SAPIENZA
UNIVERSITÀ DI ROMA

Algorithms for data streams

Irene Finocchi

finocchi@di.uniroma1.it

http://www.dsi.uniroma1.it/∼finocchi/

May 9, 2012

Outline

Warnings

Goals:

- give a flavor of the theoretical results and techniques of data stream algorithmics
- only a representative sample of each topic: many other problems, algorithms, and techniques not covered in these lectures (non-exhaustive overview at the end of the talk)

Warnings

Goals:

- give a flavor of the theoretical results and techniques of data stream algorithmics
- only a representative sample of each topic: many other problems, algorithms, and techniques not covered in these lectures (non-exhaustive overview at the end of the talk)

Math contents: some probability ahead (e.g., Chernoff bounds). Will introduce basic tools along the way.

Warnings

Goals:

- give a flavor of the theoretical results and techniques of data stream algorithmics
- only a representative sample of each topic: many other problems, algorithms, and techniques not covered in these lectures (non-exhaustive overview at the end of the talk)

Math contents: some probability ahead (e.g., Chernoff bounds). Will introduce basic tools along the way.

Request

- If you get bored, ask questions

Warnings

Goals:

- give a flavor of the theoretical results and techniques of data stream algorithmics
- only a representative sample of each topic: many other problems, algorithms, and techniques not covered in these lectures (non-exhaustive overview at the end of the talk)

Math contents: some probability ahead (e.g., Chernoff bounds). Will introduce basic tools along the way.

Request

- If you get bored, ask questions
- If you get lost, ask questions

Warnings

Goals:

- give a flavor of the theoretical results and techniques of data stream algorithmics
- only a representative sample of each topic: many other problems, algorithms, and techniques not covered in these lectures (non-exhaustive overview at the end of the talk)

Math contents: some probability ahead (e.g., Chernoff bounds). Will introduce basic tools along the way.

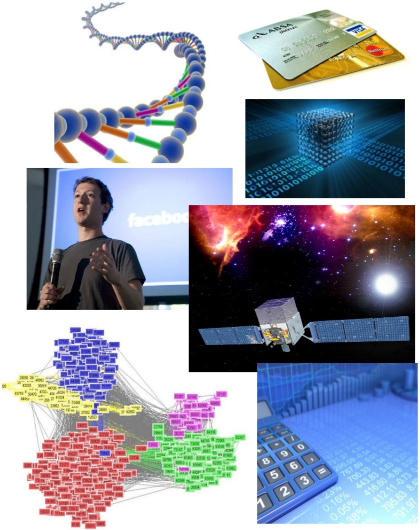
Request

- If you get bored, ask questions
- If you get lost, ask questions
- If you'd like to ask questions, ask questions

Massive data

Data is growing faster than our ability to store and index it:

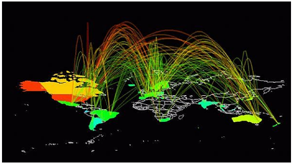
- networking: phone call networks, Internet, social networks
- scientific data: astronomical data, genome sequences, GIS geo-spatial data
- economic transactions: credit cards, online auctions
- ...



Network management

Monitoring flow of IP packets through the routers (Internet traffic):

- how many IP addresses used a given link in the last month?
- which are the top 100 IP addresses in terms of traffic?
- which destinations use most bandwidth?

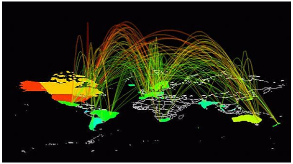


- what's the average duration of an IP session?
- which hosts have similar usage patterns (clusters)?
- does traffic distribution change in different periods of time?

Network management

Monitoring flow of IP packets through the routers (Internet traffic):

- how many IP addresses used a given link in the last month?
- which are the top 100 IP addresses in terms of traffic?
- which destinations use most bandwidth?



- what's the average duration of an IP session?
- which hosts have similar usage patterns (clusters)?
- does traffic distribution change in different periods of time?

Up to 1 Billion packets per hour per router
Many hundreds of routers per ISP

⇒ Many terabytes of data per hour!

Sensor data

Sensors with GPS unit deployed in the ocean:

- Each sensor reports surface height (4-byte real number) every tenth of second
- Base station receives 3.5 MB per day per sensor

Sensor data

Sensors with GPS unit deployed in the ocean:

- Each sensor reports surface height (4-byte real number) every tenth of second
- Base station receives 3.5 MB per day per sensor

What about a million sensors?

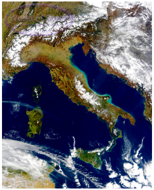
3.5 TB of data per day, coming at a high rate

A million sensors isn't very many: roughly one sensor per 150 square miles of ocean...

More streams...

Image data

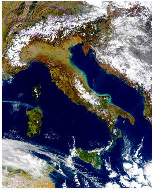
- satellites send down to earth many TBs of images per day
- surveillance cameras produce roughly one image per second: London has about six millions such cameras



More streams...

Image data

- satellites send down to earth many TBs of images per day
- surveillance cameras produce roughly one image per second: London has about six millions such cameras



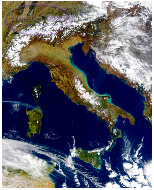
Web traffic

Google receives several hundreds million search queries per day

More streams...

Image data

- satellites send down to earth many TBs of images per day
- surveillance cameras produce roughly one image per second: London has about six millions such cameras



Web traffic

Google receives several hundreds million search queries per day

Economic trend analysis

- in online auction systems, users continuously submit bids for items and items for auction

Issues in data stream processing

Some features common to all these applications:

- huge volumes of data (terabytes, even petabytes)
- records arrive at a rapid rate
- need to monitor data continuously to support exploratory analyses and to detect correlations, patterns, rare events, fraud, intrusion, unusual activities

Issues in data stream processing

Some features common to all these applications:

- huge volumes of data (terabytes, even petabytes)
- records arrive at a rapid rate
- need to monitor data continuously to support exploratory analyses and to detect correlations, patterns, rare events, fraud, intrusion, unusual activities

Many problems about streaming data would be easy to solve if we had enough memory, but require new techniques for realistic data rates and sizes

What can be computed without even storing the input?

Basic data stream model

Data stream = sequence $\sigma = \langle a_1, a_2, \ldots a_m \rangle$
of tokens drawn from universe $[n] = \{1, 2, \ldots n\}$

Basic data stream model

Data stream = sequence $\sigma = \langle a_1, a_2, \ldots a_m \rangle$
of tokens drawn from universe $[n] = \{1, 2, \ldots n\}$

Input parameters: m and n

1. Stream σ is massively long. Stream length m is:
   - typically unknown
   - possibly infinite

Basic data stream model

Data stream = sequence $\sigma = \langle a_1, a_2, \ldots a_m \rangle$
of tokens drawn from universe $[n] = \{1, 2, \ldots n\}$

Input parameters: m and n

1. Stream σ is massively long. Stream length m is:
   - typically unknown
   - possibly infinite
2. Universe size n is also typically very large
   (e.g., IP addresses, URLs, item prices)

Performance metrics

Minimize space, passes, and processing time upon token arrivals

Performance metrics

Minimize space, passes, and processing time upon token arrivals

1. Use a sublinear amount of space s:

$$s = o(\min\{n, m\})$$

where $s =$ bits of random-access working memory

Performance metrics

Minimize space, passes, and processing time upon token arrivals

1. Use a sublinear amount of space s:

$$s = o(\min\{n, m\})$$

where s = bits of random-access working memory

2. Make p passes over the data, for some small integer p (no random access to tokens)

Performance metrics

Minimize space, passes, and processing time upon token arrivals

1. Use a sublinear amount of space s:
$$s = o(\min\{n, m\})$$
where s = bits of random-access working memory
2. Make p passes over the data, for some small integer p (no random access to tokens)
3. Use small per-item processing time t

Performance metrics

Minimize space, passes, and processing time upon token arrivals

1. Use a sublinear amount of space s:

$$s = o(\min\{n, m\})$$

   where s = bits of random-access working memory

2. Make p passes over the data, for some small integer p (no random access to tokens)

3. Use small per-item processing time t

"You're looking for the Holy Grail? Have you tried Ebay?"

Performance metrics

Minimize space, passes, and processing time upon token arrivals

1. Use a sublinear amount of space s:
$$s = o(\min\{n, m\})$$
where s = bits of random-access working memory
2. Make p passes over the data, for some small integer p (no random access to tokens)
3. Use small per-item processing time t

$$\begin{cases} s = O(\log m + \log n) \\ \text{Happy if } s = O(polylog(\min\{n, m\})) \\ p = 1 \\ t = O(1) \end{cases}$$

Token frequencies

Data stream = sequence $\sigma = \langle a_1, a_2, ...a_m \rangle$
of tokens drawn from universe $[n] = \{1, 2, ...n\}$

Token frequencies

Data stream = sequence $\sigma = \langle a_1, a_2, \ldots a_m \rangle$
of tokens drawn from universe $[n] = \{1, 2, \ldots n\}$

σ represents a multiset of items and implicitly defines a frequency vector

$$f = \langle f_1, f_2, \ldots f_n \rangle$$

where f_i = number of occurrences of item $i \in [n]$ in σ

Example

If $\sigma = \langle 2, 1, 2, 1, 5, 2, 3, 2 \rangle$ and $n = 5$, then $f = \langle 2, 4, 1, 0, 1 \rangle$

Token frequencies

Data stream = sequence $\sigma = \langle a_1, a_2, ...a_m \rangle$
of tokens drawn from universe $[n] = \{1, 2, ...n\}$

σ represents a multiset of items and implicitly defines a frequency vector

$$f = \langle f_1, f_2, ...f_n \rangle$$

where f_i = number of occurrences of item $i \in [n]$ in σ

Example

If $\sigma = \langle 2, 1, 2, 1, 5, 2, 3, 2 \rangle$ and $n = 5$, then $f = \langle 2, 4, 1, 0, 1 \rangle$

In many streaming problems, wish to compute some statistical properties of the multiset: e.g., majority token (if any), most frequent items, or number of distinct items

Variations of the basic setup

Data stream =
sequence of **tuples** $\sigma = \langle (a_1, c_1), (a_2, c_2), \ldots \rangle$
where $(a_i, c_i) \in [n] \times \{-F, \ldots, F\}$

Upon arrival of (a_i, c_i)), update frequency $f_{a_i} = f_{a_i} + c_i$

New role for m: $m = \sum_{j=1}^{n} f_j$

Variations of the basic setup

Data stream =
sequence of **tuples** $\sigma = \langle (a_1, c_1), (a_2, c_2), \dots \rangle$
where $(a_i, c_i) \in [n] \times \{-F, \dots, F\}$

Upon arrival of (a_i, c_i)), update frequency $f_{a_i} = f_{a_i} + c_i$

New role for m: $m = \sum_{j=1}^{n} f_j$

Basic data stream model: $c_i = 1$ (m = stream length)

Variations of the basic setup

Data stream =
sequence of **tuples** $\sigma = \langle (a_1, c_1), (a_2, c_2), \dots \rangle$
where $(a_i, c_i) \in [n] \times \{-F, \dots, F\}$

Upon arrival of (a_i, c_i)), update frequency $f_{a_i} = f_{a_i} + c_i$

New role for m: $m = \sum_{j=1}^{n} f_j$

Basic data stream model: $c_i = 1$ ($m =$ stream length)

Cash register model: $c_i > 0$ (items can only arrive, their frequencies can be incremented by variable amounts)

Variations of the basic setup

Data stream =
sequence of **tuples** $\sigma = \langle (a_1, c_1), (a_2, c_2), \ldots \rangle$
where $(a_i, c_i) \in [n] \times \{-F, \ldots, F\}$

Upon arrival of (a_i, c_i)), update frequency $f_{a_i} = f_{a_i} + c_i$

New role for m: $m = \sum_{j=1}^{n} f_j$

Basic data stream model: $c_i = 1$ (m = stream length)

Cash register model: $c_i > 0$ (items can only arrive, their frequencies can be incremented by variable amounts)

Turnstile model: generic c_i (items can arrive and depart from the multiset)

Historical remarks

Origin in the 70s (seminal paper by Munro & Paterson, STOC'78)

Historical remarks

Origin in the 70s (seminal paper by Munro & Paterson, STOC'78)

Gained popularity in the last fifteen years:

- theoretical interest:
  - easy-to-state, but hard-to-solve problems
  - links to other theory areas and to novel computing paradigms (MapReduce)

Historical remarks

Origin in the 70s (seminal paper by Munro & Paterson, STOC'78)

Gained popularity in the last fifteen years:

- theoretical interest:
  - easy-to-state, but hard-to-solve problems
  - links to other theory areas and to novel computing paradigms (MapReduce)
- practical appeal: fast and effective solutions, wide applicability

Historical remarks

Origin in the 70s (seminal paper by Munro & Paterson, STOC'78)

Gained popularity in the last fifteen years:

- theoretical interest:
  - easy-to-state, but hard-to-solve problems
  - links to other theory areas and to novel computing paradigms (MapReduce)
- practical appeal: fast and effective solutions, wide applicability

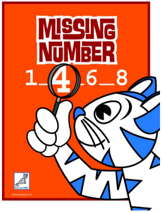
Alon, Matias & Szegedy: Gödel prize (2005) for their paper on frequency moments approximation (STOC'96, JCSS'99), foundational work for streaming and sketching algorithms

Three puzzles

Data stream challenges

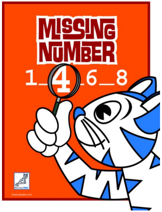
The missing number puzzle

$\pi = \langle \pi_1, \pi_2, ...\pi_{n-1} \rangle$
is a permutation of $[1, n]$
with one number missing



The missing number puzzle

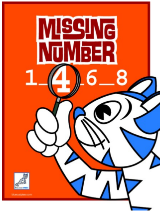
$\pi = \langle \pi_1, \pi_2, \ldots \pi_{n-1} \rangle$
is a permutation of $[1, n]$
with one number missing



What's the missing number?

The missing number puzzle

$\pi = \langle \pi_1, \pi_2, \ldots \pi_{n-1} \rangle$
is a permutation of $[1, n]$
with one number missing

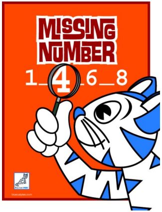


What's the missing number?

Constraint: Carole has limited memory: she can only use $O(\log n)$ bits

The missing number puzzle

$\pi = \langle \pi_1, \pi_2, \ldots \pi_{n-1} \rangle$
is a permutation of $[1, n]$
with one number missing



What's the missing number?

Constraint: Carole has limited memory: she can only use $O(\log n)$ bits

$$\frac{n(n-1)}{2} - \sum_{i=1}^{n-1} \pi_i$$

Two missing numbers

Now π has two missing numbers, x and y: find them, but use only $O(\log n)$ bits!

Two missing numbers

Now π has two missing numbers, x and y: find them, but use only $O(\log n)$ bits!

Track $\begin{cases} S = \frac{n(n+1)}{2} - \sum_{i=1}^{n-2} \pi_i \\ P = n! - \prod_{i=1}^{n-2} \pi_i \end{cases}$

Solve equations $x + y = S$ and $x\,y = P$

Two missing numbers

Now π has two missing numbers, x and y: find them, but use only $O(\log n)$ bits!

Track
$$\begin{cases} S = \frac{n(n+1)}{2} - \sum_{i=1}^{n-2} \pi_i \\ P = n! - \prod_{i=1}^{n-2} \pi_i \end{cases}$$

Solve equations $x + y = S$ and $x\,y = P$

How many bits? $\Omega(\log n!) = \Omega(n \log n)$

Two missing numbers

Now π has two missing numbers, x and y: find them, but use only $O(\log n)$ bits!

Two missing numbers

Now π has two missing numbers, x and y: find them, but use only $O(\log n)$ bits!

$$\text{Track} \begin{cases} S_1 = \frac{n(n-1)}{2} - \sum_{i=1}^{n-2} \pi_i \\ S_2 = \frac{n(n+1)(2n+1)}{6} - \sum_{i=1}^{n-2} \pi_i^2 \end{cases}$$

Solve equations $x + y = S_1$ and $x^2 + y^2 = S_2$

Two missing numbers

Now π has two missing numbers, x and y: find them, but use only $O(\log n)$ bits!

Track $\begin{cases} S_1 = \frac{n(n-1)}{2} - \sum_{i=1}^{n-2} \pi_i \\ S_2 = \frac{n(n+1)(2n+1)}{6} - \sum_{i=1}^{n-2} \pi_i^2 \end{cases}$

Solve equations $x + y = S_1$ and $x^2 + y^2 = S_2$

How many bits? $O(\log n^3) = O(\log n)$

Lesson 1

Some problems can be deterministically solved in:

- logarithmic space
- one pass

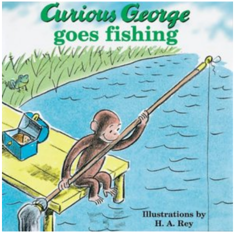
Lesson 1

Some problems can be deterministically solved in:

- logarithmic space
- one pass

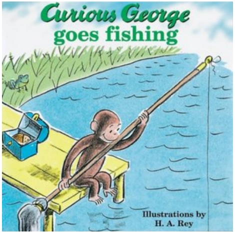
Most of the times, we're not so lucky

Fishing



$U = \{1, \ldots u\}$ fish species in the universe

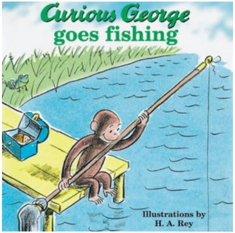
Fishing



$U = \{1, ...u\}$ fish species in the universe

$a_t \in U$ fish species caught at time t

Fishing

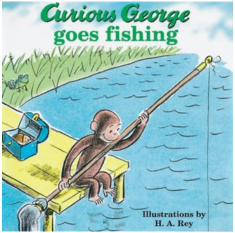


$U = \{1, \ldots u\}$ fish species in the universe

$a_t \in U$ fish species caught at time t

$f_t[j] = |\{a_i \mid a_i = j, i \leq t\}|$ frequency of species j up to time t

Fishing



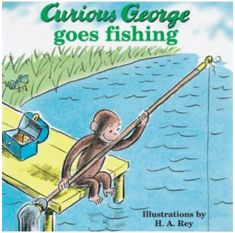
$U = \{1, \ldots u\}$ fish species in the universe

$a_t \in U$ fish species caught at time t

$f_t[j] = |\{a_i \mid a_i = j, i \leq t\}|$ frequency of species j up to time t

j is rare iff $f_t[j] = 1$

Fishing



$U = \{1, \ldots u\}$ fish species in the universe

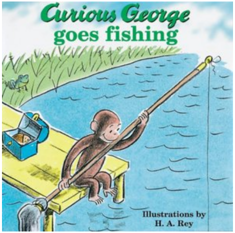
$a_t \in U$ fish species caught at time t

$f_t[j] = |\{a_i \mid a_i = j, i \leq t\}|$ frequency of species j up to time t

j is rare iff $f_t[j] = 1$

Rarity of catch at time t: $\rho_t = \dfrac{|\{j \mid f_t[j] = 1\}|}{u} = \dfrac{R_t}{u}$

Fishing



$U = \{1, \dots u\}$ fish species in the universe

$a_t \in U$ fish species caught at time t

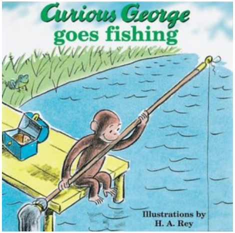
$f_t[j] = |\{a_i \mid a_i = j, i \leq t\}|$ frequency of species j up to time t

j is rare iff $f_t[j] = 1$

Rarity of catch at time t: $\rho_t = \frac{|\{j \mid f_t[j] = 1\}|}{u} = \frac{R_t}{u}$

- George is curious and wants to compute rarity

Fishing



$U = \{1, \ldots u\}$ fish species in the universe

$a_t \in U$ fish species caught at time t

$f_t[j] = |\{a_i \mid a_i = j, i \le t\}|$ frequency of species j up to time t

j is rare iff $f_t[j] = 1$

Rarity of catch at time t: $\rho_t = \frac{|\{j \mid f_t[j] = 1\}|}{u} = \frac{R_t}{u}$

- George is curious and wants to compute rarity
- $2u$-bit vector would suffice
- … but George's suitcase has $o(u)$ size

Deterministic fish rarity

George cannot compute ρ_t precisely with a deterministic algorithm using only o(u) bits

By contradiction

Deterministic fish rarity

George cannot compute ρ_t precisely with a deterministic algorithm using only o(u) bits

By contradiction

Let $S \subseteq U$ be a set of species: no duplicates, $|S| = \Theta(u)$

Need $\Omega(|S|) = \Omega(u)$ bits to represent S

If claim is false, could break information theoretic lower bound

Deterministic fish rarity

George cannot compute ρ_t precisely with a deterministic algorithm using only o(u) bits

By contradiction

Let $S \subseteq U$ be a set of species: no duplicates, $|S| = \Theta(u)$

Need $\Omega(|S|) = \Omega(u)$ bits to represent S

If claim is false, could break information theoretic lower bound

To retrieve S, for each $i \in U$, stream $\langle S, i \rangle$ to George and compare ρ_t and ρ_{t+1}:

Deterministic fish rarity

George cannot compute ρ_t precisely with a deterministic algorithm using only o(u) bits

By contradiction

Let $S \subseteq U$ be a set of species: no duplicates, $|S| = \Theta(u)$

Need $\Omega(|S|) = \Omega(u)$ bits to represent S

If claim is false, could break information theoretic lower bound

To retrieve S, for each $i \in U$, stream $\langle S, i \rangle$ to George and compare ρ_t and ρ_{t+1}:

- if $i \notin S$, then $R_{t+1} = R_t + 1$ and $\rho_{t+1} > \rho_t$

Deterministic fish rarity

George cannot compute ρ_t precisely with a deterministic algorithm using only o(u) bits

By contradiction

Let $S \subseteq U$ be a set of species: no duplicates, $|S| = \Theta(u)$

Need $\Omega(|S|) = \Omega(u)$ bits to represent S

If claim is false, could break information theoretic lower bound

To retrieve S, for each $i \in U$, stream $\langle S, i\rangle$ to George and compare ρ_t and ρ_{t+1}:

- if $i \notin S$, then $R_{t+1} = R_t + 1$ and $\rho_{t+1} > \rho_t$
- if $i \in S$, then $R_{t+1} = R_t - 1$ and $\rho_{t+1} < \rho_t$

Deterministic fish rarity

George cannot compute ρ_t precisely with a deterministic algorithm using only o(u) bits

By contradiction

Let $S \subseteq U$ be a set of species: no duplicates, $|S| = \Theta(u)$

Need $\Omega(|S|) = \Omega(u)$ bits to represent S

If claim is false, could break information theoretic lower bound

To retrieve S, for each $i \in U$, stream $\langle S, i \rangle$ to George and compare ρ_t and ρ_{t+1}:

- if $i \notin S$, then $R_{t+1} = R_t + 1$ and $\rho_{t+1} > \rho_t$
- if $i \in S$, then $R_{t+1} = R_t - 1$ and $\rho_{t+1} < \rho_t$

Hence ρ decreases $\Leftrightarrow i \in S$

Randomized fish rarity (1/2)

George can approximate ρ_t using $2k = o(u)$ bits

Sampling:

- pick k random fish species
- maintain rarity $c_1[t], \dots c_k[t]$ of each sampled species (2 bits)
- Return $\widetilde{\rho}_t = \dfrac{|\{i \in [1,k] \mid c_i[t] = 1\}|}{k} = \dfrac{\widetilde{R_t}}{k}$

Randomized fish rarity (1/2)

George can approximate ρ_t using $2k = o(u)$ bits

Sampling:

- pick k random fish species
- maintain rarity $c_1[t], \dots c_k[t]$ of each sampled species (2 bits)
- Return $\widetilde{\rho}_t = \dfrac{|\{i \in [1,k] \mid c_i[t] = 1\}|}{k} = \dfrac{\widetilde{R_t}}{k}$

Claim: $E[\widetilde{\rho}_t] = \rho_t$

Randomized fish rarity (1/2)

George can approximate ρ_t using $2k = o(u)$ bits

Sampling:

- pick k random fish species
- maintain rarity $c_1[t], \dots c_k[t]$ of each sampled species (2 bits)
- Return $\widetilde{\rho}_t = \dfrac{|\{i \in [1,k] \mid c_i[t] = 1\}|}{k} = \dfrac{\widetilde{R_t}}{k}$

Claim: $E[\widetilde{\rho}_t] = \rho_t$

- If ρ_t large enough, $\widetilde{\rho}_t$ is a good estimate for ρ_t with arbitrarily small precision and good probability
- Requires more advanced probabilistic tools: examples later

Randomized fish rarity (2/2)

$$\widetilde{\rho}_t = \frac{|\{i \in [1,k] \mid c_i[t] = 1\}|}{k} = \frac{\widetilde{R_t}}{k}$$

$$E[\widetilde{\rho}_t] = \rho_t$$

Randomized fish rarity (2/2)

$$\widetilde{\rho}_t = \frac{|\{i \in [1,k] \mid c_i[t] = 1\}|}{k} = \frac{\widetilde{R_t}}{k}$$

$$E[\widetilde{\rho}_t] = \rho_t$$

Y_i indicator variable: $\begin{cases} Y_i = 1 & \text{if } c_i[t] = 1 \\ Y_i = 0 & \text{otherwise} \end{cases}$

Randomized fish rarity (2/2)

$$\widetilde{\rho}_t = \frac{|\{i \in [1,k] \mid c_i[t] = 1\}|}{k} = \frac{\widetilde{R_t}}{k}$$

$$E[\widetilde{\rho}_t] = \rho_t$$

$$Y_i \text{ indicator variable: } \begin{cases} Y_i = 1 & \text{if } c_i[t] = 1 \\ Y_i = 0 & \text{otherwise} \end{cases}$$

$$Pr\{Y_i = 1\} = Pr\{\text{the i-th sampled species is rare}\} = \frac{R_t}{u} = \rho_t$$

Randomized fish rarity (2/2)

$$\widetilde{\rho}_t = \frac{|\{i \in [1,k] \mid c_i[t] = 1\}|}{k} = \frac{\widetilde{R_t}}{k}$$

$$E[\widetilde{\rho}_t] = \rho_t$$

Y_i indicator variable: $\begin{cases} Y_i = 1 & \text{if } c_i[t] = 1 \\ Y_i = 0 & \text{otherwise} \end{cases}$

$$Pr\{Y_i = 1\} = Pr\{\text{the i-th sampled species is rare}\} = \frac{R_t}{u} = \rho_t$$

$$\Rightarrow \quad E[Y_i] = \rho_t$$

Randomized fish rarity (2/2)

$$\widetilde{\rho}_t = \frac{|\{i \in [1,k] \mid c_i[t] = 1\}|}{k} = \frac{\widetilde{R_t}}{k}$$

$$E[\widetilde{\rho}_t] = \rho_t$$

$$Y_i \text{ indicator variable: } \begin{cases} Y_i = 1 & \text{if } c_i[t] = 1 \\ Y_i = 0 & \text{otherwise} \end{cases}$$

$$Pr\{Y_i = 1\} = Pr\{\text{the i-th sampled species is rare}\} = \frac{R_t}{u} = \rho_t$$

$$\Rightarrow \quad E[Y_i] = \rho_t$$

$$\Rightarrow \quad E[\widetilde{R_t}] = \sum_{i=1}^{k} E[Y_i] = k\rho_t$$

Randomized fish rarity (2/2)

$$\widetilde{\rho}_t = \frac{|\{i \in [1,k] \mid c_i[t] = 1\}|}{k} = \frac{\widetilde{R_t}}{k}$$

$$E[\widetilde{\rho}_t] = \rho_t$$

Y_i indicator variable: $\begin{cases} Y_i = 1 & \text{if } c_i[t] = 1 \\ Y_i = 0 & \text{otherwise} \end{cases}$

$$Pr\{Y_i = 1\} = Pr\{\text{the i-th sampled species is rare}\} = \frac{R_t}{u} = \rho_t$$

$$\Rightarrow \quad E[Y_i] = \rho_t$$

$$\Rightarrow \quad E[\widetilde{R_t}] = \sum_{i=1}^{k} E[Y_i] = k\rho_t$$

$$\Rightarrow \quad E[\widetilde{\rho}_t] = \frac{E[\widetilde{R_t}]}{k} = \rho_t$$

Lesson 2

It is often impossible to solve problems precisely and deterministically in small (sublinear) space

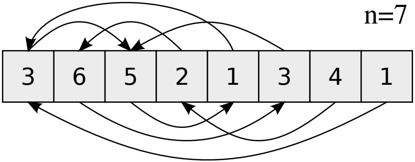
Randomization and approximation greatly help:

- find an answer correct within some factor (guarantee that $\widetilde{\rho}$ is within 10% of ρ)
- allow a small probability of failure (answer is correct, except with probability 1 in 10,000)

Pointer and chaser

Paul has $n+1$ pointers

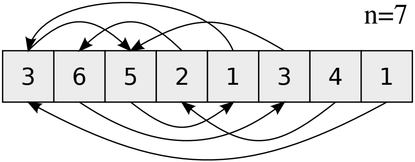
For each pointer i, he points to a position $P[i] \in [1, n]$



Pointer and chaser

Paul has $n+1$ pointers

For each pointer i, he points to a position $P[i] \in [1, n]$

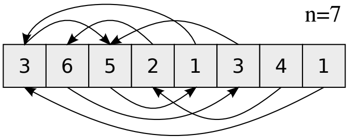


Carole has to guess *any* duplicate pointer

Constraints:

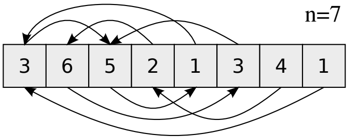
- $O(\log n)$ bits
- $O(n)$ queries
- cannot move items

Repeated scans



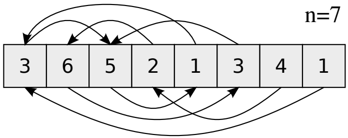
1. Trivial solution
   - for each i, count how many j are such that P[j]=i
   - $O(\log n)$ bits, but $O(n^2)$ queries

Repeated scans

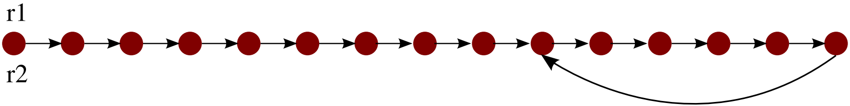


1. Trivial solution
   - for each i, count how many j are such that P[j]=i
   - $O(\log n)$ bits, but $O(n^2)$ queries
2. Better solution
   - if # of items below $n/2 >$ # of items above $n/2$
     then search for duplicates $< n/2$
     else search for duplicates $\geq n/2$
   - $O(\log n)$ bits and passes, $O(n \log n)$ queries
3. With $O(\log n)$ bits, $\Omega(\log n/\log\log n)$ passes are needed

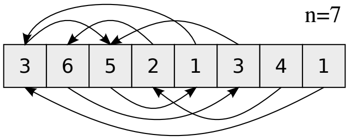
Random access helps



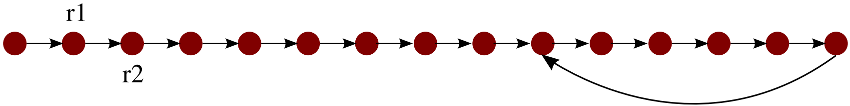
- Chase pointers, starting from position $n+1$
- Problem equivalent to finding a loop in a linked list
- Can be solved in $O(n)$ time with just 2 pointers!



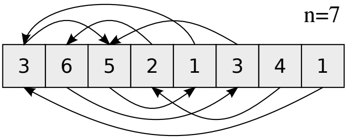
Random access helps



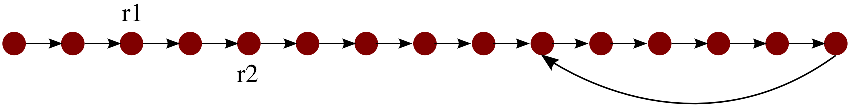
- Chase pointers, starting from position $n+1$
- Problem equivalent to finding a loop in a linked list
- Can be solved in $O(n)$ time with just 2 pointers!



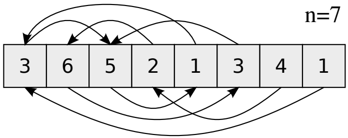
Random access helps



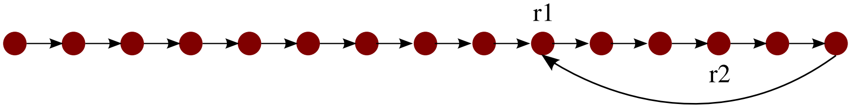
- Chase pointers, starting from position $n + 1$
- Problem equivalent to finding a loop in a linked list
- Can be solved in $O(n)$ time with just 2 pointers!



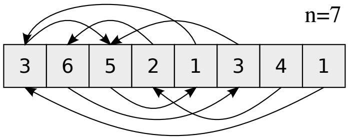
Random access helps



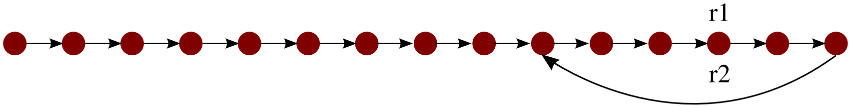
- Chase pointers, starting from position $n+1$
- Problem equivalent to finding a loop in a linked list
- Can be solved in $O(n)$ time with just 2 pointers!



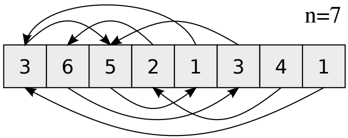
Random access helps



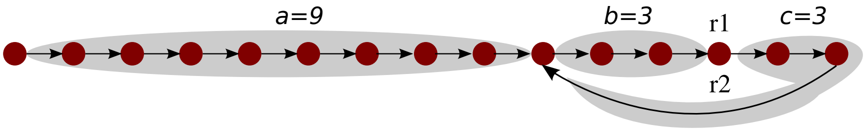
- Chase pointers, starting from position $n+1$
- Problem equivalent to finding a loop in a linked list
- Can be solved in $O(n)$ time with just 2 pointers!



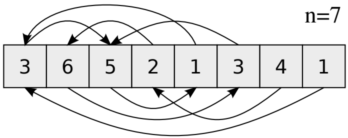
Random access helps



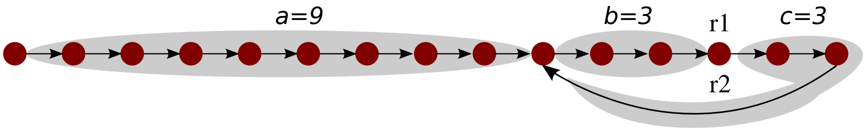
- Chase pointers, starting from position $n+1$
- Problem equivalent to finding a loop in a linked list
- Can be solved in $O(n)$ time with just 2 pointers!



Random access helps



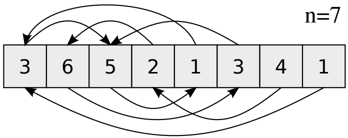
- Chase pointers, starting from position $n+1$
- Problem equivalent to finding a loop in a linked list
- Can be solved in $O(n)$ time with just 2 pointers!



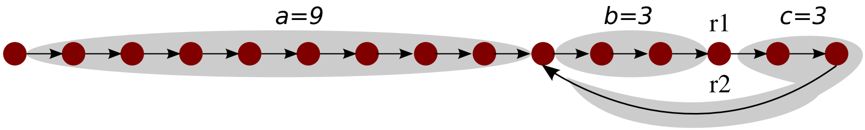
$$\begin{cases} a+b=t \\ a+k(b+c)+b=2t \end{cases}$$

t and k known

Random access helps



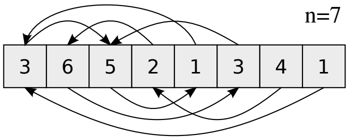
- Chase pointers, starting from position $n+1$
- Problem equivalent to finding a loop in a linked list
- Can be solved in $O(n)$ time with just 2 pointers!



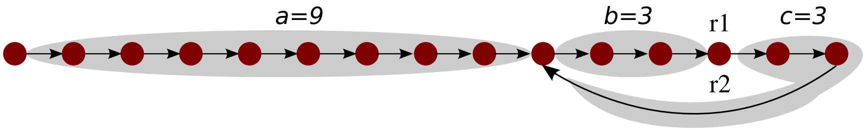
$$\begin{cases} a+b=t \\ a+k(b+c)+b=2t \end{cases} \Rightarrow \begin{cases} a+b=t \\ b+c=t/k \end{cases}$$

t and k known

Random access helps



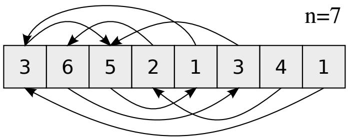
- Chase pointers, starting from position $n+1$
- Problem equivalent to finding a loop in a linked list
- Can be solved in $O(n)$ time with just 2 pointers!



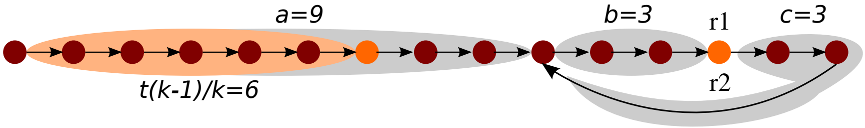
$$\begin{cases} a+b=t \\ a+k(b+c)+b=2t \end{cases} \Rightarrow \begin{cases} a+b=t \\ b+c=t/k \end{cases} \Rightarrow a=c+\frac{k-1}{k}t$$

t and k known

Random access helps



- Chase pointers, starting from position $n+1$
- Problem equivalent to finding a loop in a linked list
- Can be solved in $O(n)$ time with just 2 pointers!

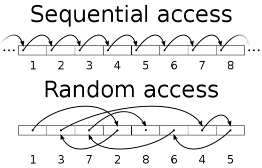


$$\begin{cases} a+b=t \\ a+k(b+c)+b=2t \end{cases} \Rightarrow \begin{cases} a+b=t \\ b+c=t/k \end{cases} \Rightarrow a=c+\frac{k-1}{k}t$$

t and k known

Lesson 3

Tokens come as a stream: no random access

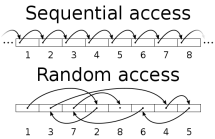


Sometimes impossible to achieve the same bounds as in the RAM model

Recap on lessons

Typically impossible to solve problems precisely and deterministically in small (sublinear) space

Randomize and approximate!

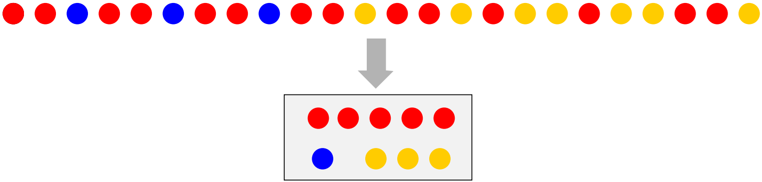


Sequential data access makes things harder

Sampling

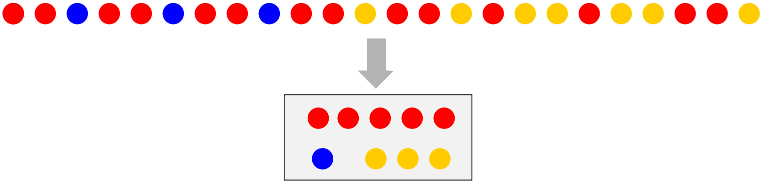
Working with less

Why sampling?



- Basic problem: sample s items uniformly from a stream
- Answer queries (e.g., compute fish species rarity) on the sample
- Utility depends on the problem: in some cases, sampling-based approaches not effective unless taking large (almost linear) samples

Why sampling?



- Basic problem: sample s items uniformly from a stream
- Answer queries (e.g., compute fish species rarity) on the sample
- Utility depends on the problem: in some cases, sampling-based approaches not effective unless taking large (almost linear) samples

> How can we sample uniformly
> if we don't know in advance how long is the stream?
> When do we sample a stream token?

Reservoir sampling

1. Add to S the first s stream items
2. Upon seeing x_i at time, sample x_i with probability s/i
3. If x_i added to S, evict a random item from S (other than x_i)

Sample is uniform

At any time t and for each $i \leq t$, it holds: $Pr\{x_i \in_t S\} = \dfrac{s}{t}$

Reservoir sampling

1. Add to S the first s stream items
2. Upon seeing x_i at time, sample x_i with probability s/i
3. If x_i added to S, evict a random item from S (other than x_i)

Sample is uniform

At any time t and for each $i \leq t$, it holds: $Pr\{x_i \in_t S\} = \frac{s}{t}$

Warmup analysis: $s = 1$

$$Pr\{x_i \in_t S\} =$$

$$= Pr\{x_i \text{ sampled at time } i\} \times Pr\{x_i \text{ survives up to time } t\} =$$

$$= \frac{1}{i} \times \frac{i}{i+1} \times \frac{i+1}{i+2} \times \ldots \times \frac{t-2}{t-1} \times \frac{t-1}{t} = \frac{1}{t}$$

Arbitrary sample size s: analysis

Sample is uniform: $Pr\{x_i \in_t S\} = \frac{s}{t}$

Arbitrary sample size s: analysis

Sample is uniform: $Pr\{x_i \in_t S\} = \frac{s}{t}$

By induction on t (base step: $t \leq s$)

Arbitrary sample size s: analysis

Sample is uniform: $Pr\{x_i \in_t S\} = \frac{s}{t}$

By induction on t (base step: $t \leq s$)

How does S change at time t when x_t arrives?

Arbitrary sample size s: analysis

Sample is uniform: $Pr\{x_i \in_t S\} = \frac{s}{t}$

By induction on t (base step: $t \leq s$)

How does S change at time t when x_t arrives?

1. $Pr\{x_t \text{ added to } S\} = \frac{s}{t}$

Arbitrary sample size s: analysis

Sample is uniform: $Pr\{x_i \in_t S\} = \frac{s}{t}$

By induction on t (base step: $t \leq s$)

How does S change at time t when x_t arrives?

1. $Pr\{x_t \text{ added to } S\} = \frac{s}{t}$
2. Inductive hypothesis: $Pr\{x_i \in_{t-1} S\} = \frac{s}{t-1}$

Arbitrary sample size s: analysis

Sample is uniform: $Pr\{x_i \in_t S\} = \frac{s}{t}$

By induction on t (base step: $t \leq s$)

How does S change at time t when x_t arrives?

1. $Pr\{x_t \text{ added to } S\} = \frac{s}{t}$
2. Inductive hypothesis: $Pr\{x_i \in_{t-1} S\} = \frac{s}{t-1}$
3. $Pr\{x_i \in_t S \,|\, x_t \text{ added to } S\} = Pr\{x_i \in_{t-1} S \text{ and not evicted}\} =$
   $= \frac{s}{t-1}\left(1 - \frac{1}{s}\right)$

Arbitrary sample size s: analysis

Sample is uniform: $Pr\{x_i \in_t S\} = \frac{s}{t}$

By induction on t (base step: $t \leq s$)

How does S change at time t when x_t arrives?

1. $Pr\{x_t \text{ added to } S\} = \frac{s}{t}$
2. Inductive hypothesis: $Pr\{x_i \in_{t-1} S\} = \frac{s}{t-1}$
3. $Pr\{x_i \in_t S \,|\, x_t \text{ added to } S\} = Pr\{x_i \in_{t-1} S \text{ and not evicted}\} = \frac{s}{t-1}\left(1 - \frac{1}{s}\right)$
4. $Pr\{x_i \in_t S \,|\, x_t \text{ not added to } S\} = Pr\{x_i \in_{t-1} S\} = \frac{s}{t-1}$

Arbitrary sample size s: analysis

Sample is uniform: $Pr\{x_i \in_t S\} = \frac{s}{t}$

By induction on t (base step: $t \leq s$)

How does S change at time t when x_t arrives?

1. $Pr\{x_t \text{ added to } S\} = \frac{s}{t}$
2. Inductive hypothesis: $Pr\{x_i \in_{t-1} S\} = \frac{s}{t-1}$
3. $Pr\{x_i \in_t S \,|\, x_t \text{ added to } S\} = Pr\{x_i \in_{t-1} S \text{ and not evicted}\} =$ $= \frac{s}{t-1}\left(1 - \frac{1}{s}\right)$
4. $Pr\{x_i \in_t S \,|\, x_t \text{ not added to } S\} = Pr\{x_i \in_{t-1} S\} = \frac{s}{t-1}$

By combining conditional probabilities:

$$Pr\{x_i \in_t S\} = \frac{s}{t}\frac{s}{t-1}\left(1 - \frac{1}{s}\right) + \left(1 - \frac{s}{t}\right)\frac{s}{t-1} = \frac{s}{t}$$

Optimizations and drawbacks

Skip numbers

Instead of flipping a coin at each stream element, generate number of elements to be skipped before the next element is added to S [Vitter 85]

Optimizations and drawbacks

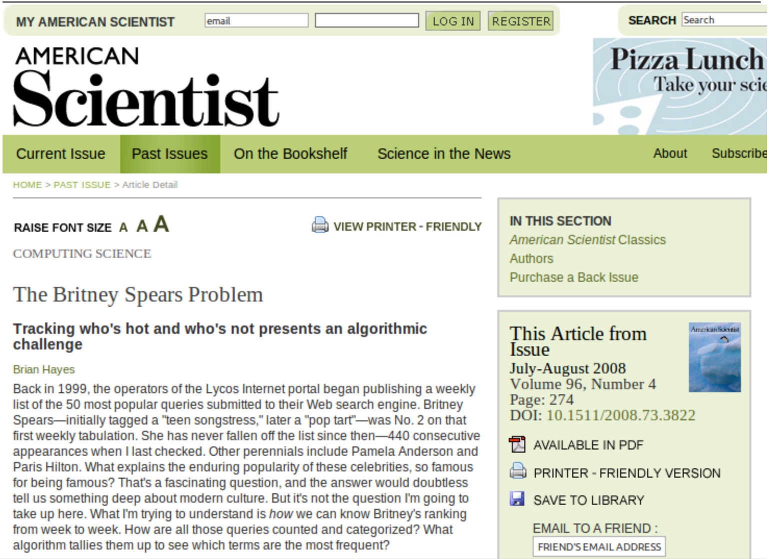
Skip numbers

Instead of flipping a coin at each stream element, generate number of elements to be skipped before the next element is added to S [Vitter 85]

Other issues:

- Frequently occurring values are a wasteful use of the available sample space: concise sampling [Gibbons and Matias '98]
- Runs into difficulties in the presence of data deletions: [Babcock *et al.* '02]
- Hard to parallelize on multiple streams: how do we sample if more than one item comes at any time? Min-wise sampling [Nath *et al.* '04]

The Britney Spears problem...



MY AMERICAN SCIENTIST email LOG IN REGISTER SEARCH Search

AMERICAN Scientist

Pizza Lunch
Take your sci

Current Issue | Past Issues | On the Bookshelf | Science in the News | About | Subscribe

HOME > PAST ISSUE > Article Detail

RAISE FONT SIZE A A A VIEW PRINTER - FRIENDLY

COMPUTING SCIENCE

The Britney Spears Problem

Tracking who's hot and who's not presents an algorithmic challenge

Brian Hayes

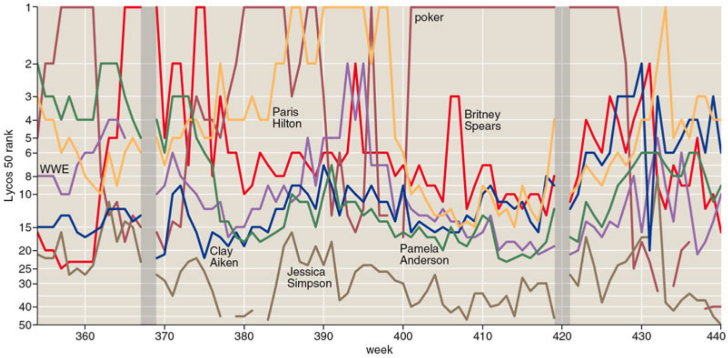
Back in 1999, the operators of the Lycos Internet portal began publishing a weekly list of the 50 most popular queries submitted to their Web search engine. Britney Spears—initially tagged a "teen songstress," later a "pop tart"—was No. 2 on that first weekly tabulation. She has never fallen off the list since then—440 consecutive appearances when I last checked. Other perennials include Pamela Anderson and Paris Hilton. What explains the enduring popularity of these celebrities, so famous for being famous? That's a fascinating question, and the answer would doubtless tell us something deep about modern culture. But it's not the question I'm going to take up here. What I'm trying to understand is *how* we can know Britney's ranking from week to week. How are all those queries counted and categorized? What algorithm tallies them up to see which terms are the most frequent?

IN THIS SECTION
American Scientist Classics
Authors
Purchase a Back Issue

This Article from Issue
July-August 2008
Volume 96, Number 4
Page: 274
DOI: 10.1511/2008.73.3822

AVAILABLE IN PDF
PRINTER - FRIENDLY VERSION
SAVE TO LIBRARY
EMAIL TO A FRIEND :
FRIEND'S EMAIL ADDRESS

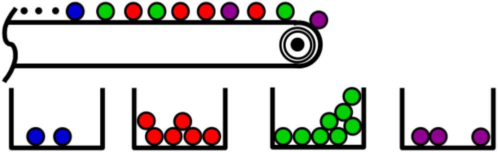
... tracking who's hot and who's not



"... can't just pay attention to a few popular subjects, because you can't know in advance which ones are going to rank near the top. To be certain of catching every new trend as it unfolds, you have to monitor *all* the incoming queries – and their variety is unbounded. "

Heavy hitters

Given a stream of n items, find those that appear "most frequently"



E.g., items occurring more than 1% of the time

- Formally "hard" in small space, so allow approximation
- No false negatives: return all items with count $\geq \varphi n$
- "Good" false positives: no item with count $< (\varphi - \varepsilon)n$ is returned (error $\varepsilon \in (0, 1)$, $\varepsilon \ll \varphi$)
- Related problem: estimate each frequency with error $\pm\varepsilon n$

Why heavy hitters?

- Many practical applications: mining of search logs, analysis of network data, DBMS optimization...
- Core streaming problem: connections with entropy estimation, itemsets mining, compressed sensing
- Extensive research: scores of streaming papers on frequent items and its variations

We'll see a counter-based algorithm named Sticky sampling:

1. probabilistic, sampling-based approach
2. correct with probability $\geq 1 - \delta$, with $\delta \in (0, 1)$ user-specified probability of failure

Sticky sampling

Intuition

It should be possible to estimate frequent items by a good sample

Data structure S: set of pairs $\langle x, f_e(x)\rangle$, where

- $f_e(x)$ estimated frequency of x
- $f(x)$ true frequency

Query algorithm: at time n report items $x \in S$ such that $f_e(x) \geq (\varphi - \varepsilon)n$

Update algorithm works in rounds:

- each round distinguished by a (fixed) sampling rate r
- sampling rate adjusted between rounds so that probability of sampling a stream item decreases as stream gets longer

Update algorithm

Structure of r-rate round

For each stream item x:

1. if $x \in S$, then increase $f_e(x)$ by 1
2. if $x \notin S$, sample x with probability $\frac{1}{r}$: if x sampled, add pair $\langle x, 1 \rangle$ to S

At the end of a round:

1. double sampling rate r (r increases geometrically)
2. adjust estimated frequencies so that S is transformed into exactly the state it would have been in, if new rate $2r$ had been used from the beginning

Adjusting frequencies

Assume x sampled at time k with probability $\frac{1}{r}$:

- $f_e(x) =$ exact number of occurrences of x after time k
- with smaller sampling probability $(\frac{1}{2r})$, x will be sampled at one of the later occurrences
- simulate all coin tosses not done with sampling rate r

For each $\langle x, f_e(x)\rangle \in S$ repeatedly toss a coin:

1. first coin toss unbiased ($\frac{1}{2}$, makes probability of sampling x at time $k = \frac{1}{2r}$)
2. next coin tosses biased with probability $\frac{1}{2r}$
3. for each unsuccessful coin toss, decrease $f_e(x)$ by 1
4. stop when coin toss successful or $f_e(x) = 0$ (in this case remove x from S)

Round length

Recall: $\begin{cases} \varphi = \text{frequency threshold} \\ \varepsilon = \text{frequency error} \\ \delta = \text{algorithm failure probability} \end{cases}$

Let $t = \frac{1}{\varepsilon} \log \frac{1}{\varphi\delta}$

	2t	2t	4t	8t	...
rate	1	2	4	8	...

r-rate round has length rt (except for $r = 1$)

expected sample size: $2t$ (we'll prove)

A technical lemma

For each rate $r \geq 2$, let n be the number of stream items considered up to the r-rate round. It holds:

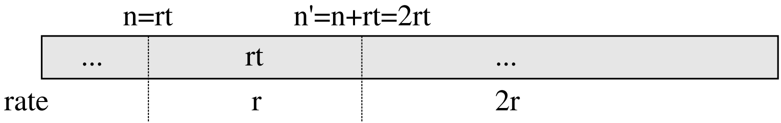
$$\frac{1}{r} \geq \frac{t}{n}$$

A technical lemma

For each rate $r \geq 2$, let n be the number of stream items considered up to the r-rate round. It holds:

$$\frac{1}{r} \geq \frac{t}{n}$$

By induction, at the beginning of r-rate round $n = rt$:



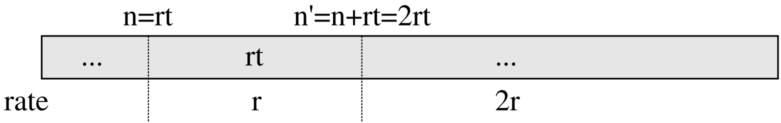
Hence during the round $n \geq rt$

A technical lemma

For each rate $r \geq 2$, let n be the number of stream items considered up to the r-rate round. It holds:

$$\frac{1}{r} \geq \frac{t}{n}$$

By induction, at the beginning of r-rate round $n = rt$:



Hence during the round $n \geq rt$

Expected sample size at the end of r-rate round $= \dfrac{n'}{r} = 2t$

Analysis (1/2)

For any $\varphi, \varepsilon, \delta \in (0, 1)$, with $\varepsilon < \varphi$, Sticky Sampling computes the heavy hitters with probability $\geq 1 - \delta$

Analysis (1/2)

For any $\varphi, \varepsilon, \delta \in (0,1)$, with $\varepsilon < \varphi$, Sticky Sampling computes the heavy hitters with probability $\geq 1 - \delta$

1. Good false positives: items with frequency $< (\varphi - \varepsilon)n$ are not returned
   $f(x) < (\varphi - \varepsilon)n \Rightarrow f_e(x) < (\varphi - \varepsilon)n$, since $f_e(x) \leq f(x)$

Analysis (1/2)

For any $\varphi, \varepsilon, \delta \in (0, 1)$, with $\varepsilon < \varphi$, Sticky Sampling computes the heavy hitters with probability $\geq 1 - \delta$

1. Good false positives: items with frequency $< (\varphi - \varepsilon)n$ are not returned
   $f(x) < (\varphi - \varepsilon)n \Rightarrow f_e(x) < (\varphi - \varepsilon)n$, since $f_e(x) \leq f(x)$
2. No false negatives: all items with frequency $\geq \varphi n$ are returned

Analysis (1/2)

For any $\varphi, \varepsilon, \delta \in (0,1)$, with $\varepsilon < \varphi$, Sticky Sampling computes the heavy hitters with probability $\geq 1-\delta$

1. Good false positives: items with frequency $< (\varphi-\varepsilon)n$ are not returned
   $f(x) < (\varphi-\varepsilon)n \Rightarrow f_e(x) < (\varphi-\varepsilon)n$, since $f_e(x) \leq f(x)$
2. No false negatives: all items with frequency $\geq \varphi n$ are returned

y_1 ... y_k frequent items: $\quad f(y_i) \geq \varphi n \quad \forall i$

$$\Rightarrow k \leq \frac{1}{\varphi}$$

$Pr\{\exists$ false negative$\} = \quad Pr\{\exists y_i : \; y_i \text{ not returned}\} \leq \sum_{i=1}^{k} Pr\{y_i \text{ not returned}\}$

Analysis (2/2)

$$
\begin{aligned}
Pr\{y_i \text{ not returned}\} = \quad & Pr\{f_e(y_i) < (\varphi - \varepsilon)n\} = \\
& Pr\{\text{at least } \varepsilon n \text{ unsuccessful coin tosses}\} \leq \\
& \left(1 - \frac{1}{r}\right)^{\varepsilon n} \leq \left(1 - \frac{t}{n}\right)^{\varepsilon n} \leq e^{-t\varepsilon}
\end{aligned}
$$

Analysis (2/2)

$$
\begin{aligned}
Pr\{y_i \text{ not returned}\} = \;& Pr\{f_e(y_i) < (\varphi - \varepsilon)n\} = \\
& Pr\{\text{at least } \varepsilon n \text{ unsuccessful coin tosses}\} \leq \\
& \left(1 - \frac{1}{r}\right)^{\varepsilon n} \leq \left(1 - \frac{t}{n}\right)^{\varepsilon n} \leq e^{-t\varepsilon}
\end{aligned}
$$

Hence:

$$
\begin{aligned}
Pr\{\exists \text{ false negative}\} \;& \leq \sum_{i=1}^{k} Pr\{y_i \text{ not returned}\} \leq \\
& \leq k e^{-t\varepsilon} \leq \frac{e^{-t\varepsilon}}{\varphi} = \delta \quad \text{by definition of } t
\end{aligned}
$$

Sketching streams

Sketches

- Not every problem can be solved with sampling
  E.g., counting distinct items in a stream: need to sample a large fraction of items to know if they are all same or different
- Sketches take advantage that the algorithm can "see" all the data even if it can't "remember" it all

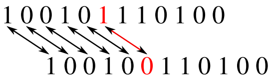
Sketch = linear transform of the input (exploit hashing)

Sampling and sketching ideas at the heart of stream mining:

- A sample is a quite general representative of the data set
- Sketches tend to be tailored to a specific problem (e.g., distinct items)

Warmup example

Problem: test if two asynchronous binary streams are equal



To test in small space: pick a random hash function h and test $h(\sigma_1) = h(\sigma_2)$:

- no false negatives: if $\sigma_1 = \sigma_2$ then $h(\sigma_1) = h(\sigma_2)$
- small chance of false positive: it may be $h(\sigma_1) = h(\sigma_2)$ for $\sigma_1 \neq \sigma_2$ with very small probability

Compute $h(\sigma_1)$ and $h(\sigma_2)$ incrementally as new bits arrive (Karp-Rabin fingerprints)

Distinct items

Count of the number of distinct items seen in the stream

Trivial solution: maintain set of encountered items through its characteristic vector

$O(1)$ processing time but $\Theta(u)$ space, where $u =$ universe size

- Exact/deterministic algorithms need $\Omega(u)$ bits of space
- Approximate randomized algorithms use $O(\log u)$ bits of space

FM-sketch [Flajolet & Martin '85]

Sampling not appropriate here: we'll build a data summary (sketch)

Universal hashing

- Idea: select a hash function at random from a family $\mathcal{H}$ of hash functions with a certain mathematical property
- Guarantee: low number of collisions in expectation, even if the data is chosen by an adversary

2-universal hashing

$\mathcal{H}$ is a 2-universal family (set) of hash functions $h : U \rightsquigarrow D$ if, for all $x, y \in U$, $x \neq y$:

$$Pr_{h \in \mathcal{H}}\{h(x) = h(y)\} \leq \frac{1}{|D|}$$

Strongly 2-universal hashing

$\mathcal{H}$ is strongly 2-universal if, for all $x \neq y \in U$ and $a, b \in D$:

$$Pr_{h \in \mathcal{H}}\{h(x) = a \ \& \ h(y) = b\} = \frac{1}{|D|^2}$$

FM skecth: probabilistic counter

Two useful functions:

- $h : U \rightsquigarrow [0, u-1]$ drawn from a family of strongly 2-universal hash functions

  Transforms values of the universe into integers uniformly distributed over the set of binary strings of length $\log u$

- $t : [0, u-1] \rightsquigarrow [1, \log u]$ gives the number $t(i)$ in the binary representation of i

  E.g., $t(5_{10}) = t(00101_2) = 2$

FM sketch: counter C of $\log u$ bits

Counter update: upon seeing stream item x, set $C[t(h(x))] = 1$

Query algorithm: return 2^R, where $R \in [1, \log u]$ is the position of the rightmost 1 in C

E.g., if $C = 1110100$, then $R = 5$: returns 32

Intuition

h distributes items of the universe U uniformly on $[0, u-1]$:
important to avoid adversarial streams

- How many values in $[0, u-1]$ have exactly 0 trailing 0s? $u/2$
- How many values have exactly 1 trailing 0? $u/4$
- How many values have exactly 2 trailing 0s? $u/8$ ...

Hence, if the stream contains D distinct values:

- $D/2$ will be mapped to the first bit of C
- $D/4$ to the second bit
- $D/8$ to the third bit ...

We expect the first $\log D$ counter bits will be set to 1
Hence $R \approx \log D$ and $2^R \approx D$

Geometric distribution over counter bits

- $|\text{values with exactly } j \text{ trailing 0s}| = \frac{u}{2^{j+1}}$
- $|\text{values with} \geq j \text{ trailing 0s}| = 1 + \sum_{i=j}^{\log u - 1} \frac{u}{2^{j+1}} = 2^{\log u - j}$
- W_x indicator random variable: $W_x = 1$ iff $t(h(x)) \geq j$

  $Pr\{W_x = 1\} = Pr\{t(h(x)) \geq j\} = \frac{2^{\log u - j}}{u} = 2^{-j}$

  since h distributes items uniformly over $[0, u-1]$
- $E[W_x] = 2^{-j}$
- $Var[W_x] = E[W_x^2] - E[W_x]^2 = 2^{-j} - 2^{-2j} < 2^{-j} = E[W_x]$

$$E[W_x] = 2^{-j} \quad \text{and} \quad Var[W_x] < E[W_x]$$

Geometric distribution over counter bits

- $Z_j =$ number of stream items x s.t. $t(h(x)) \geq j$
  $= \sum_{x \in U \cap \Sigma} W_x$
- $E[Z_j] = \sum_{x \in U \cap \Sigma} E[W_x] = \sum_{x \in U \cap \Sigma} 2^{-j} = \frac{D}{2^j}$
- Due to pairwise independence of W_x and W_y,
  $Var[W_x + W_y] = Var[W_x] + Var[W_y]$
- $Var[Z_j] = \sum_{x \in U \cap \Sigma} Var[W_x] < \sum_{x \in U \cap \Sigma} E[W_x] = E[Z_j]$

$$E[Z_j] = \frac{D}{2^j} \quad \text{and} \quad Var[Z_j] < E[Z_j]$$

- $R = \max j$ such that $Z_j > 0$

Probability of overestimating

Let $c > 2$. $Pr\{2^R > cD\} = ?$

By Markov's inequality (Z_j takes only non-negative values):

$$Pr\{Z_j \geq 1\} \leq \frac{E[Z_j]}{1} = \frac{D}{2^j} \tag{1}$$

$$\begin{aligned} 2^R > cD \quad & \Rightarrow \exists j \text{ such that } C[j] = 1 \ \& \ 2^j > cD \\ & \Rightarrow C[j] = 1 \ \& \ j > \log_2(cD) \\ & \Rightarrow Z_{\log_2(c\,D)} \geq 1 \end{aligned}$$

Thus:

$$Pr\{2^R > cD\} \leq Pr\{\, Z_{\log_2(c\,D)} \geq 1 \,\} \leq_{(1)} \frac{D}{2^{\log_2(c\,D)}} = \frac{1}{c}$$

Probability of underestimating

Let $c > 2$. $Pr\left\{2^R < \frac{D}{c}\right\} = ?$

By Chebyshev inequality (Z_j takes only non-negative values):

$$\begin{aligned} Pr\{Z_j = 0\} &= Pr\{|Z_j - E[Z_j]| \geq E[Z_j]\} \\ &\leq \frac{Var[Z_j]}{E[Z_j]^2} < \frac{1}{E[Z_j]} = \frac{2^j}{D} \end{aligned} \tag{2}$$

$$\begin{aligned} 2^R < \tfrac{D}{c} &\Rightarrow C[p] = 0 \;\; \forall p \geq \log_2(D/c) \\ &\Rightarrow Z_{\log_2(D/c)} = 0 \end{aligned}$$

Thus:

$$Pr\left\{2^R < \frac{D}{c}\right\} \leq Pr\{\, Z_{\log_2(D/c)} = 0 \,\} \leq_{(2)} \frac{2^{\log_2(D/c)}}{D} = \frac{1}{c}$$

Distinct items: summing up

Let D be the exact number of distinct values and let 2^R be the output of the probabilistic counter.

For any $c > 2$, the probability that 2^R is not between D/c and $c\,D$ is at most $2/c$.

Frequency moments

Stream $\Sigma = \langle x_1, x_2, \dots x_n \rangle$ of tokens drawn from universe U

$f_i = |\{j : x_j = i\}|$

k-th frequency moment F_k of Σ

$$F_k = \sum_{i \in U} f_i^k$$

Useful statistical information:

- F_0 = distinct items
- F_1 = stream length
- F_2 = Gini's index (skew of the data)
- F_∞ related to maximum frequency element, i.e., $\max_{i \in U} f_i$

AMS sketch for F_2

Fundamental technique introduced by Alon, Matias, and Szegedy

AMS sketches = randomized linear projections

Define a random variable Z such that $E[Z^2] = F_2$:

- select at random a hash function $\xi : U \rightsquigarrow \{-1, +1\}$ from a family of 4-wise independent hash functions
- $Z = \sum_{u \in U} f_u \, \xi(u)$

  random linear projection (inner product) of frequency vector $\langle f_1, f_2, \ldots f_u \rangle$ with random vector $\{-1, +1\}^u$
- Z incrementally updated upon arrival of x_t by adding $\xi(x_t)$

AMS sketch: expectation

$$Z = \sum_{u \in U} f_u \, \xi(u)$$

$\xi : U \rightsquigarrow \{-1, +1\}$ 4-wise independent

$$E[\xi(i)] = (-1)\tfrac{1}{2} + (1)\tfrac{1}{2} = 0$$

$$\begin{aligned}
E[Z^2] &= E\left[\left(\sum_{i \in U} f_i \, \xi(i)\right)^2\right] \\
&= E\left[\sum_{i \in U} f_i^2 \, (\xi(i))^2 + 2 \sum_{i \neq j \in U} f_i f_j \, \xi(i)\xi(j)\right] \\
&= \sum_{i \in U} f_i^2 E\left[(\xi(i))^2\right] + 2 \sum_{i \neq j \in U} f_i f_j \, E\left[\xi(i)\xi(j)\right] \\
&= \sum_{i \in U} f_i^2 = F_2
\end{aligned}$$

since $(\xi(i))^2 = 1$ and by pair-wise independence

$$E\left[\xi(i)\xi(j)\right] = E\left[\xi(i)\right] E\left[\xi(j)\right] = 0 \cdot 0 = 0$$

Median of the averages

Still need small variance and good confidence:

- Compute μ random variables $Y_1, \ldots, Y_\mu$ and output their median Y as the estimator for F_2
- Each Y_i is the average of α independent, identically distributed random variables X_{ij} computed as random linear projections

Averaging X_{ij} implies each Y_i has small variance

Computing Y as the median of the Y_i allows it to boost confidence using Chernoff bounds

F_2: summing up

For every $\lambda, \delta > 0$, there exists a randomized algorithm that computes a number Y that deviates from F_2 by more than λF_2 with probability at most δ.

The algorithm uses only

$$O\left(\frac{\log(1/\delta)}{\lambda^2}(\log u + \log n)\right)$$

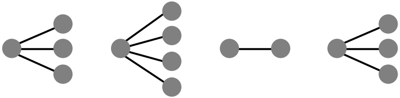
memory bits and performs one pass over the data.

Similar results for frequency moments F_k, with $k > 2$

Mining graphs

Models for graph streams

- $G = (V, E)$ graph with $|V| = n$ nodes and $|E| = m$ edges, possibly weighted
- Observe edges of G in a stream, one by one
- What order do we see the edges in?
  - Arbitrary (adversarial) order
  - Incidence streams: all edges incident to one vertex appear sequentially (easier, stronger bounds)



- How many passes over the data can we take (one or many?)
- How much space?

Counting triangles

- Finding frequent graph patterns and dense subgraphs are basic tools in the analysis of the structure of large networks (e.g., social networks, Web graph)
- Exact triangle counting reduces to matrix multiplication: unfeasible even for networks of medium size
- Resort to random sampling
- We'll present an algorithm for the arbitrary order model

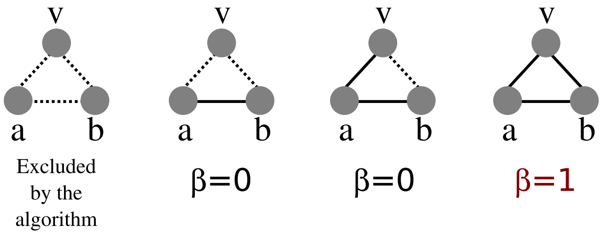
A 3-pass algorithm

Algorithm SampleTriangle

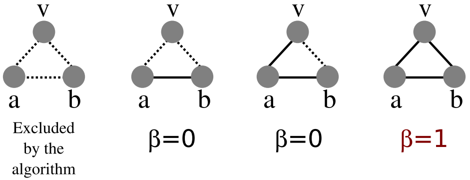
1st pass. Count number of edges m in the stream

2nd pass. Sample an edge $e = (a, b)$ uniformly from E and a node v uniformly from $V \setminus \{a, b\}$

3rd pass. If $(a, v) \in E$ and $(b, v) \in E$ then $\beta = 1$, else $\beta = 0$



A useful property



T_i = triples with i edges, $0 \le i \le 3$

$$E[\beta] = \frac{3|T_3|}{m \cdot (n-2)} = \frac{3|T_3|}{|T_1| + 2|T_2| + 3|T_3|}$$

- $m \cdot (n-2)$ ways to select an edge (a, b) and a node $v \neq a, b$
- $i|T_i|$ ways to select a triple with i edges, $i > 0$

The complete 3-pass algorithm

- Start s parallel instances of algorithm SampleTriangle, where

$$s \geq \frac{3}{\varepsilon^2} \frac{|T_1| + 2|T_2| + 3|T_3|}{|T_3|} \ln\left(\frac{2}{\delta}\right)$$

- Each instance returns a value β_i
- Return $\widetilde{T_3} = \left(\frac{1}{s}\sum_{i=1}^{s} \beta_i\right) \frac{m \cdot (n-2)}{3}$ as an estimation for T_3

$E[\widetilde{T_3}] = |T_3|$ because $E[\beta_i] = \frac{3|T_3|}{m \cdot (n-2)}$

OK, but how far from the mean?

Chernoff bounds

- $X_1, X_2, \ldots X_n$ independent Bernoulli trials: X_i indicator random variable, $Pr\{X_i = 1\} = p$, X_i all independent
- $X = \sum_{i=1}^{n} X_i$
- $E[X] = \mu = n\,p$

Lower tail bound

For any $\varepsilon \in (0, 1]$ $\quad Pr\{X < (1 - \varepsilon)\mu\} < e^{-\frac{\mu\varepsilon^2}{2}}$

Upper tail bound

For any $\varepsilon \in (0, 1]$ $\quad Pr\{X > (1 + \varepsilon)\mu\} < e^{-\frac{\mu\varepsilon^2}{3}}$

Triangle counting analysis

In triangle counting, $X = \sum_{i=1}^{s} \beta_i$ and $p = \frac{3|T_3|}{|T_1|+2|T_2|+3|T_3|}$

$$Pr\{X < (1-\varepsilon)ps \mid\mid X > (1+\varepsilon)ps\} \quad < e^{-\frac{ps\varepsilon^2}{2}} + e^{-\frac{ps\varepsilon^2}{3}}$$

Triangle counting analysis

In triangle counting, $X = \sum_{i=1}^{s} \beta_i$ and $p = \frac{3|T_3|}{|T_1|+2|T_2|+3|T_3|}$

$$\begin{aligned} Pr\{X < (1-\varepsilon)ps \mid\mid X > (1+\varepsilon)ps\} &< e^{-\frac{ps\varepsilon^2}{2}} + e^{-\frac{ps\varepsilon^2}{3}} \\ &\leq 2e^{-\frac{sp\varepsilon^2}{3}} \end{aligned}$$

Triangle counting analysis

In triangle counting, $X = \sum_{i=1}^{s} \beta_i$ and $p = \frac{3|T_3|}{|T_1|+2|T_2|+3|T_3|}$

$$
\begin{aligned}
Pr\{X < (1-\varepsilon)ps \mid\mid X > (1+\varepsilon)ps\} \quad &< e^{-\frac{ps\varepsilon^2}{2}} + e^{-\frac{ps\varepsilon^2}{3}} \\
&\le 2e^{-\frac{sp\varepsilon^2}{3}} \le \delta
\end{aligned}
$$

as long as $s \ge \frac{3}{\varepsilon^2} \frac{|T_1|+2|T_2|+3|T_3|}{|T_3|} \ln\left(\frac{2}{\delta}\right)$

Triangle counting analysis

In triangle counting, $X = \sum_{i=1}^{s} \beta_i$ and $p = \frac{3|T_3|}{|T_1|+2|T_2|+3|T_3|}$

$$
\begin{aligned}
Pr\{X < (1-\varepsilon)ps \mid\mid X > (1+\varepsilon)ps\} &< e^{-\frac{ps\varepsilon^2}{2}} + e^{-\frac{ps\varepsilon^2}{3}} \\
&\leq 2e^{-\frac{sp\varepsilon^2}{3}} \leq \delta
\end{aligned}
$$

as long as $s \geq \frac{3}{\varepsilon^2} \frac{|T_1|+2|T_2|+3|T_3|}{|T_3|} \ln\left(\frac{2}{\delta}\right)$

$$
X < (1-\varepsilon)ps \Leftrightarrow \underbrace{\left(\frac{1}{s}\sum_{i=1}^{s}\beta_i\right)\frac{m\cdot(n-2)}{3}}_{\widetilde{T_3}} < (1-\varepsilon)\underbrace{p\frac{m(n-2)}{3}}_{T_3}
$$

Triangle counting analysis

In triangle counting, $X = \sum_{i=1}^{s} \beta_i$ and $p = \frac{3|T_3|}{|T_1|+2|T_2|+3|T_3|}$

$$\begin{aligned} Pr\{X < (1-\varepsilon)ps \mid\mid X > (1+\varepsilon)ps\} &< e^{-\frac{ps\varepsilon^2}{2}} + e^{-\frac{ps\varepsilon^2}{3}} \\ &\leq 2e^{-\frac{sp\varepsilon^2}{3}} \leq \delta \end{aligned}$$

as long as $s \geq \frac{3}{\varepsilon^2} \frac{|T_1|+2|T_2|+3|T_3|}{|T_3|} \ln\left(\frac{2}{\delta}\right)$

$$X < (1-\varepsilon)ps \Leftrightarrow \underbrace{\left(\frac{1}{s}\sum_{i=1}^{s}\beta_i\right)\frac{m\cdot(n-2)}{3}}_{\widetilde{T_3}} < (1-\varepsilon)\,\underbrace{p\frac{m(n-2)}{3}}_{T_3}$$

Similarly $X > (1+\varepsilon)ps \Leftrightarrow \widetilde{T_3} > (1+\varepsilon)\,T_3$

Improvements and extensions

- Expected constant time:
  1. when edge (a, b) and node v sampled, hash missing edges (a, v) and (b, v) to a set M
  2. in the third pass, lookup each edge (x, y) in M, and mark it if present
  3. triangles determined in a postprocessing step

Improvements and extensions

- Expected constant time:
  1. when edge (a, b) and node v sampled, hash missing edges (a, v) and (b, v) to a set M
  2. in the third pass, lookup each edge (x, y) in M, and mark it if present
  3. triangles determined in a postprocessing step

- 1-pass: exploit reservoir sampling

Improvements and extensions

- Expected constant time:
  1. when edge (a, b) and node v sampled, hash missing edges (a, v) and (b, v) to a set M
  2. in the third pass, lookup each edge (x, y) in M, and mark it if present
  3. triangles determined in a postprocessing step
- 1-pass: exploit reservoir sampling
- Other minors and cliques of size α

Improvements and extensions

- Expected constant time:
  1. when edge (a, b) and node v sampled, hash missing edges (a, v) and (b, v) to a set M
  2. in the third pass, lookup each edge (x, y) in M, and mark it if present
  3. triangles determined in a postprocessing step
- 1-pass: exploit reservoir sampling
- Other minors and cliques of size α
- Better space bounds for incidence streams

Semi-streaming model

- For many graph problems *space* $\times$ *passes* $= \Omega(n)$, even using randomization and approximation

  $\Rightarrow$ Cannot achieve $O(1)$ passes and polylog working space

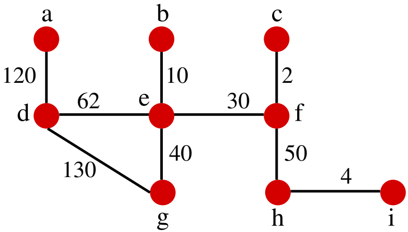
- *Semi-streaming* model: polylog space requirement is relaxed

> working memory size $O(n \text{ polylog } n)$
> for input graph with n nodes
>
> enough space to store nodes, not enough for edges

- Problems solvable in semi-streaming: spanners, matching, diameter estimation...

Maximum weight matching

- Edge weighted, undirected graph $G(V, E, w)$
- No two edges in a matching have a common endpoint



Optimization problem: find a maximum weight matching M^*

1-pass semi-streaming algorithm with approximation ratio $1/6$:

$$w(M) \geq \frac{w(M^*)}{6}$$

where M returned matching

Semi-streaming algorithm

Data structure: matching M maintained in main memory

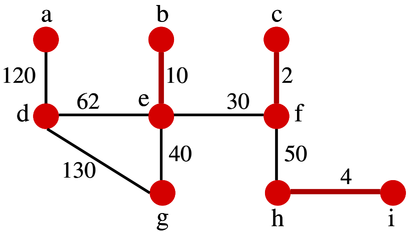
Query algorithm: return M

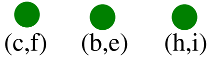
Update algorithm: upon arrival of edge e, consider set $C \subseteq M$ of conflicting edges (edges in M that share an endpoint with e)

- if $w(e) > 2w(C)$, replace C with $\{e\}$ in M
- if $w(e) \leq 2w(C)$), ignore e

Replacement forest

$$\Sigma = \langle\ (c,f,2)\ (b,e,10)\ (h,i,4)\ (e,f,30)\ (h,f,50)\ (e,g,40)\ (d,e,62)\ (a,d,120)\ (d,g,130)\ \rangle$$



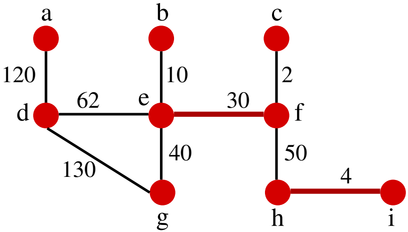


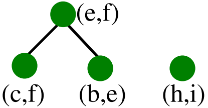
Replacement forest

Every edge $e \in M$ is root of a replacement tree T_e

Replacement forest

$$\Sigma = \langle\ (c, f, 2)\ (b, e, 10)\ (h, i, 4)\ (e, f, 30)\ (h, f, 50)$$
$$(e, g, 40)\ (d, e, 62)\ (a, d, 120)\ (d, g, 130)\ \rangle$$





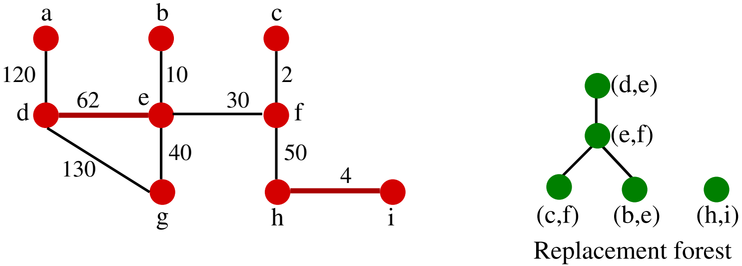
Replacement forest

Every edge $e \in M$ is root of a replacement tree T_e

$R(e) =$ nodes in T_e except for root e

Replacement forest

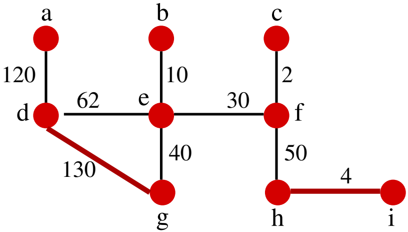
$$\Sigma = \langle\ (c,f,2)\ (b,e,10)\ (h,i,4)\ (e,f,30)\ (h,f,50)$$
$$(e,g,40)\ (d,e,62)\ (a,d,120)\ (d,g,130)\ \rangle$$

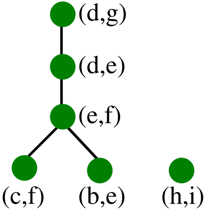


Every edge $e \in M$ is root of a replacement tree T_e
$R(e) =$ nodes in T_e except for root e

Replacement forest

$$\Sigma = \langle\ (c, f, 2)\ (b, e, 10)\ (h, i, 4)\ (e, f, 30)\ (h, f, 50)\ (e, g, 40)\ (d, e, 62)\ (a, d, 120)\ (d, g, 130)\ \rangle$$





Replacement forest

Every edge $e \in M$ is root of a replacement tree T_e
$R(e)$ = nodes in T_e except for root e

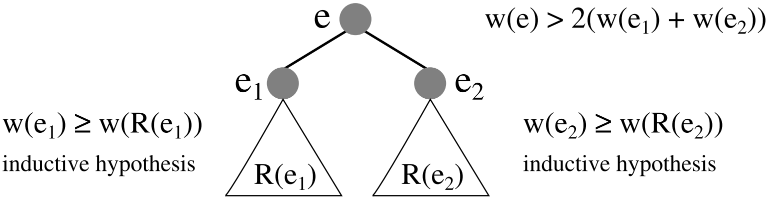
Replacement edges have small weight

$$w(R(e)) \leq w(e)$$

Replacement edges have small weight

$$w(R(e)) \leq w(e)$$

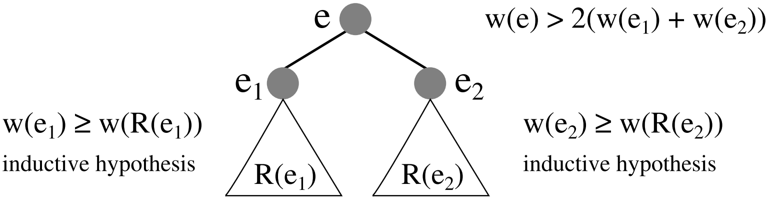
By induction:



Replacement edges have small weight

$$w(R(e)) \leq w(e)$$

By induction:



$$w(e) > 2w(e_1) + 2w(e_2) \geq$$
$$\geq w(e_1) + w(R(e_1)) + w(e_2) + w(R(e_2)) = w(R(e))$$

A charging scheme for M^*

M^* maximum weight matching

H = history edges part of the matching at some point

Charge weight of M^* to H. For each $o \in M^*$:

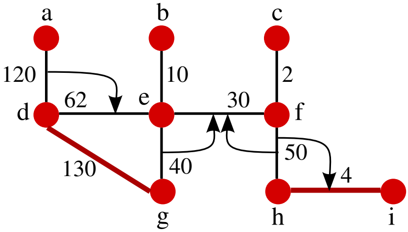
1. $o \in H$: charge $w(o)$ to o itself
2. $o \notin H$:
   - C = edges conflicting with o it was examined for insertion: $w(o) \leq 2\, w(C)$, since o was not inserted
   - If $C = \{e\}$: charge $w(o) \leq 2w(e)$ to e
   - If $C = \{e_1, e_2\}$: charge
     - $\dfrac{w(o)w(e_1)}{w(e_1) + w(e_2)} \leq 2\, w(e_1)$ to e_1
     - $\dfrac{w(o)w(e")}{w(e') + w(e")} \leq 2\, w(e_2)$ to e_2

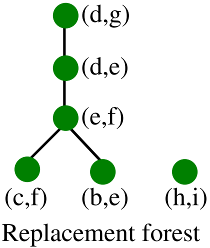
(a) Charge of $o \in M^*$ to any edge $e \in H \leq 2\, w(e)$

Initial charging

$$\Sigma = \langle\ (c,f,2)\ (b,e,10)\ (h,i,4)\ (e,f,30)\ (h,f,50)\ (e,g,40)\ (d,e,62)\ (a,d,120)\ (d,g,130)\ \rangle$$

$$M^* = \{(a,d),(e,g),(h,f)\}$$

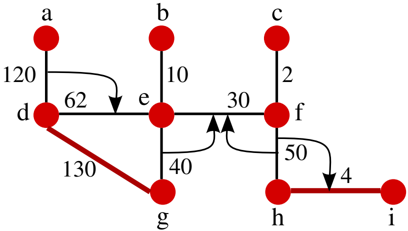


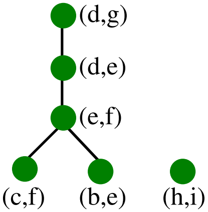


(b) Any edge of H charged by at most two edges of M^*, one per endpoint.

Charging redistribution

If $o \in M^*$ charges $e \in H$, e replaced by $e' \in H$, e' and o incident, transfer charge of o from e to e'.





Replacement forest

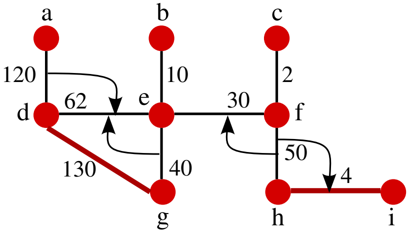
(a) Charge of $o \leq 2\,w(e) \leq 2\,w(e')$

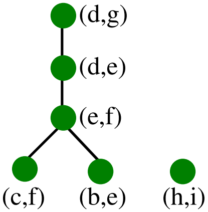
(b) Any edge of H charged by at most two edges of M^*, one per endpoint (redistribution preserves incidence)

(c) Each edge $e \in H \setminus M$ charged by at most one edge in M^*

Charging redistribution

If $o \in M^*$ charges $e \in H$, e replaced by $e' \in H$, e' and o incident, transfer charge of o from e to e'.





Replacement forest

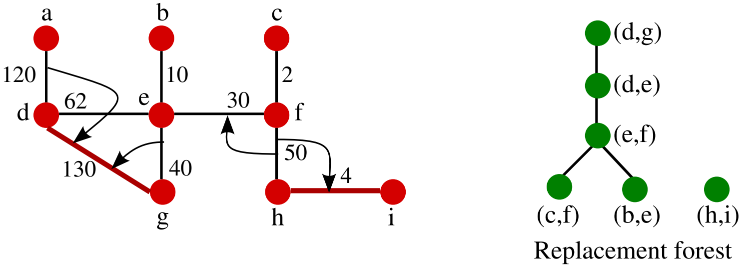
(a) Charge of $o \leq 2\,w(e) \leq 2\,w(e')$

(b) Any edge of H charged by at most two edges of M^*, one per endpoint (redistribution preserves incidence)

(c) Each edge $e \in H \setminus M$ charged by at most one edge in M^*

Charging redistribution

If $o \in M^*$ charges $e \in H$, e replaced by $e' \in H$, e' and o incident, transfer charge of o from e to e'.



(a) Charge of $o \leq 2\,w(e) \leq 2\,w(e')$

(b) Any edge of H charged by at most two edges of M^*, one per endpoint (redistribution preserves incidence)

(c) Each edge $e \in H \setminus M$ charged by at most one edge in M^*

Analysis: summing up

- Charge of $o \in M^*$ to any edge $e \in H \leq 2\,w(e)$
- Edges in $H \setminus M$ charged by at most one edge in M^*
- Edges in M charged by at most two edges in M^*

$$w(M^*) \leq \sum_{x \in H \setminus M} 2w(x) + \sum_{e \in M} 4w(e)$$

Analysis: summing up

- Charge of $o \in M^*$ to any edge $e \in H \leq 2\,w(e)$
- Edges in $H \setminus M$ charged by at most one edge in M^*
- Edges in M charged by at most two edges in M^*

$$w(M^*) \leq \sum_{x \in H \setminus M} 2w(x) + \sum_{e \in M} 4w(e)$$

Since $H \setminus M = \cup_{e \in M} R(e)$:

$$w(M^*) \leq \sum_{x \in H \setminus M} 2w(x) + \sum_{e \in M} 4w(e) = \sum_{e \in M} 2w(R(e)) + \sum_{e \in M} 4w(e)$$

Analysis: summing up

- Charge of $o \in M^*$ to any edge $e \in H \leq 2\,w(e)$
- Edges in $H \setminus M$ charged by at most one edge in M^*
- Edges in M charged by at most two edges in M^*

$$w(M^*) \leq \sum_{x \in H \setminus M} 2w(x) + \sum_{e \in M} 4w(e)$$

Since $H \setminus M = \cup_{e \in M} R(e)$:

$$w(M^*) \leq \sum_{x \in H \setminus M} 2w(x) + \sum_{e \in M} 4w(e) = \sum_{e \in M} 2w(R(e)) + \sum_{e \in M} 4w(e)$$

Since replacement edges have small weight $w(R(e)) \leq w(e)$:

$$w(M^*) \leq \sum_{e \in M} 6w(e) = 6w(M)$$

Lower bounds

Communication complexity

Important technique for proving streaming lower bounds:
reducing communication complexity problems to streaming problems

Lower bounds known in communication complexity yield streaming lower bounds

Example related to triangle counting:

To determine whether $T_3 > 0$, we need $\Omega(n^2)$ space, even using a randomized algorithm

$T_3 =$ number of triangles

2-player set-disjointness

Alice has $n \times n$ matrix A
Bob has $n \times n$ matrix B

$$A = \begin{array}{|c|c|c|}\hline 0 & 1 & 0 \\ \hline 0 & 0 & 0 \\ \hline 0 & 0 & 1 \\ \hline \end{array} \qquad B = \begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline 1 & 1 & 0 \\ \hline 0 & 0 & 1 \\ \hline \end{array}$$

Alice and Bob wish to determine if $A \cap B \neq \emptyset$

$$A \cap B \neq \emptyset \quad \Leftrightarrow \quad \exists i, j : A[i,j] = 1 \text{ and } B[i,j] = 1$$

By a communication complexity lower bound, this requires $\Omega(n^2)$ bits even for protocols that are correct with probability 3/4

Is $T_3 > 0$? Graph construction

Alice has $n \times n$ matrix A
Bob has $n \times n$ matrix B

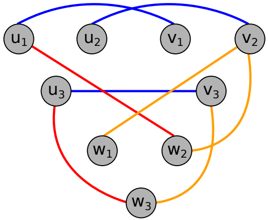
$A \cap B \neq \emptyset$?

A

0	1	0
0	0	0
0	0	1

B

0	0	0
1	1	0
0	0	1



Build graph $G = (V, E)$ as follows:

- $V = \{u_1, u_2, \dots u_n\} \cup \{v_1, v_2, \dots v_n\} \cup \{w_1, w_2, \dots w_n\}$
- $E = \{(u_i, v_i) : i \in [1, n]\} \cup \{(u_i, w_j) : A[i, j] = 1\} \cup \{(v_i, w_j) : B[i, j] = 1\}$

Triangles can only have the form $\langle u_i, v_i, w_j \rangle$

G contains a triangle $\Leftrightarrow \exists j : A[i, j] = 1$ and $B[i, j] = 1$

The reduction

$\mathcal{A}$ = s-bit streaming algorithm that determines whether $T_3 > 0$

Use $\mathcal{A}$ to solve set disjointness as follows:

1. Alice creates a stream with blue and red edges, and runs the algorithm on the stream
2. Then she sends s bits (her memory content) to Bob
3. Bob runs the algorithm, starting from Alice memory content, on the remaining yellow edges
4. He finally communicates 1 bit (the result) to Alice

Communication: $s + 1$ bits

$\Rightarrow s = \Omega(n^2)$

Conclusions

More streaming algorithms...

Many others fundamentals have been studied, not covered here

- Different stream data types:
  - geometric data (location streams)
  - permutations
  - graphs and hypergraphs
- Different streaming models:
  - time-conscious models: sliding windows, exponential decay
  - non adversarial models: random order streams, skewed streams
- Different streaming scenarios:
  - distributed computations
  - sensor network computations

Directions: time-conscious models

Which is more popular between *Star Wars - Episode IV* (1977) and *Mission Impossible - Ghost Protocol* (2011)?

Are N tickets sold in each of the last 20 years better than N tickets sold in the last week?

Recent past in some cases more important than distant past
$\Rightarrow$ windowed streaming:

- fixed size window
- decaying window: influence of items on the result decreases exponentially

Directions: graphs

Rich graph structure in Web data: conversations, friendships, video, images...

Billions of dollar industry applications rely on analyzing Web info

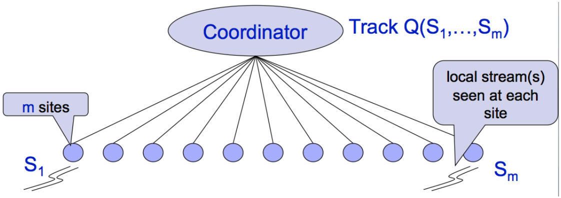
Graph problems are very challenging:

- More dense graph problems in semi-streaming (so far, matching, spanners, shortest paths and diameter)
- Space/passes tradeoffs: reduce or annotate the stream, taking multiple passes on less and less elements
- Look at graphs as matrices: can we compute fundamental properties such as eigenvalues?
- Many natural graph questions are "hard" in standard models: more realistic and tractable models?

Directions: distributed streams

Data progressively seen from distributed sources, a central monitor (coordinator) needs to estimate some quantity

Goal: minimize total number of bits communicated by the distributed streams to the coordinator



- Can we continuously track a (global) query over streams while bounding the communication with the coordinator?
- Can we design stream summary data structures that can be combined to summarize the union of streams?

Directions: beyond adversarial order

In practice, not all frequency distributions are worst case

Can we prove stronger algorithmic results for:

- Skewed data (e.g., "Zipfian" distribution)
- Small-world scale-free models for graphs
- Random order streams
- Semi-random streams: can we develop algorithms whose performance degrades smoothly as the stream ordering becomes "less-random"?

Results in these lectures: references

- *Reservoir sampling.* J. S. Vitter. Random Sampling with a Reservoir, ACM Transactions on Mathematical Software, 11(1), 37-57, 1985
- *Heavy hitters.* G. S. Manku & R. Motwani. Approximate Frequency Counts over Data Streams, VLDB 2002
- *Distinct items.* P. Flajolet, G. N. Martin. Probabilistic Counting Algorithms for Data Base Applications. J. Comput. Syst. Sci. 1985
- *Frequency moments.* N. Alon, Y. Matias and M. Szegedy. The Space Complexity of Approximating the Frequency Moments. J. Comput. Syst. Sci. 1999
- *Triangle counting.* L. Buriol, G. Frahling, S. Leonardi, A. Marchetti-Spaccamela, & C. Sohler. Counting Triangles in Data Streams. PODS 2006
- *Weighted matching.* J. Feigenbaum, S. Kannan, A. McGregor, S. Suri, J. Zhang. On graph problems in a semi-streaming model. Theor. Comput. Sci. 2005

Online resources

Too many papers to be comprehensive... Some surveys and interesting pointers:

1. Data streams: algorithms and applications, S. Muthukrishnan
   `http://www.cs.rutgers.edu/~muthu/`
2. Sketch techniques for massive data, G. Cormode
   Continuous distributed monitoring: a short survey, G. Cormode
   `http://dimacs.rutgers.edu/~graham/`
3. Algorithms for data streams, C. Demetrescu & I. Finocchi
   `twiki.di.uniroma1.it/pub/Ing_algo/WebHome/DFchapter08.pdf`
4. Andrew McGregor's crash course and blog
   `http://polylogblog.wordpress.com/2010/09/08/some-slides/`
5. IITK Workshop on Algorithms for Processing Massive Data Sets, IIT-Kanpur, India, 2009
   `http://www2.cse.iitk.ac.in/~fsttcs/2009/wapmds/`
6. Open problems in data streams, property testing, and related topics, Indyk *et al.*, 2011 (the Bertinoro and Kanpur lists)
   `http://polylogblog.wordpress.com/category/open-problems/`

Thanks!