Chapter 37

Streaming

> I don't know why it should be, I am sure; but the sight of another man asleep in bed when I am up, maddens me. It seems to me so shocking to see the precious hours of a man's life - the priceless moments that will never come back to him again - being wasted in mere brutish sleep.
>
> Jerome K. Jerome, Three men in a boat

By Sariel Har-Peled, November 28, 2018[1]

37.1. How to sample a stream

Imagine that you are given a stream of elements $s_1, s_2, \ldots$, and you need to sample k numbers from this stream (say, without repetition) – assume that you do not know the length of the stream in advance, and furthermore, you have only $O(k)$ space available for you. How to do that efficiently?

There are two natural schemes:

(A) Whenever an element arrives, generate a random number for it in the range $[0,1]$. Maintain a heap with the k elements with the lowest priority. Implemented naively this requires $O(\log k)$ comparisons after each insertion, but it is not difficult to improve this to $O(1)$ comparisons in the amortized sense per insertion. Clearly, the resulting set is the desired random sample

(B) Let S_t be the random sample maintained in the tth iteration. When the ith element arrives, the algorithm flip a coin that is heads with probability $\min(1, k/i)$. If the coin is heads then it inserts s_i to S_{i-1} to get S_i. If S_{i-1} already have k elements, then first randomly delete one of the elements.

Theorem 37.1.1. *Given a stream of elements, one can uniformly sample k elements (without repetition), from the stream using $O(k)$ space, where $O(1)$ time is spent for handling each incoming element.*

Proof: We implement the scheme (B) above. We only need to argue that this is a uniform random sample. The claim trivially hold for $i = k$. So assume the claim holds for $i < t$, and we need to prove that the set after getting tth element is still a uniform random sample.

So, consider a specific set $K \subseteq \{s_1, \ldots, s_t\}$ of k elements. The probability of K to be a random sample of size k from a set of t elements is $1/\binom{t}{k}$. We need to argue that this probability remains the same for this scheme.

So, if $s_t \notin K$, then we have

$$\mathbb{P}[K = S_t] = \mathbb{P}[K = S_{t-1} \text{ and } s_t \text{ was not inserted}] = \frac{1}{\binom{t-1}{k}}\left(1 - \frac{k}{t}\right) = \frac{k!(t-1-k)!(t-k)}{(t-1)!t} = \frac{1}{\binom{t}{k}}.$$

If $s_t \in K$, then

$$\mathbb{P}\big[K = S_t\big] = \mathbb{P}\left[\begin{array}{c} K \setminus \{s_t\} \subseteq S_{t-1}, \\ s_t \text{ was inserted} \\ \text{and } S_{t-1} \setminus K \text{ thrown out of } S_{t-1} \end{array}\right] = \frac{t-1-(k-1)}{\binom{t-1}{k}} \frac{k}{t} \frac{1}{k} = \frac{(t-k)k!(t-1-k)!}{(t-1)!t} = \frac{1}{\binom{t}{k}},$$

as desired. Indeed, there are $t-1-(k-1)$ subsets of size k of $\{s_1, \ldots, s_{t-1}\}$ that contains $K \setminus \{s_t\}$ – since we fix $k-1$ of the $t-1$ elements. ∎

37.2. Sampling and median selection

Let $B[1, \ldots, n]$ be a set of n numbers. We would like to estimate the median, without computing it outright. A natural idea, would be to pick k elements $e_1, \ldots, e_k$ randomly from B, and return their median as the guess for the median of B.

In the following, let $R_B(t)$ be the tth smallest number in the array B.

Observation 37.2.1. *For $\varepsilon \in (0,1)$, we have that $\frac{1}{1-\varepsilon} \geq 1 + \varepsilon$.*

Lemma 37.2.2. *Let $\varepsilon \in (0, 1/2)$, and let $k = \lceil \frac{12}{\varepsilon^2} \ln \frac{2}{\delta} \rceil$. Let Z be the median of the random sample of B of size k. We have that*

$$\mathbb{P}\left[R_B\left(\frac{1-\varepsilon}{2}n\right) \leq Z \leq R_B\left(\frac{1+\varepsilon}{2}n\right)\right] \geq 1 - \delta.$$

Namely, with probability at least $1-\delta$, the returned value Z is $(\varepsilon/2)n$ positions away from the true median.

Proof: Let $L = R_B((1-\varepsilon)n/2)$. Let $X_i = 1$ if and only if $e_i \leq L$. We have that

$$\mathbb{P}[X_i = 1] = \frac{(1-\varepsilon)n/2}{n} = \frac{1-\varepsilon}{2}.$$

As such, setting $Y = \sum_{i=1}^{k} X_i$, we have

$$\mu = \mathbb{E}[Y] = \frac{1-\varepsilon}{2}k \geq \frac{k}{4} \geq \frac{3}{\varepsilon^2} \ln \frac{2}{\delta}.$$

One case is that the algorithm fails, if $Y \geq k/2$. We have that

$$\mathbb{P}[Y \geq k/2] = \mathbb{P}\left[Y \geq \frac{1/2}{(1-\varepsilon)/2} \cdot \frac{1-\varepsilon}{2}k\right] = \mathbb{P}[Y \geq (1+\varepsilon)\mu] \leq \exp\big(-\varepsilon^2\mu/3\big) \leq \exp\left(-\varepsilon^2 \cdot \frac{3}{\varepsilon^2} \ln \frac{2}{\delta}\right) \leq \frac{\delta}{2}.$$

by Chernoff's inequality (see Theorem 37.5.1).

This implies that $\mathbb{P}\big[R_B((1-\varepsilon)n/2) > Z\big] \leq \delta/2$.

The claim now follows by realizing that by symmetry (i.e., revering the order), we have that $\mathbb{P}\big[Z > R_B((1+\varepsilon)n/2)\big] \leq \delta/2$, and putting these two inequalities together. ∎

The above already implies that we can get a good estimate for the median. We need something somewhat stronger – we state it without proof since it follows by similarly mucking around with Chernoff's inequality.

Lemma 37.2.3. *Let $\varepsilon \in (0,1/2)$, let B an array of n elements, and let $S = \{e_1, \ldots, e_k\}$ be a set of k samples picked uniformly and randomly from B. Then, for some absolute constant c, and an integer k, such that $k \geq \left\lceil \frac{c}{\varepsilon^2} \ln \frac{1}{\delta} \right\rceil$, we have that*

$$\mathbb{P}\left[R_S(k_-) \leq R_B(n/2) \leq R_S\left(k^+\right)\right] \geq 1 - \delta.$$

for $k_- = \lfloor (1-\varepsilon)k/2 \rfloor$, and $k^+ = \lfloor (1+\varepsilon)k/2 \rfloor$.

One can prove even a stronger statement:

$$\mathbb{P}\left[R_B((1-2\varepsilon)n/2) \leq R_S((1-\varepsilon)k/2) \leq R_B(n/2) \leq R_S((1+\varepsilon)k/2) \leq R_B((1+2\varepsilon)n/2)\right] \geq 1 - \delta$$

(the constant c would have to be slightly bigger).

37.2.1. A median selection with few comparisons

The above suggests a natural algorithm for computing the median (i.e., the element of rank $n/2$ in B). Pick a random sample S of $k = O(\sqrt{n} \log n)$ elements. Next, sort S, and pick the elements L and R of ranks $(1-\varepsilon)k$ and $(1+\varepsilon)k$ in S, respectively. Next, scan the elements, and compare them to L and R, and keep only the elements that are between. In the end of this process, we have computed:

(A) α: The rank of the number L in the set B.

(B) $T = \{x \in B \mid L \leq x \leq H\}$.

Compute, by brute force (i.e., sorting) the element of rank $n/2 - \alpha$ in T. Return it as the desired median. If $n/2 - \alpha$ is negative, then the algorithm failed, and it tries again.

Lemma 37.2.4. *The above algorithm performs $2n + O(n^{3/4} \log n)$ comparisons, and reports the median. This holds with high probability.*

Proof: Set $\varepsilon = 1/n^{1/4}$, and $\delta = 1/n^{O(1)}$, and observe that Lemma 37.2.3 implies that with probability $\geq 1 - 1/\delta$, we have that the desired median is between L and H. In addition, Lemma 37.2.3 also implies that $|T| \leq 4\varepsilon n \leq 4n^{3/4}$, which readily implies the correctness of the algorithm.

As for the bound on the number of comparisons, we have, with high probability, that the number of comparisons is

$$O(|S| \log |S| + |T| \log |T|) + 2n = O\left(\sqrt{n} \log^2 n + n^{3/4} \log n\right) + 2n,$$

since deciding if an element is between L and H requires two comparisons. ∎

Lemma 37.2.5. *The above algorithm can be modified to perform $(3/2)n + O(n^{3/4} \log n)$ comparisons, and reports the median correctly. This holds with high probability.*

Proof: The trick is to randomly compare each element either first to L or first to H with equal probability. For elements that are either smaller than L or bigger than H, this requires $(3/2)n$ comparisons in expectation. Thus improving the bound from $2n$ to $(3/2)n$. ∎

Remark 37.2.6. Note, that if we know, as in this case, that L and H are in the middle, than it is not needed to do the random comparisons trick used above – indeed, just regular algorithm would work. This trick makes sense only if do not know the rank of L and H in the real array, but only know that they are close together. Then, the random comparisons trick does work better than the deterministic approach.

Lemma 37.2.7. *Consider a stream B of n numbers, and assume we can make two passes over the data. Then, one can compute exactly the median of B using:*
(I) $O(n^{3/4})$ *space.*
(II) $1.5n + O(n^{3/4} \log n)$ *comparisons.*
The algorithm reports the median correctly, and it succeeds with high probability.

Proof: Implement the above algorithm, using the random sampling from Theorem 37.1.1. ∎

37.3. Big data and the streaming model

Here, we are interested in doing some computational tasks when the amount of data we have to handle is quite large (think terabytes or larger). The main challenge in many of these cases is that even reading the data once is expensive. Running times of $O(n \log n)$ might not be acceptable. Furthermore, in many cases, we can load all the data into memory.

In the ***streaming*** model, one reads the data as it comes in, but one can not afford to keep all the data. A natural example would be a internet router, which has gazillion of packets going through it every minute. We might still be interested in natural questions about these packets, but we want to do this without storing all the packets.

37.4. Heavy hitters

Imagine a stream $s_1, \ldots$, where elements might repeat, and we would like to maintain a list of elements that appear at least εn times. We present a simple but clever scheme that maintains such a list.

The algorithm. To this end, let

$$k = \lceil 1/\varepsilon \rceil .$$

At each point in time, we maintain a set S of k elements, with a counter for each element. Let S_t be the version of S after t were inserted. When s_{t+1} arrives, we increase its counter if it is already in S_t. If $|S_t| < k$, then we just insert s_{t+1} to the set, and set its counter to 1. Otherwise, $|S_t| = k$ and $s_{t+1} \notin S_t$. We then decrease all the k counters of elements in S_t by 1. If a counter of an element in S_{t+1} is zero, then we delete it from the set.

37.5. Chernoff inequality

Proving the specific form of Chernoff's inequality we need is outside our scope. The interested reader is referred to notes here `https://sarielhp.org/p/notes/16/chernoff/chernoff.pdf`. We next state what we need:

Theorem 37.5.1. *Let $X_1, \ldots, X_n$ be n independent Bernoulli trials, where $\mathbb{P}[X_i = 1] = p_i$, and $\mathbb{P}[X_i = 0] = 1 - p_i$, for $i = 1, \ldots, n$. Let $X = \sum_{i=1}^{b} X_i$, and $\mu = \mathbb{E}\big[X\big] = \sum_i p_i$. For $\delta, \in (0,1)$, we have*

$$\mathbb{P}\big[X > (1+\delta)\mu\big] < \exp\big(-\mu\delta^2/3\big).$$

Bibliography