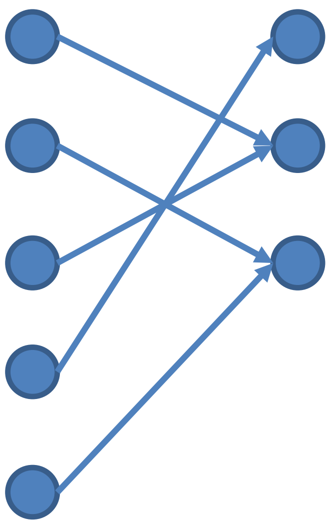
Algorithms at Scale

(Week 4)

Puzzle of the Day:



A bipartite graph:

- m left nodes, n right nodes [numbered 1 to n]
- Each left node has degree 1.

Given $\log(m)$ space, a stream of edges:

1. $m = n + 1$, max right degree is 2, min right degree is 1.
   See stream once. Find *the* edge with degree 2.
2. $m = n + 2$, max right degree is 2, min right degree is 1.
   See stream once. Find two edges with degree 2.
3. $m = n + 1$, max right degree is m. See stream $\log(m)$ times.
   Find *any* edge with degree > 1.

Summary

Last Week: Property Testing

Sorting: Is this array sorted?

- Gap-style question: sorted or far from sorted?

Yao's Lemma:

- Key technique for proving lower bound.
- Show that testing if something is sorted has inherent cost.

CS5234 Part 1: Sublinear Time

Approximation:

Toy example 2: Faction of 1's?

- Additive $\pm\,\varepsilon$ approximation

Number of connected components in a graph.

- Additive $\pm\,\varepsilon$ approximation.

Weight of MST

- Multiplicative $(1 \pm \varepsilon)$ approximation.

Size of maximal matching

- Additive $\pm\,\varepsilon$ approximation.

PAC learning

- Approximate concept.

Property Testing:

Toy example 1: array all 0's?

- All 0's or far from all 0's?

Is the graph connected?

- Connected or far from connected?

Image properties

- Is the image divisible?
- Is the image a rectangle?
- Is the image convex?

Is the array sorted?

- Sorted or far from sorted?

CS5234 Part 1: Sublinear Time

Appr

Toy e

•

Num
com

•

Weig

•

?

Size o

• Additive ± ε approximation.

PAC learning

• Approximate concept.

Is the array sorted?

• Sorted or far from sorted?

Basic recipe: sampling

1. Identify *local* property.
2. Sample from dataset, measure local property.
3. Show E[out] = goal (i.e., unbiased estimator).
4. Show answer is close to expectation (via Chernoff, Hoeffding, Markov, Chebychev).

CS5234 Part 2: Sublinear Space

Data arrives in a stream: $S = s_1, s_2, \ldots, s_T$

Examples:

- Twitter tweet stream
  - ⇒ On average, how many hashtags per tweet?
  - ⇒ How many unique users tweet per day?
  - ⇒ On average, how many times does a user tweet per day?
- Facebook friend updates
  - ⇒ How many connected components in the Facebook graph?
  - ⇒ Is the Facebook graph k-connected?
  - ⇒ How many triangles are there in the Facebook graph?
- Sensor data
  - ⇒ What is the average temperature for region xxx?
- Stock market
  - ⇒ What was the most traded stock in December 2017?
  - ⇒ What was the average stock price of MSFT in 2018?
  - ⇒ If MSFT went up, did GOOGL go up or down?

Summary

Today: Data

Counting distinct elements:

- How many items in the stream?

Item frequencies:

- How often does an item appear in a stream?

Heavy hitters:

- Identify the most frequent items

Statistics

- Average, median, etc.

Next Weeks: Graphs

Connectivity:

- Is the graph connected?

MST:

- Find an MST

Matching:

- Approximate the maximal matching.

Shortest paths:

- Approximate the shortest paths in a graph.

Triangles:

- How many triangles in a graph?

Announcements / Reminders

Problem sets:

Problem Set 3 will be released tonight.

Frequencies / Heavy Hitters

Given a stream of items:

$$S = s_1, s_2, \ldots, s_m$$

Assume:

- length of stream: m
- allowable space: small (e.g., logarithmic)

Example:
[A, B, B, D, A, B, B, E, H, B, J, B, B, B, A, A]

m = 16
count(A) = 4
count(B) = 8
count(J) = 1

heavy(1/2) = {B}
heavy(1/4) = {A,B}

Find:

- count(x) : number of times x appears in stream.
- heavy hitters : every item that appears at least εm times.

Parameter: ε

Frequencies / Heavy Hitters

Given a stream of items:

$S = s_1, s_2, \ldots, s_m$

Assume:

- length of s
- allowable s
  (e.g., logar

Find:

- count(x) : number of times x appears in stream.
- heavy hitters : every item that appears at least εm times.

Parameter: ε

Example:
[A, B, B, D, A, B, B, E, H, B, J, B, B, B, A, A]

m = 16
count(A) = 4
count(B) = 8

Impossible

(prove it)

Frequencies / Heavy Hitters

Given a stream of items:

$S = s_1, s_2, \ldots, s_m$

Assume:

- length of stream: m
- allowable space: small
  (e.g., logarithmic)
- $N(x)$ = number of times times x appears in stream.

Find:

- count(x) : $N(x) - \varepsilon m \leq \text{count}(x) \leq N(x) + \varepsilon m$
- heavy hitters : return
  1. every item that appears $\geq 2\varepsilon m$ times.
  2. no item that appears $< \varepsilon m$ times.

Frequencies / Heavy Hitters

count(x) : $N(x) - \varepsilon m \leq \text{count}(x) \leq N(x) + \varepsilon m$

Example:
[A, B, B, D, A, B, B, E, H, B, J, B, B, B, A, A]

$m = 16$
$\varepsilon = 1/8$
N(A) = 4
N(B) = 8
N(J) = 1
N(L) = 0

count(A) = 6
count(A) = 2

count(B) = 10

count(J) = 0
count(L) = 2

Frequencies / Heavy Hitters

heavy hitters : return

1. every item that appears $\geq 2\varepsilon m$ times.
2. no item that appears $< \varepsilon m$ times.

Example:
[A, B, B, D, A, B, B, D, H, B, J, B, B, B, A, A]

$m = 16$
$\varepsilon = 1/8$
$N(A) = 4$
$N(B) = 8$
$N(D) = 2$
$N(J) = 1$

must return: A, B

may return: D

may NOT return: J

Challenge: Small Space

With arbitrary space:

- Maintain m counters.
- Use a hash table.
- Use a counting Bloom filter

With small space:

- Cannot maintain counts of all items!
- Try to maintain counts only for frequent items.

Misra-Gries Algorithm

Key parameter: k

Set P of <item, count> pairs.

For each u in stream S:

1. if <u, c> is in set P, increment c.
2. else add <u, 1> to set P.
3. if |P| > k, decrement count c for each item.
4. Remove all items from P with count c=0.

Count(x):

1. if <x, c> is in set P, return c.
2. else return 0.

Misra-Gries Algorithm

Set P of <item, count> pairs.

For each u in stream S:

1. if <u, c> is in set P, increment c.
2. else add <u, 1> to set P.
3. if |P| > k, decrement count c for each item.
4. Remove all items from P with count c=0.

Example stream (k=2):

2, 5, 7, 2, 2, 5, 5, 5, 5, 7, 2

Misra-Gries Algorithm

Set P of <item, count> pairs.

For each u in stream S:

1. if <u, c> is in set P, increment c.
2. else add <u, 1> to set P.
3. if |P| > k, decrement count c for each item.
4. Remove all items from P with count c=0.

Example stream (k=2):

2, 5, 7, 2, 2, 5, 5, 5, 5, 7, 2

(2, 1)

Misra-Gries Algorithm

Set P of <item, count> pairs.

For each u in stream S:

1. if <u, c> is in set P, increment c.
2. else add <u, 1> to set P.
3. if |P| > k, decrement count c for each item.
4. Remove all items from P with count c=0.

Example stream (k=2):

2, 5, 7, 2, 2, 5, 5, 5, 5, 7, 2

(2, 1)
(5, 1)

Misra-Gries Algorithm

Set P of <item, count> pairs.

For each u in stream S:

1. if <u, c> is in set P, increment c.
2. else add <u, 1> to set P.
3. if |P| > k, decrement count c for each item.
4. Remove all items from P with count c=0.

Example stream (k=2):

2, 5, 7, 2, 2, 5, 5, 5, 5, 7, 2

(2, 1)
(5, 1)
(7, 1)

Misra-Gries Algorithm

Set P of <item, count> pairs.

For each u in stream S:

1. if <u, c> is in set P, increment c.
2. else add <u, 1> to set P.
3. if |P| > k, decrement count c for each item.
4. Remove all items from P with count c=0.

Example stream (k=2):

2, 5, 7, 2, 2, 5, 5, 5, 5, 7, 2

(2, 0)
(5, 0)
(7, 0)

Misra-Gries Algorithm

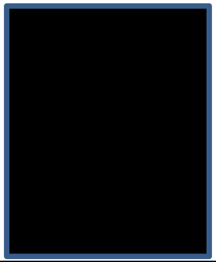
Set P of <item, count> pairs.

For each u in stream S:

1. if <u, c> is in set P, increment c.
2. else add <u, 1> to set P.
3. if $|P| > k$, decrement count c for each item.
4. Remove all items from P with count c=0.

Example stream (k=2):

2, 5, 7, 2, 2, 5, 5, 5, 5, 7, 2



Misra-Gries Algorithm

Set P of <item, count> pairs.
For each u in stream S:

1. if <u, c> is in set P, increment c.
2. else add <u, 1> to set P.
3. if |P| > k, decrement count c for each item.
4. Remove all items from P with count c=0.

Example stream (k=2):
2, 5, 7, 2, 2, 5, 5, 5, 5, 7, 2

(2, 1)

Misra-Gries Algorithm

Set P of <item, count> pairs.

For each u in stream S:

1. if <u, c> is in set P, increment c.
2. else add <u, 1> to set P.
3. if |P| > k, decrement count c for each item.
4. Remove all items from P with count c=0.

Example stream (k=2):

2, 5, 7, 2, 2, 5, 5, 5, 5, 7, 2

(2, 2)

Misra-Gries Algorithm

Set P of <item, count> pairs.

For each u in stream S:

1. if <u, c> is in set P, increment c.
2. else add <u, 1> to set P.
3. if |P| > k, decrement count c for each item.
4. Remove all items from P with count c=0.

Example stream (k=2):

2, 5, 7, 2, 2, 5, 5, 5, 5, 7, 2

(2, 2)
(5, 1)

Misra-Gries Algorithm

Set P of <item, count> pairs.

For each u in stream S:

1. if <u, c> is in set P, increment c.
2. else add <u, 1> to set P.
3. if |P| > k, decrement count c for each item.
4. Remove all items from P with count c=0.

Example stream (k=2):

2, 5, 7, 2, 2, 5, 5, 5, 5, 7, 2

(2, 2)
(5, 4)

Misra-Gries Algorithm

Set P of <item, count> pairs.

For each u in stream S:

1. if <u, c> is in set P, increment c.
2. else add <u, 1> to set P.
3. if |P| > k, decrement count c for each item.
4. Remove all items from P with count c=0.

Example stream (k=2):

2, 5, 7, 2, 2, 5, 5, 5, 5, 7, 2

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

Misra-Gries Algorithm

Set P of <item, count> pairs.

For each u in stream S:

1. if <u, c> is in set P, increment c.
2. else add <u, 1> to set P.
3. if |P| > k, decrement count c for each item.
4. Remove all items from P with count c=0.

Example stream (k=2):

2, 5, 7, 2, 2, 5, 5, 5, 5, 7, 2

(2, 1)
(5, 3)
(7, 0)

Misra-Gries Algorithm

Set P of <item, count> pairs.

For each u in stream S:

1. if <u, c> is in set P, increment c.
2. else add <u, 1> to set P.
3. if |P| > k, decrement count c for each item.
4. Remove all items from P with count c=0.

Example stream (k=2):

2, 5, 7, 2, 2, 5, 5, 5, 5, 7, 2

(2, 1)
(5, 3)

Misra-Gries Algorithm

Set P of <item, count> pairs.

For each u in stream S:

1. if <u, c> is in set P, increment c.
2. else add <u, 1> to set P.
3. if |P| > k, decrement count c for each item.
4. Remove all items from P with count c=0.

Example stream (k=2):

2, 5, 7, 2, 2, 5, 5, 5, 5, 7, 2

(2, 2)
(5, 3)

Misra-Gries Algorithm

Set P of <item, count> pairs.

For each u in stream S:

1. if <u, c> is in set P, increment c.
2. else add <u, 1> to set P.
3. if $|P| > k$, decrement count c for each item.
4. Remove all items from P with count $c=0$.

Example stream (k=2):

2, 5, 7, 2, 2, 5, 5, 5, 5, 7, 2

(2, 2)
(5, 3)

Claim: space = $O(k \log(m))$

(Count all the bits you need to store k counts up to m)

Misra-Gries Algorithm

Set P of <item, count> pairs.

For each u in stream S:

1. if <u, c> is in set P, increment c.
2. else add <u, 1> to set P.
3. if |P| > k, decrement count c for each item.
4. Remove all items from P with count c=0.

Example stream (k=2):

2, 5, 7, 2, 2, 5, 5, 5, 5, 7, 2

Is the answer good?

(2, 2)
(5, 3)

Misra-Gries Algorithm

Set P of <item, count> pairs.

For each u in stream S:

1. if <u, c> is in set P, increment c.
2. else add <u, 1> to set P.
3. if |P| > k, decrement count c for each item.
4. Remove all items from P with count c=0.

Example stream (k=2):

2, 5, 7, 2, 2, 5, 5, 5, 5, 7, 2

(2, 2)
(5, 3)

Is the answer good?

- count(7) = 0

Misra-Gries Algorithm

Set P of <item, count> pairs.

For each u in stream S:

1. if <u, c> is in set P, increment c.
2. else add <u, 1> to set P.
3. if |P| > k, decrement count c for each item.
4. Remove all items from P with count c=0.

Example stream (k=2):

2, 5, 7, 2, 2, 5, 5, 5, 5, 7, 2

(2, 2)
(5, 3)

Is the answer good?

- count(7) = 0
- count(2) = 2

Misra-Gries Algorithm

Set P of <item, count> pairs.

For each u in stream S:

1. if <u, c> is in set P, increment c.
2. else add <u, 1> to set P.
3. if |P| > k, decrement count c for each item.
4. Remove all items from P with count c=0.

Example stream (k=2):

2, 5, 7, 2, 2, 5, 5, 5, 5, 7, 2

(2, 2)
(5, 3)

Is the answer good?

- count(7) = 0
- count(2) = 2
- count(5) = 3

Misra-Gries Algorithm

Set P of <item, count> pairs.

For each u in stream S:

1. if <u, c> is in set P, increment c.
2. else add <u, 1> to set P.
3. if |P| > k, decrement count c for each item.
4. Remove all items from P with count c=0.

Claim: count(x) ≤ N(x)

Why? Only increment <x, c> at most N(x) times.

Misra-Gries Algorithm

Set P of <item, count> pairs.

For each u in stream S:

1. if <u, c> is in set P, increment c.
2. else add <u, 1> to set P.
3. if $|P| > k$, decrement count c for each item.
4. Remove all items from P with count $c=0$.

Claim: $\text{count}(x) \geq N(x) - (m/k)$

Misra-Gries Algorithm

Claim: $count(x) \geq N(x) — (m/k)$

Proof:

1. Count of x is incremented $N(x)$ times total.

Misra-Gries Algorithm

Claim: count(x) ≥ N(x)—(m/k)

<u>Proof:</u>

1. Count of x is incremented N(x) times total.
2. Total number of increments is m = $\sum_x N(x)$.

Misra-Gries Algorithm

Claim: count(x) ≥ N(x)—(m/k)

<u>Proof:</u>

1. Count of x is incremented N(x) times total.
2. Total number of increments is m = $\sum_x N(x)$.
3. When count(x) is decremented, at least k other items are *also* decremented.

Misra-Gries Algorithm

Claim: count(x) ≥ N(x)—(m/k)

<u>Proof:</u>

1. Count of x is incremented N(x) times total.
2. Total number of increments is m = $\sum_x N(x)$.
3. When count(x) is decremented, at least k other items are *also* decremented.
4. At most m decrements in total.

Misra-Gries Algorithm

Claim: count(x) ≥ N(x)—(m/k)

<u>Proof:</u>

1. Count of x is incremented N(x) times total.
2. Total number of increments is m = $\sum_x N(x)$.
3. When count(x) is decremented, at least k other items are *also* decremented.
4. At most m decrements in total.
5. So count(x) is decremented at most m/k times.

Misra-Gries Algorithm

Claim: space $= O(k \log(m))$

Claim: $N(x) \geq \text{count}(x) \geq N(x) - (m/k)$

Misra-Gries Algorithm

Claim: space = O(k log(m))

Claim: N(x) ≥ count(x) ≥ N(x)—(m/k)

$\boxed{\text{Choose } k = 1/\varepsilon}$

Misra-Gries Algorithm

Claim: space $= O(k \log(m))$

Claim: $N(x) \geq \text{count}(x) \geq N(x) - (m/k)$

$\boxed{\text{Choose } k = 1/\varepsilon}$

Claim: space $= O(1/\varepsilon)$

Claim: $N(x) \geq \text{count}(x) \geq N(x) - \varepsilon m$

Heavy Hitters

How to use Misra-Gries to solve Heavy Hitters problem?

Heavy Hitters

How to use Misra-Gries to solve Heavy Hitters problem?

Return x if $\text{count}(x) \geq \varepsilon m$

Heavy Hitters

How to use Misra-Gries to solve Heavy Hitters problem?

Return x if count(x) $\geq \varepsilon m$

Condition 1: if $N(x) \geq 2\varepsilon m$, then include x.

Heavy Hitters

How to use Misra-Gries to solve Heavy Hitters problem?

Return x if $\text{count}(x) \geq \varepsilon m$

Condition 1: if $N(x) \geq 2\varepsilon m$, then include x.

➔ $\text{count}(x) \geq 2\varepsilon m - \varepsilon m = \varepsilon m$

➔ return x

Heavy Hitters

How to use Misra-Gries to solve Heavy Hitters problem?

Return x if count(x) $\geq \varepsilon m$

Condition 2: if $N(x) < \varepsilon m$, then DO NOT include x.

Heavy Hitters

How to use Misra-Gries to solve Heavy Hitters problem?

Return x if $\text{count}(x) \geq \varepsilon m$

Condition 2: if $N(x) < \varepsilon m$, then DO NOT include x.

➔ $\text{count}(x) < \varepsilon m$

➔ DO NOT return x

Frequencies / Heavy Hitters

Given a stream of items:

$$S = s_1, s_2, \ldots, s_m$$

Assume:

- length of stream: m
- allowable space: $O(\log(m)/\varepsilon)$
- $N(x)$ = number of times times x appears in stream.

Find:

- count(x) : $N(x) - \varepsilon m \leq \text{count}(x) \leq N(x) + \varepsilon m$
- heavy hitters : return
  1. every item that appears $\geq 2\varepsilon m$ times.
  2. no item that appears $< \varepsilon m$ times.

Summary

Today: Data

Item frequencies:

- How often does an item appear in a stream?

Heavy hitters:

- Identify the most frequent items

Counting distinct elements:

- How many items in the stream?

Statistics

- Average, median, etc.

Next Weeks: Graphs

Connectivity:

- Is the graph connected?

MST:

- Find an MST

Matching:

- Approximate the maximal matching.

Shortest paths:

- Approximate the shortest paths in a graph.

Triangles:

- How many triangles in a graph?

Number of Distinct Items

Given a stream of items:

$S = s_1, s_2, \ldots, s_m$

Assume:

- length of stream: m
- allowable space: small (e.g., logarithmic)

Example:
[A, B, B, D, A, B, B, E, H, B, J, B, B, B, A, A]

m = 16

distinct = 6
distinct(1/3) >= 4
distinct(1/3) <= 8

Find:

- distinct : number of distinct items in stream.
- distinct(ε): $(1\pm \varepsilon)$ approximation with probability at least $(1-\delta)$.

Parameters: ε, δ

Challenge: Small Space

With arbitrary space:

- Use a hash table.
- Use a Bloom filter

With small space:

- Can you solve it with Misra-Gries?

Challenge: Small Space

With arbitrary space:

- Use a hash table.
- Use a Bloom filter

With small space:

- Can you solve it with Misra-Gries? NO
  - Cannot distinguish 0 from 1 appearance.
- Need another trick…

Challenge: Small Space

Trick 1: Hash Function

- Assume a hash function $h(x) \rightarrow [1,N]$.
- Assume it is perfectly random, i.e., each item x is mapped to a random item in $[1,N]$.

Key points:

- Every time you see x it is mapped to the same hash.
- Collisions are still possible!

Challenge: Small Space

Trick 1: Hash Function

- Assume a hash function $h(x) \rightarrow [1,N]$.
- Assume it is perfectly random, i.e., each item x is mapped to a random item in $[1,N]$.

Key points:

- Every time you see x it is mapped to the same hash.
- Collisions are still possible!
- How much space to store hash function?

Challenge: Small Space

Trick 1: Hash Function

- Assume a hash function $h(x) \rightarrow [1,N]$.
- Assume it is perfectly random, i.e., each item x is mapped to a random item in $[1,N]$.

Key points:

- Every time you see x it is mapped to the same hash.
- Collisions are still possible!
- How much space to store hash function?
  - Need $\Omega(n \log N)$ bits!
  - Need to store hash value for each possible item.
- Can use *k-wise-independent* hash functions instead.

Challenge: Small Space

Trick 1.1: Hash Function

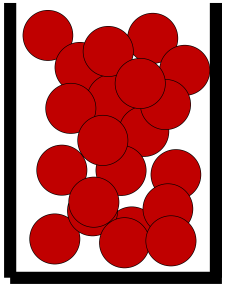
- Assume a hash function $h(x) \rightarrow [0,1]$.
- Assume it is perfectly random, i.e., each item x is mapped to a random item in $[0,1]$.

Key points:

- To simplify the math, let's map to $[0,1]$ instead.
- Easy to translate to discrete model. (Exercise!)

Challenge: Small Space

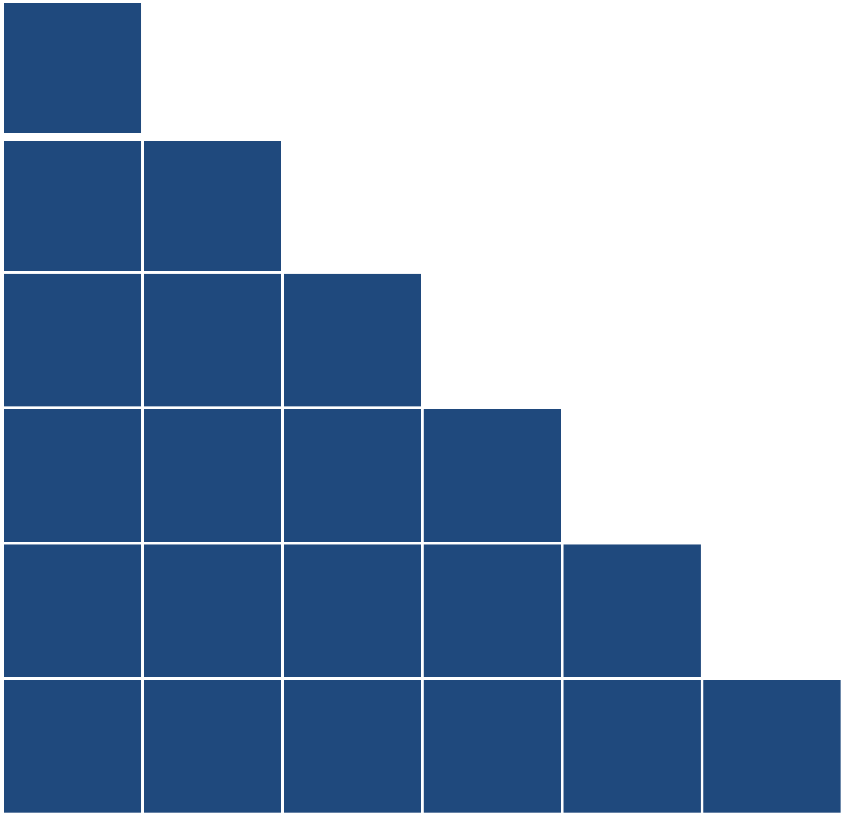
Trick 2: minimum of a set of random variables



Imagine a bucket of balls.

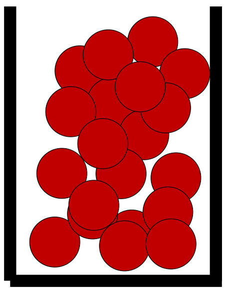
Roll the balls down the stairs.

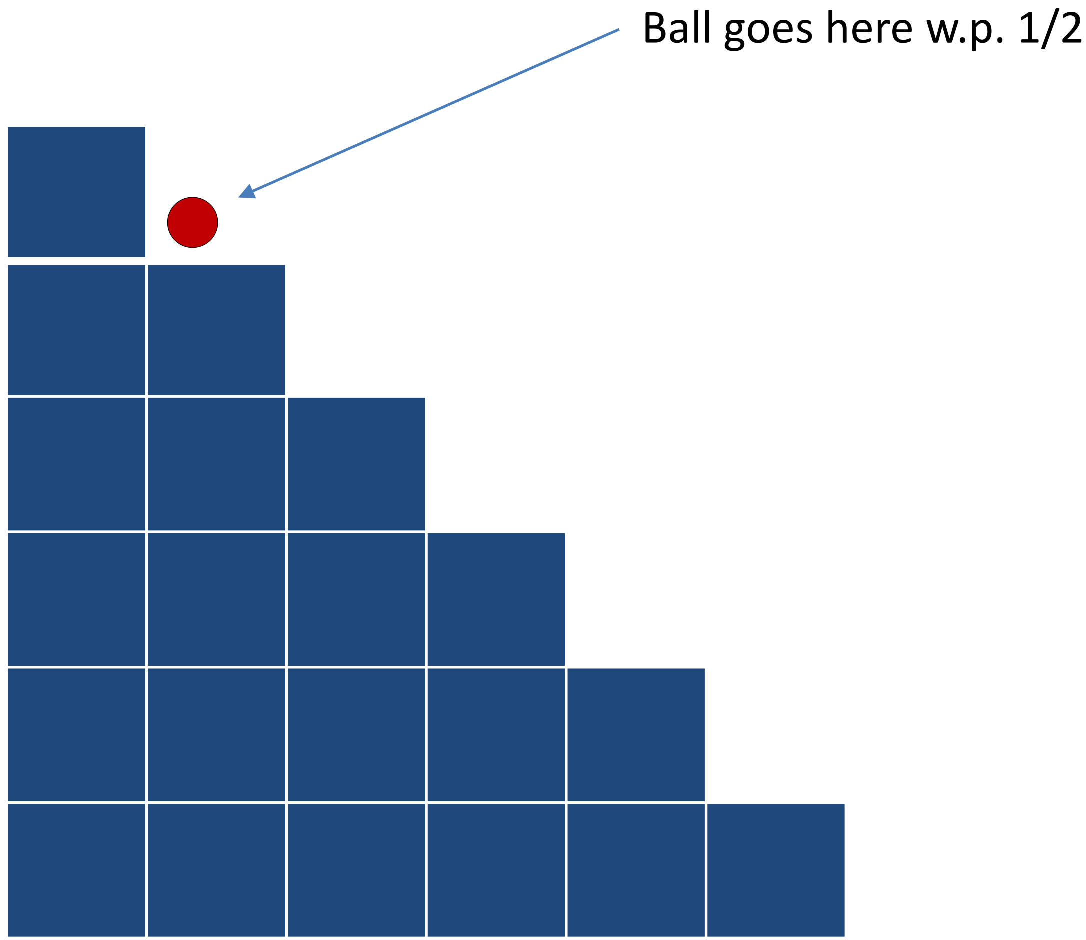
Each ball stops at each step with probability ½.



Challenge: Small Space

Trick 2: minimum of a set of random variables





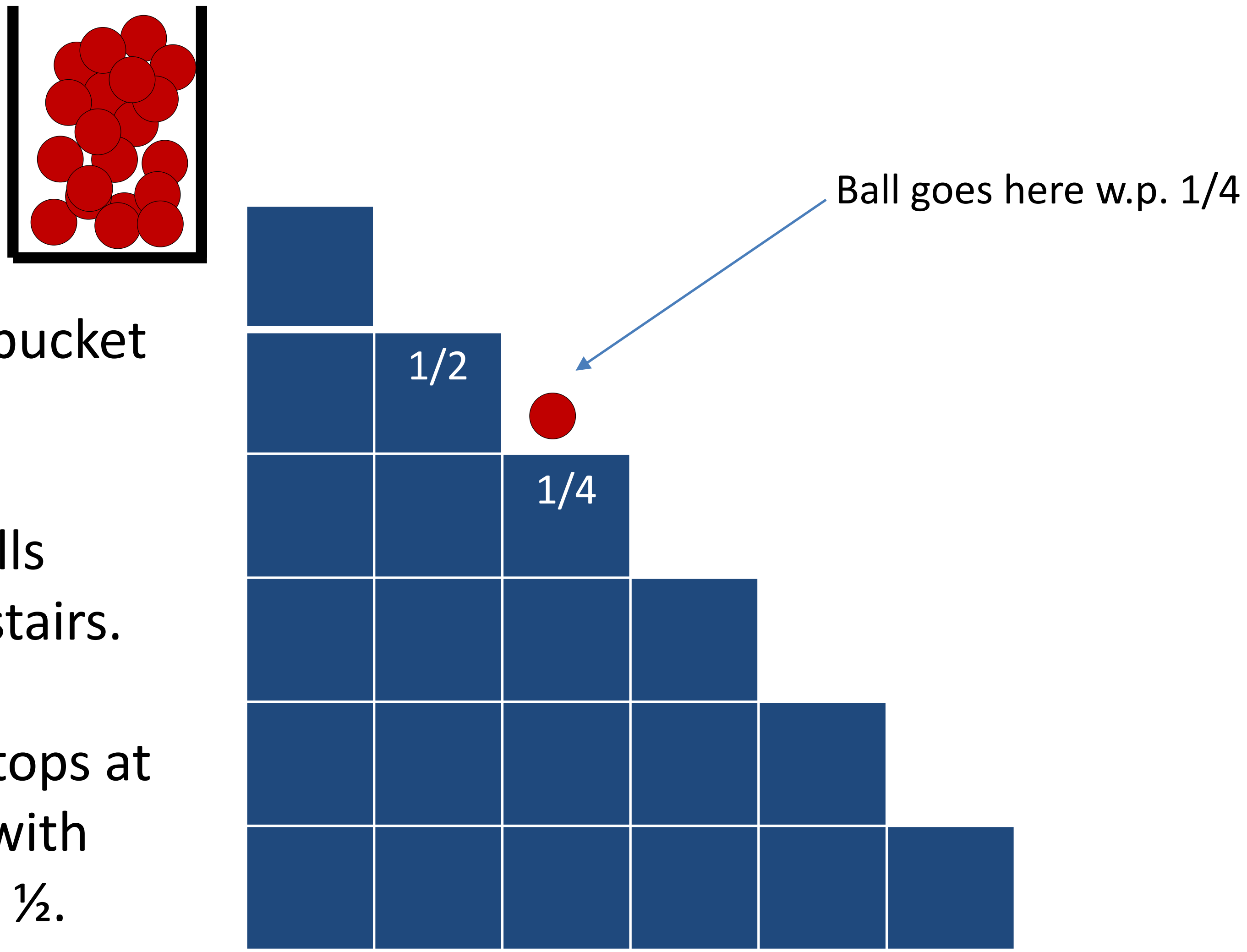
Imagine a bucket of balls.

Roll the balls down the stairs.

Each ball stops at each step with probability ½.

Challenge: Small Space

Trick 2: minimum of a set of random variables



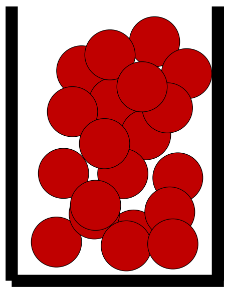
Imagine a bucket of balls.

Roll the balls down the stairs.

Each ball stops at each step with probability ½.

Challenge: Small Space

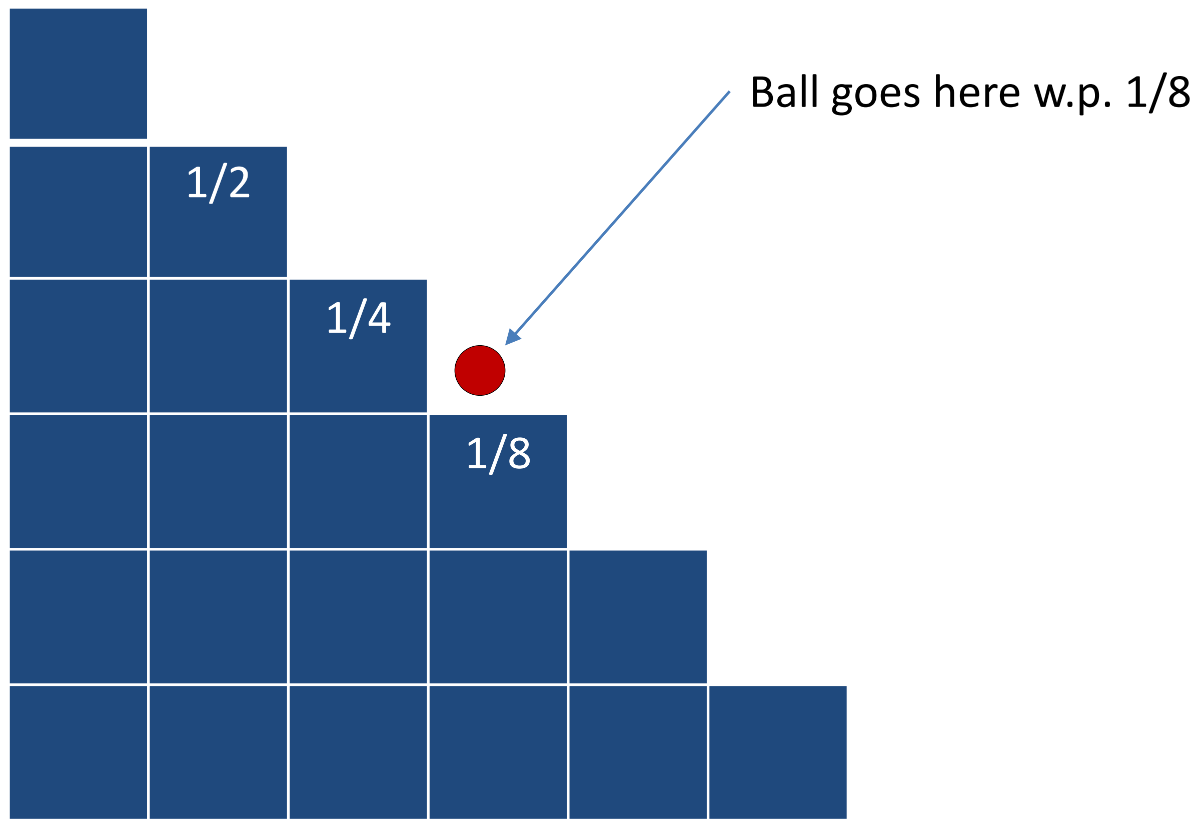
Trick 2: minimum of a set of random variables



Imagine a bucket of balls.

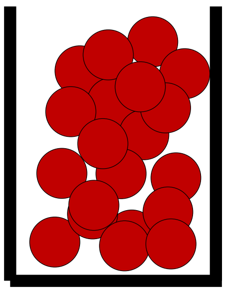
Roll the balls down the stairs.

Each ball stops at each step with probability ½.



Challenge: Small Space

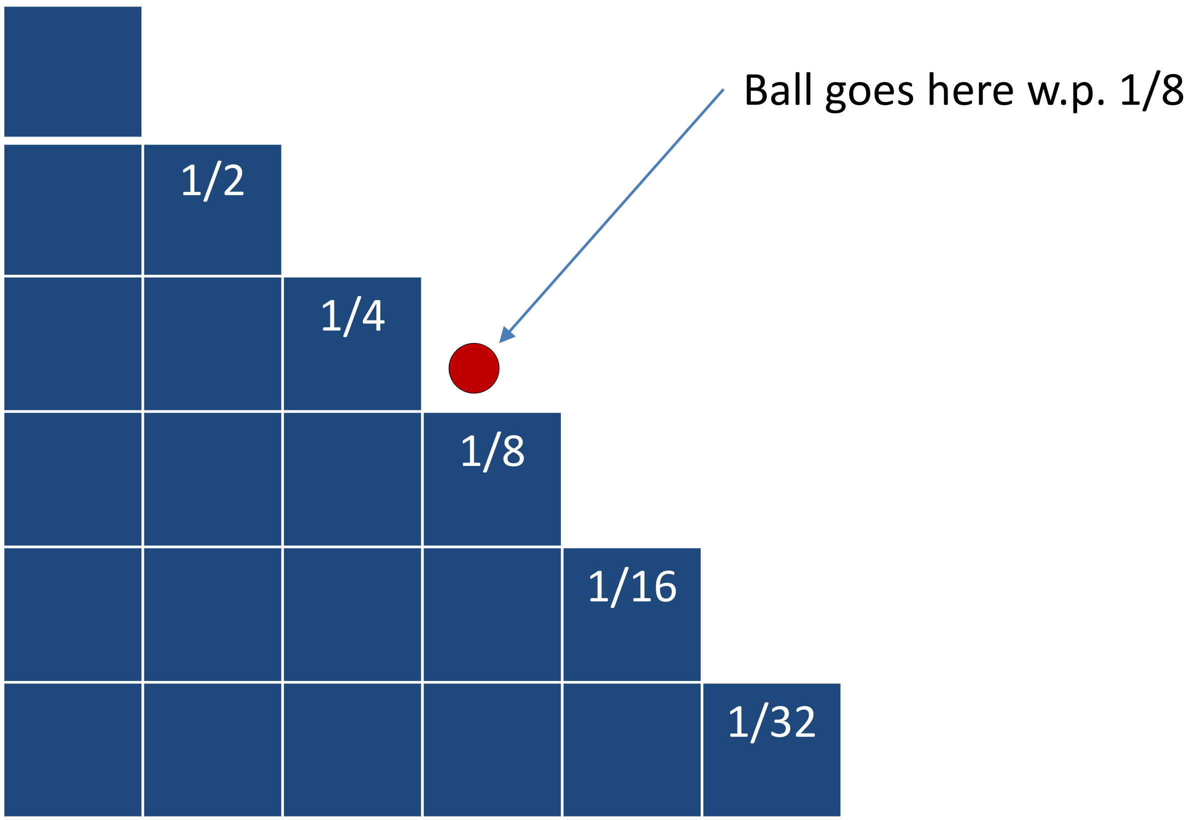
Trick 2: minimum of a set of random variables



Imagine a bucket of balls.

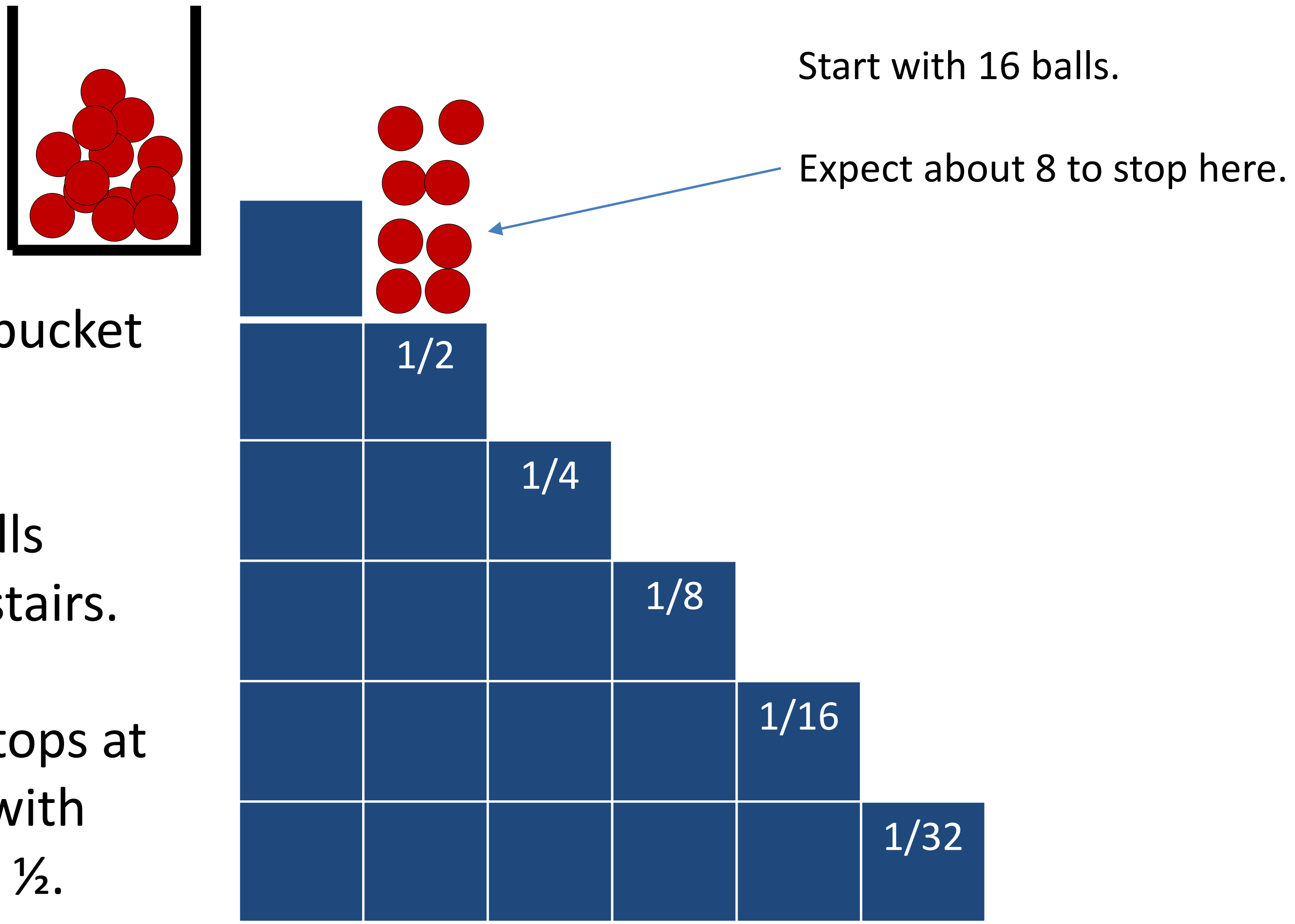
Roll the balls down the stairs.

Each ball stops at each step with probability ½.



Challenge: Small Space

Trick 2: minimum of a set of random variables



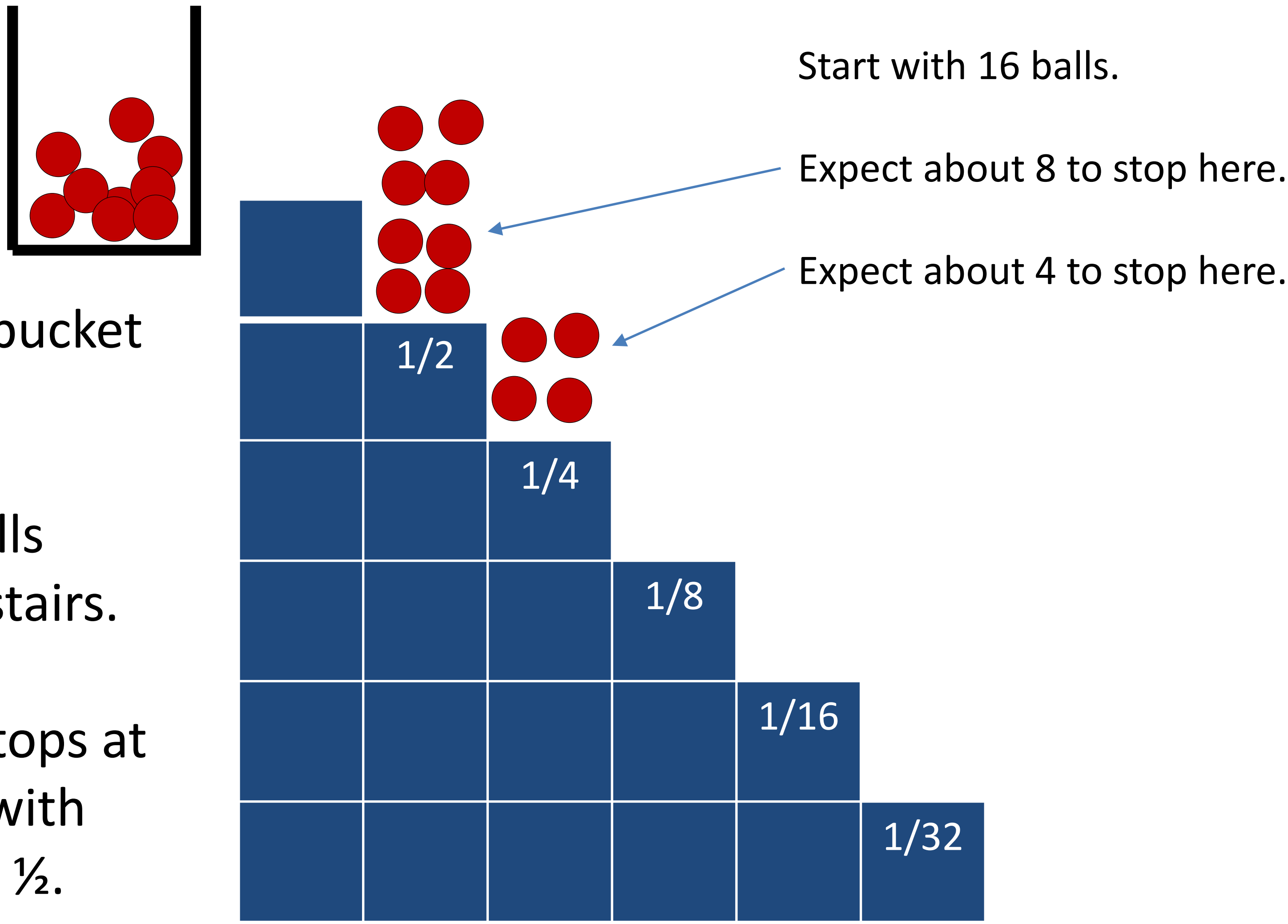
Imagine a bucket of balls.

Roll the balls down the stairs.

Each ball stops at each step with probability ½.

Challenge: Small Space

Trick 2: minimum of a set of random variables



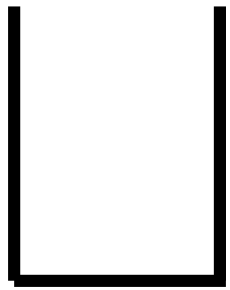
Imagine a bucket of balls.

Roll the balls down the stairs.

Each ball stops at each step with probability ½.

Challenge: Small Space

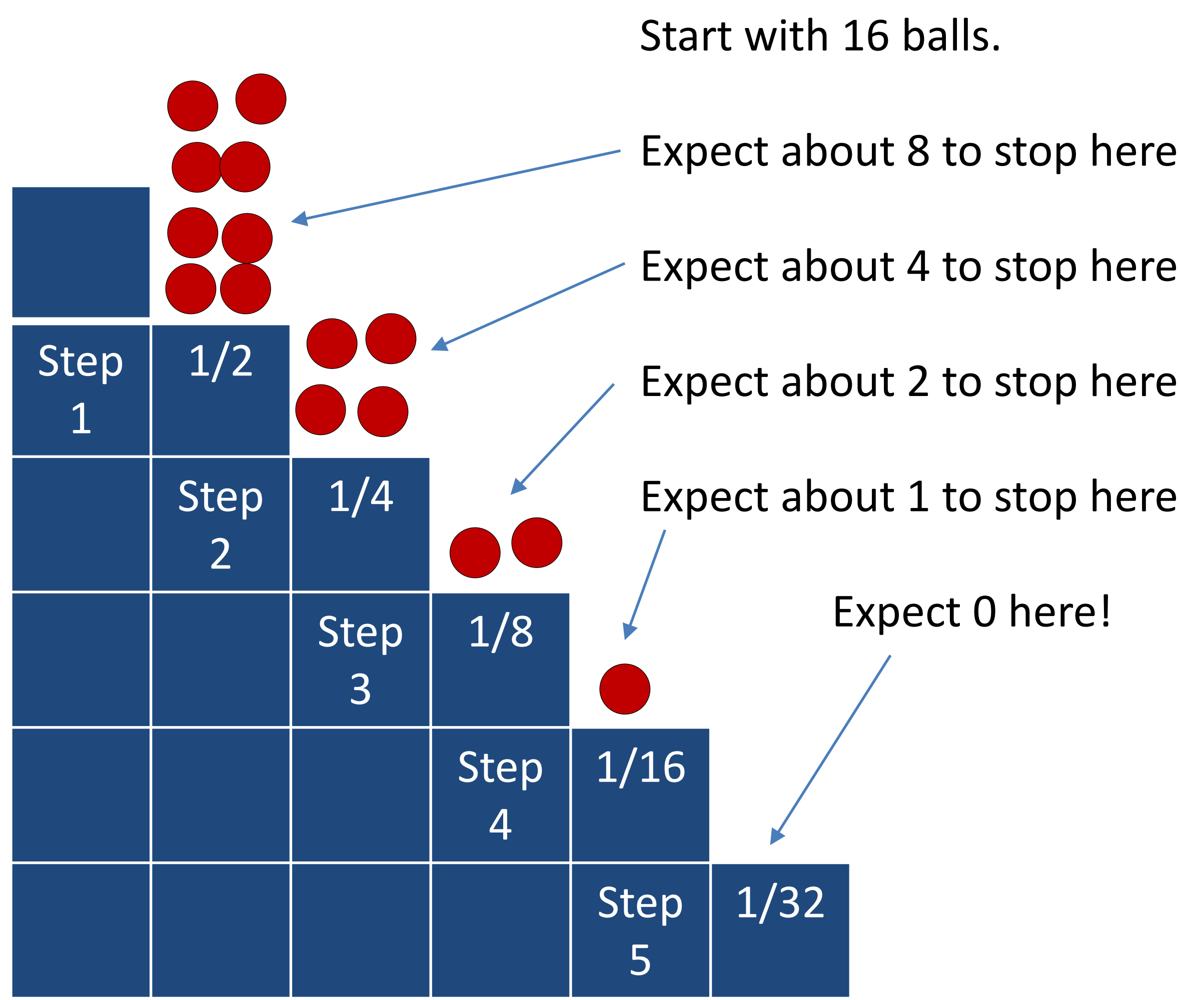
Trick 2: minimum of a set of random variables



Imagine a bucket of balls.

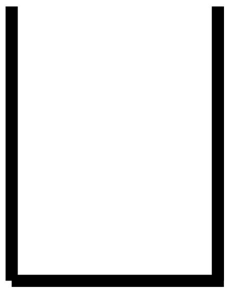
Roll the balls down the stairs.

Each ball stops at each step with probability ½.

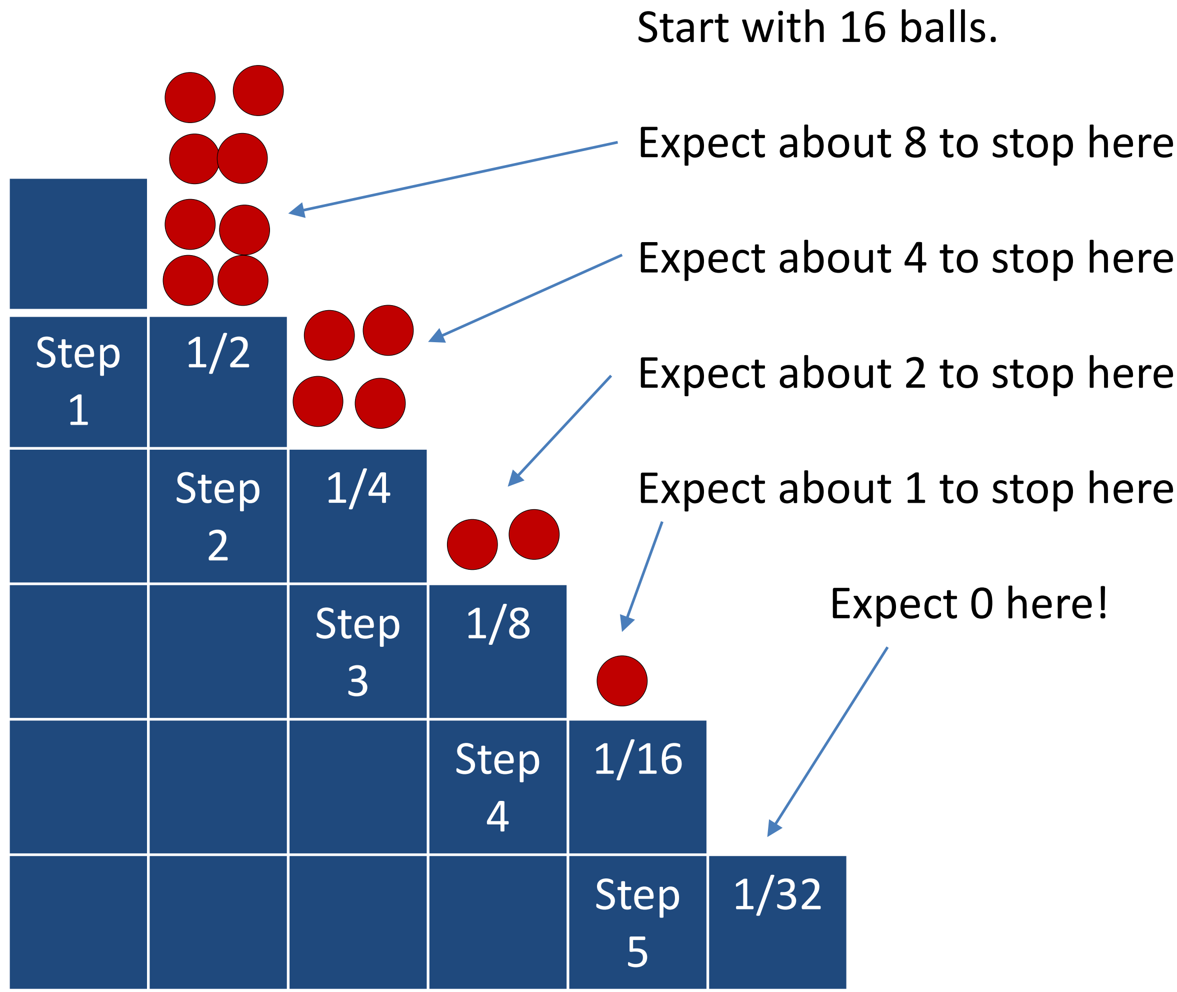


Challenge: Small Space

Trick 2: minimum of a set of random variables

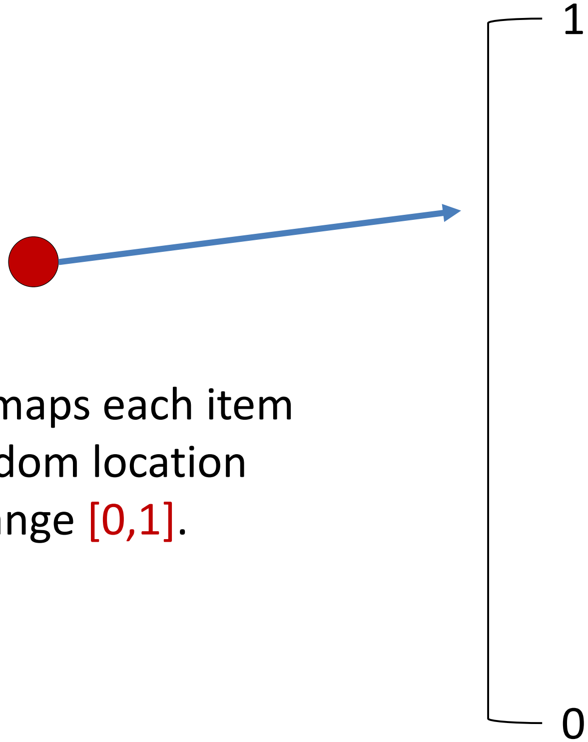


If last bucket containing at least one ball is step j, estimate that there are 2^j balls in total!



Challenge: Small Space

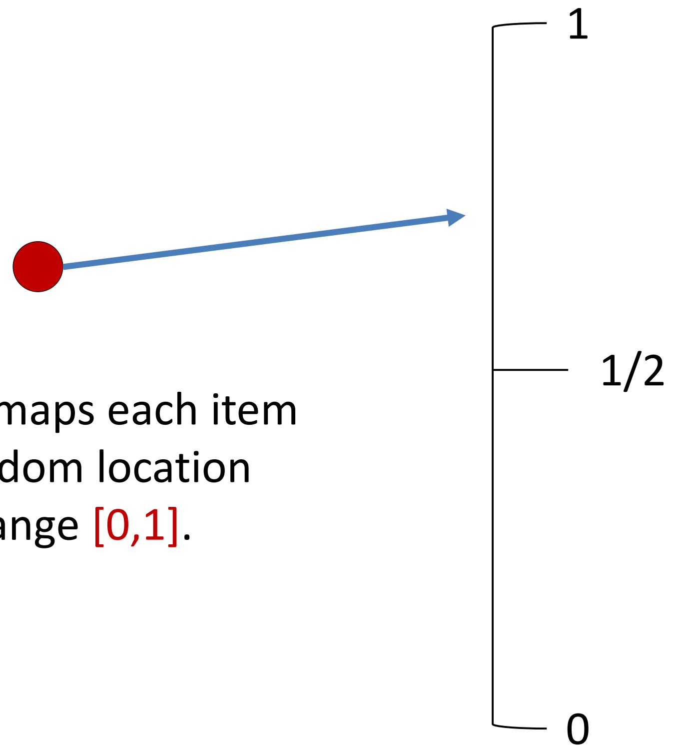
Trick 2: minimum of a set of random variables



Hash h maps each item to a random location in the range [0,1].

Challenge: Small Space

Trick 2: minimum of a set of random variables

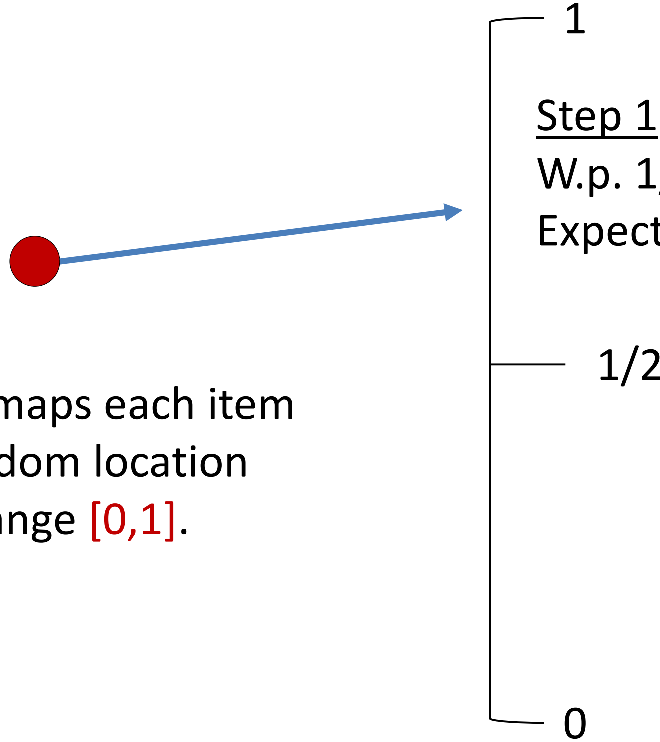


What fraction are mapped to the range [1/2, 1]?

Hash h maps each item to a random location in the range [0,1].

Challenge: Small Space

Trick 2: minimum of a set of random variables



<u>Step 1</u>

W.p. 1/2, h(x) ∈ [1/2, 1]

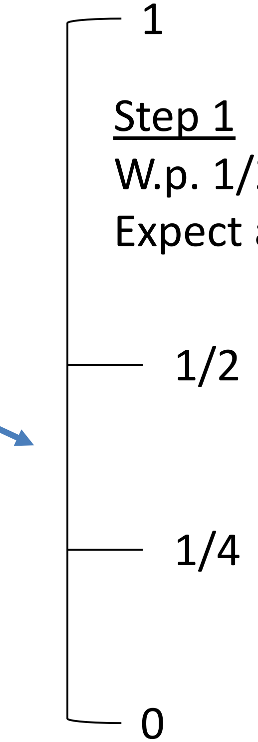
Expect about half the items to map here.

Hash h maps each item to a random location in the range [0,1].

Challenge: Small Space

Trick 2: minimum of a set of random variables

Hash h maps each item to a random location in the range [0,1].

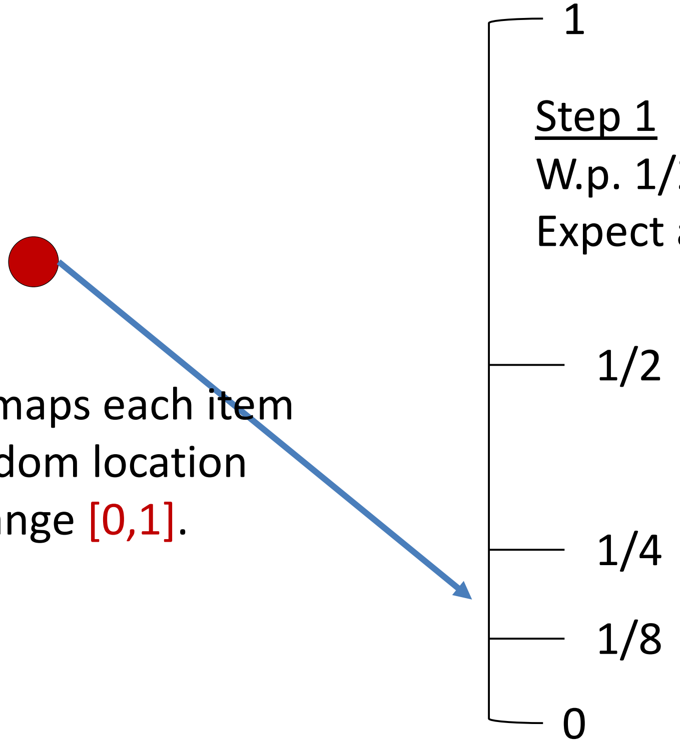


<u>Step 1</u>
W.p. 1/2, h(x) ∊ [1/2, 1]
Expect about 1/2 the items to map here.

<u>Step 2</u>
W.p. 1/4, h(x) ∊ [1/4, 1/2]
Expect about 1/4 the items to map here.

Challenge: Small Space

Trick 2: minimum of a set of random variables



Step 1
W.p. 1/2, h(x) ∊ [1/2, 1]
Expect about 1/2 the items to map here.

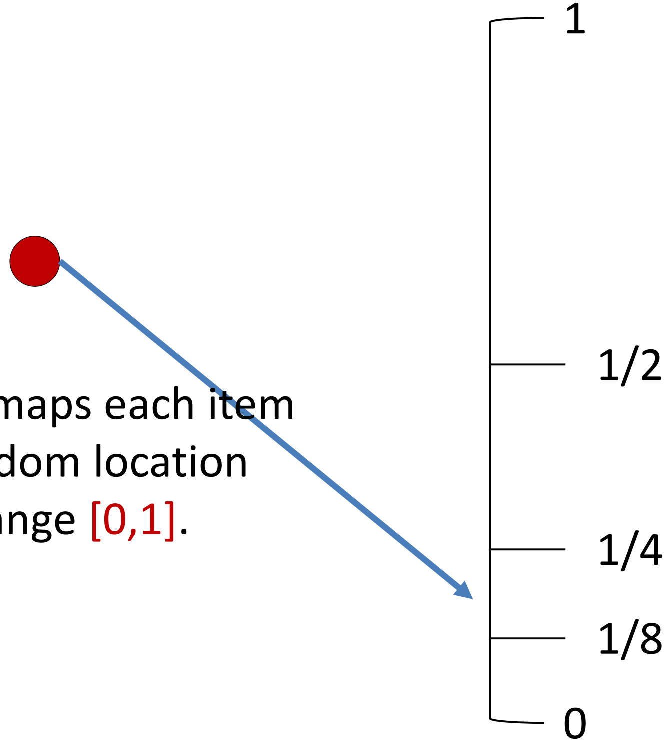
Hash h maps each item
to a random location
in the range [0,1].

Step 2
W.p. 1/4, h(x) ∊ [1/4, 1/2]
Expect about 1/4 the items to map here.

Step 3
W.p. 1/8, h(x) ∊ [1/8, 1/4]
Expect about 1/8 the items to map here.

Challenge: Small Space

Trick 2: minimum of a set of random variables



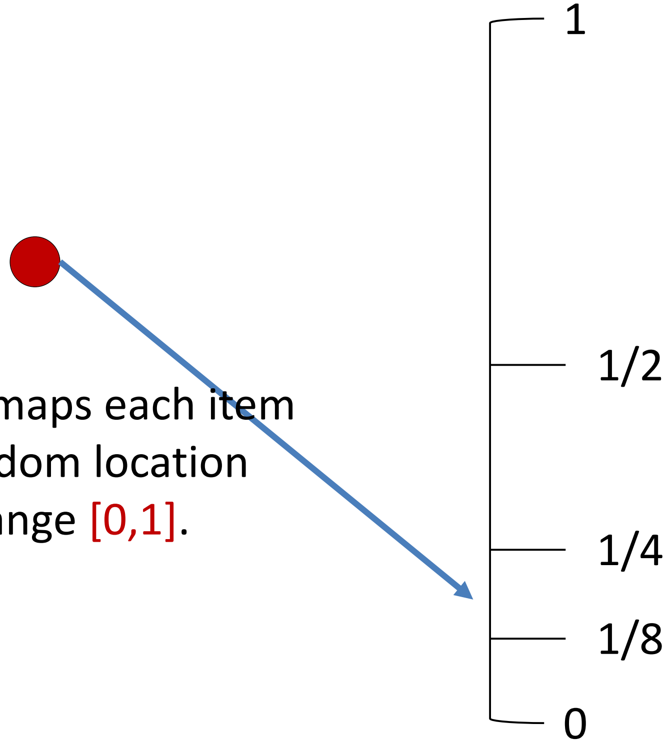
Hash h maps each item to a random location in the range [0,1].

<u>In general</u>

If there are n distinct items, we expect $n/2^j$ items to map to the region $[1/2^j, 1/2^{j-1}]$.

Challenge: Small Space

Trick 2: minimum of a set of random variables



Hash h maps each item to a random location in the range [0,1].

<u>In general</u>

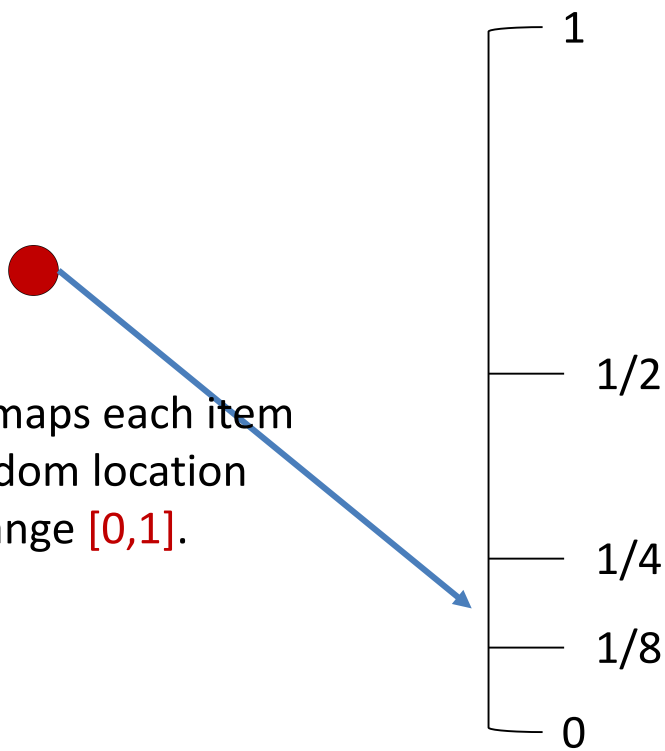
If there are n distinct items, we expect $n/2^j$ items to map to the region $[1/2^j, 1/2^{j-1}]$.

<u>Idea</u>

If $[1/2^j, 1/2^{j-1}]$ is the smallest range containing at least one item, then return 2^j.

Challenge: Small Space

Trick 2: minimum of a set of random variables



Hash h maps each item to a random location in the range [0,1].

<u>In general</u>

If there are n distinct items, we expect $n/2^j$ items to map to the region $[1/2^j, 1/2^{j-1}]$.

<u>Idea</u>

If $[1/2^j, 1/2^{j-1}]$ is the smallest range containing at least one item, then return 2^j.

<u>Simpler Idea</u>

If x is the minimum hash value, return $1/x$.

Flajolet-Martin (FM) Algorithm

Let $x = 1$.

For each u in stream S:

    if $h(u) < x$ then $x = h(u)$

Return $1/x - 1$.

Flajolet-Martin (FM) Algorithm

Analysis: $\mathrm{E}[x]$

Flajolet-Martin (FM) Algorithm

Assume items $s_1, s_2, \ldots,$
Assume t distinct items.

Analysis: $$\mathrm{E}[x] = \int_0^1 \mathrm{P{\scriptsize R}}[x \geq \lambda]\, \mathrm{d}\lambda$$

Flajolet-Martin (FM) Algorithm

Assume items $s_1, s_2, \ldots,$
Assume t distinct items.

Analysis: $$\mathrm{E}[x] = \int_0^1 \mathrm{P\scriptsize{R}}[x \geq \lambda]\ \mathrm{d}\lambda$$

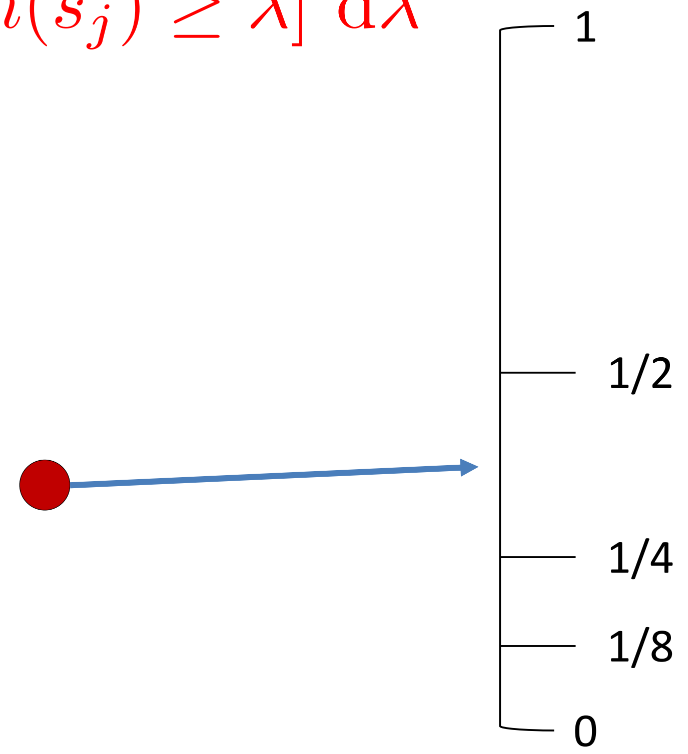
<u>Discrete definition of expectation</u>

$$\begin{aligned}
\mathrm{E}[X] &= \sum_{z=1}^{\infty} z \Pr[X = z] \\
&= \sum_{z=1}^{\infty} \sum_{j=1}^{z} \Pr[X = z] \\
&= \sum_{j=1}^{\infty} \sum_{z=j}^{\infty} \Pr[X = z] \\
&= \sum_{j=1}^{\infty} \Pr[X \geq j]
\end{aligned}$$

Flajolet-Martin (FM) Algorithm

Assume items $s_1, s_2, \ldots,$
Assume t distinct items.

Analysis:
$$\mathrm{E}[x] = \int_0^1 \Pr[x \geq \lambda]\, \mathrm{d}\lambda$$
$$= \int_0^1 \Pr[\forall j \;:\; h(s_j) \geq \lambda]\, \mathrm{d}\lambda$$

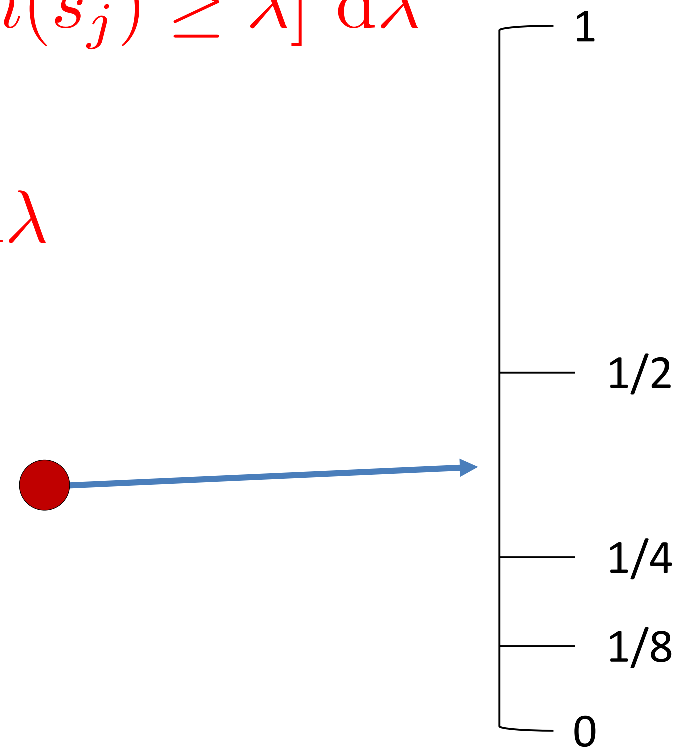


Flajolet-Martin (FM) Algorithm

Assume items s_1, s_2, …,
Assume t distinct items.

Analysis:

$$
\begin{aligned}
\mathrm{E}[x] &= \int_0^1 \textsc{Pr}[x \geq \lambda]\ \mathrm{d}\lambda \\
&= \int_0^1 \textsc{Pr}[\forall j\ :\ h(s_j) \geq \lambda]\ \mathrm{d}\lambda \\
&= \int_0^1 (1-\lambda)^t\ \mathrm{d}\lambda
\end{aligned}
$$



Note key assumption:
Each item is hashed independently!

Flajolet-Martin (FM) Algorithm

Assume items s_1, s_2, …,
Assume t distinct items.

Analysis:
$$
\begin{aligned}
\mathrm{E}[x] &= \int_0^1 \mathrm{P}\textsc{r}[x \geq \lambda]\ \mathrm{d}\lambda \\
&= \int_0^1 \mathrm{P}\textsc{r}[\forall j\ :\ h(s_j) \geq \lambda]\ \mathrm{d}\lambda \\
&= \int_0^1 (1-\lambda)^t\ \mathrm{d}\lambda \\
&= \left.\frac{-(1-\lambda)^{t+1}}{t+1}\right|_0^1
\end{aligned}
$$

Flajolet-Martin (FM) Algorithm

Assume items $s_1, s_2, \ldots,$
Assume t distinct items.

Analysis:

$$
\begin{aligned}
\mathrm{E}[x] &= \int_0^1 \Pr[x \geq \lambda]\, \mathrm{d}\lambda \\
&= \int_0^1 \Pr[\forall j \;:\; h(s_j) \geq \lambda]\, \mathrm{d}\lambda \\
&= \int_0^1 (1-\lambda)^t \; \mathrm{d}\lambda \\
&= \left. \frac{-(1-\lambda)^{t+1}}{t+1} \right|_0^1 \\
&= \frac{1}{t+1}
\end{aligned}
$$

Flajolet-Martin (FM) Algorithm

Assume items s_1, s_2, …,
Assume t distinct items.

Conclusion: $$\mathrm{E}[X] = \frac{1}{t+1}$$

Flajolet-Martin (FM) Algorithm

Assume items s_1, s_2, …,
Assume t distinct items.

Conclusion: $$\mathrm{E}[X] = \frac{1}{t+1}$$

$$\begin{aligned}
\mathrm{E}[FM] &= \mathrm{E}[1/X - 1] \\
&= \mathrm{E}[1/X] - 1 \\
&= \frac{1}{\mathrm{E}[X]} - 1 \\
&= \frac{1}{1/(t+1)} - 1 \\
&= t + 1 - 1 \\
&= t
\end{aligned}$$

Flajolet-Martin (FM) Algorithm

Assume items $s_1, s_2, \ldots,$
Assume t distinct items.

Conclusion: $$\mathrm{E}[X] = \frac{1}{t+1}$$

$$\begin{aligned}
\mathrm{E}[FM] &= \mathrm{E}[1/X - 1] \\
&= \mathrm{E}[1/X] - 1 \\
&= \frac{1}{\mathrm{E}[X]} - 1 \\
&= \frac{1}{1/(t+1)} - 1 \\
&= t + 1 - 1 \\
&= t
\end{aligned}$$

Cannot invert expected values!

$\mathrm{E}[1/X] \neq 1/\mathrm{E}[X]$

Flajolet-Martin (FM) Algorithm

Assume items s_1, s_2, …,
Assume t distinct items.

Variance: $\textsc{Var}[x] = \mathrm{E}[x^2] - \mathrm{E}[x]^2$

$$\mathrm{E}[x]^2 = \frac{1}{(t+1)^2}$$

Flajolet-Martin (FM) Algorithm

Assume items $s_1, s_2, \ldots,$
Assume t distinct items.

Variance: $$\mathrm{E}[x^2] = \int_0^1 \Pr[x^2 \geq \lambda]\, \mathrm{d}\lambda$$

Flajolet-Martin (FM) Algorithm

Assume items s_1, s_2, …,
Assume t distinct items.

Variance:

$$\mathrm{E}[x^2] = \int_0^1 \mathrm{Pr}[x^2 \geq \lambda]\,\mathrm{d}\lambda$$

$$= \int_0^1 \mathrm{Pr}[x \geq \sqrt{\lambda}]\,\mathrm{d}\lambda$$

Flajolet-Martin (FM) Algorithm

Assume items $s_1, s_2, \ldots,$
Assume t distinct items.

Variance: $$\mathrm{E}[x^2] = \int_0^1 \textsc{Pr}[x^2 \geq \lambda]\, \mathrm{d}\lambda$$
$$= \int_0^1 \textsc{Pr}[x \geq \sqrt{\lambda}]\, \mathrm{d}\lambda$$
$$= \int_0^1 \textsc{Pr}[\forall j \ : \ h(s_j) \geq \sqrt{\lambda}]\, \mathrm{d}\lambda$$

Flajolet-Martin (FM) Algorithm

Assume items $s_1, s_2, \ldots,$
Assume t distinct items.

Variance:

$$\begin{aligned}
\mathrm{E}[x^2] &= \int_0^1 \Pr[x^2 \geq \lambda]\, \mathrm{d}\lambda \\
&= \int_0^1 \Pr[x \geq \sqrt{\lambda}]\, \mathrm{d}\lambda \\
&= \int_0^1 \Pr[\forall j \;:\; h(s_j) \geq \sqrt{\lambda}]\, \mathrm{d}\lambda \\
&= \int_0^1 \left(1 - \sqrt{\lambda}\right)^t \mathrm{d}\lambda
\end{aligned}$$

Flajolet-Martin (FM) Algorithm

Assume items $s_1, s_2, \ldots,$
Assume t distinct items.

Variance:

$$
\begin{aligned}
\mathrm{E}[x^2] &= \int_0^1 \textsc{Pr}[x^2 \geq \lambda]\ \mathrm{d}\lambda \\
&= \int_0^1 \textsc{Pr}[x \geq \sqrt{\lambda}]\ \mathrm{d}\lambda \\
&= \int_0^1 \textsc{Pr}[\forall j\ :\ h(s_j) \geq \sqrt{\lambda}]\ \mathrm{d}\lambda \\
&= \int_0^1 \left(1 - \sqrt{\lambda}\right)^t\ \mathrm{d}\lambda \\
&= \int_1^0 u^t \left(-2(1-u)\right)\ \mathrm{d}u
\end{aligned}
$$

$$
\begin{aligned}
u &= 1 - \sqrt{\lambda} \\
\lambda &= (1-u)^2
\end{aligned}
$$

$$\mathrm{d}\lambda = -2(1-u)\mathrm{d}u$$

Flajolet-Martin (FM) Algorithm

Assume items s_1, s_2, …,
Assume t distinct items.

Variance:
$$\mathrm{E}[x^2] = \int_0^1 \mathrm{P\scriptsize R}[x^2 \geq \lambda]\, \mathrm{d}\lambda$$

$$= \int_1^0 u^t \left(-2(1-u)\right)\ \mathrm{d}u$$

$$= 2\int_0^1 u^t\ \mathrm{d}u - 2\int_0^1 u^{t+1}\ \mathrm{d}u$$

Flajolet-Martin (FM) Algorithm

Assume items s_1, s_2, …,
Assume t distinct items.

Variance: $$\mathrm{E}[x^2] = \int_0^1 \mathrm{P{\scriptstyle R}}[x^2 \geq \lambda]\ \mathrm{d}\lambda$$

$$= \int_1^0 u^t\,(-2(1-u))\ \mathrm{d}u$$

$$= 2\int_0^1 u^t\ \mathrm{d}u - 2\int_0^1 u^{t+1}\ \mathrm{d}u$$

$$= 2\frac{u^{t+1}}{t+1}\ \Big|_0^1 - 2\frac{u^{t+2}}{t+2}\ \Big|_0^1$$

Flajolet-Martin (FM) Algorithm

Assume items $s_1, s_2, \ldots,$
Assume t distinct items.

Variance: $$\mathrm{E}[x^2] = \int_0^1 \Pr[x^2 \geq \lambda]\, \mathrm{d}\lambda$$

$$= \int_1^0 u^t \left(-2(1-u)\right)\, \mathrm{d}u$$

$$= 2\int_0^1 u^t\, \mathrm{d}u - 2\int_0^1 u^{t+1}\, \mathrm{d}u$$

$$= 2\frac{u^{t+1}}{t+1}\Big|_0^1 - 2\frac{u^{t+2}}{t+2}\Big|_0^1$$

$$= \frac{2}{t+1} - \frac{2}{t+2}$$

Flajolet-Martin (FM) Algorithm

Assume items s_1, s_2, …,
Assume t distinct items.

Variance: $\textsc{Var}[x] = \mathrm{E}[x^2] - \mathrm{E}[x]^2$

$$
\begin{aligned}
\textsc{Var}[x] &= \frac{2}{t+1} - \frac{2}{t+2} - \frac{1}{(t+1)^2} \\
&= \frac{2}{(t+1)(t+2)} - \frac{1}{(t+1)^2} \\
&\leq \frac{1}{(t+1)^2}
\end{aligned}
$$

Flajolet-Martin+ (FM+) Algorithm

1. Run a copies of FM: get $X_1, X_2, \ldots, X_a$
2. Compute average: $Z = \frac{1}{a}\sum_{j=1}^{a} X_j$
3. Return $(1/Z)-1$.

Flajolet-Martin (FM) Algorithm

Analysis:

$$\begin{aligned}
\mathrm{E}[Z] &= \frac{1}{a}\mathrm{E}\left[\sum_{j=1}^{a} X_j\right] \\
&= \frac{1}{a}\sum_{j=1}^{a}\mathrm{E}[X_j] \\
&= \frac{1}{a}a\frac{1}{t+1} \\
&= \frac{1}{t+1}
\end{aligned}$$

Flajolet-Martin (FM) Algorithm

Analysis: $$\textsc{Var}[Z] = \frac{1}{a^2}\sum_{j=1}^{a}\textsc{Var}[X_j]$$
$$= \frac{1}{a^2}a\frac{t}{(t+1)^2(t+2)}$$
$$\leq \frac{1}{a(t+1)^2}$$

Flajolet-Martin (FM) Algorithm

Chebychev's Inequality:

Let Y be a random variable.

$$\Pr\left[|Y - \mathrm{E}[Y]| \geq t\right] \leq \frac{\textsc{Var}[Y]}{t^2}$$

Note:

- More general than Chernoff: holds for all Y.
- Weaker than Chernoff: less tight bound.

Flajolet-Martin (FM) Algorithm

Analysis:

$$\Pr\left[\left|Z - \frac{1}{t+1}\right| \geq \epsilon\left(\frac{1}{t+1}\right)\right] \leq \textsc{Var}[Z]\frac{(t+1)^2}{\epsilon^2}$$

Flajolet-Martin (FM) Algorithm

Analysis:

$$\Pr\left[\left|Z - \frac{1}{t+1}\right| \geq \epsilon\left(\frac{1}{t+1}\right)\right] \leq \textsc{Var}[Z]\frac{(t+1)^2}{\epsilon^2}$$

$$\leq \frac{1}{a(t+1)^2}\frac{(t+1)^2}{\epsilon^2}$$

Flajolet-Martin (FM) Algorithm

Analysis:

$$\Pr\left[\left|Z - \frac{1}{t+1}\right| \geq \epsilon\left(\frac{1}{t+1}\right)\right] \leq \textsc{Var}[Z]\frac{(t+1)^2}{\epsilon^2}$$
$$\leq \frac{1}{a(t+1)^2}\frac{(t+1)^2}{\epsilon^2}$$
$$\leq \frac{1}{a\epsilon^2}$$

Flajolet-Martin (FM) Algorithm

Analysis:

$$\Pr\left[\left|Z - \frac{1}{t+1}\right| \geq \epsilon\left(\frac{1}{t+1}\right)\right] \leq \textsc{Var}[Z]\frac{(t+1)^2}{\epsilon^2}$$

$$\leq \frac{1}{a(t+1)^2}\frac{(t+1)^2}{\epsilon^2}$$

$$\leq \frac{1}{a\epsilon^2}$$

$$\boxed{a = \frac{4}{\epsilon^2}}$$

Flajolet-Martin (FM) Algorithm

Analysis:

$$\Pr\left[\left|Z - \frac{1}{t+1}\right| \geq \epsilon\left(\frac{1}{t+1}\right)\right] \leq \textsc{Var}[Z]\frac{(t+1)^2}{\epsilon^2}$$

$$\leq \frac{1}{a(t+1)^2}\frac{(t+1)^2}{\epsilon^2}$$

$$\leq \frac{1}{a\epsilon^2}$$

$$\boxed{a = \frac{4}{\epsilon^2}}$$

$$\leq 1/4$$

Flajolet-Martin (FM) Algorithm

What have we shown?

$$\Pr\left[\left|Z - \frac{1}{t+1}\right| \geq \epsilon\left(\frac{1}{t+1}\right)\right] \leq 1/4$$

That is, w.p. at least 3/4:

1) $Z \geq \dfrac{1-\epsilon}{t+1}$

2) $Z \leq \dfrac{1+\epsilon}{t+1}$

Flajolet-Martin (FM) Algorithm

FM+ returns: $$\frac{1}{Z} - 1 \geq \frac{t+1}{1+\epsilon} - 1$$
$$\geq (t+1)(1-\epsilon) - 1$$
$$\geq t(1-2\epsilon)$$

Recall with probability at least 3/4:

$$Z \leq \frac{1+\epsilon}{t+1}$$

Recall: for $0 < x < 1/2$

$$\frac{1}{1-x} = 1 + x + x^2 + \ldots \leq 1 + 2x$$

$$\frac{1}{1+x} = 1 - x + x^2 - x^2 + \ldots \geq 1 - x$$

Flajolet-Martin (FM) Algorithm

FM+ returns: $$\frac{1}{Z} - 1 \le \frac{t+1}{1-\epsilon} - 1$$
$$\le (t+1)(1+2\epsilon) - 1$$
$$\le t(1+4\epsilon)$$

Recall with probability at least 3/4:

$$Z \ge \frac{1-\epsilon}{t+1}$$

Recall: for 0 < x < 1/2

$$\frac{1}{1-x} = 1 + x + x^2 + \ldots \le 1 + 2x$$

$$\frac{1}{1+x} = 1 - x + x^2 - x^2 + \ldots \ge 1 - x$$

Flajolet-Martin+ (FM+) Algorithm

1. Run a copies of FM: get $X_1, X_2, \ldots, X_a$
2. Compute average: $Z = \frac{1}{a}\sum_{j=1}^{a} X_j$
3. Return (1/Z)-1.

Not done yet…. Better than ¾ probability?

Flajolet-Martin++ (FM++) Algorithm

1. Run b copies of FM: get $Y_1, Y_2, \ldots, Y_b$

2. Return median($Y_1, Y_2, \ldots, Y_b$)

$$Y_j = (1/Z_j) - 1$$

Flajolet-Martin++ (FM++) Algorithm

1. Run b copies of FM: get $Y_1, Y_2, \ldots, Y_b$
2. Return median($Y_1, Y_2, \ldots, Y_b$)

Define random variables: $x_1, x_2, \ldots, x_b$

$x_j = 1$ if $|Y_j - t| \leq 4\varepsilon t$

$x_j = 0$ otherwise

Note: x_j are independent, 0/1 random variables!

Flajolet-Martin++ (FM++) Algorithm

1. Run b copies of FM: get $Y_1, Y_2, \ldots, Y_b$
2. Return median$(Y_1, Y_2, \ldots, Y_b)$

Define random variables: $x_1, x_2, \ldots, x_b$

$x_j = 1$ if $|Y_j - t| \leq 4\varepsilon t$ ← $\Pr[x_j = 1] = ??$

$x_j = 0$ otherwise

Flajolet-Martin++ (FM++) Algorithm

1. Run b copies of FM: get $Y_1, Y_2, \ldots, Y_b$

2. Return median($Y_1, Y_2, \ldots, Y_b$)

Define random variables: $x_1, x_2, \ldots, x_b$

$x_j = 1$ if $|Y_j - t| \leq 4\varepsilon t$ ← $\Pr[x_j = 1] \geq 3/4$

$x_j = 0$ otherwise

Flajolet-Martin++ (FM++) Algorithm

1. Run b copies of FM: get $Y_1, Y_2, \ldots, Y_b$

2. Return median($Y_1, Y_2, \ldots, Y_b$)

Define random variables: $x_1, x_2, \ldots, x_b$

$x_j = 1$ if $|Y_j - t| \leq 4\varepsilon t$ ← $\Pr[x_j = 1] \geq 3/4$

$x_j = 0$ otherwise

$$\mathrm{E}[x_j] \geq 3/4$$

Flajolet-Martin++ (FM++) Algorithm

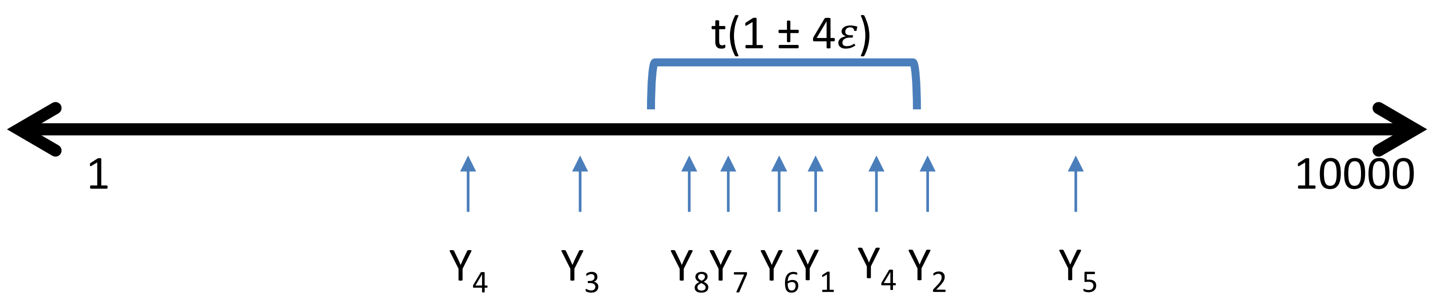
1. Run b copies of FM: get $Y_1, Y_2, \ldots, Y_b$
2. Return median($Y_1, Y_2, \ldots, Y_b$)

When is the median the “right” answer?

Flajolet-Martin++ (FM++) Algorithm

1. Run b copies of FM: get $Y_1, Y_2, \ldots, Y_b$
2. Return median($Y_1, Y_2, \ldots, Y_b$)

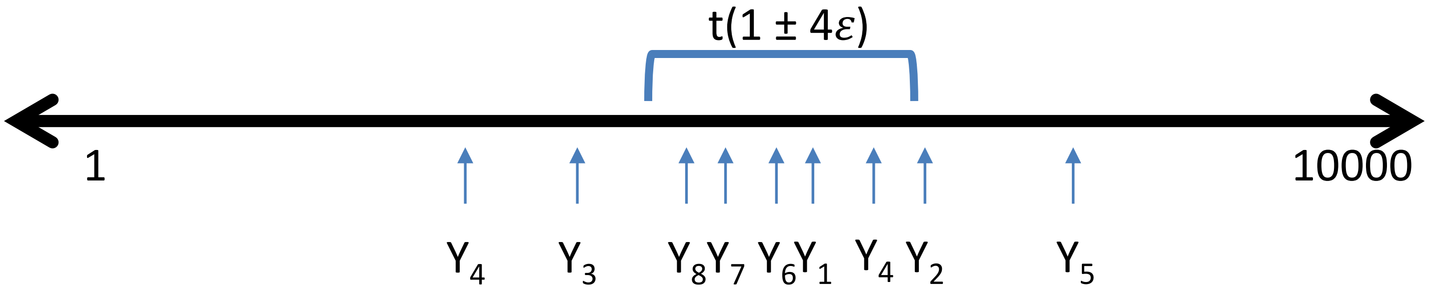
When is the median the “right” answer?



Flajolet-Martin++ (FM++) Algorithm

1. Run b copies of FM: get $Y_1, Y_2, \ldots, Y_b$
2. Return median($Y_1, Y_2, \ldots, Y_b$)

When is the median the "right" answer?

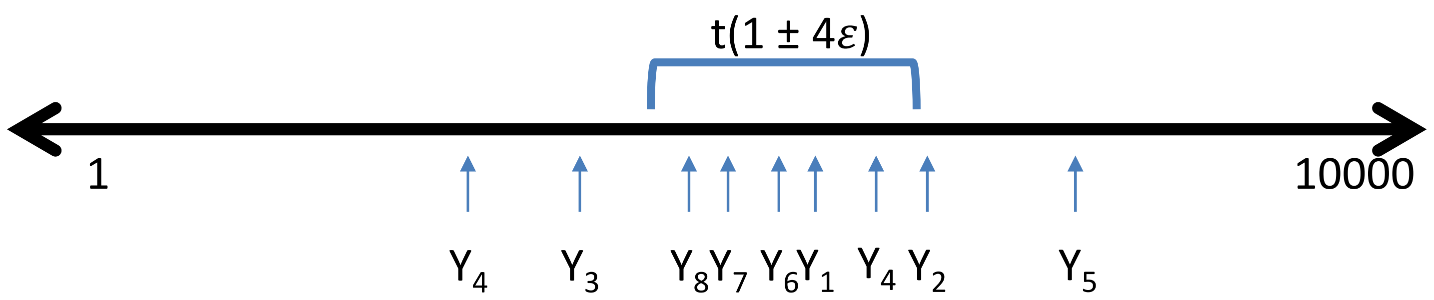


Claim: If > 1/2 of the Y_j's are "good", then the median is good.

Flajolet-Martin++ (FM++) Algorithm

1. Run b copies of FM: get $Y_1, Y_2, \ldots, Y_b$
2. Return median($Y_1, Y_2, \ldots, Y_b$)

When is the median the "right" answer?



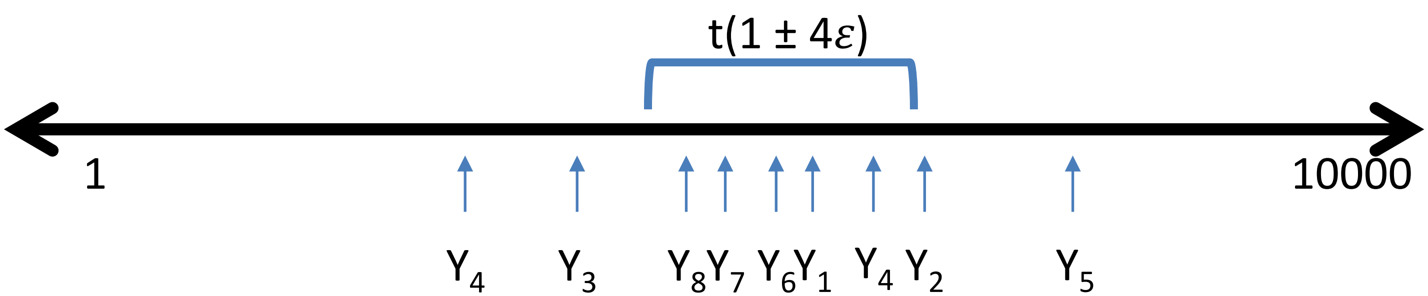
Claim: If $\sum_{i=1}^{b} x_i \geq b/2$ then median is "right."

Flajolet-Martin++ (FM++) Algorithm

Chernoff Bound:

$$\Pr\left[\sum_{i=1}^{b} x_j < (1-\rho)\mu\right] \leq e^{-\mu\rho^2/3}$$

When is the median the "right" answer?



Claim: If $\sum_{i=1}^{b} x_i \geq b/2$ then median is "right."

Flajolet-Martin++ (FM++) Algorithm

Chernoff Bound:

$$\Pr\left[\sum_{i=1}^{b} x_j < (1-\rho)\mu\right] \leq e^{-\mu\rho^2/3}$$

$$\Pr\left[\sum_{i=1}^{b} x_j < (1-1/3)(3b/4)\right] \leq e^{-(1/3)^2(3b/4)/3}$$

$$\leq e^{-b/36}$$

Flajolet-Martin++ (FM++) Algorithm

Chernoff Bound:

$$\Pr\left[\sum_{i=1}^{b} x_j < (1-\rho)\mu\right] \leq e^{-\mu\rho^2/3}$$

$$\Pr\left[\sum_{i=1}^{b} x_j < \underbrace{(1-1/3)(3b/4)}_{\text{b/2}}\right] \leq e^{-(1/3)^2(3b/4)/3}$$

$$\leq e^{-b/36}$$

Flajolet-Martin++ (FM++) Algorithm

Chernoff Bound:

$$\Pr\left[\sum_{i=1}^{b} x_j < (1-\rho)\mu\right] \leq e^{-\mu\rho^2/3}$$

$$\Pr\left[\sum_{i=1}^{b} x_j < (1-1/3)(3b/4)\right] \leq e^{-(1/3)^2(3b/4)/3}$$

$$\leq e^{-b/36} \leq \delta$$

b/2

$$\boxed{\text{Choose } b = 36\ln(2/\delta)}$$

Flajolet-Martin++ (FM++) Algorithm

1. Run b copies of FM: get $Y_1, Y_2, \ldots, Y_b$
2. Return median($Y_1, Y_2, \ldots, Y_b$)

Conclusion:

With probability at least: $1 - \delta$

the FM++ algorithm returns an answer in the range:

$$t(1 \pm 4\epsilon)$$

Summary

Today: Data

Counting distinct elements:

- How many items in the stream?

Item frequencies:

- How often does an item appear in a stream?

Heavy hitters:

- Identify the most frequent items

Statistics

- Average, median, etc.

Next Weeks: Graphs

Connectivity:

- Is the graph connected?

MST:

- Find an MST

Matching:

- Approximate the maximal matching.

Shortest paths:

- Approximate the shortest paths in a graph.

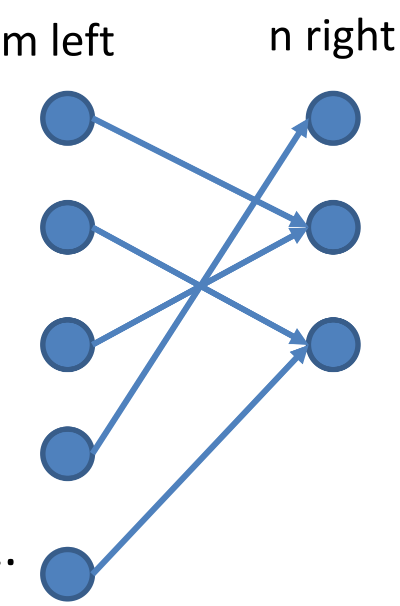
Triangles:

- How many triangles in a graph?

Algorithms at Scale

(Week 4)

Puzzle of the Day:



A bipartite graph:

- m left nodes, n right nodes
- Each left node has degree 1.

Given $\log(m)$ space, a stream of edges:

1. $m = n + 1$, max right degree is 2, min right degree is 1.
   See stream once. Find *the* edge with degree 2.
2. $m = n + 2$, max right degree is 2, min right degree is 2.
   See stream once. Find two edges with degree 2.
3. $m = n + 1$, max right degree is m. See stream $\log(m)$ times.
   Find *any* edge with degree > 1.

Questions to think about:

1) Imagine a stream of tweets.
   You have no idea how long the stream of tweets is.
   How do you sample k items from the stream?

2) Do you have to download at all the tweets to sample k?
   How many tweets do you have to look at?

3) Tweets may have hashtags. Give an algorithm for finding the average number of hashtags in a tweet. (Note: each hashtag must have at least 2 characters, and tweets are at most 280 characters.)

Questions to think about:

Here is an algorithm for approximate counting:

1. $X = 0$
2. For each item in the stream, increment X with probability $p(X)$.
3. Return $f(X)$

What is a good choice of $p(X)$ and $f(X)$ to get very, very small space usage?

(Since we can trivially count in $\log(n)$ space, the goal is to do better than $\log(n)$!)