CSCE 689: Special Topics in Modern Algorithms for Data Science

Lecture 9

Samson Zhou

Presentation Schedule

- September 25: Team DAP, Team Bokun, Team Jason
- September 27: Galaxy AI, Team STMI
- September 29: Team JAC

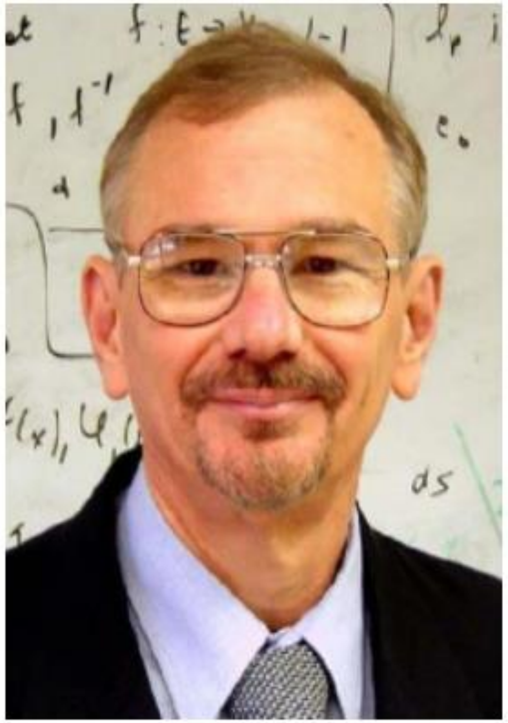
Texas A&M University
Mathematics

Previous Lecture

Featured programs ▾

ABOUT	PEOPLE	RESEARCH	ACADEMICS	SERVICES	OUTREACH	NEWS & EVENTS
welcome	faculty · staff	areas	courses & general info	administration	programs	news
employment	visitors	seminars	undergraduate	computing	friends & alumni	calendar
contact	grad students	lecture series	graduate	resources		conferences
			math placement (mpe)			

Faculty »
Staff »
Visiting faculty »
Retired faculty »
Graduate students »



Bill Johnson

A.G.and M.E. Owen Chair and Distinguished Professor

Office Blocker 525F

Fax +1 979 845 6028

Email w-johnson <at> tamu.edu

URL https://people.tamu.edu/~w-johnson/

Education Ph.D. Iowa State University, 1969
B.A. Southern Methodist University, 1966

Research Area Banach spaces, nonlinear functional analysis, probability theory

Last Time: Johnson-Lindenstrauss Lemma

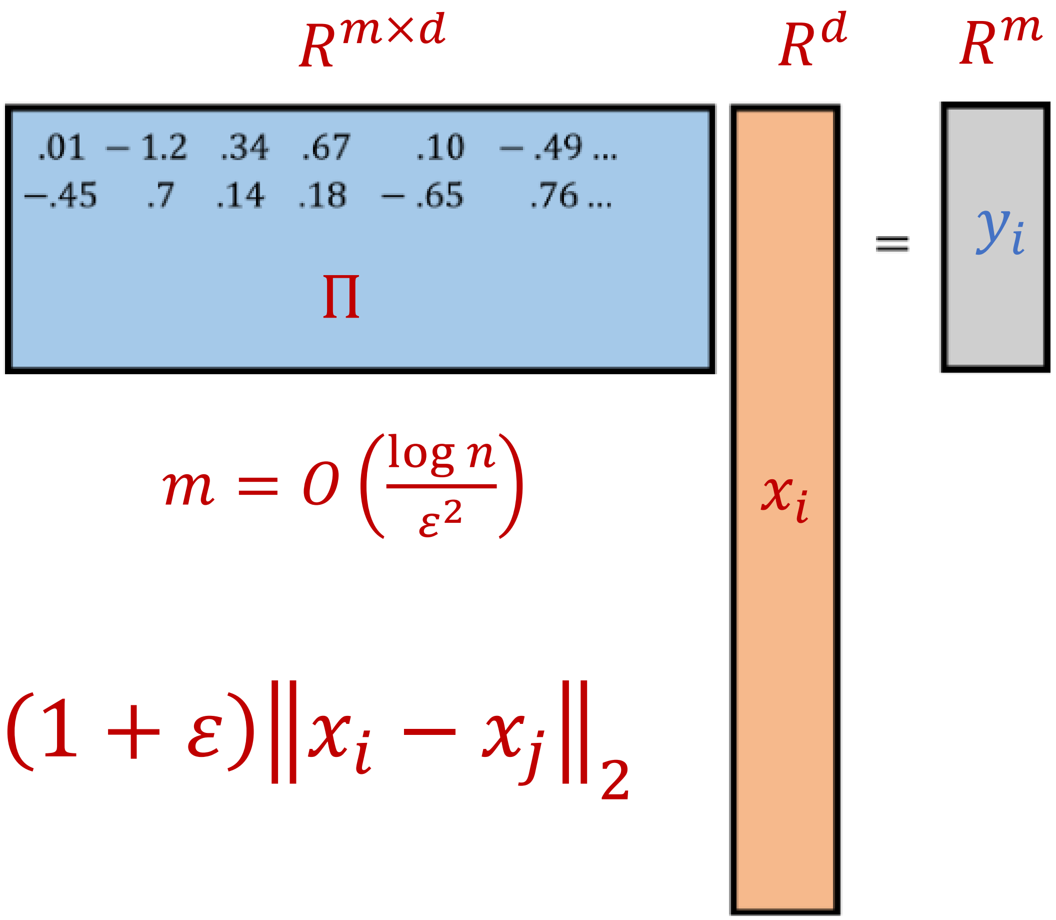
- Johnson-Lindenstrauss Lemma: Given $x_1, \ldots, x_n \in R^d$ and an accuracy parameter $\varepsilon \in [0,1)$, there exists a linear map $\Pi: R^d \to R^m$ with $m = O\left(\frac{\log n}{\varepsilon^2}\right)$ so that if $y_i = \Pi x_i$, then for all $i, j \in [n]$:

$$(1-\varepsilon)\left\|x_i - x_j\right\|_2 \leq \left\|y_i - y_j\right\|_2 \leq (1+\varepsilon)\left\|x_i - x_j\right\|_2$$

- Moreover, if each entry of Π is drawn from $\frac{1}{\sqrt{m}}N(0,1)$, then Π satisfies the guarantee with high probability

Last Time: Johnson-Lindenstrauss Lemma

- Given $x_1, \ldots, x_n \in R^d$ and $\Pi \in R^{m \times d}$ with $m = O\left(\frac{\log n}{\varepsilon^2}\right)$ and each entry drawn from $\frac{1}{\sqrt{m}} N(0,1)$ and setting $y_i = \Pi x_i$, then with high probability, for all $i, j \in [n]$:



$$(1 - \varepsilon)\|x_i - x_j\|_2 \leq \|y_i - y_j\|_2 \leq (1 + \varepsilon)\|x_i - x_j\|_2$$

- Π is called a random projection

Last Time: Johnson-Lindenstrauss Lemma

- Johnson-Lindenstrauss Lemma: Given $x_1, \dots, x_n \in R^d$ and $\Pi \in R^{m \times d}$ with $m = O\left(\frac{\log n}{\varepsilon^2}\right)$ and each entry drawn from $\frac{1}{\sqrt{m}} N(0,1)$ and setting $y_i = \Pi x_i$, then with high probability, for all $i, j \in [n]$:

$$(1-\varepsilon)\left\|x_i - x_j\right\|_2 \leq \left\|y_i - y_j\right\|_2 \leq (1+\varepsilon)\left\|x_i - x_j\right\|_2$$

- Distributional Johnson-Lindenstrauss Lemma: Given $\Pi \in R^{m \times d}$ with $m = O\left(\frac{\log 1/\delta}{\varepsilon^2}\right)$ and each entry drawn from $\frac{1}{\sqrt{m}} N(0,1)$, then for any $x \in R^d$ and setting $y = \Pi x$, then with probability at least $1 - \delta$

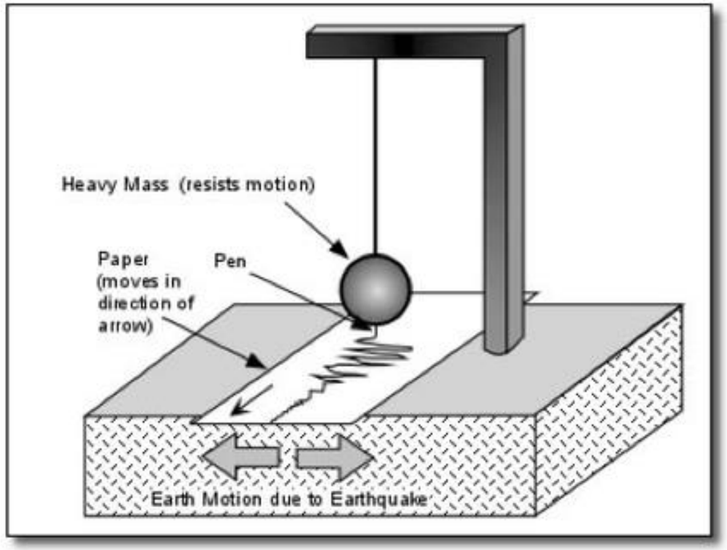
$$(1-\varepsilon)\|x\|_2 \leq \|y\|_2 \leq (1+\varepsilon)\|x\|_2$$

The Streaming Model

- Scenario: We are given a massive dataset that arrives in a continuous stream, which we would like to analyze – but we do not have enough space to store all the items

The Streaming Model

- Scientific observations: images from telescopes (Event Horizon Telescope collected 1 petabyte, i.e., 1024 terabytes, of data from a five-day observing campaign), readings from seismometer arrays monitoring and predicting earthquake activity

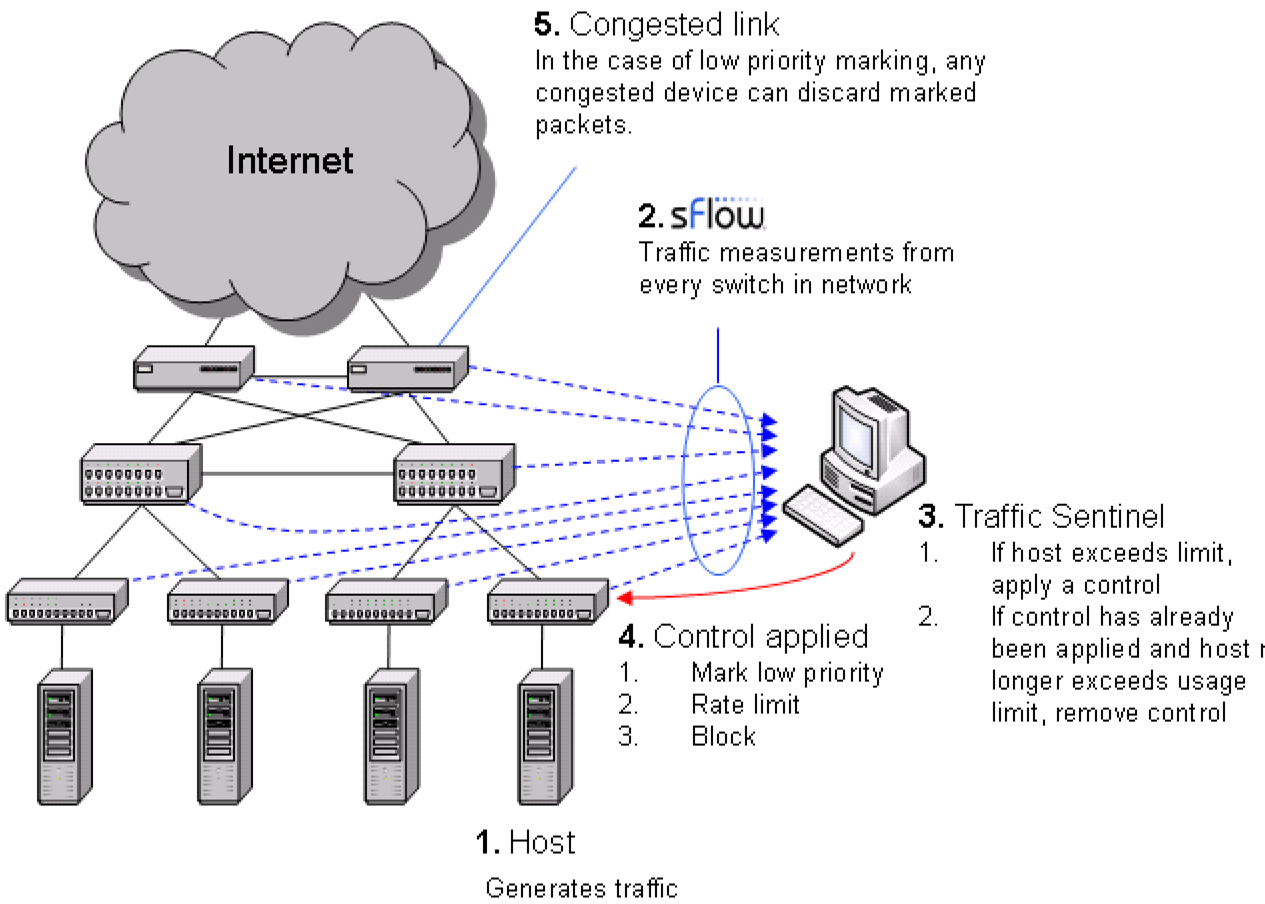


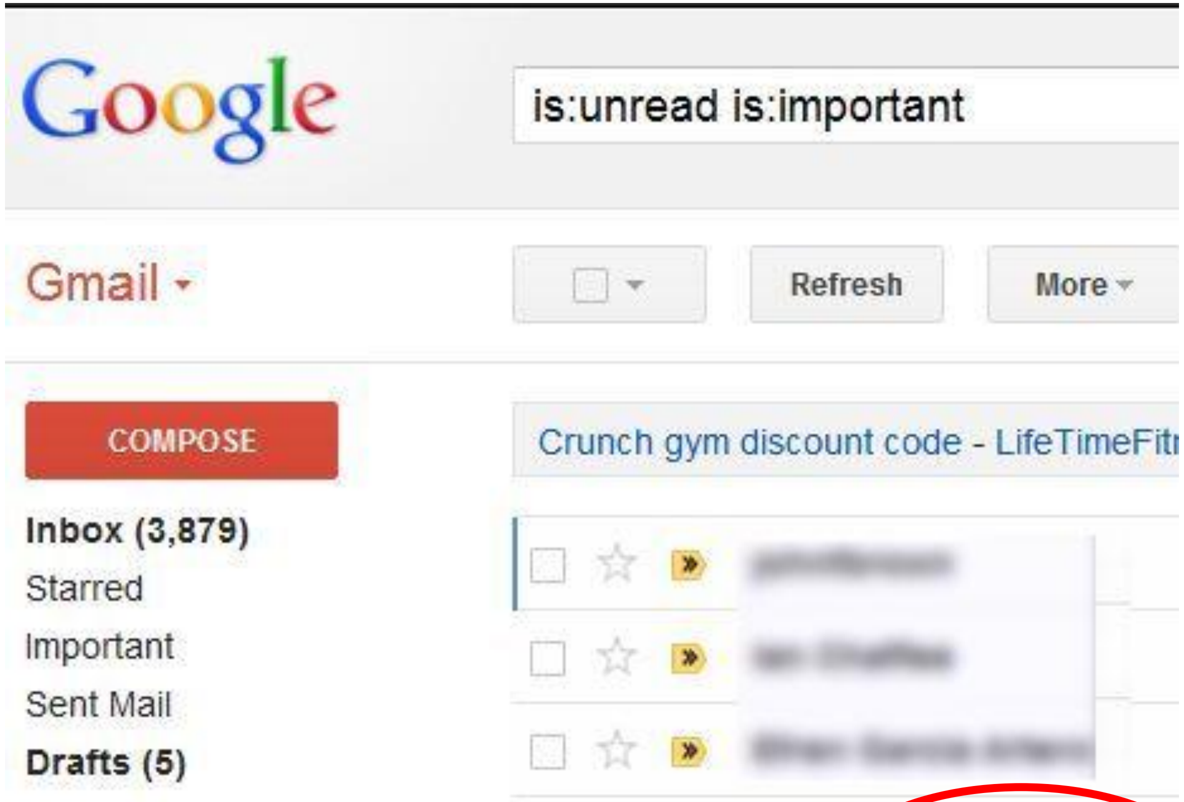
The Streaming Model

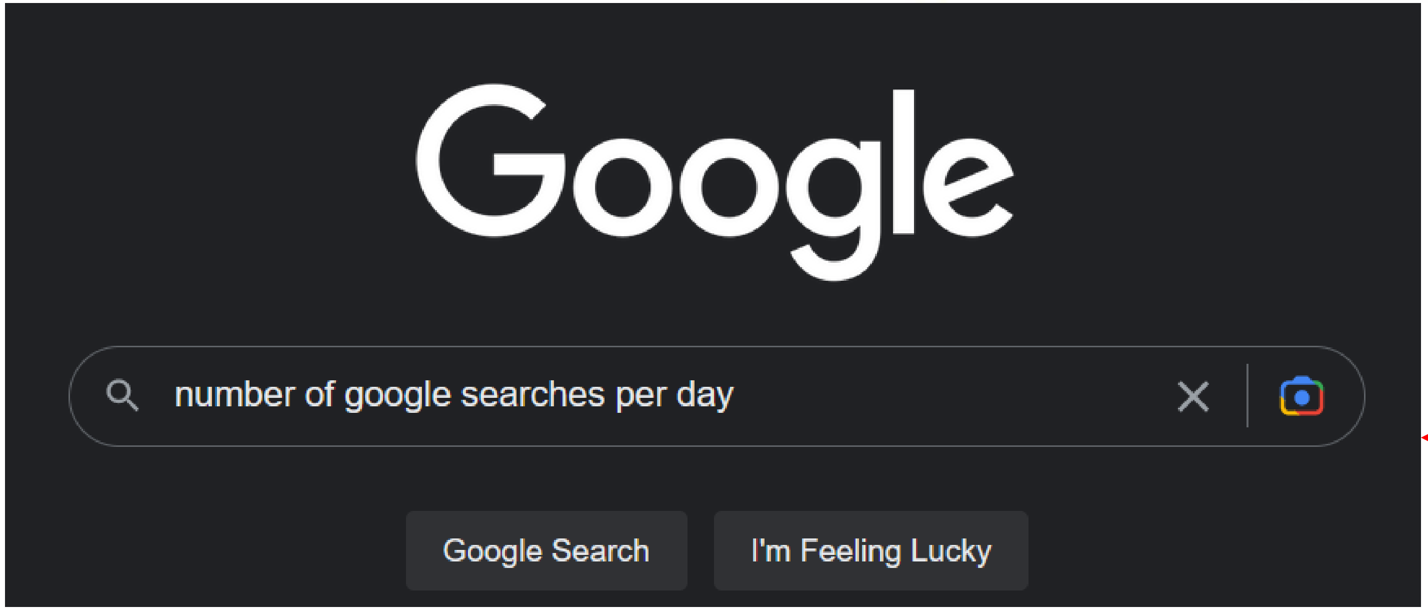
- Internet of Things (IoT): home automation (security cameras, smart devices), medical care (health monitoring devices, pacemakers), traffic cameras and travel time sensors (smart cities), electrical grid monitoring

The Streaming Model

- Financial markets
- Traffic network monitoring







330 billion daily e-mails

3 billion monthly active users

8.5 billion daily Google searches

The Streaming Model

- Scenario: We are given a massive dataset that arrives in a continuous stream, which we would like to analyze – but we do not have enough space to store all the items

- Typically the data must be compressed on-the-fly
- Store a data structure from which we can still learn useful information

The Streaming Model

- Input: Elements of an underlying data set S, which arrive sequentially
- Output: Evaluation (or approximation) of a given function
- Goal: Use space *sublinear* in the size m of the input S

1 0 1 1 1 0 0 1

IP Address	Extended IP Address	Sorted IP Address
15.231.156.11	015.231.156.011	015.231.156.011
55.188.89.38	055.188.089.038	055.188.089.038
82.102.176.196	082.102.176.196	082.102.176.196
111.89.188.4	111.089.188.004	111.089.188.004
111.197.241.108	111.197.241.108	111.197.241.108
114.122.13.1	114.122.013.001	114.122.013.001
114.122.102.3	114.122.102.003	114.122.102.003
122.12.11.5	122.012.011.005	122.012.011.005
125.245.42.185	125.245.042.185	125.245.042.185
139.72.251.251	139.072.251.251	139.072.251.251
148.179.4.219	148.179.004.219	148.179.004.219
152.227.163.70	152.227.163.070	152.227.163.070
188.133.95.141	188.133.095.141	188.133.095.141
192.144.1.16	192.144.001.016	192.144.001.016
200.173.128.224	200.173.128.224	200.173.128.224
232.111.123.221	232.111.123.221	232.111.123.221
236.154.17.169	236.154.017.169	236.154.017.169

The Streaming Model

- Input: Elements of an underlying data set S, which arrive sequentially
- Output: Evaluation (or approximation) of a given function
- Goal: Use space *sublinear* in the size m of the input S

- Compared to traditional algorithmic design, which focuses on minimizing runtime, the big question here is how much space is needed to answer queries of interest

Sampling

- Suppose we see a stream of elements from $[n]$. How do we uniformly sample one of the positions of the stream?

47 72 81 10 14 33 51 29 54 9 36 46 10

Sampling

- Suppose we see a stream of elements from $[n]$. How do we uniformly sample one of the positions of the stream?

47 72 81 10 14 33 51 29 54 9 36 46 10

Reservoir Sampling

- Suppose we see a stream of elements from $[n]$. How do we uniformly sample one of the positions of the stream?
- [Vitter 1985]: Initialize $s = \perp$
- On the arrival of element i, replace s with x_i with probability $\frac{1}{i}$

47 72 81 10 14 33 51 29 54 9 36 46 10

Reservoir Sampling

- Suppose the stream has length m. What is the probability that $s = x_t$ for fixed $t \in [m]$?

47 72 81 10 14 33 51 29 54 9 36 46 10

Reservoir Sampling

- Suppose the stream has length m. What is the probability that $s = x_t$ for fixed $t \in [m]$?
- Must have chosen $s = x_t$ at time t AND must have never updated s afterwards

47 72 81 10 14 33 51 29 54 9 36 46 10

Reservoir Sampling

- Suppose the stream has length m. What is the probability that $s = x_t$ for fixed $t \in [m]$?
- Must have chosen $s = x_t$ at time t AND must have never updated s afterwards
- Must have chosen $s = x_t$ at time t AND did not update s at time $t + 1$ AND did not update s at time $t + 2$ AND did not update s at time $t + 3$ AND … AND did not update s at time m

Reservoir Sampling

- Must have chosen $s = x_t$ at time t
- AND did not update s at time $t + 1$
- AND did not update s at time $t + 2$
- AND did not update s at time $t + 3$
- AND …
- AND did not update s at time m

Reservoir Sampling

- Must have chosen $s = x_t$ at time t
- AND did not update s at time $t + 1$
- AND did not update s at time $t + 2$
- AND did not update s at time $t + 3$
- AND …
- AND did not update s at time m

Happens with probability $\frac{1}{t}$

Reservoir Sampling

- Must have chosen $s = x_t$ at time t
- AND did not update s at time $t + 1$
- AND did not update s at time $t + 2$
- AND did not update s at time $t + 3$
- AND …
- AND did not update s at time m

Happens with probability $\frac{1}{t}$

Happens with probability $1 - \frac{1}{t+1}$

Reservoir Sampling

- Must have chosen $s = x_t$ at time t
- AND did not update s at time $t+1$
- AND did not update s at time $t+2$
- AND did not update s at time $t+3$
- AND …
- AND did not update s at time m

Happens with probability $\frac{1}{t}$

Happens with probability $1 - \frac{1}{t+1}$

Happens with probability $1 - \frac{1}{t+2}$

Reservoir Sampling

- Must have chosen $s = x_t$ at time t
- AND did not update s at time $t + 1$
- AND did not update s at time $t + 2$
- AND did not update s at time $t + 3$
- AND …
- AND did not update s at time m

Happens with probability $\frac{1}{t}$

Happens with probability $1 - \frac{1}{t+1}$

Happens with probability $1 - \frac{1}{t+2}$

Happens with probability $1 - \frac{1}{m}$

Reservoir Sampling

- Must have chosen $s = x_t$ at time t
- AND did not update s at time $t+1$
- AND did not update s at time $t+2$
- AND did not update s at time $t+3$
- AND …
- AND did not update s at time m

Happens with probability $\frac{1}{t}$

Happens with probability $1 - \frac{1}{t+1}$

Happens with probability $1 - \frac{1}{t+2}$

Happens with probability $1 - \frac{1}{m}$

$$\Pr[s = x_t] = \frac{1}{t} \times \frac{t}{t+1} \times \frac{t+1}{t+2} \times \cdots \times \frac{m-1}{m} = \frac{1}{m}$$

Frequency Vector

- Given a set S of m elements from $[n]$, let f_i be the frequency of element i. (How often it appears)

$$1\ 1\ 2\ 1\ 2\ 1\ 1\ 2\ 3 \rightarrow [5, 3, 1, 0] \coloneqq f$$

Frequent Items

- Given a set S of m elements from $[n]$, let f_i be the frequency of element i. (How often it appears)

f_1	f_2	f_3	f_4	f_5	f_6	f_7
10	0	1	1	2	0	9

- Goal: Given a set S of m elements from $[n]$ that induces a frequency vector f, find the “large” coordinates of f

Frequent Items

- Data mining: Finding top products/viral objects, e.g., Google searches, Amazon products, YouTube videos, etc.
- Traffic network monitoring: Finding IP addresses with high volume traffic, e.g., detecting distributed denial of service (DDoS) attacks, network anomalies)
- Database design: Finding iceberg queries, i.e., items in a database with high volume of queries

- Want fast response and running list of frequent items, i.e., cannot process entire database for each query/update

Frequent Items

- Goal: Given a set S of m elements from $[n]$ and a parameter k, output the k elements i with the largest frequency f_i

f_1	f_2	f_3	f_4	f_5	f_6	f_7
10	0	1	1	2	0	9

- Return the k elements with the largest frequency

- Natural approach: store the count for each item and return the k elements with the largest frequency

Frequent Items

- Goal: Given a set S of m elements from $[n]$ and a parameter k, output the k elements i with the largest frequency f_i

f_1	f_2	f_3	f_4	f_5	f_6	f_7
10	0	1	1	2	0	9

- Return the k elements with the largest frequency

- Natural approach: store the count for each item and return the k elements with the largest frequency, uses $O(n)$ space

- MUST USE LINEAR SPACE

Frequent Items

- Goal: Given a set S of m elements from $[n]$ and a parameter k, output the items from $[n]$ that have frequency at least $\frac{m}{k}$

f_1	f_2	f_3	f_4	f_5	f_6	f_7
10	0	1	1	2	0	9

- How many items can be returned? At most k coordinates with frequency at least $\frac{m}{k}$

- For $k = 20$, want items that are at least 5% of the stream

Frequent Items

- Goal: Given a set S of m elements from $[n]$ and a parameter $k = 2$, output the items from $[n]$ that have frequency at least $\frac{m}{2}$

- Find the item that forms the majority of the stream

1

3

3

1

$$-2$$

1

1

7

1

1

3

3

1

1

3

3

Majority

- Goal: Given a set $S = \{x_1, \dots, x_m\}$ of m elements from $[n]$ and a parameter $k = 2$, output the items from $[n]$ that have frequency at least $\frac{m}{2}$
- Initialize item $V = 1$ with count $c = 0$
- For updates $1, \dots, m$:
  - If $c = 0$, set $V = x_i$
  - Else if $V = x_i$, increment counter c by setting $c = c + 1$
  - Else if $V \neq x_i$, decrement counter c by setting $c = c - 1$

Majority

- Initialize item $V = 1$ with count $c = 0$
- For updates $1, \ldots, m$:
  - If $c = 0$, set $V = x_i$
  - Else if $V = x_i$, increment counter c by setting $c = c + 1$
  - Else if $V \neq x_i$, decrement counter c by setting $c = c - 1$

- Let M be the true majority element
- Let z be a helper variable with $z = +1$ when $V = M$ and $z = -1$ when $V \neq M$

Majority

- Let M be the true majority element
- Let z be a helper variable with $z = +1$ when $V = M$ and $z = -1$ when $V \neq M$

- Since M is the majority, then z is positive at the end of the stream, so algorithm ends with $V = M$

- $O(\log m + \log n)$ bits of space
- $O(\log n)$ bits of space for $m \leq n^{\alpha}$ for fixed constant α
- For simplicity, let's assume $m = \Theta(n)$