Data stream

Danupon Nanongkai

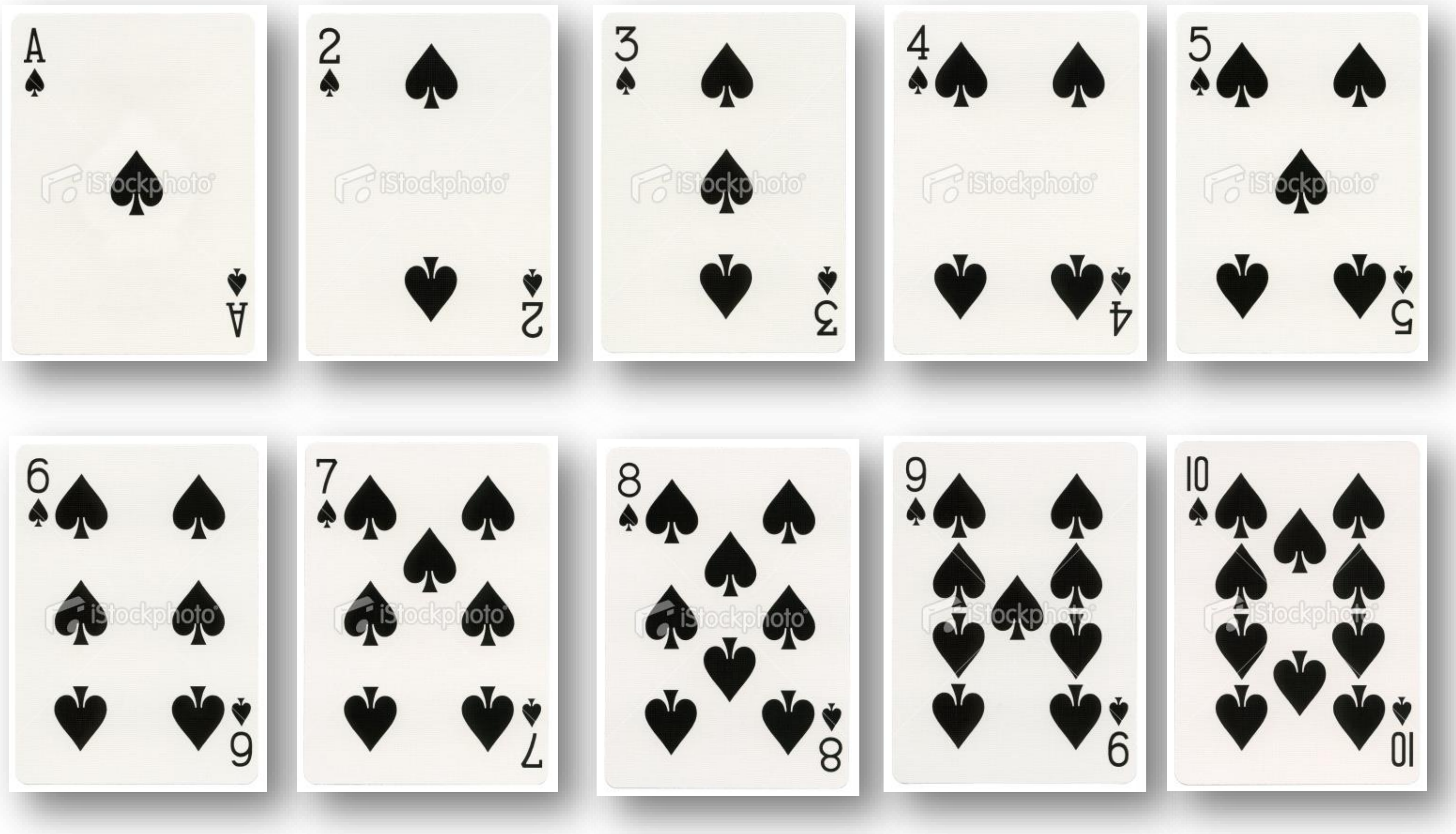
Part 1

One-pass streaming model

Part 1.1

Warm-up: Finding a missing number

There are 10 cards: A, 2, 3, …, 10



There are 10 cards: A, 2, 3, …, 10

What is the removed card?

I will show you one number at a time

I will show you one number at a time

I will show you one number at a time

I will show you one number at a time

I will show you one number at a time

I will show you one number at a time

I will show you one number at a time

I will show you one number at a time

I will show you one number at a time

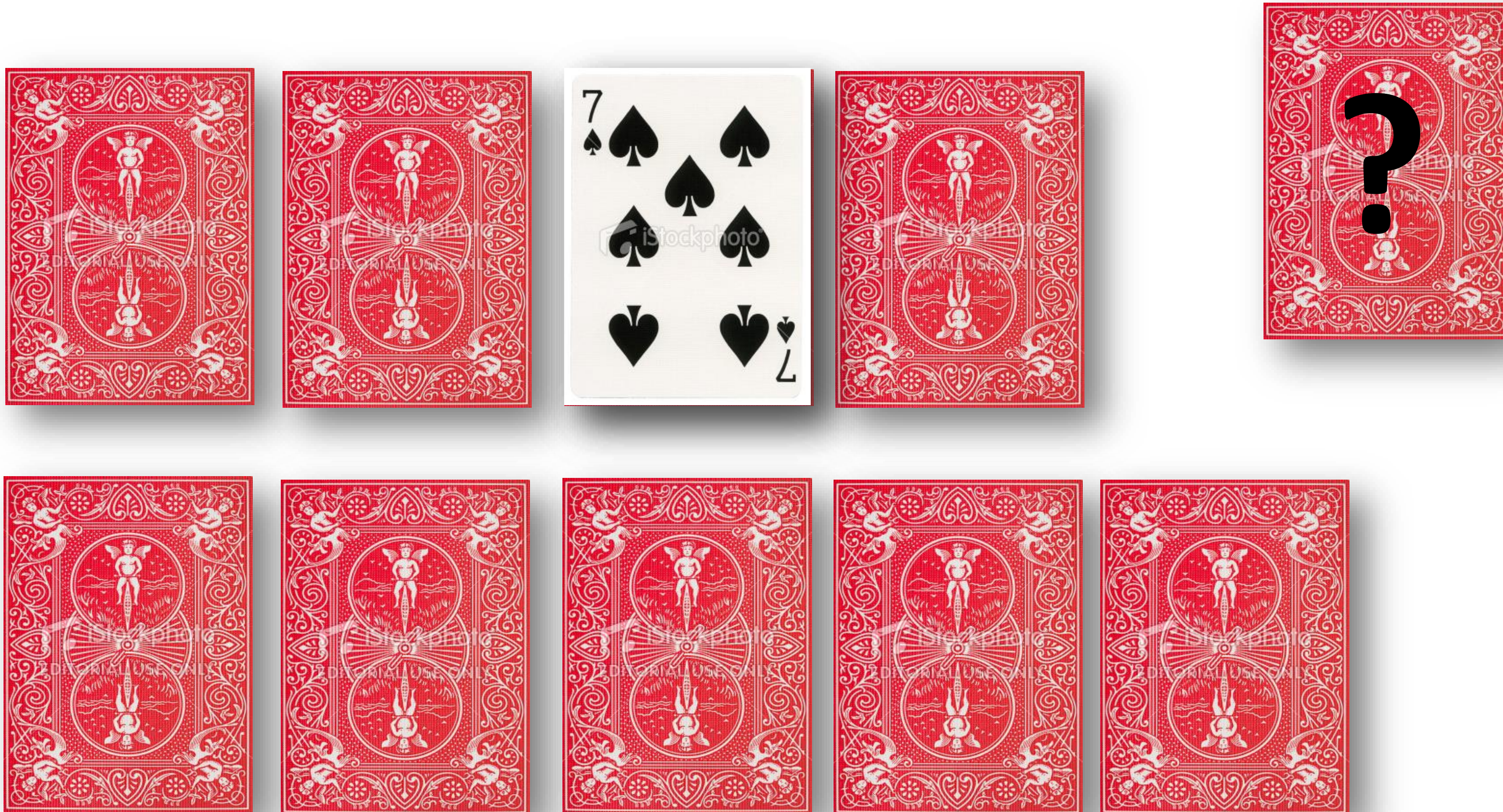
Let’s try again

- Hint: It is enough to just memorize one number

I will show you one number at a time

I will show you one number at a time

I will show you one number at a time



I will show you one number at a time

I will show you one number at a time

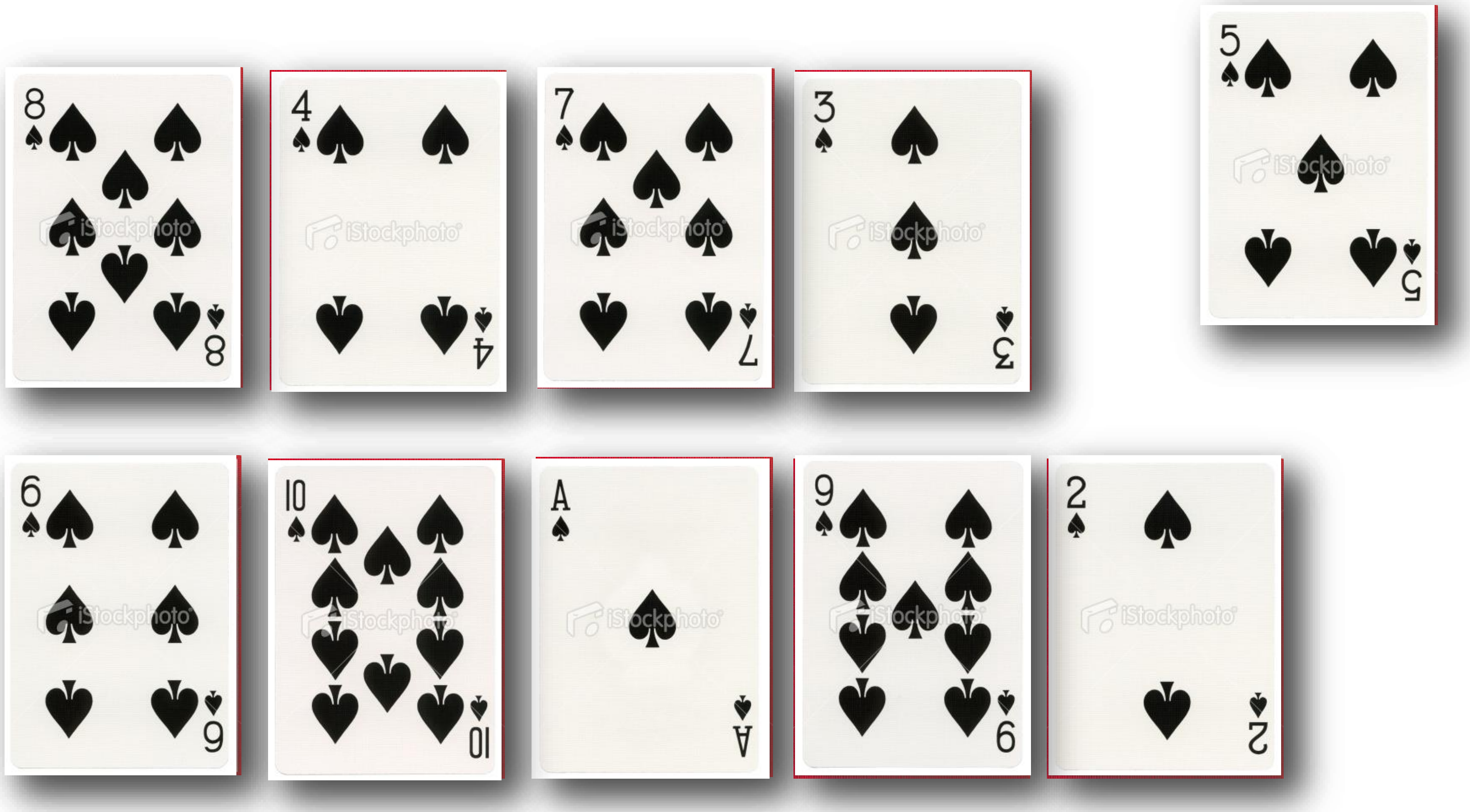
I will show you one number at a time

I will show you one number at a time

I will show you one number at a time

I will show you one number at a time

Answer

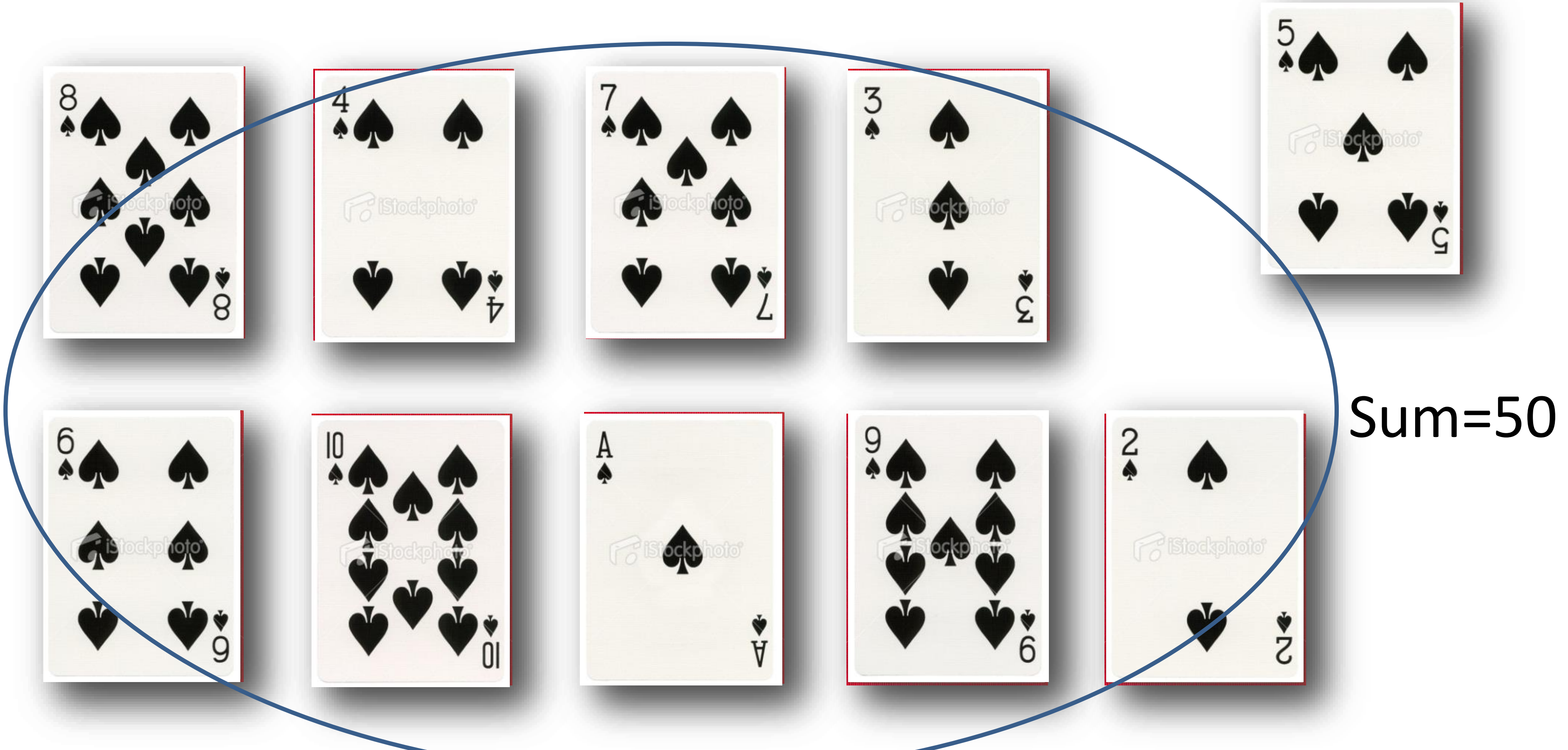


Algorithm

- Memorize the sum of all numbers
- 1+2+3+…+10 = 55

Algorithm

- Memorize the sum of all numbers
- 1+2+3+…+10 = 55



What are the removed cards?

What are the removed cards?

Remove **x** cards → remember **x** numbers

(Let me know after the class if you need a hint)

Further extensions

- The same number could appear many times
- I can say “insert number i” and “delete number j”
- This becomes a research problem with applications.
  - Yaron Minsky, Ari Trachtenberg, Richard Zippel: **Set reconciliation with nearly optimal communication complexity**. IEEE Transactions on Information Theory 49(9): 2213-2218 (2003)

Data Stream Model

Data Stream Model

1 3 5 9 2 10 11 12 20

O(polylog n) RAM

n = range of input numbers

Data Stream Model

1 3 5 9 2 10 11 12 20

O(polylog n) RAM

n = range of input numbers

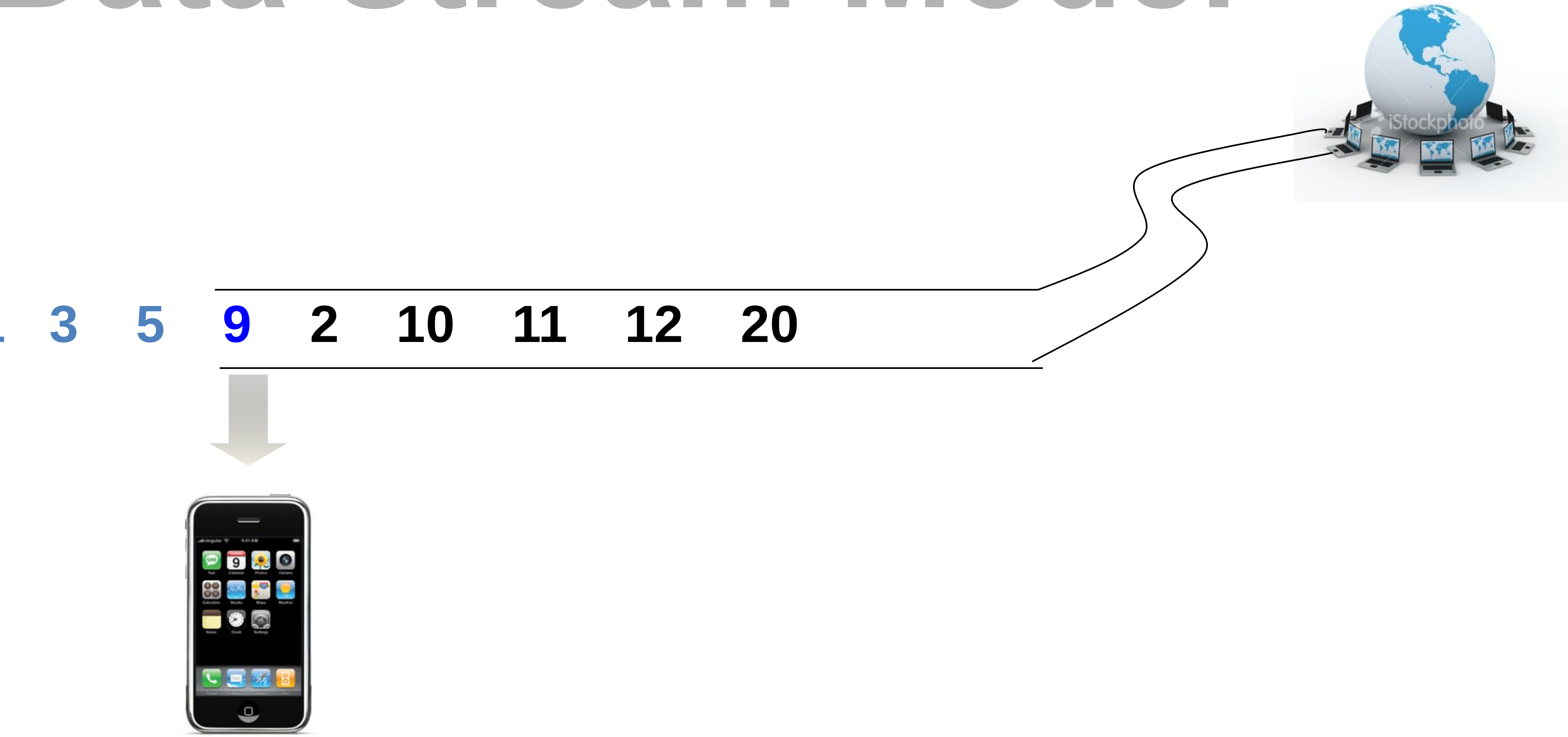
Data Stream Model

1 3 **5** 9 2 10 11 12 20

O(polylog n) RAM

n = range of input numbers

Data Stream Model



O(polylog n) RAM

n = range of input numbers

Applications

- Synchronization in distributed systems
- Network monitoring and traffic engineering
- Sensor networks, RFID tags
- Telecom call records
- Financial applications
- Web logs and click-streams
- Manufacturing processes
- Databases (Data Stream Management Systems)

Part 1.2
Majority

Majority Problem

- Given **m** numbers: $a_1, a_2, \ldots, a_m$
  - Example: 10, 5, 2, 1, 1 ,5, 1, 1, 1
- If there is a number x that appears $> m/2$ times then output x.
  - Example: 10, 5, 2, 1, 1 ,5, 1, 1, 1
  - Output "1"
- Otherwise, output anything
- Example: 10, 5, 2, 1, 1 ,5, 1, 5, 1
  - Can output either "1", "2", "5" or "10"

Hint: Remember 2 numbers:

$ID \in \{a_1, \ldots, a_m\}$ and counter

$c \leq m$

Misra-Gries Algorithm (1982)

- Maintain a counter and an ID.
  - If new item is same as stored ID, increment counter.
  - Otherwise, decrement the counter.
  - If counter 0, store new item with count = 1.

- If counter > 0, then its item is the only candidate for majority.

Misra-Gries Algorithm (1982)

- Initially, $c = 0$
- When we read a_i from stream:
  1. If $c = 0$ then set $ID = a_i$ and $c = 1$
  2. Else if $ID = a_i$ then $c = c + 1$
  3. Else $(ID \neq a_i)$ then $c = c - 1$
- After we finish reading the stream:
  - If $c = 0$ then answer "NONE"
  - Else answer ID

Example:

	2	9	9	9	7	6	4	9	9	9	3	9
ID	2	2	9	9	9	9	4	4	9	9	9	9
count	1	0	1	2	1	0	1	0	1	2	1	2

Exercise 1: 10, 5, 2, 1, 1 ,5, 1, 1, 1

Exercise 2: 10, 1, 2, 1, 1 ,5, 1, 5, 5

Next:

1. Why the algorithm is correct?
2. Find numbers that appear $>m/3$ times

Why is the algorithm correct?

If x occurs $> m/2$ times then …

- ID will be x at some point (why?)
  - Because there are not enough number to keep it away
- It’s counter cannot be 0 (why?)
  - Because there are not enough number to decrease it
- Imagine: 10,5,2,1,1,5,1,1,1 → yyyxxyxxx
  - x will outnumber y at some point

More formal analysis

If x occurs $> m/2$ times then …

- One ID will be x at some point (why?)
  - Let's say that a_i is **bad** if $ID \neq x$ after reading a_i

		☹						☹				
Example: x="9"	2	9	9	9	7	6	4	9	9	9	3	9
ID	2	2	9	9	9	9	4	4	9	9	9	9
count	1	0	1	2	1	0	1	0	1	2	1	2

  - There could be $< \mathrm{m}/2$ bad a_i (why?)
    - » Because a_i will decrease counter of other number $\mathrm{y} \neq x$.
    - » How many such y are there?
      - <m/2

More formal analysis (2)

If x occurs $> m/2$ times then …

- One ID will be x at some point (why?)
- x's counter cannot be 0 (why?)
  - Let's say that a_i is **bad** if $ID \neq x$ after reading a_i **OR** $ID \neq x$ after reading a_j for some j>i

Example: x="9"

		☹	☹	☹				☹				
	2	9	9	9	7	6	4	9	9	9	3	9
ID	2	2	9	9	9	9	4	4	9	9	9	9
count	1	0	1	2	1	0	1	0	1	2	1	2

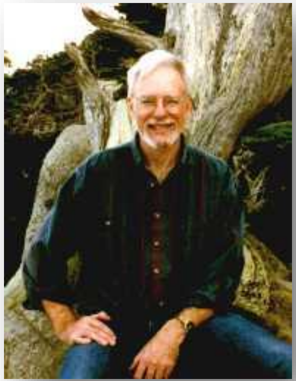
  - There could be <m/2 bad a_i (why?)
    - » Because a_i will decrease counter of other number y $\neq x$ OR prevent y $\neq x$ from being counted.
  - There is at least one good a_i. (why?)
    - » Because x appears >m/2 times.
    - » The good a_i is the one that survives and makes the counter > 0

A tale

- Discovered by Boyer and Moore in 1980
- Repeatedly rejected for publication
- Misra and Gries extended to “finding frequent items” (next)
- People always refer to this algorithm as Misra-Gries Algorithm
- Finally published in 1991

- Read the full story at http://blogs.law.harvard.edu/pamphlet/2011/09/23/tales-of-peer-review-episode-1-boyer-and-moores-mjrty-algorithm/

Boyer



Moore

Misra

Gries

Part 1.3
Finding frequent items

Finding frequent items

- Given **k** and **m** numbers: $a_1, a_2, \ldots, a_m$
- Goal: Output **k** numbers $x_1, x_2, \ldots, x_k$ such that if there is a number y that appears $> m/(k+1)$ times then $x_i = y$ for some i.
  - How many such numbers are there?
  - Majority is when k=1

  - Example 1: k=2, m=11: 10, 5, 2, 1, 1 ,5, 1, 5, 1, 5, 1
  - Output “1” and “5”

  - Example 2: k=2, m=11: 10, 5, 2, 1, 1 ,5, 1, **4**, 1, **3**, 1
  - Output “1” and “i”, for any number i

Misra-Gries Algorithm (1982)

- Maintain **k** items and their counters.
  - If next item x is one of the k, increment its counter.
  - Else if there is a zero counter, put x there with count = 1
  - Else (all counters non-zero) decrement **all** k counters

- The survived k items are the only candidate frequent items

Example 1: k=2, m=11: 10, 5, 2, 1, 1 ,5, 1, 5, 1, 5, 1

Example 2: k=2, m=11: 10, 5, 2, 1, 1 ,5, 1, **4**, 1, **3**, 1

Can you prove correctness yourself?

Correctness

If x occurs $> m/(k+1)$ times then …

- One ID will be x at some point (why?)
  - Let's say that a_i is **bad** if $ID \neq x$ after reading a_i
  - There could be $< \mathrm{m}/(k+1)$ bad a_i (why?)
    - » Because a_i will decrease counters of k other number $\mathrm{y}_1 \neq x, \ldots, y_k \neq x$.
    - » How many times each $\mathrm{y_i}$ can appear?
      - One of them appears $< \mathrm{m}/(k+1)$ times

Correctness (2)

If x occurs $> m/(k+1)$ times then …

- One ID will be x at some point (why?)
- x's counter cannot be 0 (why?)
  - Let's say that a_i is **bad** if $ID \neq x$ after reading a_i **OR** $ID \neq x$ after reading a_j for some j>i
  - There could be $< \mathrm{m}/(k+1)$ bad a_i (why?)
    - » Because a_i will decrease counters of k other numbers $y_1 \neq x, \ldots, y_k \neq x$ **OR** *prevent k other numbers from being counted*
  - There is at least one good a_i. (why?)
    - » Because x appears >m/(k+1) times.
    - » The good a_i is the one that survives and makes the counter > 0

Theorem

- For any x, let f_x be the frequency of x in $a_1, a_2, \ldots, a_m$.
- For any k, Misra-Gries algorithm (with k counters) can output f_x' such that
$$f_x - \frac{m}{k} \leq f_x' \leq f_x$$
- Can you prove this?

Part 1.4

L_0 and L_1 Sampling

L_1 Sampling

L_1 Sampling Problem

- This is one of the popular job interview problems
- We are given $a_1, a_2, \ldots \in \{1, \ldots, n\}$
- After we read a_i, we want to maintain a random number from $a_1, \ldots, a_i$.
- Example: Example: 1 5 1 1 1
  - Output 1 with probability 4/5 and 5 with probability 1/5
- Do this by maintaining **one number** plus the count of stream size

L_1 Sampling Algorithm

- Initially, $x = a_1$. When read a_i, let $x = a_i$ with probability $\frac{1}{i}$
- Can you prove that this algorithm works?
- This is called “reservoir sampling”.

Application of L_1 Sampling

Approximate median ("middle number"):

- Sample k numbers from the stream.
- Output the median of sampled numbers
- If k is large enough, then the output will be close to the middle.
  - Can you analyze it?
- More generally: quantile *approximation*
  - A number x is an ε-approximate α-quantile if its rank is in $[\alpha-\varepsilon, \alpha+\varepsilon]$.
  - Many cool algorithms.

L_0 Sampling

L_0 Sampling Problem

- We are given $a_1, a_2, \ldots \in \{1, \ldots, n\}$
- After we read a_i, let **S** be the set of numbers that appeared in $a_1, a_2, \ldots, a_i$.
- Output one number in S uniformly at random.
- Example: 1 5 1 1 1
  - L_0 sampling outputs 1 and 5 with same probability
  - L_1 sampling outputs 1 with probability 4/5 and 5 with probability 1/5

Assumption on Random Hash Function

- Assume that for any k, we can construct a **random hash function** $h: \{1..n\} \rightarrow \{1..k\}$.
- That is, for any i $\in \{1..n\}$ and j $\in \{1..k\}$, $\Pr[h(i) = j] = 1/k$.
- Moreover, h uses small space
  - i.e. $O(\log n)$ space.
  - We will see how to construct it later

* In fact, we won't actually have such a random hash function, but something close enough

Warm-up case: Algorithm

- Assume know **k**, the number of distinct elements that will appear.
- Example: 1 5 1 1 1 → k=2

Algorithm:

- Initially, construct random hash $h: \{1..n\} \to \{1..k\}$.
- When read a_i: Remember a_i if $h(a_i) = 1$.
- If there are more than one number remembered: FAIL.
- Otherwise, output the remembered number.

Warm-up case: Intuition

Intuition:

- Let **S** be the set of k numbers that appear in stream
- Question: In expectation, how many numbers in S should be mapped by h to 1? That is $E[x \in S \mid h(x) = 1] =?$
- Answer: 1 since there are k places these k numbers can map to.
- So, there is a “good” chance that there is only one number remembered.

Analysis:

- Next time (need some concentration bounds).

Part 1.5
Number of distinct elements

Number of distinct elements

- Given a stream $a_1, \dots, a_m$ where $a_i \in [n]$
  - $[n] = \{1, \dots, n\}$
- How many distinct elements are there in the stream?
- Example 1: 10, 5, 2, 1, 1 ,5, 1, 5, 1, 5, 1
- Example 2: 10, 5, 2, 1, 1 ,5, 1, **4**, 1, **3**, 1

- Assume: there is a family of hash functions
$$\mathrm{H} = \{h: [n] \to [n]\}$$
such that each h uses only $O(\log n)$ space
  - For now, let's not worry how it works

Algorithm

Preprocess:

- Pick a random $h \in H$.
  - For any a and b, $\Pr[h(a) = b] = 1/n$
- $z = 0$

When read $\boldsymbol{a_i}$:

- If $zeros\big(h(a_i)\big) > z$ then $z = zeros(h(a_i))$
  - $zeros(a)$=number of "rightmost zeros" in binary
  - Example: $8 = 1000_2$ so $zeros(8) = 3$

Output $2^{z+\frac{1}{2}}$

Example

- 10, 5, 2, 1, 1 ,5, 1, 5, 1, 5, 1

Intuition

- Pick k random numbers $X_1, \dots, X_k$
- What is the expected largest number of rightmost zeros?
  - $E[\max(zeros(X_1), \dots, zeros(X_k))] = ?$
- Imagine that you generate each bit of X_i by tossing a coin
- When things are "perfectly uniform":
  - Half of them will be $010 \dots 0010\mathbf{1}$
  - Half of the rest will be $010 \dots 010\mathbf{1}0$
  - Half of the rest will be $100 \dots 00\mathbf{1}00$
- If there are k distinct elements then $\mathrm{z} = \max(zeros(X_1), \dots, zeros(X_k))$ "should" be $\log k$
- So, we answer 2^z but we multiply $\sqrt{2}$ to offset some errors

Proof: All I know about probability

- $Var[A] = Var[(A - E[A])^2] = E[A^2] - (E[A])^2$
- If $A = \sum_i B_i$'s and B_i are independent then $Var[A] = \sum_i Var[B_i]$
  - In contrast, $\mathrm{E}[A] = \sum_i E[B_i]$ even when B_i's are NOT independent
- Markov's inequality:

$$\Pr[A \geq t] \leq E[A]/t$$

- Chebyshev's inequality:

$$\Pr[|A - E[A]| \geq t] \leq Var[A]/t^2$$

- Chernoff's inequality: If we repeat A for n times and let $B = \sum_i A_i$

$$\Pr[B > (1 + t)E[B]] \leq 2^{-(1+t)E[B]}$$

(Useful when $E[B]$ is large, e.g., tossing coin n times)

Theorem

- The algorithm outputs d' such that
$$\Pr\left[\frac{d}{3} \leq d' \leq 3\right] \geq 1 - 2^{\Omega(k)}$$
- Here, d is the correct answer
- This is 3-approximation!
- Can we get a smaller approximation?
- Yes: Try replace "3" by something else and go through the analysis again.

History

- Flajolet-Martin 1985
- Alon-Matias-Szegedy 1999
  - Known as AMS algorithm(s)
  - Algorithms and lower bounds for frequency moment problem (F_p)
  - Distinct element problem is F_0
  - Won Godel prize in 2005
  - Foundation of data stream (along with Henzinger-Ragahavan-Rajagopalan'99 and Munro-Paterson'78)

Flajolet

Martin

Alon

Matias

Szegedy

Resources

- Yaron Minsky, Ari Trachtenberg, Richard Zippel: Set reconciliation with nearly optimal communication complexity. IEEE Transactions on Information Theory 49(9): 2213-2218 (2003)
- Jayaev Misra, David Gries: Finding Repeated Elements. Sci. Comput. Program. 2(2): 143-152 (1982)
- S. Muthukrishnan's book http://algo.research.googlepages.com/eight.ps
- Amit Chakrabarti's lecture note http://www.cs.dartmouth.edu/~ac/Teach/CS49-Fall11/Notes/lecnotes.pdf

Part 2

Multi-pass streaming model

What is multi-pass stream?

Multi-pass Streaming model

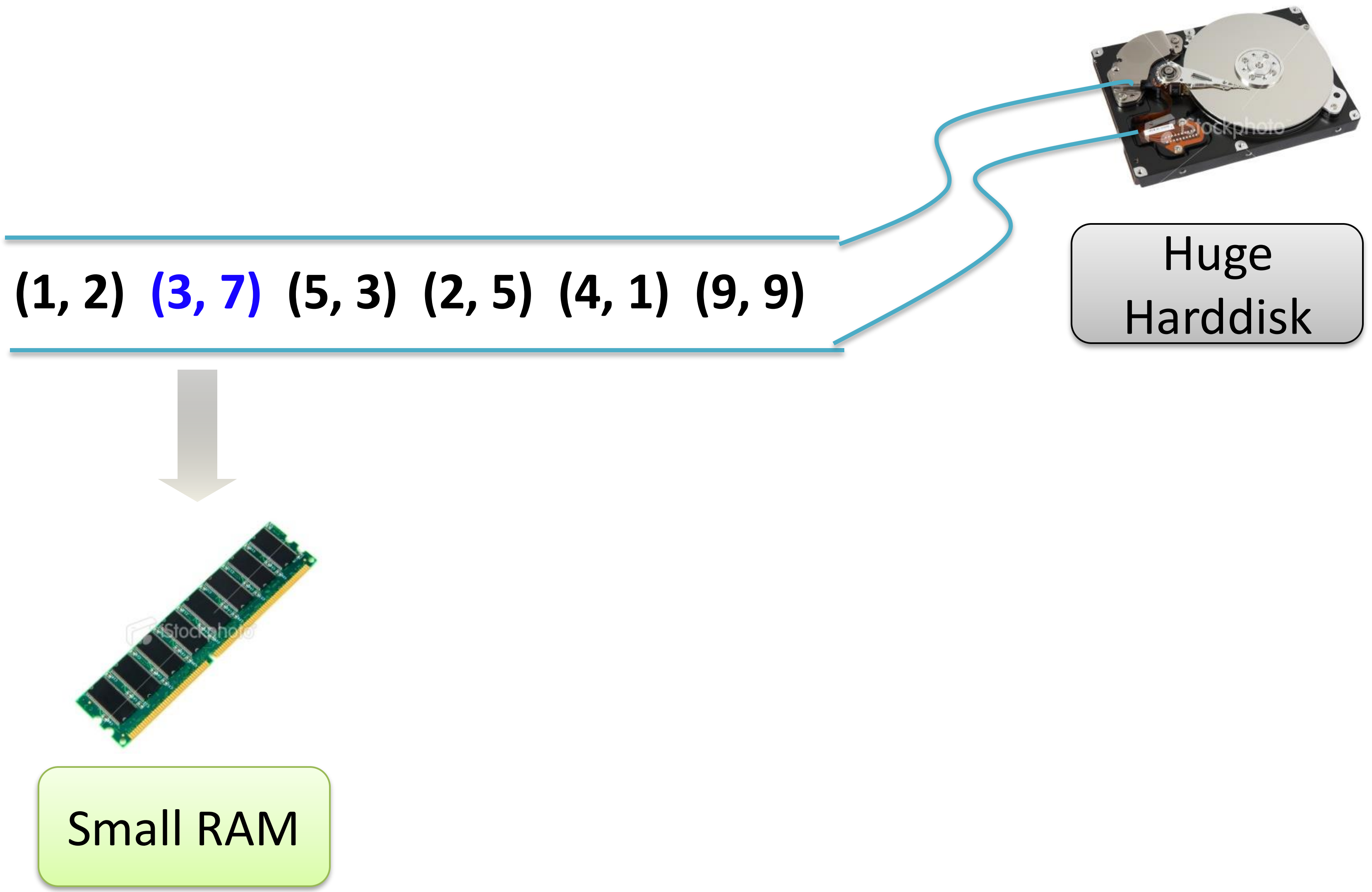
numbers, points, edges, etc

Huge Harddisk

(1, 2) (3, 7) (5, 3) (2, 5) (4, 1) (9, 9)

Small RAM

Multi-pass Streaming model



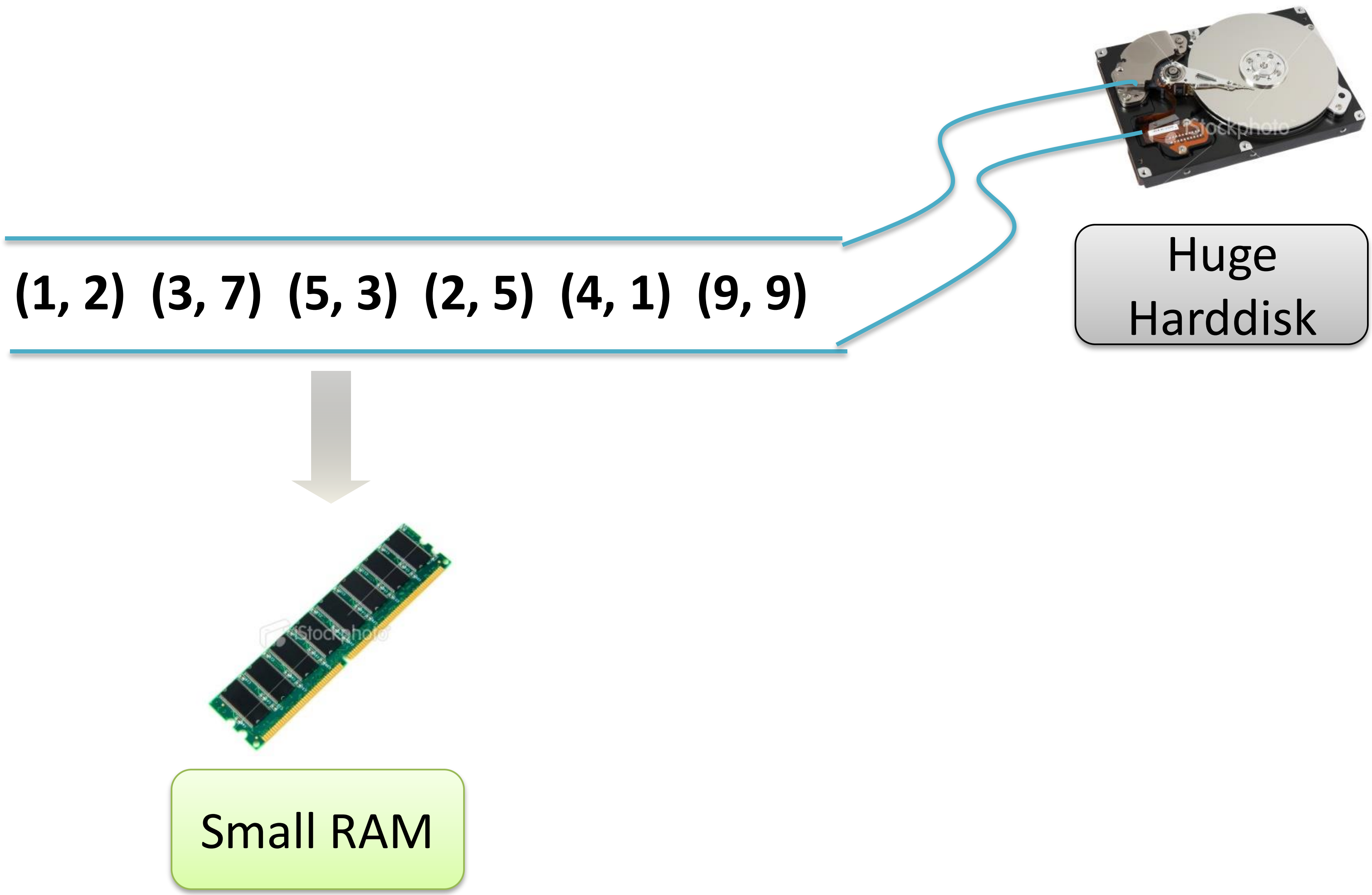
Multi-pass Streaming model

Huge Harddisk

(1, 2) (3, 7) (5, 3) (2, 5) (4, 1) (9, 9)

Small RAM

Multi-pass Streaming model



Multi-pass Streaming model

Huge Harddisk

(1, 2) (3, 7) (5, 3) (2, 5) (4, 1) (9, 9)

Small RAM

2^{nd} pass

Multi-pass Streaming model

Huge Harddisk

(1, 2) (3, 7) (5, 3) (2, 5) (4, 1) (9, 9)

Small RAM

3rd pass

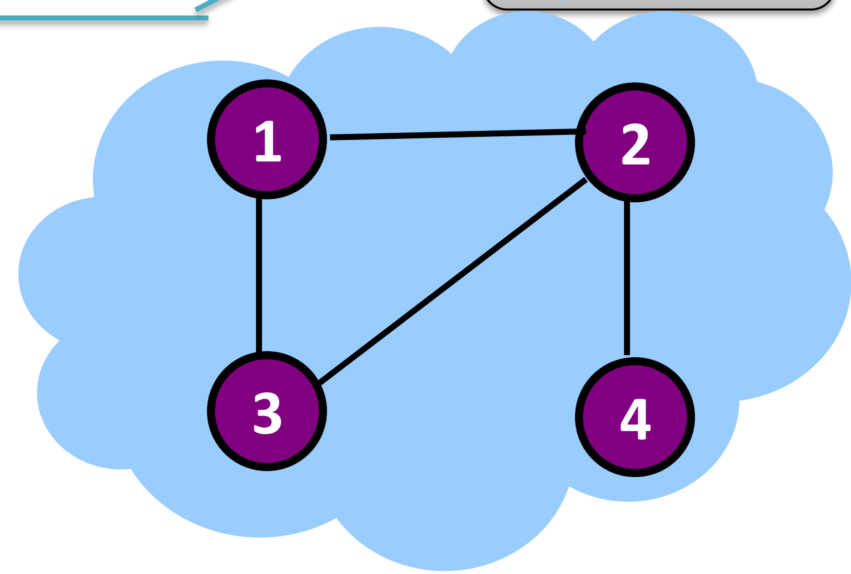
Multi-pass Graph Streaming Model

Multi-pass **Graph** Streaming model

Huge harddisk

(1, 2) (2, 4) (1, 3) (2, 3)

Small RAM



Unweighted Maximum Matching

- Given a graph as a steam, find a matching of maximum size
- Can you find this in 1 pass and $O(n^2 \text{polylog } n)$ space?

Unweighted Maximum Matching

- Given a graph as a steam, find a matching of maximum size
- Can you find this in 1 pass and $O(n^2 \text{polylog } n)$ space?
- 2-approximation in 1 pass $O(n \text{ polylog } n)$ space
- How?

Extension

- Can get $(1+\varepsilon)$-approximation using O(n polylog n) space and *constant* passes (depending on ε)
- Quite complicated and we will skip it here
- When the space is O(n polylog n), some people called it by a fancy name “semi-stream”

Multi-pass Geometric Streaming Model

Multi-pass **Geometric** Streaming model

Huge Harddisk

(1, 2) (3, 7) (5, 3) (2, 5) (4, 1) (9, 9)

Small RAM

Multi-pass Geometric Streaming model

Huge Harddisk

(1, 2) (3, 7) (5, 3) (2, 5) (4, 1) (9, 9)

Small RAM

Multi-pass **Geometric** Streaming model

(1, 2) (3, 7) (5, 3) (2, 5) (4, 1) (9, 9)

Huge Harddisk

Small RAM

Multi-pass **Geometric** Streaming model

Huge Harddisk

(1, 2) (3, 7) (5, 3) (2, 5) (4, 1) (9, 9)

Small RAM

Multi-pass **Geometric** Streaming model

Huge Harddisk

(1, 2) (3, 7) (5, 3) (2, 5) (4, 1) (9, 9)

Small RAM

2^{nd} pass

Multi-pass **Geometric** Streaming model

Huge Harddisk

(1, 2) (3, 7) (5, 3) (2, 5) (4, 1) (9, 9)

Small RAM

3rd pass

Example: Skyline computation

Skyline shows all “interesting” cars

	Car	MPG	HP
p1	Toyota Prius	51	134
~~p2~~	~~Honda Civic Hybrid~~	~~40~~	~~110~~
p3	Ford Fusion	41	191
p4	Nissan Altima Hybrid	35	198
~~p5~~	~~Volkswagen Jetta TDI~~	~~30~~	~~140~~

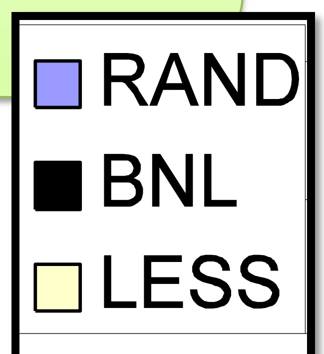
dominate

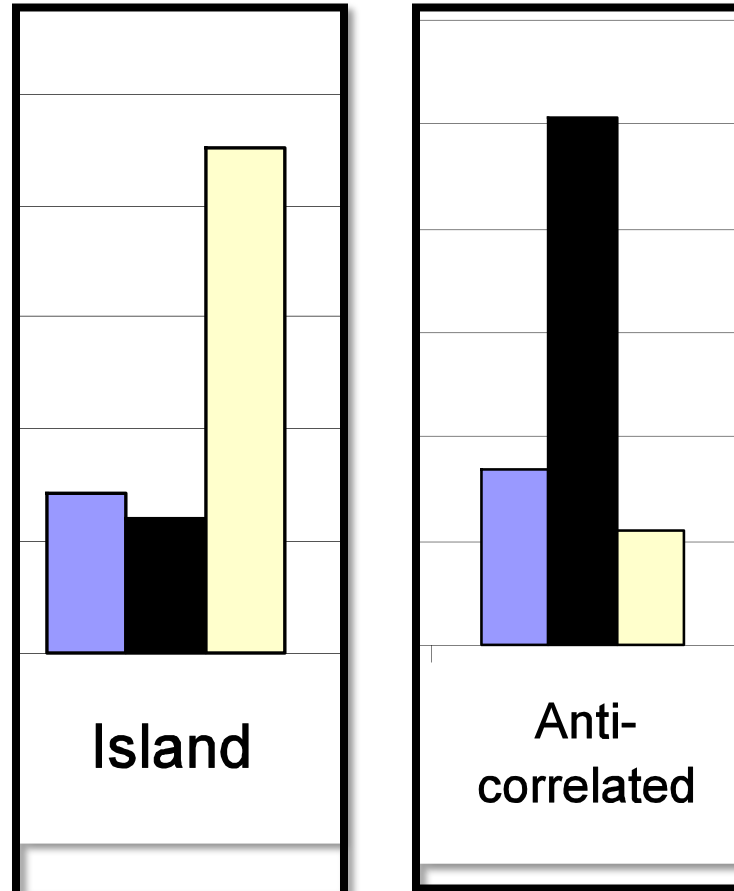
dominate

Experimental Results

<u>Average case</u>

- No clear winner between BNL and LESS
- RAND is always close to the winner





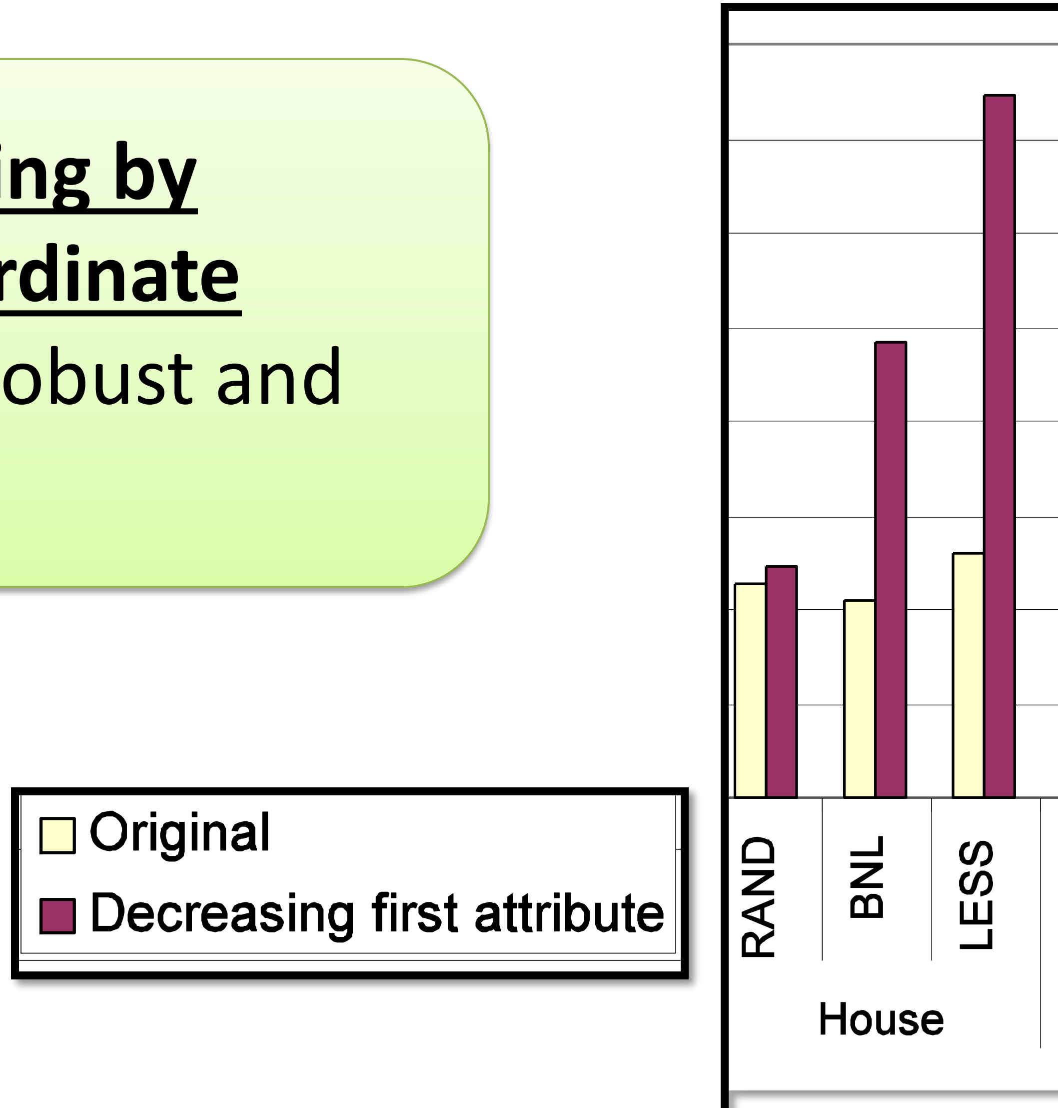
RAND

BNL & LESS

Experimental Results

<u>"Worse": After sorting by decreasing first coordinate</u>

- RAND is the most robust and usually fastest

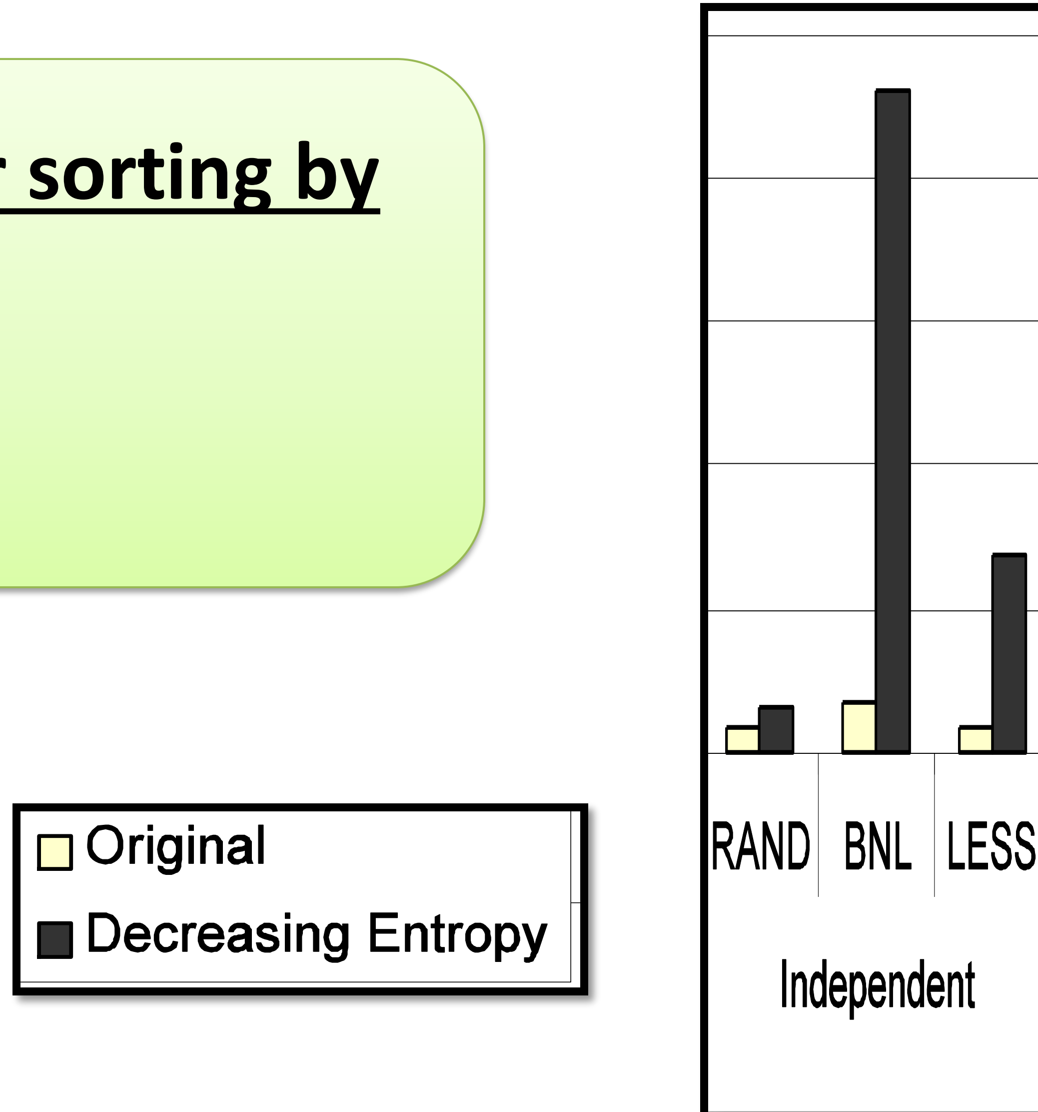


RAND

BNL & LESS

Experimental Results

<u>"Even Worse": After sorting by "entropy"</u>



RAND

BNL & LESS