = 0.0001$, $\epsilon = 0.1$ for example, and $n = 10^{21}$. If we keep an EH for each element of the SpaceSaving, the memory is in the order of

$10^3 \cdot 10 \cdot \log(10^{21}) \simeq 700,000$ times the constant implicit in the $O()$ of the EH sketch. (a tighter estimate can be done as in the previous problem).

Alternatively, a Reservoir sampling sketch could also be used for keeping a sample of the pairs (origin,destination) seen recently, and rebuild the EH and/or SpaceSaving at regular times. The theory is that this is less efficient for small $\epsilon$.