Streaming and sketching
The main idea

Sometimes data is too big to fit into memory...

**Sketching**

Represent data by a structure of sublinear size
Preferably $O(1)$; at most $n^{1-\varepsilon}$ for some $\varepsilon > 0$
Keep enough information to solve what you want either exactly or approximately

**Streaming**

Read through data once, in some given order
Use a sketch to represent information about it
Produce a result from the sketch
Example: Sketch for the average

Sketch for the average of a collection of numbers: pair \((n, t)\)

\(n\) is the number of elements in the collection

\(t\) is the sum of the elements

Operations:

- Incorporate new element \(x\) into collection:
  
  Add one to \(n\) and add \(x\) to \(t\)

- Remove element from collection:
  
  Subtract one from \(n\) and subtract \(x\) from \(t\)

- Compute average of collection:
  
  Return \(t/n\)

Total space is \(O(1)\) regardless of how big \(n\) is

(Time/operation is also \(O(1)\) but our main concern is space)
Example: Streaming average

To compute the average of a sequence $S$:
- Initialize an empty sketch for the average
- For each $x$ in the sequence, incorporate $x$ into the sketch
- Return the average of the sketch

Time for an $n$-element sequence is $O(n)$
Working space (not counting the space for the input) is $O(1)$

This works even when the sequence is generated somehow in a way that does not involve local storage (e.g. internet traffic)
Cash register versus turnstile

Two different variations of streaming algorithms:

Cash register:
Money goes in, doesn’t come back out

Meaning for streaming: Input has insertions, no removals

Turnstile:
People can either enter or exit

Meaning for streaming: Input has both insertions and removals

Streaming average was presented in cash register model but the same sketch also works for turnstile model
Sketches can sometimes be combined to produce information about multiple data sets.

Example: Can produce an average-sketch for the concatenation of two sequences by adding the sketches of the two sequences.

If we have $n$ sequences, of total length $N$, we can compute all pairwise averages in time $O(n^2 + N)$, faster than computing each pairwise average separately.
Medians and reservoir sampling
Medians

Median: Sort given list of $n$ items; which one goes to position $n/2$?

Like average, an important measure of the central tendency of a sample, more robust against arbitrarily-corrupted outliers

Applications in facility location: Given $n$ points on a line, place a center minimizing the average distance to the given points
Impossibility of streaming median

No streaming algorithm can compute the median!

Consider sketch after seeing the first $n/2$ items in a stream
Depending on the remaining items, any one of these first $n/2$ could become the median
of the whole stream
So we have to remember all their values

Space is $\Omega(n)$, not $O(n^{1-\varepsilon})$ for some $\varepsilon > 0$
Approximate medians

We want the item at position $\frac{1}{2}n$ in sorted sequence

Relaxation: $\delta$-approximate median
Position should be between $(\frac{1}{2} - \delta)n$ and $(\frac{1}{2} + \delta)n$

Easy randomized method:
- Maintain a sample of $\Theta((1/\delta)^2)$ sequence members
- Return the median of the sample

Chernoff $\Rightarrow$ constant probability of being in desired position
Bigger constant in sample size $\Rightarrow$ better probability

(Later this lecture: a non-random method that is always within the desired approximation bound, but more complicated and with non-constant space complexity)
More detailed analysis of approximate medians

Let sample size be $2c/\delta^2$

Random event: one of our samples is less than the median

Expected number of random events: $c/\delta^2$.

Our estimate is too big when actual number of events is larger by a factor of $1 + 2\delta$

Chernoff (case for multiplier close to one): probability this happens is $\leq e^{-(2\delta)^2 \cdot c/\delta^2 \cdot \frac{1}{3}}$

The deltas cancel $\Rightarrow$ constant probability of success; symmetric for estimate too small

Increase $c$ for greater constant; use $c \approx \log n$ for probability $\approx 1 - \frac{1}{n}$
Reservoir sampling

How to maintain a sample of $k$ random elements (without replacement) from a longer sequence?

Store an array $A$ of size $k$, and the number $n$ (initially 0)

For each element $x$:

- If $n < k$:
  - Store $x$ in $A[n]$

- Else:
  - Choose a random integer $i$ from 0 to $n$
  - If $i < k$, store $x$ in $A[i]$

Increment $n$

More careful: one random number per sample (when to replace) instead of per input
MinHash
Hashing for reservoir sampling

We saw how to maintain a sample of \( k \) elements, using a random choice for each element.

Instead, make a single random choice, of a hash function \( h(x) \).

Sample = the \( k \) elements with the smallest values of \( h \).

Two advantages:

- Less use of (possibly slow) random number generation.
- Repeatable: if different agents use the same hash function to sample the same sets, they will get the same samples.

We can take advantage of repeatability to use this for more applications!

[Broder 1997]
Combining sets with MinHash

To compute the union:

$$\text{MinHash}(S \cup T) = \text{MinHash}(\text{MinHash}(S) \cup \text{MinHash}(T))$$

Because any element that is one of the $k$ smallest hashes in the union must also be one of the $k$ smallest in either of the smaller sets that contain it.

We can get useful information about sets $S$ and $T$ from this computation!

Key question: how many elements from $\text{MinHash}(S) \cup \text{MinHash}(T)$ survive into $\text{MinHash}(S \cup T)$?
Before the success of Google, a different company (DEC) ran a different popular search engine (AltaVista)

They had a problem: Many web pages can be found in multiple similar (but not always identical) copies

Search engines should return only one copy of the same page, so user can see other results without getting swamped by duplicates

⇒ Need to quickly test similarity of web pages
“Bag of words” model: represent any web page by the set of words used in it

Similar documents (web pages) should have similar words

So after converting web pages to bags of words, the search engine problem becomes: quickly test similarity of pairs of sets
Jaccard similarity

A standard measure of how similar two sets are:

\[ J(S, T) = \frac{|S \cap T|}{|S \cup T|} \]

Scale-invariant (normalized for how big the two sets are)

Ranges from 0 to 1
- 0 when sets are disjoint
- close to 0 when sets are not very similar
- close to 1 when sets are similar
- 1 when sets are identical
Estimating Jaccard similarity

Theorem:

\[
J(S, T) = \frac{1}{k} E \left[ |\text{MinHash}(S \cup T) \cap \text{MinHash}(S) \cap \text{MinHash}(T)| \right]
\]

The intersection consists of the elements of \( S \cap T \) that also belong to \( \text{MinHash}(S \cup T) \)

Linearity of expectation \( \Rightarrow \)

\[
E[\ldots] = \sum_{x \in S \cup T} \Pr[x \in \text{intersection of MinHashes}]
= \sum_{x \in S \cup T} \Pr[x \in S \cap T] \times \Pr[x \in \text{MinHash}(S \cup T)]
= \sum_{x \in S \cup T} \frac{|S \cap T|}{|S \cup T|} \times \frac{k}{|S \cup T|}
= |S \cup T| \times J(S, T) \times \frac{k}{|S \cup T|} = k J(S, T)
\]
How accurate is it?

Suppose we use

$$\frac{1}{k} |\text{MinHash}(S \cup T) \cap \text{MinHash}(S) \cap \text{MinHash}(T)|$$

as an estimate for $J(S, T)$

Previous analysis shows that this has the correct expected value:
it is an unbiased estimator

Our analysis decomposed the estimate into a sum of independent 0-1 random variables
(is each member of $S \cup T$ in the intersection)

$\Rightarrow$ Can use Chernoff bounds for being within $1 \pm \varepsilon$ of expectation

To get expected error $|J(S, T) - \text{estimate}| \leq \varepsilon$, set $k = \Theta(1/\varepsilon^2)$
Summary of MinHash

Main idea: sketch any set by choosing the $k$ elements that have the smallest hash values

Can reduce randomness in reservoir sampling

Sketches of unions can be computed from unions of sketches

Provides accurate estimates of Jaccard similarity

All-pairs similarity of $n$ sets with total length $N$: time $O(n^2 + N)$
Deterministic sampling
Small sketches exist (but can we find them?)

Given a sequence $S$ of $n$ numbers (allowing repetitions), define the quantile $q_S(x)$ of a number $x$ to be the fraction of sequence elements that are less than $x$.

(So median = element whose quantile is 1/2)

$\varepsilon$-approximation: subsequence $\hat{S}$ of $S$ such that $q_{\hat{S}} \approx q_S$:

$$\forall x \ |q_{\hat{S}}(x) - q_S(x)| \leq \varepsilon$$

It's easy to construct an $\varepsilon$-approximation of size $1/\varepsilon$: just choose $1/\varepsilon$ numbers equally spaced in the sorted sequence.

The median of an $\varepsilon$-approximation would accurately approximate the median of $S$. 
## Composition of epsilon-approximations

### Concatenation

If we concatenate two equal-length sequences $S$ and $T$ to form $S \, T$, and both have equal-length $\varepsilon$-approximations $\hat{S}$ and $\hat{T}$, then $\hat{S} \, \hat{T}$ is an $\varepsilon$-approximation of $S \, T$

But now it’s twice as long

### Approximation of approximation

A $\delta$-approximation of the $\varepsilon$-approximation $\hat{S} \, \hat{T}$ is a $(\delta + \varepsilon)$-approximation of size $1/\delta$

Worse approximation but short again

Idea: Use concatenation to combine approximations from subsequences and then use approximation to shorten the result
Hierarchical sketching, basic idea

For each block $B$ of $2^i$ consecutive numbers in the input stream, store an approximation, computed as follows:

- Break $B$ into two blocks of $2^{i-1}$ consecutive numbers
- Concatenate the approximations stored for these two blocks
- Compute an approximation of the concatenation
  
  (Merge two sorted lists and then take evenly-spaced items from the result)
Hierarchical sketching, parameterized

Choose an approximation parameter $\varepsilon_i$ for each $i$

For each block $B$ of $2^i$ consecutive numbers in the input stream, store an approximation, computed as follows:

- Break $B$ into two blocks of $2^{i-1}$ consecutive numbers
- Concatenate the $\varepsilon_{i-1}$-approximations stored for these two blocks
- Compute a $(\varepsilon_i - \varepsilon_{i-1})$-approximation of the concatenation
  (Merge two sorted lists and then take evenly-spaced items from the result)
Updating a hierarchical sketch

To insert a new element:

Create a new block of one element \( i = 0 \)
(even if there is already another one-element block)

While two blocks have the same size \( 2^i \):

- Merge them into one block of size \( 2^{i+1} \)
- Concatenate their \( \epsilon_i \)-approximations
- Construct a \((\epsilon_{i+1} - \epsilon_i)\)-approximation of the concatenation

Once we have merged two blocks, forget their approximations
(We only store the remaining unmerged blocks, at most one of each size)
Using a hierarchical sketch

To approximate the quantile of any value $x$:

- Use the approximations to find $q_B(x)$ for each block
- return the weighted average (weighted by block size)

To find an $\varepsilon$-approximate median:

- Find $x$ whose approximate quantile is near $1/2$. (Can be found using a weighted median in the union of block approximations.)
- Its position in the whole input is within $\varepsilon \cdot n$ positions of the median
How big are these sketches?

For any $\delta > 0$, the sum

$$\sum_{i=0}^{\infty} \frac{1}{(i + 1)^{1+\delta}}$$

has a finite sum $s_\delta$

Set

$$\varepsilon_{i+1} - \varepsilon_i = \frac{\varepsilon}{s_\delta} \cdot \frac{1}{(i + 1)^{1+\delta}}$$

$\Rightarrow$ sequence of parameters $\varepsilon_i$ converges in telescoping sum to desired $\varepsilon$

Summing resulting sketch sizes for all blocks gives space $O(\varepsilon^{-1}(\log n)^{2+\delta})$

[Bagchi et al. 2007]
Deterministic sampling and range counting in geometric data streams. ACM Trans. 

Dave Braunschweig. Internet word cloud. CC-BY-SA licensed image, October 15 2016. URL 
https://commons.wikimedia.org/wiki/File:Internet_word_cloud.png.

Andrei Z. Broder. On the resemblance and containment of documents. In Bruno Carpentieri, 
Alfredo De Santis, Ugo Vaccaro, and James A. Storer, editors, Compression and Complexity 
of SEQUENCES 1997, Positano, Amalfitan Coast, Salerno, Italy, June 11-13, 1997, 

Sara Eppstein. Smoke in a strawberry box and Mist in an egg carton.

Ingolf. Warszawa - metro linia M2 - stacja Świętokrzyska. CC-BY-SA licensed image, March 26 

Basile Morin. Credit card terminal in Laos. CC-BY-SA licensed image, April 24 2018. URL 
https://commons.wikimedia.org/wiki/File:Credit_card_terminal_in_Laos.jpg.