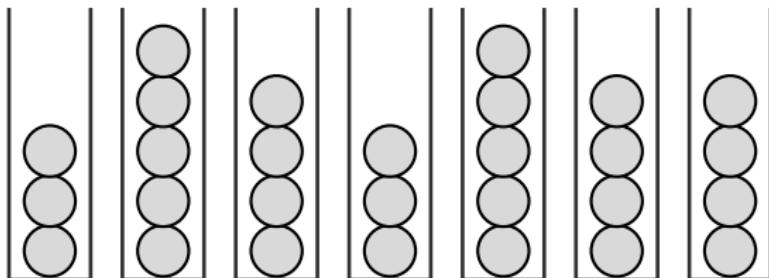


History-Independent Load Balancing



Michael A. Bender

Stony Brook University

William Kuszmaul

CMU

Elaine Shi

CMU

Rose Silver

CMU

NYT CENSORING CATASTROPHE

NYT CENSORING CATASTROPHE

Government sources differ in their accounts as to what happened when Colonel [REDACTED] tried to deliver to Mossadeq the Shah's firman dismissing him. According to General Riahi, Colonel Momtaz was on his way to the Bagh-i-Shah when he ran into Colonel [REDACTED] in the street and there-upon arrested him. According to the official communique of the Mossadeq government, [REDACTED] showed up before Mossadeq's house at 0100 hours on 16 August with four trucks full of soldiers, two jeeps, and an armored car. He claimed that

- ▶ 2000: NYT publishes internal CIA report

NYT CENSORING CATASTROPHE

Government sources differ in their accounts as to what happened when Colonel [REDACTED] tried to deliver to Mossadeq the Shah's firman dismissing him. According to General Riahi, Colonel Momtaz was on his way to the Bagh-i-Shah when he ran into Colonel [REDACTED] in the street and there-upon arrested him. According to the official communique of the Mossadeq government, [REDACTED] showed up before Mossadeq's house at 0100 hours on 16 August with four trucks full of soldiers, two jeeps, and an armored car. He claimed that

- ▶ 2000: NYT publishes internal CIA report
- ▶ Final state redacted successfully
- ▶ However, during rendering, the hidden text could be captured

NYT CENSORING CATASTROPHE

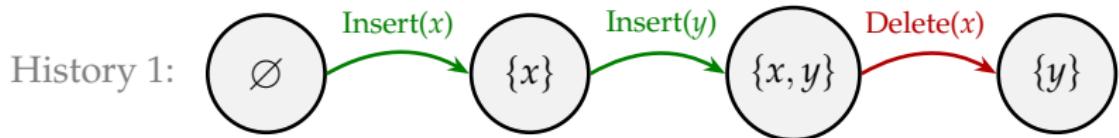
Government sources differ in their accounts as to what happened when Colonel [REDACTED] tried to deliver to Mossadeq the Shah's firman dismissing him. According to General Riahi, Colonel Momtaz was on his way to the Bagh-i-Shah when he ran into Colonel [REDACTED] in the street and there-upon arrested him. According to the official communique of the Mossadeq government, [REDACTED] showed up before Mossadeq's house at 0100 hours on 16 August with four trucks full of soldiers, two jeeps, and an armored car. He claimed that

- ▶ 2000: NYT publishes internal CIA report
- ▶ Final state redacted successfully
- ▶ However, during rendering, the hidden text could be captured

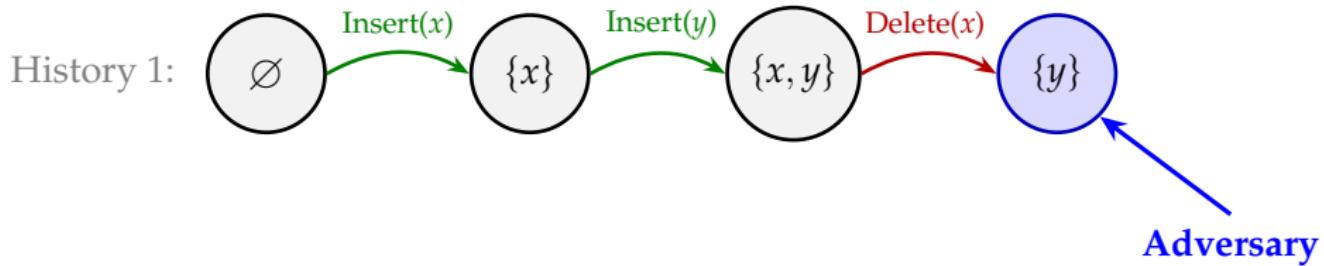
History can be just as important as content!

HISTORY-INDEPENDENT DATA STRUCTURES

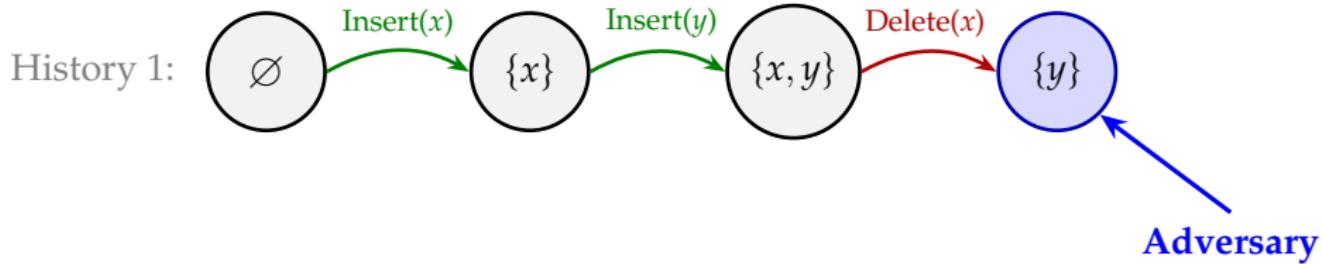
HISTORY-INDEPENDENT DATA STRUCTURES



HISTORY-INDEPENDENT DATA STRUCTURES



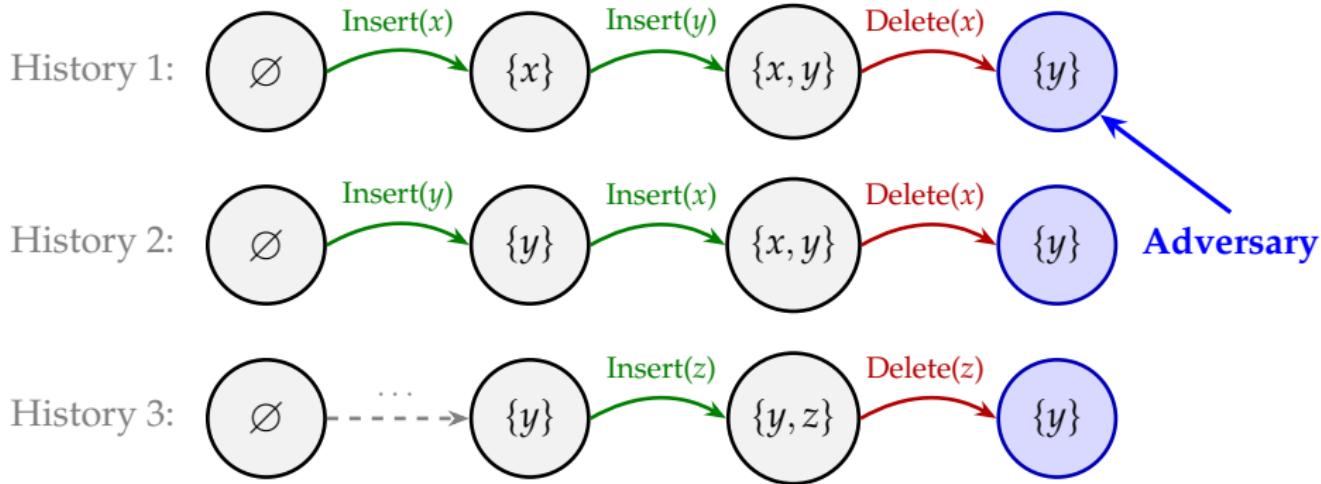
HISTORY-INDEPENDENT DATA STRUCTURES



History Independence (Micciancio '97, Naor & Teague '01)

- ▶ The state reveals only the current elements—**not the history of operations**.

HISTORY-INDEPENDENT DATA STRUCTURES



History Independence (Micciancio '97, Naor & Teague '01)

- ▶ The state reveals only the current elements—not the history of operations.

HISTORY INDEPENDENT DATA STRUCTURES

A History of Applications

Hash tables, trees, memory allocation, PMAs, graph algorithms, cache-oblivious data structures, and more.

Micciancio '97, Naor & Teague '01, Buchbinder & Petrank '03, Molnar et al. '06, Blelloch & Golovin '07,
Moran et al. '07, Naor et al. '08, Golovin '08–'10, Bajaj & Sion '13, Roche et al. '15, Bender et al. '16

HISTORY INDEPENDENT DATA STRUCTURES

A History of Applications

Hash tables, trees, memory allocation, PMAs, graph algorithms, cache-oblivious data structures, and more.

Micciancio '97, Naor & Teague '01, Buchbinder & Petrank '03, Molnar et al. '06, Blelloch & Golovin '07,
Moran et al. '07, Naor et al. '08, Golovin '08–'10, Bajaj & Sion '13, Roche et al. '15, Bender et al. '16

Yet some basic questions remain open.

HISTORY INDEPENDENT DATA STRUCTURES

A History of Applications

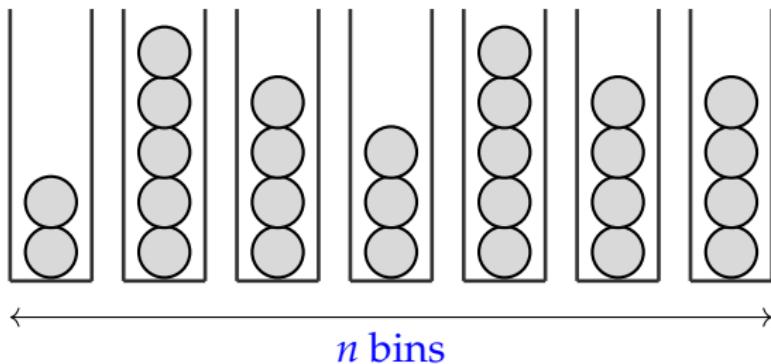
Hash tables, trees, memory allocation, PMAs, graph algorithms, cache-oblivious data structures, and more.

Micciancio '97, Naor & Teague '01, Buchbinder & Petrank '03, Molnar et al. '06, Blelloch & Golovin '07,
Moran et al. '07, Naor et al. '08, Golovin '08–'10, Bajaj & Sion '13, Roche et al. '15, Bender et al. '16

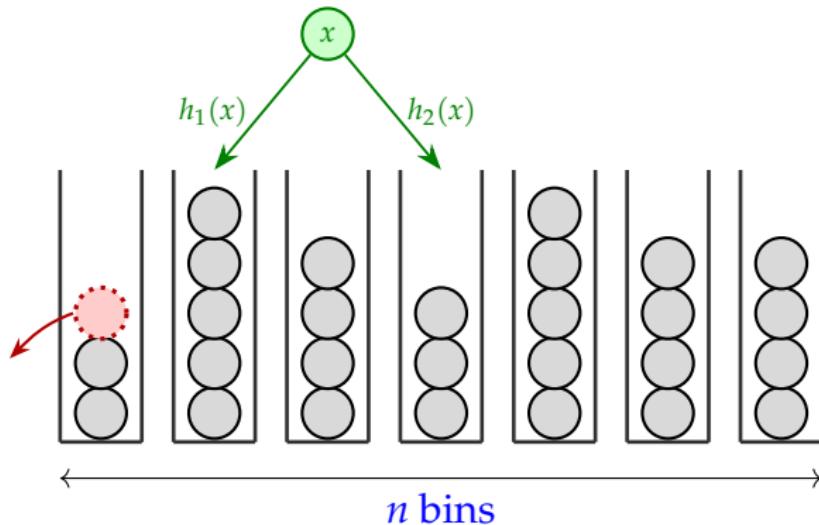
Yet some basic questions remain open.

This work: History-Independent Load Balancing

TWO-CHOICE LOAD BALANCING

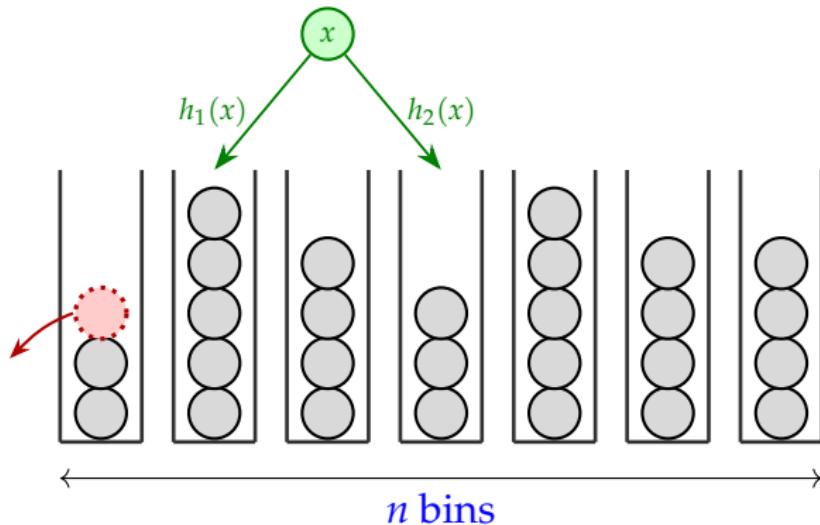


TWO-CHOICE LOAD BALANCING



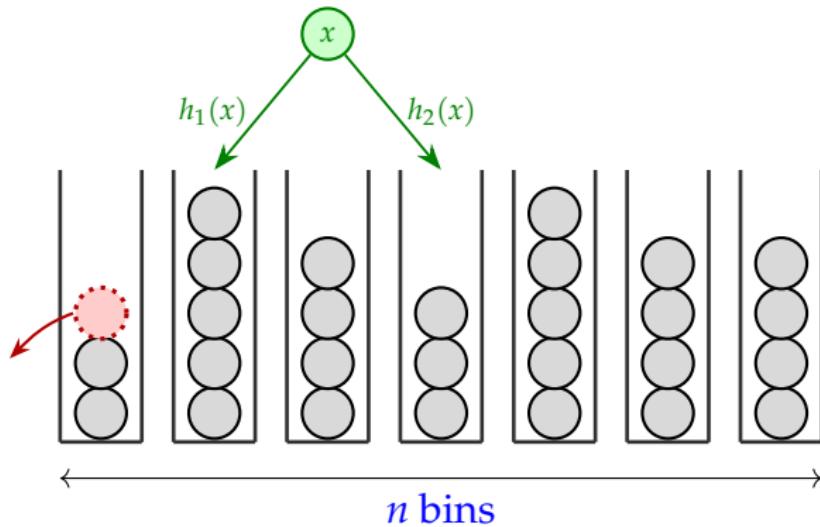
- ▶ Balls are **inserted**/**deleted**, with up to m present at a time.

TWO-CHOICE LOAD BALANCING



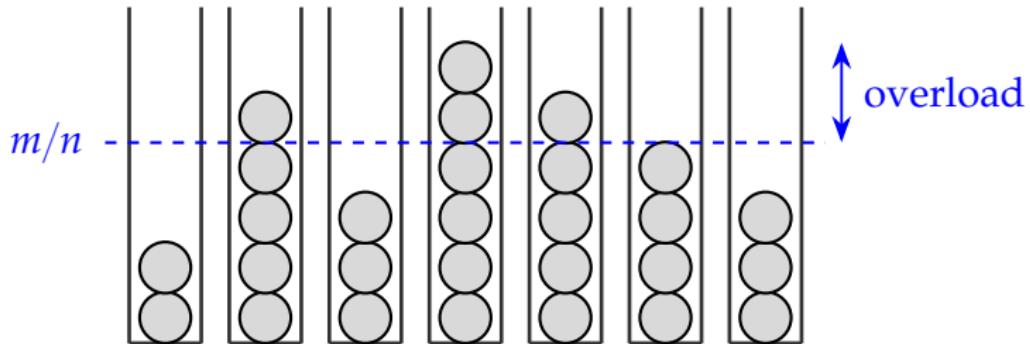
- ▶ Balls are **inserted**/**deleted**, with up to m present at a time.
- ▶ Each ball has two random bins where it can go.

TWO-CHOICE LOAD BALANCING



- ▶ Balls are **inserted**/**deleted**, with up to m present at a time.
- ▶ Each ball has two random bins where it can go.
- ▶ We must maintain a valid assignment of balls to bins.

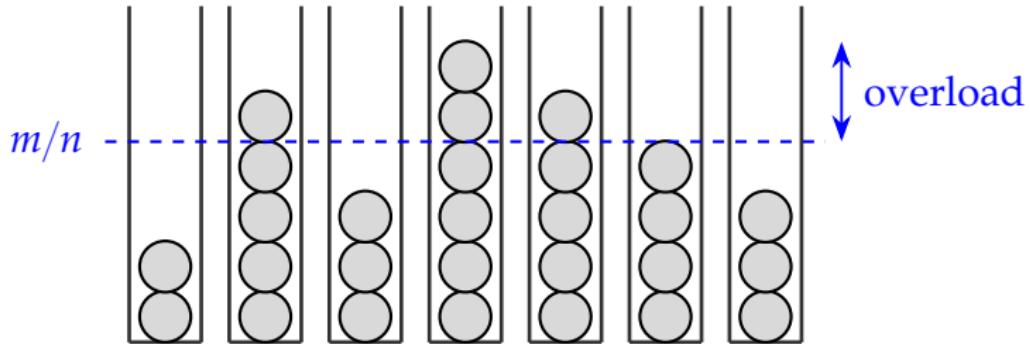
TWO GOALS



Minimize Overload:

- ▶ i.e., the amount by which the fullest bin exceeds m/n .

TWO GOALS



Minimize Overload:

- ▶ i.e., the amount by which the fullest bin exceeds m/n .

Minimize Recourse:

- ▶ i.e., the number of balls moved around on any given insertion/deletion.

PUTTING IT ALL TOGETHER

PUTTING IT ALL TOGETHER

History-Independent Load Balancing:

PUTTING IT ALL TOGETHER

History-Independent Load Balancing:

- ▶ Given the set S of balls, the assignment can be constructed from scratch.

PUTTING IT ALL TOGETHER

History-Independent Load Balancing:

- ▶ Given the set S of balls, the assignment can be constructed from scratch.
- ▶ We call this assignment A_S .

PUTTING IT ALL TOGETHER

History-Independent Load Balancing:

- ▶ Given the set S of balls, the assignment can be constructed from scratch.
- ▶ We call this assignment A_S .

Question: Does there exist a history-independent solution with small recourse and small overload?

PUTTING IT ALL TOGETHER

History-Independent Load Balancing:

- ▶ Given the set S of balls, the assignment can be constructed from scratch.
- ▶ We call this assignment A_S .

Question: Does there exist a history-independent solution with small recourse and small overload?

Our Main Result: There exists a history-independent solution with:

- ▶ High probability overload $O(1)$
- ▶ Expected recourse $O(\log \log(m/n))$

PAST WORK (NOT HISTORY INDEPENDENT)

Overload	Recourse	Reference	Caveats
$O(\log \log n)$	0	[ABKU '94] [BCSV '00]	insertion-only
$O(1)$	$O(\log(m/n))$	[Dietzfelbinger, Weidling '07]	insertion-only
$\tilde{O}(\sqrt{m/n})$	$O(1)$	[Frieze, Petti '18]	insertion-only
$O(\log(m/n))$	0	[Bansal, Kuszmaul '22]	no reinsertions
$O(1)$	$O(m/n)$	[Dietzfelbinger, Weidling '07]	

PAST WORK (NOT HISTORY INDEPENDENT)

Overload	Recourse	Reference	Caveats
$O(\log \log n)$	0	[ABKU '94] [BCSV '00]	insertion-only
$O(1)$	$O(\log(m/n))$	[Dietzfelbinger, Weidling '07]	insertion-only
$\tilde{O}(\sqrt{m/n})$	$O(1)$	[Frieze, Petti '18]	insertion-only
$O(\log(m/n))$	0	[Bansal, Kuszmaul '22]	no reinsertions
$O(1)$	$O(m/n)$	[Dietzfelbinger, Weidling '07]	
$O(1)$	$O(\log \log(m/n))$	[This Paper]	

PAST WORK (NOT HISTORY INDEPENDENT)

Overload	Recourse	Reference	Caveats
$O(\log \log n)$	0	[ABKU '94] [BCSV '00]	insertion-only
$O(1)$	$O(\log(m/n))$	[Dietzfelbinger, Weidling '07]	insertion-only
$\tilde{O}(\sqrt{m/n})$	$O(1)$	[Frieze, Petti '18]	insertion-only
$O(\log(m/n))$	0	[Bansal, Kuszmaul '22]	no reinsertions
$O(1)$	$O(m/n)$	[Dietzfelbinger, Weidling '07]	
$O(1)$	$O(\log \log(m/n))$	[This Paper]	

If we want overload $O(1)$, our result is a new state of the art!

REST OF TALK

1. A Simple Warmup
2. The Full Algorithm

Part 1: A Simple Warmup

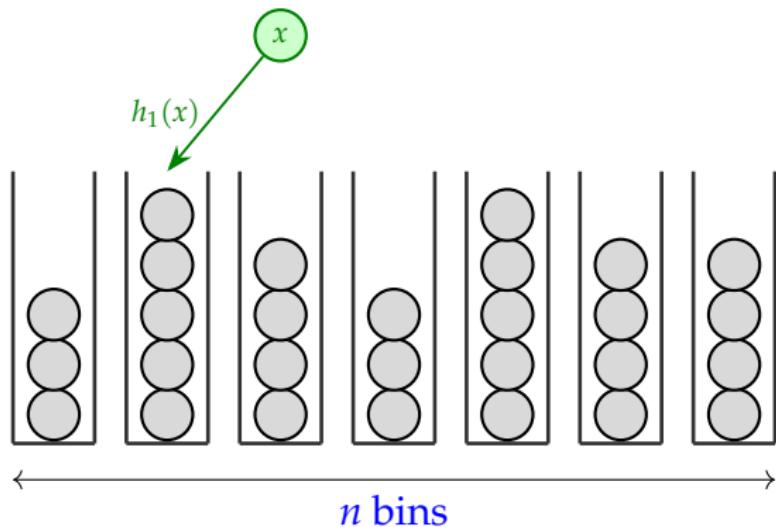
A SIMPLE WARMUP

Theorem: There exists a history-independent solution with:

- ▶ High-probability overload $\Theta(1)$ $O(\log \log n)$.
- ▶ Expected recourse $\Theta(\log \log(m/n))$ $O(m/n)$.

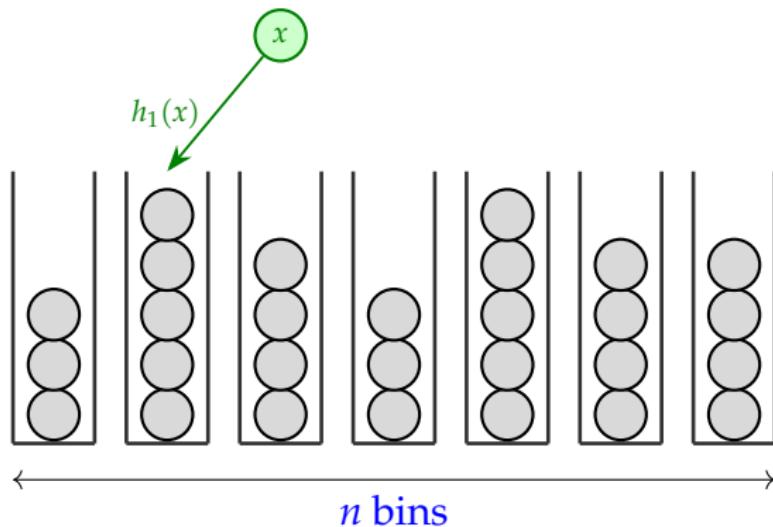
BASELINE 1: THE SINGLE-CHOICE STRATEGY

To insert a ball x , just put it in bin $h_1(x)$:



BASELINE 1: THE SINGLE-CHOICE STRATEGY

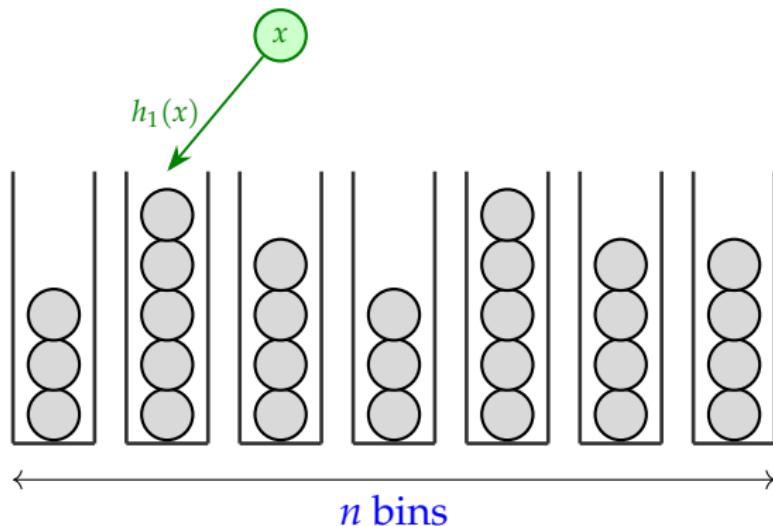
To insert a ball x , just put it in bin $h_1(x)$:



- ▶ This is history-independent ✓

BASELINE 1: THE SINGLE-CHOICE STRATEGY

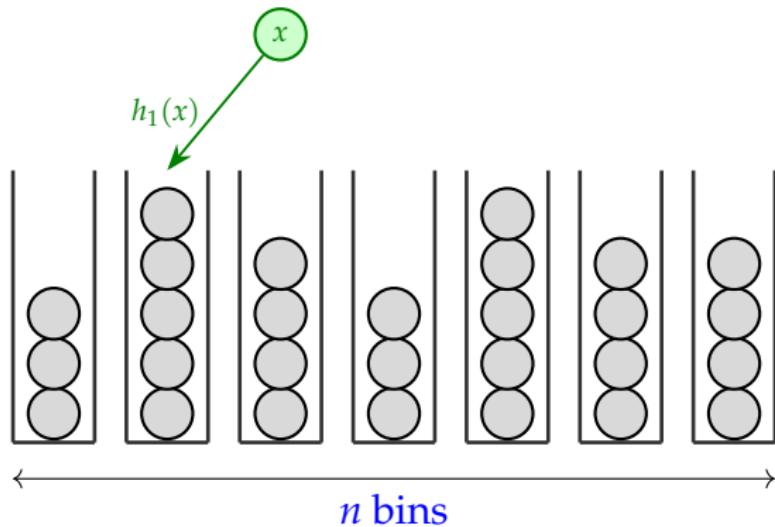
To insert a ball x , just put it in bin $h_1(x)$:



- ▶ This is history-independent ✓
- ▶ The recourse is 0 ✓

BASELINE 1: THE SINGLE-CHOICE STRATEGY

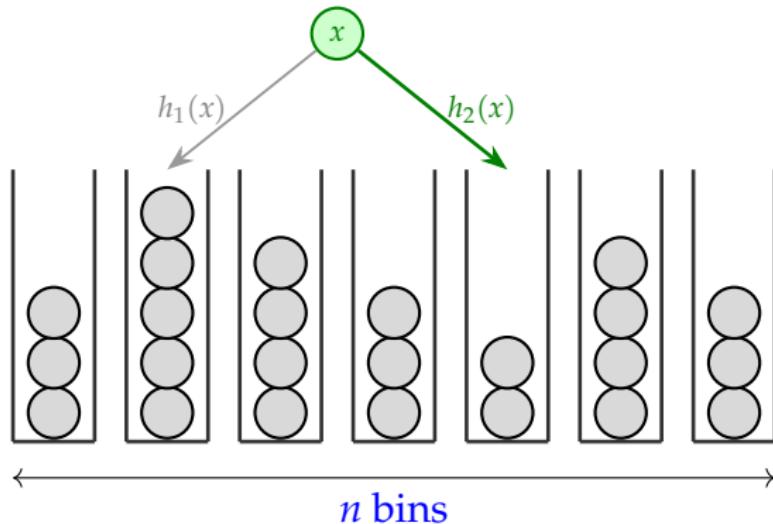
To insert a ball x , just put it in bin $h_1(x)$:



- ▶ This is history-independent ✓
- ▶ The recourse is 0 ✓
- ▶ But... the overload is huge, roughly $\sqrt{m/n}$ ✗

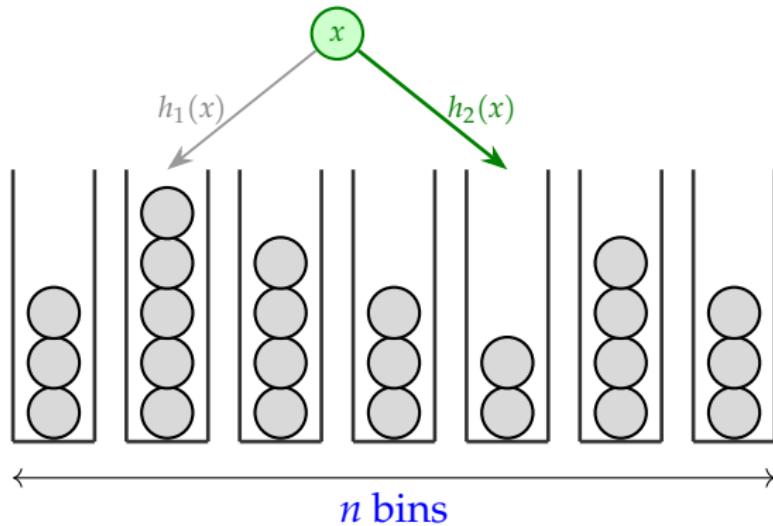
BASELINE 2: GREEDY INSERTIONS

To insert a ball x , put it in the **emptier** of its choices:



BASELINE 2: GREEDY INSERTIONS

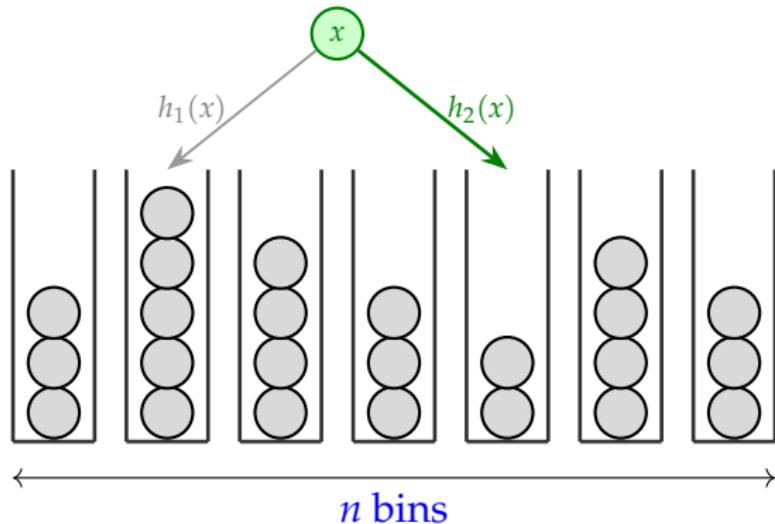
To insert a ball x , put it in the **emptier** of its choices:



- ▶ This is **not** history-independent \times

BASELINE 2: GREEDY INSERTIONS

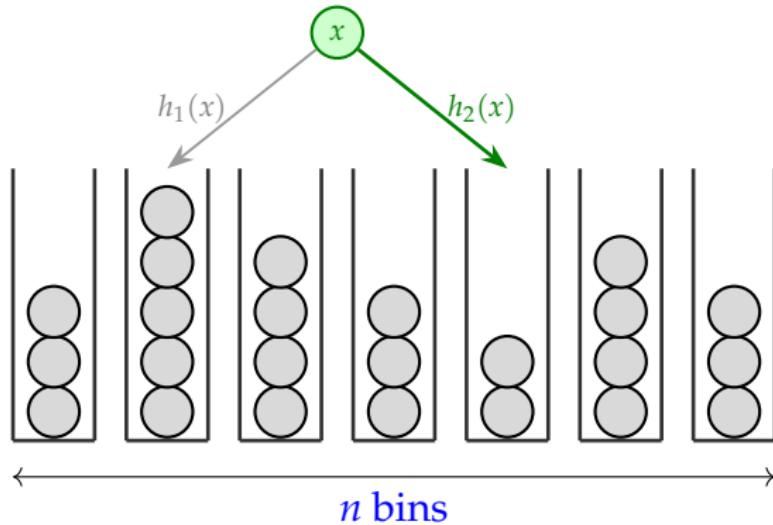
To insert a ball x , put it in the **emptier** of its choices:



- ▶ This is **not** history-independent ✗
- ▶ The recourse is 0 ✓

BASELINE 2: GREEDY INSERTIONS

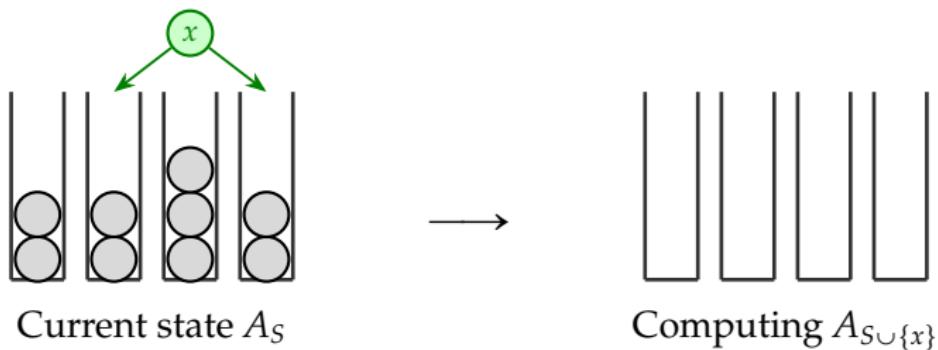
To insert a ball x , put it in the **emptier** of its choices:



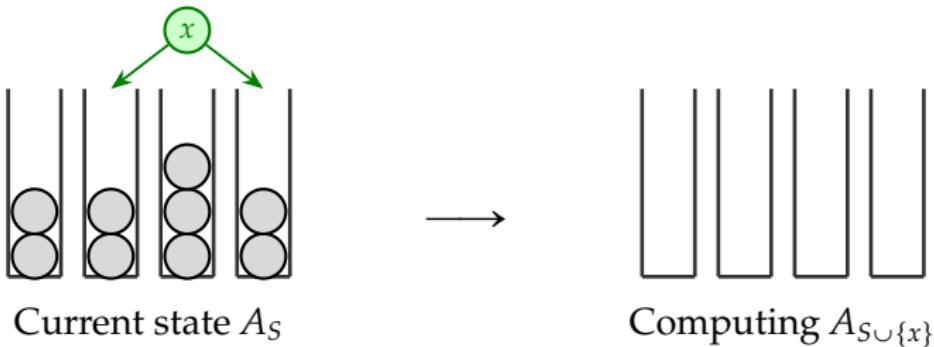
- ▶ This is **not** history-independent ✗
- ▶ The recourse is 0 ✓
- ▶ In the insertion-only case, the overload is $O(\log \log n)$ ✓
[Berenbrink, Czumaj, Steger, and Vöcking '00]

WARMUP: HISTORY-INDEPENDENT GREEDY

WARMUP: HISTORY-INDEPENDENT GREEDY



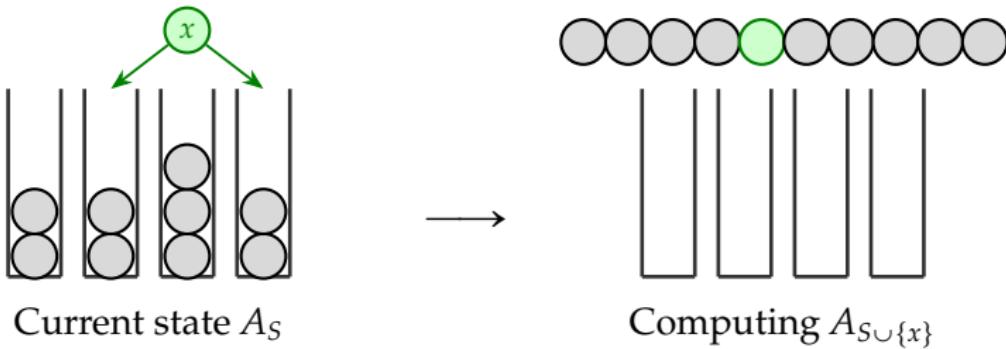
WARMUP: HISTORY-INDEPENDENT GREEDY



To compute $A_{S \cup \{x\}}$:

1. Empty out the bins.
2. Sort the balls in $S \cup \{x\}$ to get x_1, x_2, \dots
3. Insert the balls in sorted order using greedy.

WARMUP: HISTORY-INDEPENDENT GREEDY



To compute $A_{S \cup \{x\}}$:

1. Empty out the bins.
2. Sort the balls in $S \cup \{x\}$ to get x_1, x_2, \dots
3. Insert the balls in sorted order using greedy.

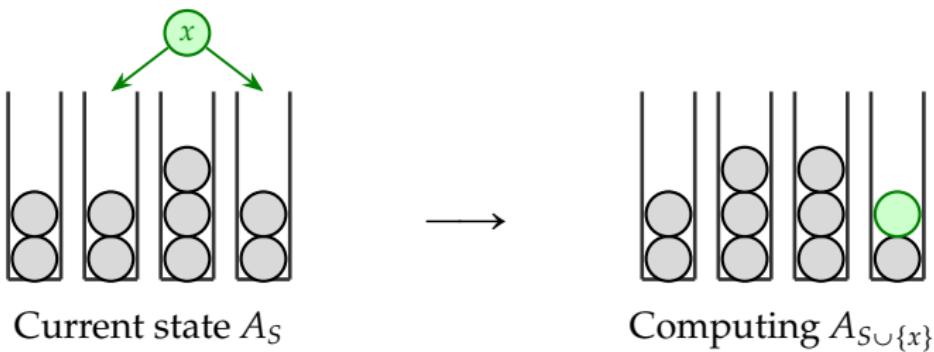
WARMUP: HISTORY-INDEPENDENT GREEDY



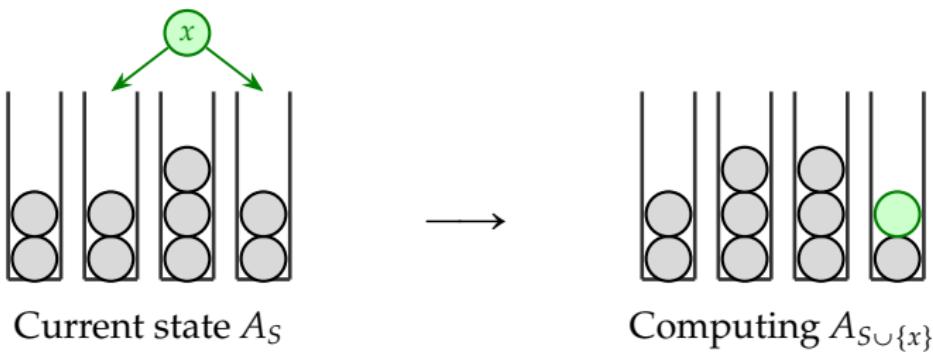
To compute $A_{S \cup \{x\}}$:

1. Empty out the bins.
2. Sort the balls in $S \cup \{x\}$ to get x_1, x_2, \dots
3. Insert the balls in sorted order using greedy.

ANALYZING HISTORY-INDEPENDENT GREEDY

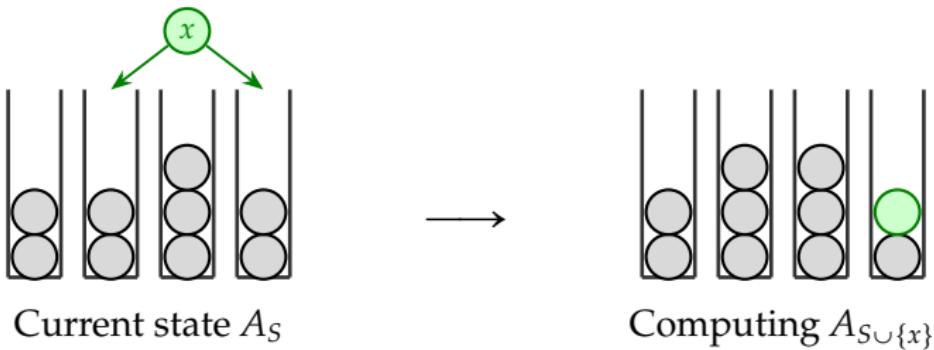


ANALYZING HISTORY-INDEPENDENT GREEDY



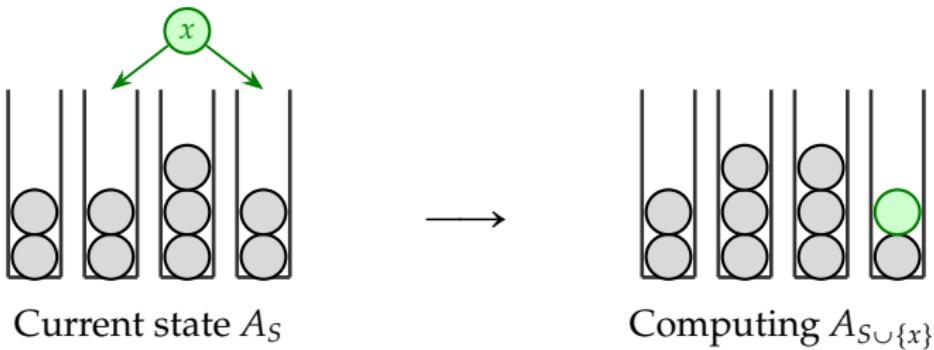
- ▶ The algorithm is history independent ✓

ANALYZING HISTORY-INDEPENDENT GREEDY



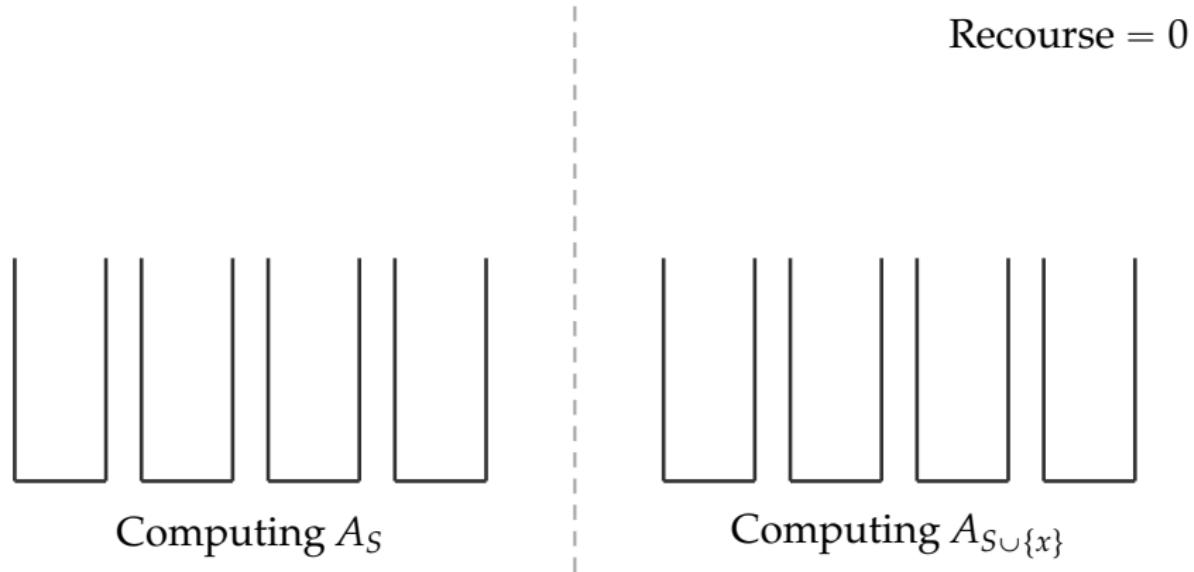
- ▶ The algorithm is history independent ✓
- ▶ The overload is $O(\log \log n)$ ✓

ANALYZING HISTORY-INDEPENDENT GREEDY



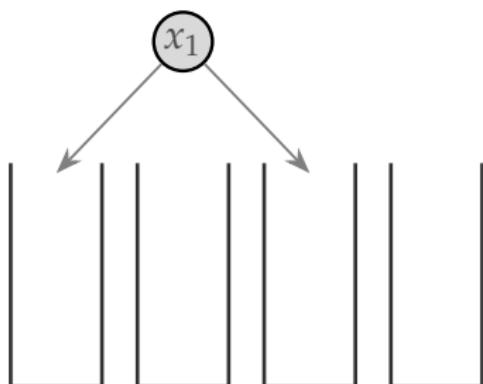
- ▶ The algorithm is history independent ✓
- ▶ The overload is $O(\log \log n)$ ✓
- ▶ What is the recourse?

ANALYZING THE RECOURSE

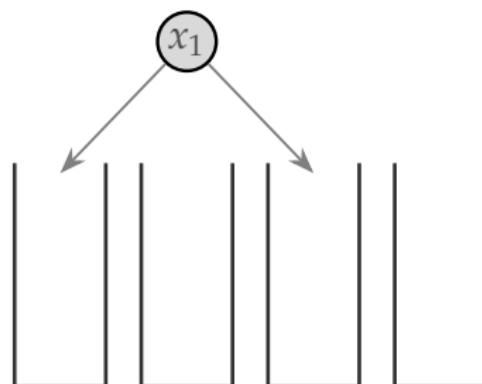


How many balls change assignments between A_S and $A_{S \cup \{x\}}$?

ANALYZING THE RECOURSE



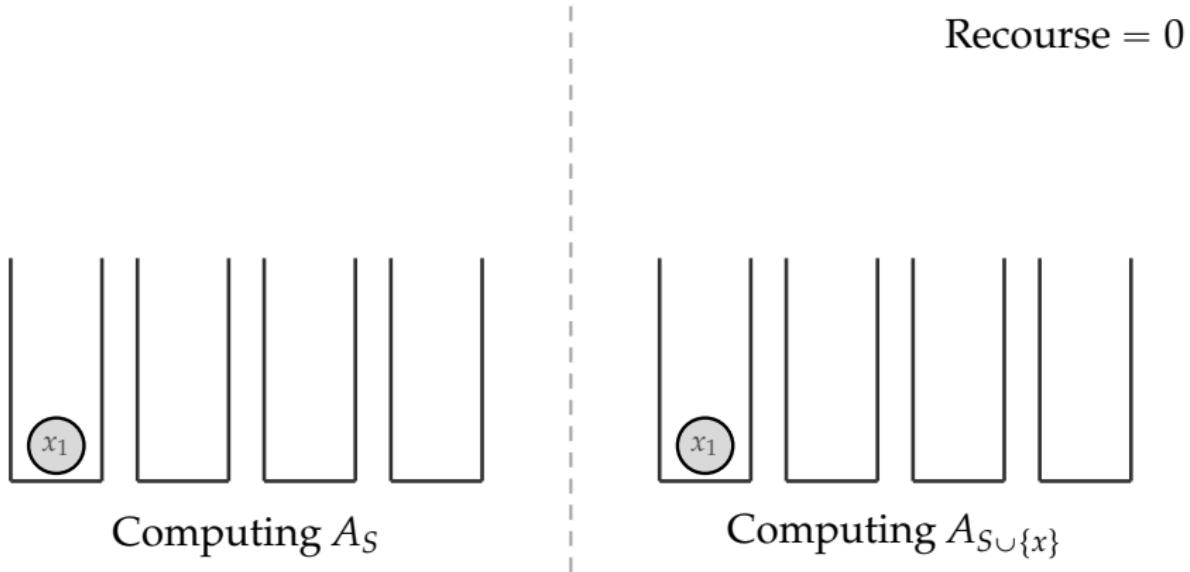
Computing A_S



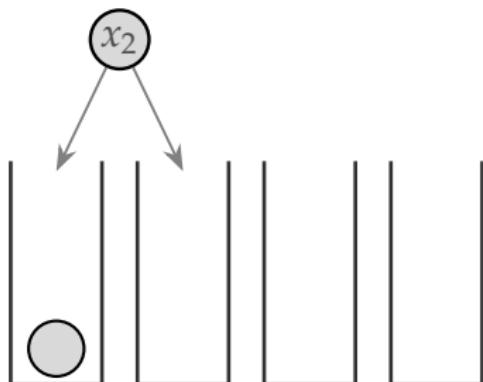
Computing $A_{S \cup \{x\}}$

Recourse = 0

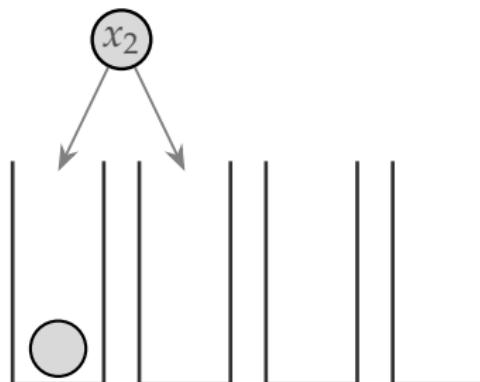
ANALYZING THE RECOURSE



ANALYZING THE RECOURSE



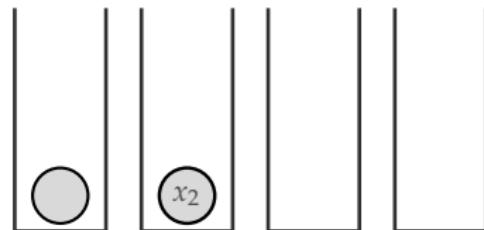
Computing A_S



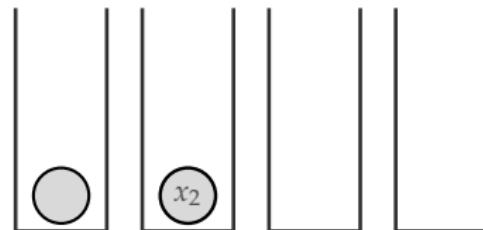
Computing $A_{S \cup \{x\}}$

Recourse = 0

ANALYZING THE RECOURSE



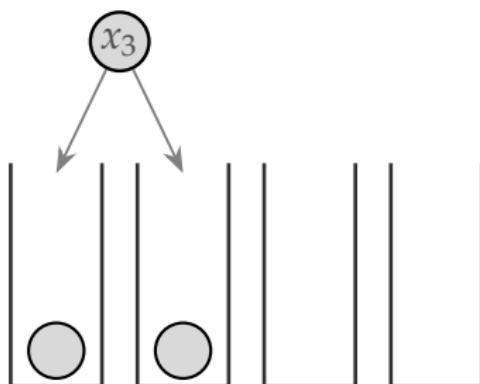
Computing A_S



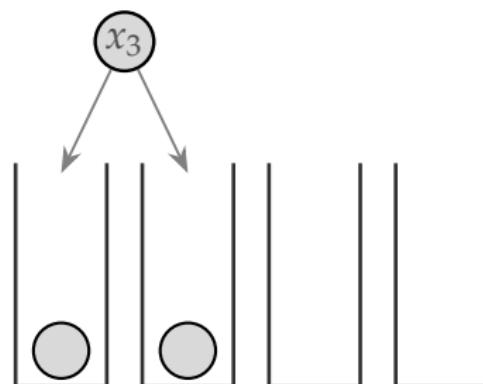
Computing $A_{S \cup \{x\}}$

Recourse = 0

ANALYZING THE RECOURSE



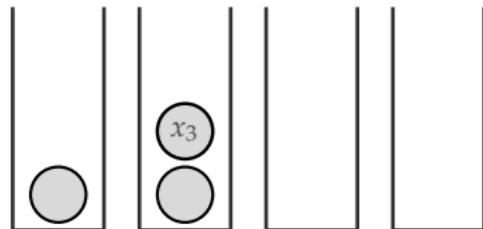
Computing A_S



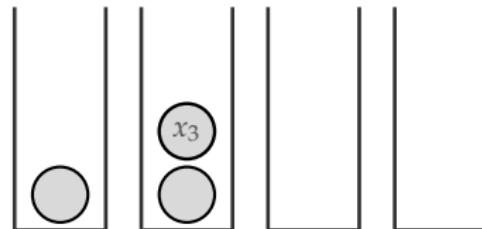
Computing $A_{S \cup \{x\}}$

Recourse = 0

ANALYZING THE RECOURSE



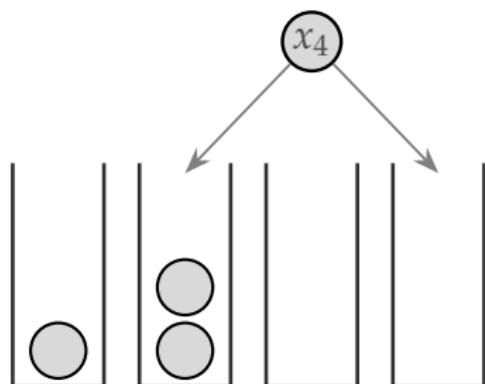
Computing A_S



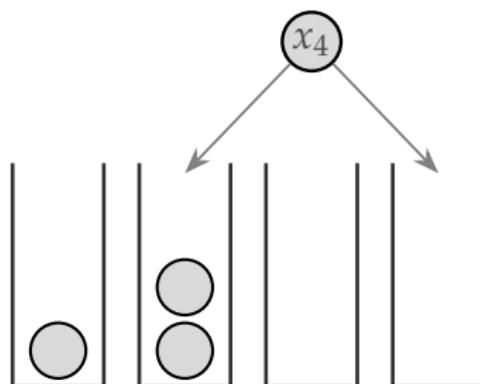
Computing $A_{S \cup \{x\}}$

Recourse = 0

ANALYZING THE RECOURSE



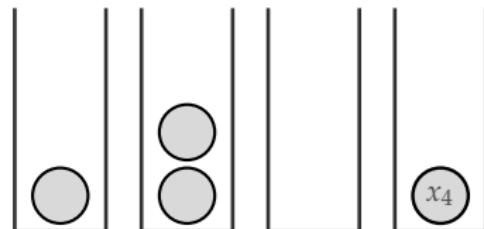
Computing A_S



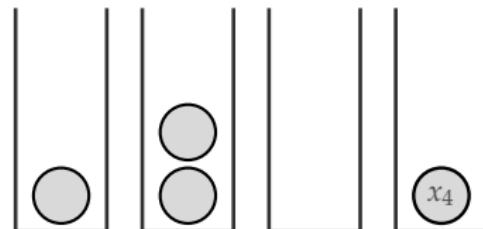
Computing $A_{S \cup \{x\}}$

Recourse = 0

ANALYZING THE RECOURSE



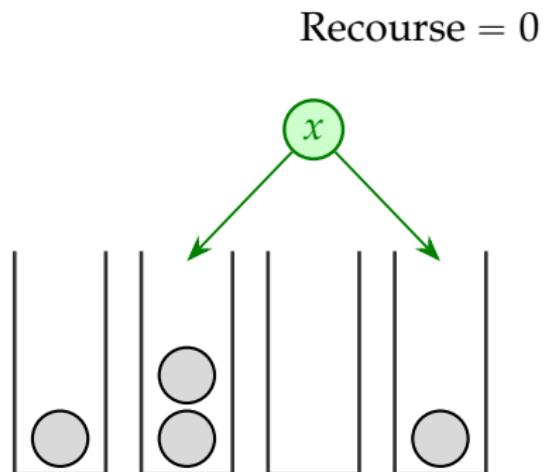
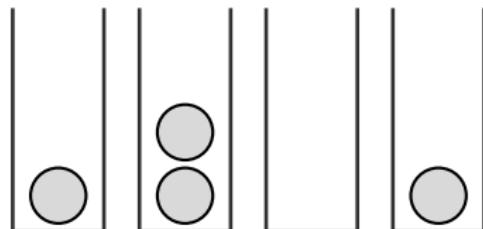
Computing A_S



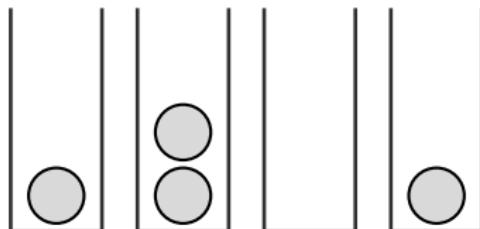
Computing $A_{S \cup \{x\}}$

Recourse = 0

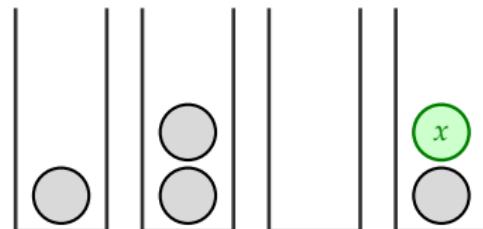
ANALYZING THE RECOURSE



ANALYZING THE RECOURSE



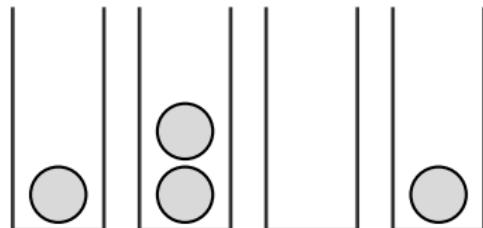
Computing A_S



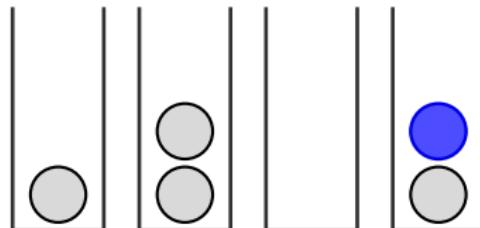
Computing $A_{S \cup \{x\}}$

Recourse = 0

ANALYZING THE RECOURSE



Computing A_S

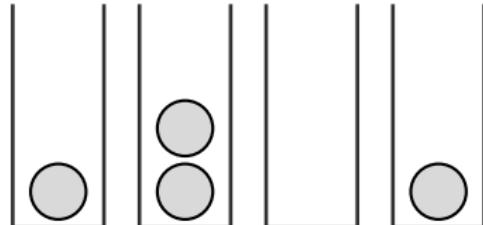


Computing $A_{S \cup \{x\}}$

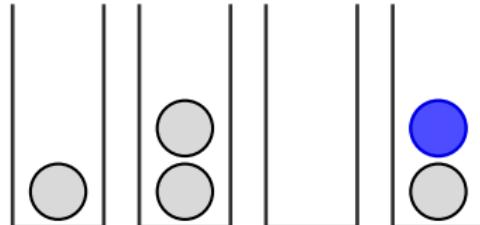
Subsequent balls will experience either:

Recourse = 0

ANALYZING THE RECOURSE



Computing A_S

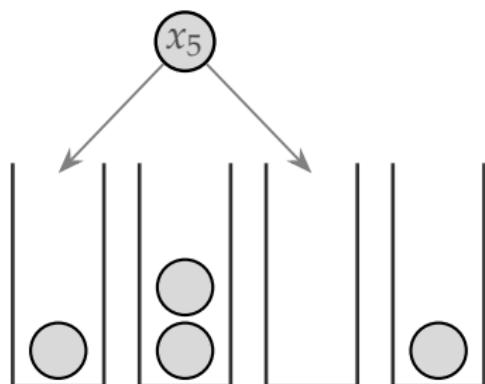


Computing $A_{S \cup \{x\}}$

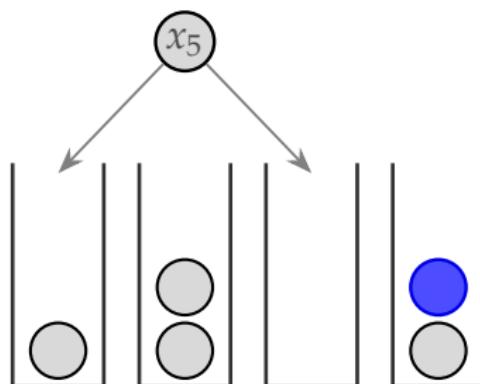
Subsequent balls will experience either:

1. No recourse

ANALYZING THE RECOURSE



Computing A_S



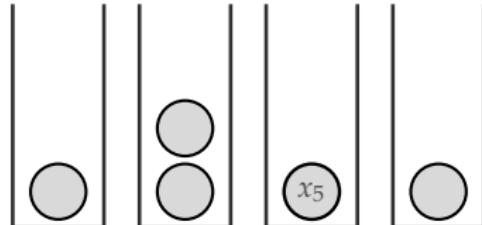
Computing $A_{S \cup \{x\}}$

Future insertions will experience either:

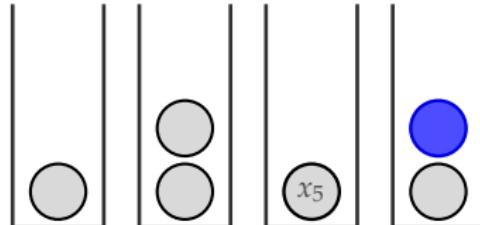
1. No recourse

Recourse = 0

ANALYZING THE RECOURSE



Computing A_S

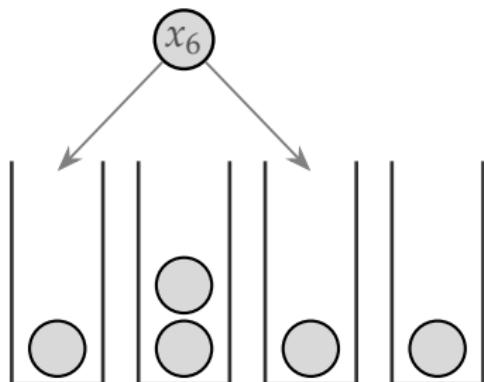


Computing $A_{S \cup \{x\}}$

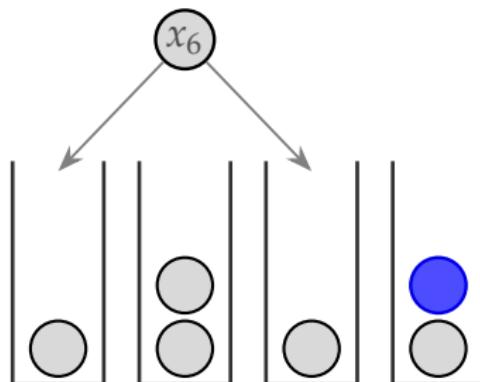
Subsequent balls will experience either:

1. No recourse

ANALYZING THE RECOURSE



Computing A_S



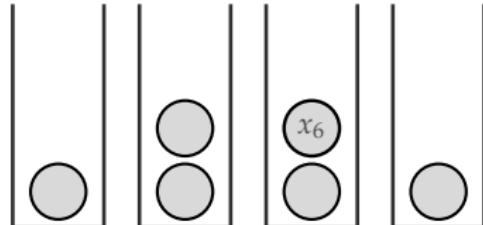
Computing $A_{S \cup \{x\}}$

Subsequent balls will experience either:

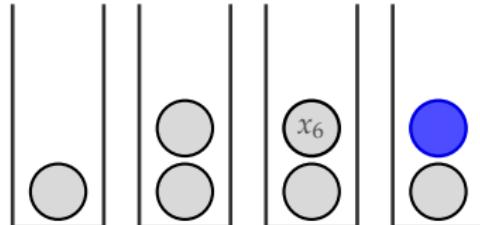
1. No recourse

Recourse = 0

ANALYZING THE RECOURSE



Computing A_S

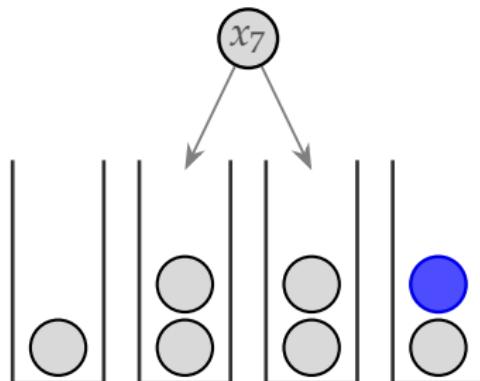
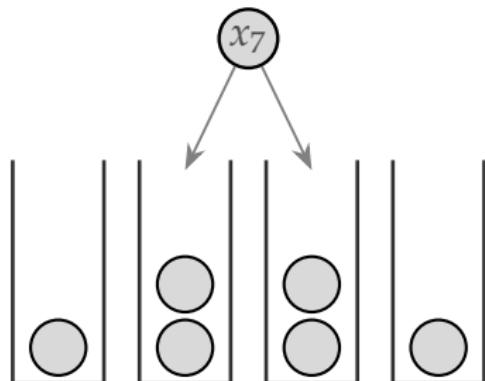


Computing $A_{S \cup \{x\}}$

Subsequent balls will experience either:

1. No recourse

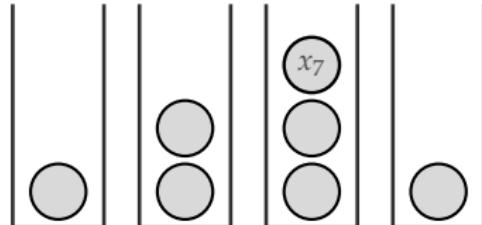
ANALYZING THE RECOURSE



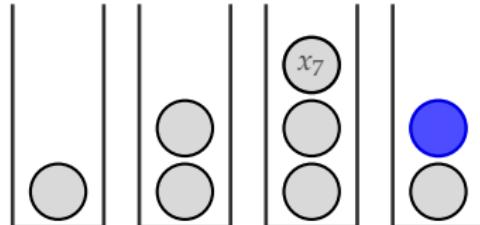
Subsequent balls will experience either:

1. No recourse

ANALYZING THE RECOURSE



Computing A_S



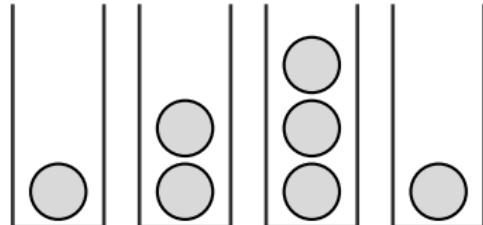
Computing $A_{S \cup \{x\}}$

Subsequent balls will experience either:

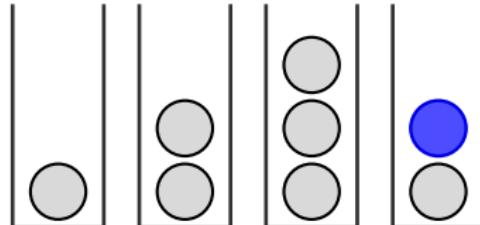
1. No recourse

$$\text{Recourse} = 0$$

ANALYZING THE RECOURSE



Computing A_S



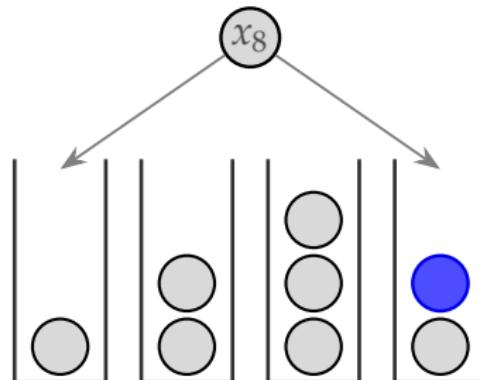
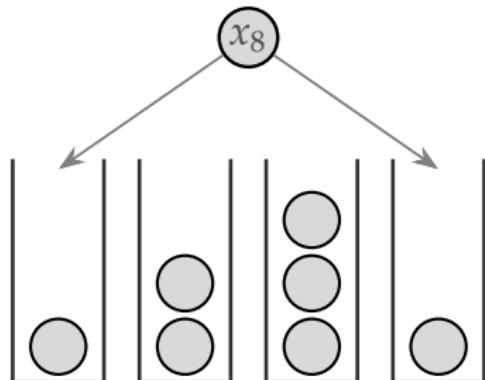
Computing $A_{S \cup \{x\}}$

Subsequent balls will experience either:

1. No recourse
2. Recourse

$$\text{Recourse} = 0$$

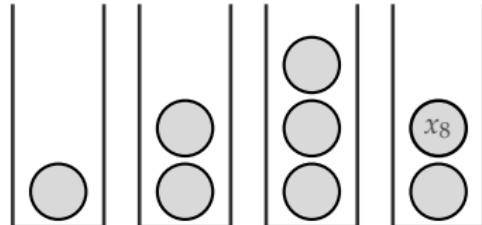
ANALYZING THE RECOURSE



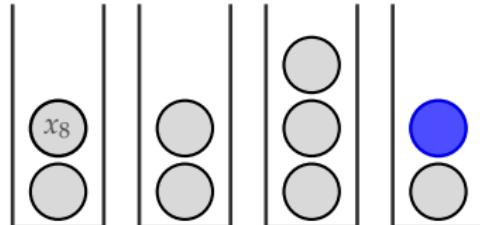
Subsequent balls will experience either:

1. No recourse
2. Recourse

ANALYZING THE RECOURSE



Computing A_S



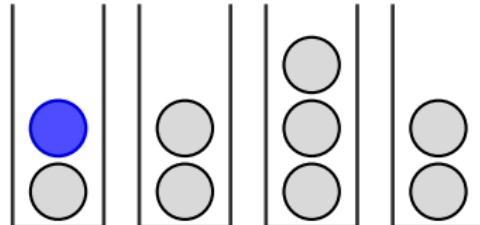
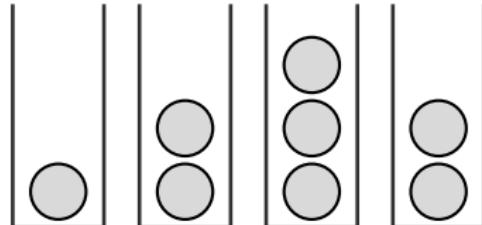
Computing $A_{S \cup \{x\}}$

Subsequent balls will experience either:

1. No recourse
2. Recourse

Recourse = 1

ANALYZING THE RECOURSE

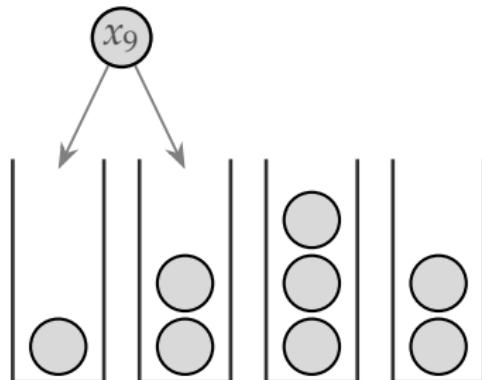


Recourse = 1

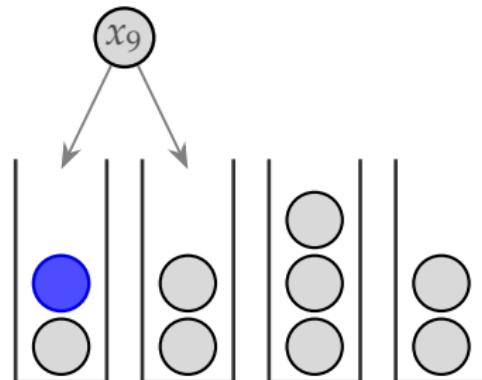
Subsequent balls will experience either:

1. No recourse
2. Recourse

ANALYZING THE RECOURSE



Computing A_S



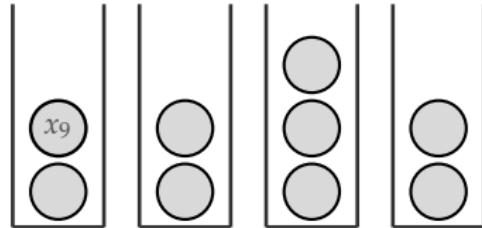
Computing $A_{S \cup \{x\}}$

Subsequent balls will experience either:

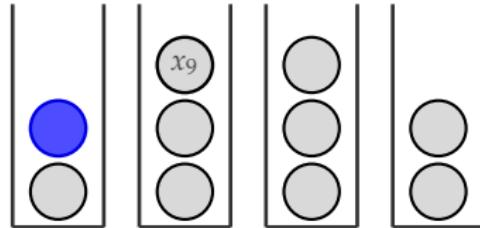
1. No recourse
2. Recourse

Recourse = 1

ANALYZING THE RECOURSE



Computing A_S



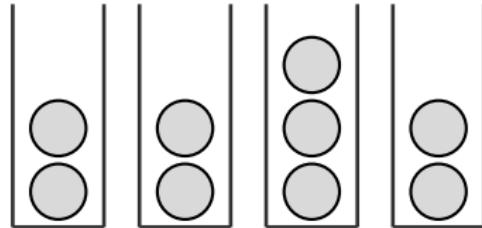
Computing $A_{S \cup \{x\}}$

Subsequent balls will experience either:

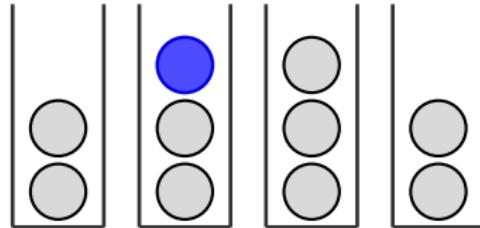
1. No recourse
2. Recourse

Recourse = 2

ANALYZING THE RECOURSE



Computing A_S



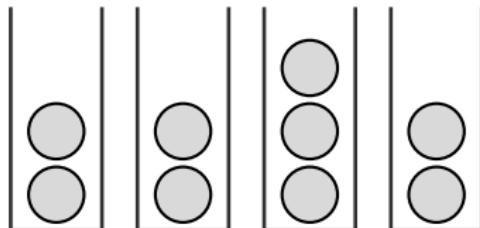
Computing $A_{S \cup \{x\}}$

Subsequent balls will experience either:

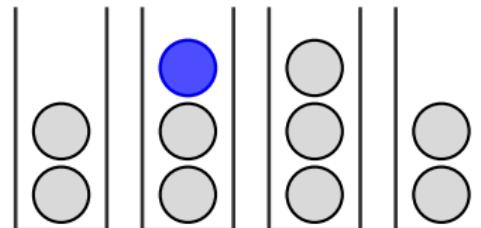
1. No recourse
2. Recourse

Recourse = 2

ANALYZING THE RECOURSE



Computing A_S

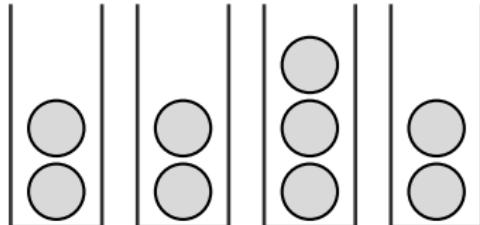


Computing $A_{S \cup \{x\}}$

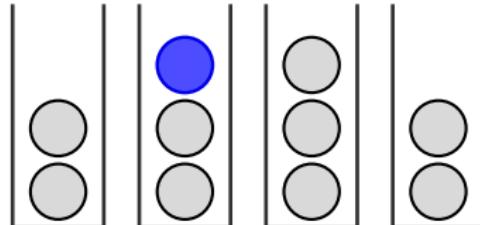
Two key observations:

Recourse = 2

ANALYZING THE RECOURSE



Computing A_S



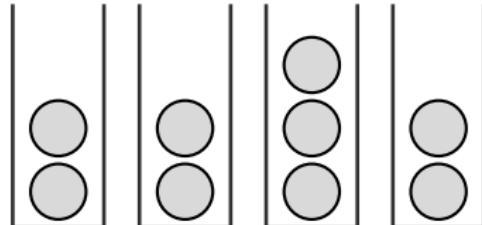
Computing $A_{S \cup \{x\}}$

Two key observations:

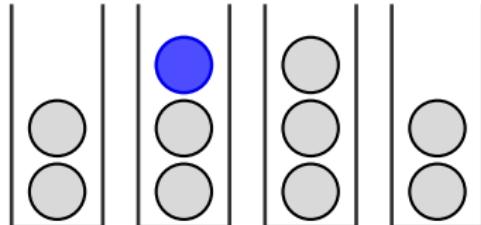
1. There's always one special bin with an extra ball

Recourse = 2

ANALYZING THE RECOURSE



Computing A_S



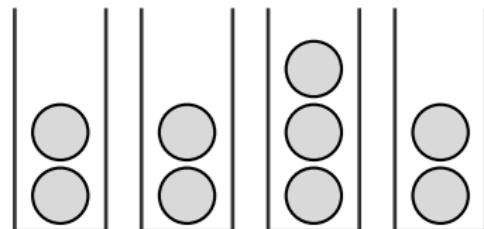
Computing $A_{S \cup \{x\}}$

Two key observations:

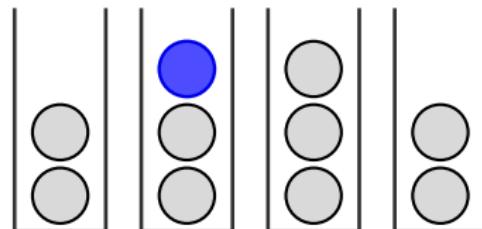
1. There's always one special bin with an extra ball
2. If a ball incurs recourse, one of its choices is the special bin

Recourse = 2

ANALYZING THE RECOURSE



Computing A_S

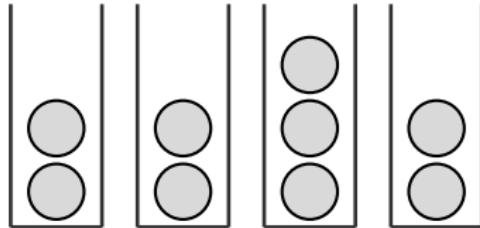


Computing $A_{S \cup \{x\}}$

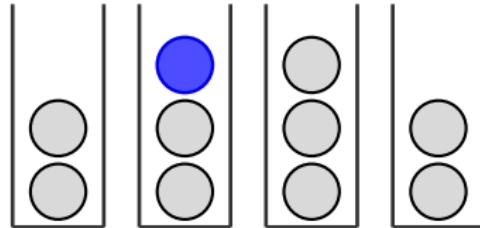
$$\Pr[\text{ball } x_i \text{ incurs recourse}] = O(1/n)$$

Recourse = 2

ANALYZING THE RECOURSE



Computing A_S



Computing $A_{S \cup \{x\}}$

$$\Pr[\text{ball } x_i \text{ incurs recourse}] = O(1/n)$$

$$\implies \mathbb{E}[\text{total recourse}] = \sum_i \Pr[\text{ball } x_i \text{ incurs recourse}] = O(m/n)$$

Recourse = 2

A SIMPLE WARMUP

Theorem: History-Independent Greedy achieves:

- ▶ High-probability overload $\Theta(1)$ $O(\log \log n)$.
- ▶ Expected recourse $\Theta(\log \log(m/n))$ $O(m/n)$.

A SIMPLE WARMUP

Theorem: History-Independent Greedy achieves:

- ▶ High-probability overload $\Theta(1) O(\log \log n)$.
- ▶ Expected recourse $\Theta(\log \log(m/n)) O(m/n)$.



We prove a matching lower bound
when $m \leq n^{2-\Omega(1)}$

REST OF TALK

1. A Simple Warmup ✓
2. The Full Algorithm

Part 2: The Full Algorithm

BAKING A CAKE



Michael



Elaine



Bill

BAKING A CAKE

Before



Michael



Elaine



Bill

BAKING A CAKE

Before



**Slice off the
excess flour!**



Michael



Elaine



Bill

BAKING A CAKE

Before



**Slice off the
excess flour!**



Michael

Spread it evenly!

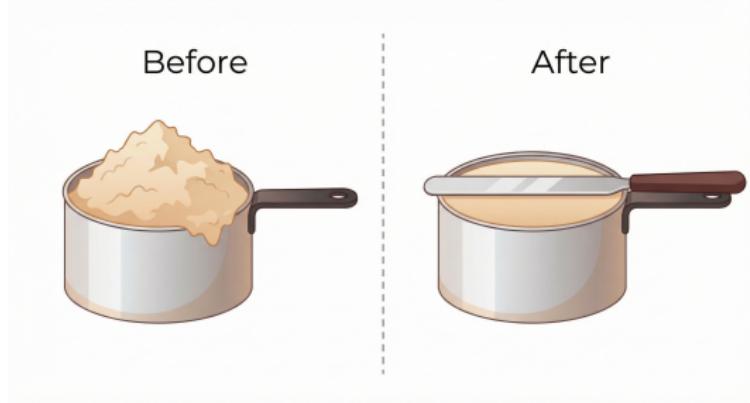


Elaine



Bill

BAKING A CAKE



**Slice off the
excess flour!**



Michael

Spread it evenly!

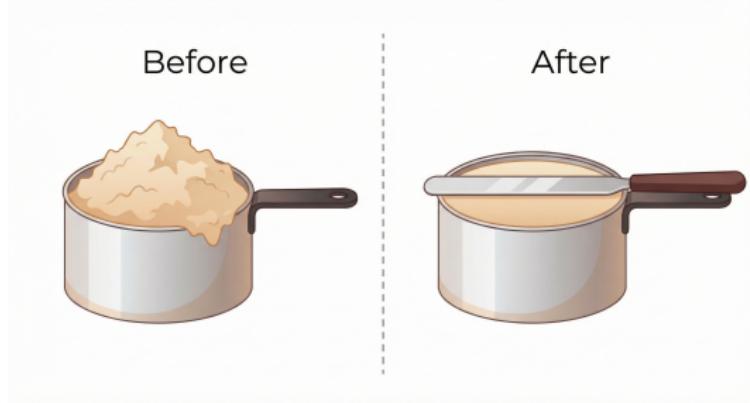


Elaine



Bill

BAKING A CAKE



**Slice off the
excess flour!**



Michael

Spread it evenly!



Elaine

You two rehearsed
this, didn't you?



Bill

BAKING A CAKE

Before



Michael



Elaine



Bill

BAKING A CAKE

Before



**Slice off the
excess frosting!**



Michael



Elaine



Bill

BAKING A CAKE

Before



Slice off the
excess frosting!



Michael

Spread it smooth!

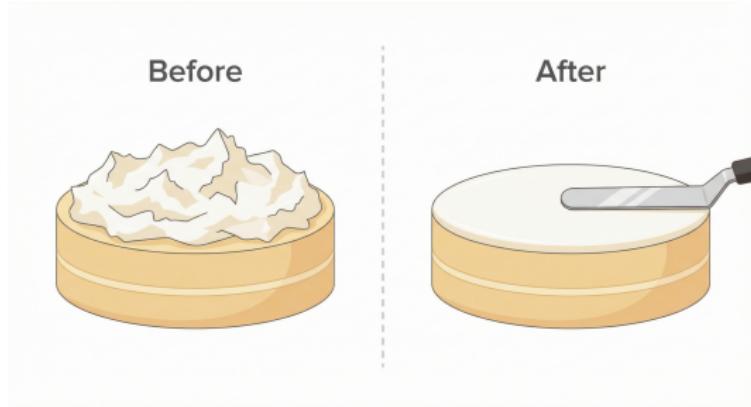


Elaine



Bill

BAKING A CAKE



**Slice off the
excess frosting!**



Michael

Spread it smooth!



Elaine



Bill

BAKING A CAKE

Before



After



**Slice off the
excess frosting!**



Michael

Spread it smooth!



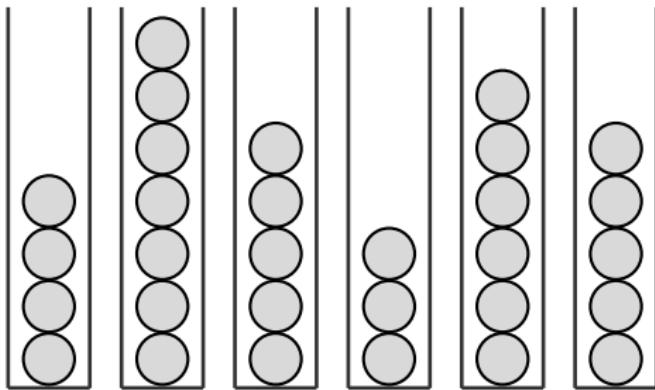
Elaine

I'm starting to
think I'm the
comic relief here...

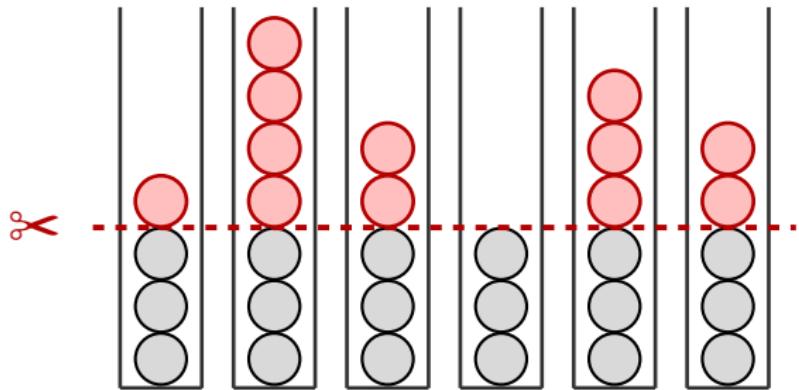


Bill

SLICE AND SPREAD

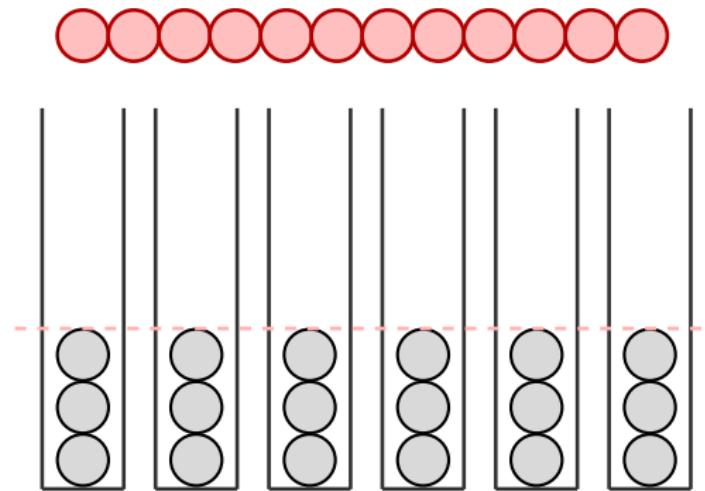


SLICE AND SPREAD



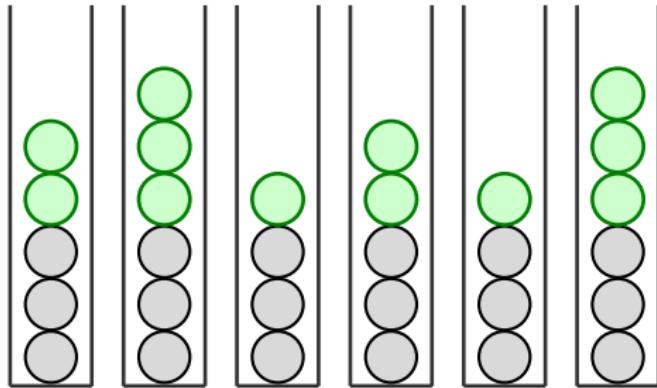
1. **Slice** off the jagged surface

SLICE AND SPREAD



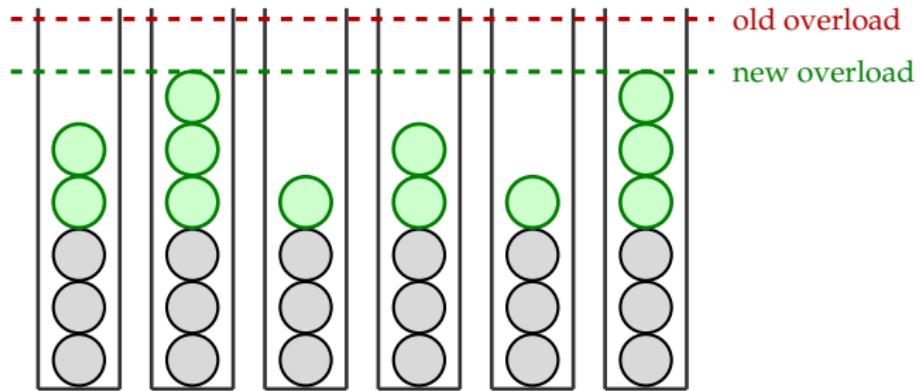
1. **Slice** off the jagged surface

SLICE AND SPREAD



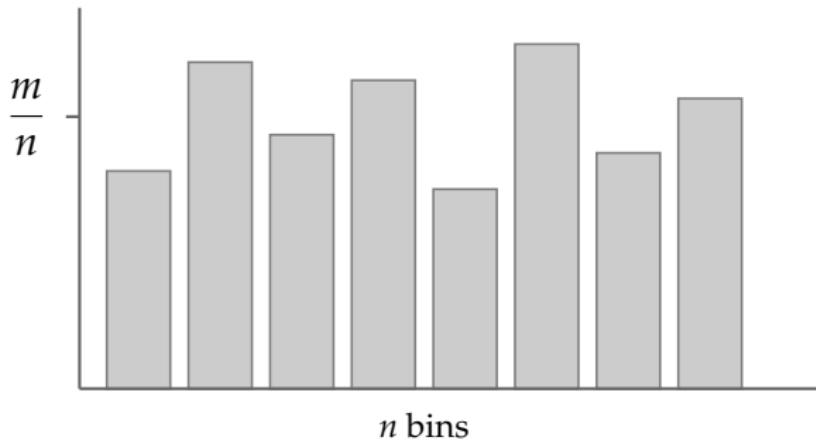
1. **Slice** off the jagged surface
2. **Spread** balls to their second-choice bins

SLICE AND SPREAD

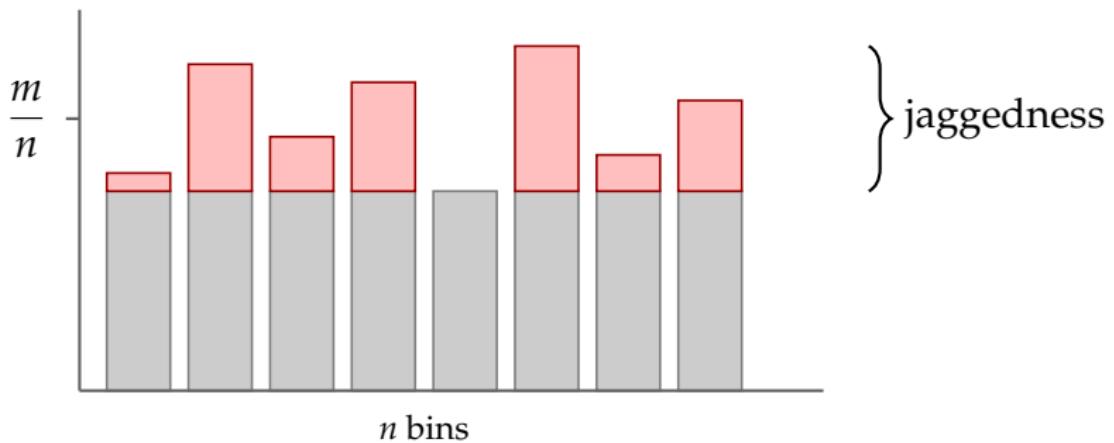


1. **Slice** off the jagged surface
2. **Spread** balls to their second-choice bins

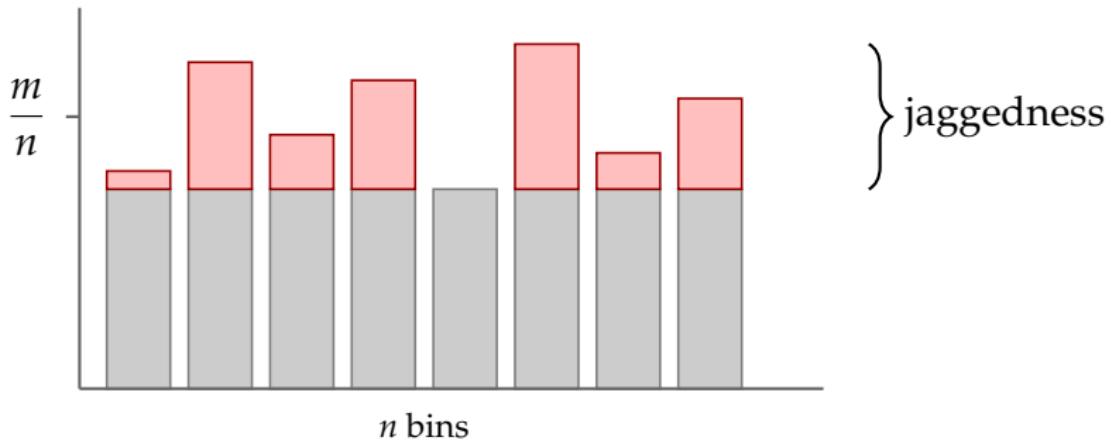
SLICE AND SPREAD REDUCES OVERLOAD



SLICE AND SPREAD REDUCES OVERLOAD

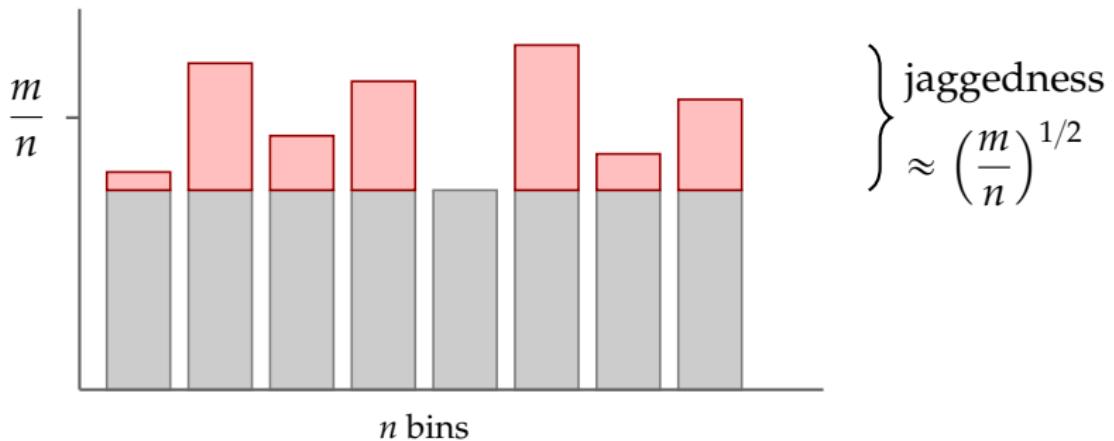


SLICE AND SPREAD REDUCES OVERLOAD



Q: How much is the jaggedness?

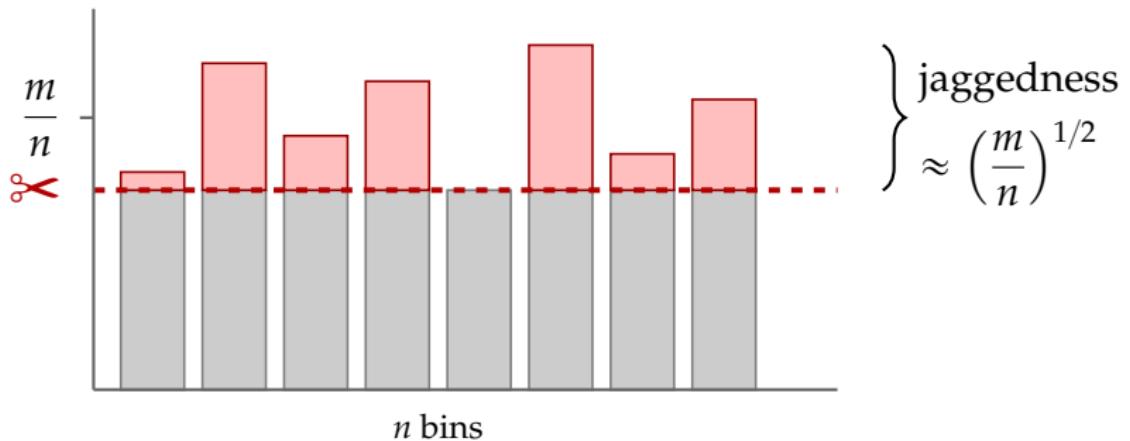
SLICE AND SPREAD REDUCES OVERLOAD



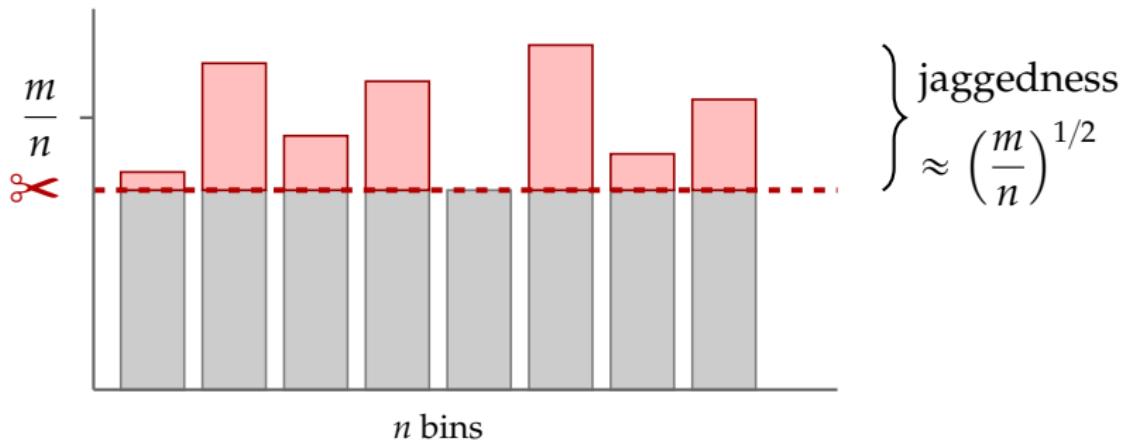
Q: How much is the jaggedness?

A: For $m \gg n$, the jaggedness is $\approx (m/n)^{1/2}$ (whp in n)

SLICE AND SPREAD REDUCES OVERLOAD

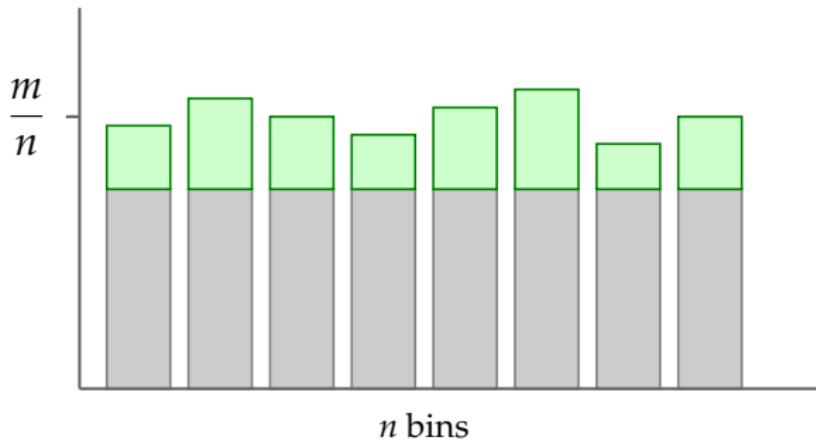


SLICE AND SPREAD REDUCES OVERLOAD



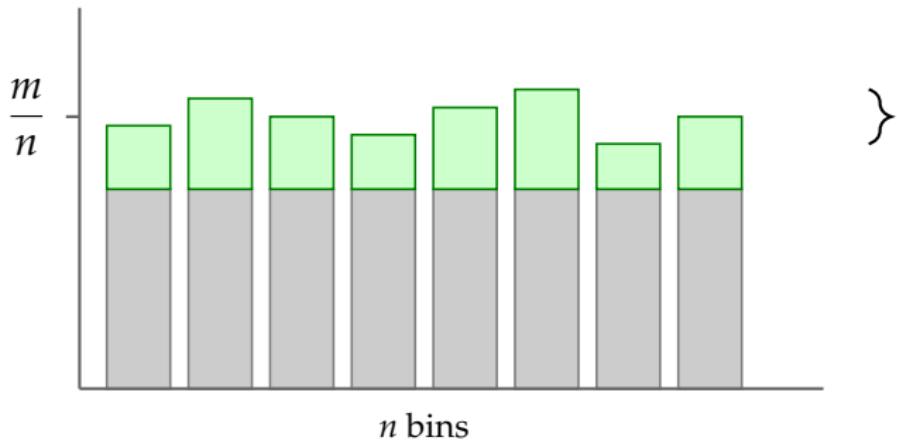
Balls sliced away: $m' \approx n \cdot (m/n)^{1/2} = (mn)^{1/2}$

SLICE AND SPREAD REDUCES OVERLOAD



Balls sliced away: $m' \approx n \cdot (m/n)^{1/2} = (mn)^{1/2}$

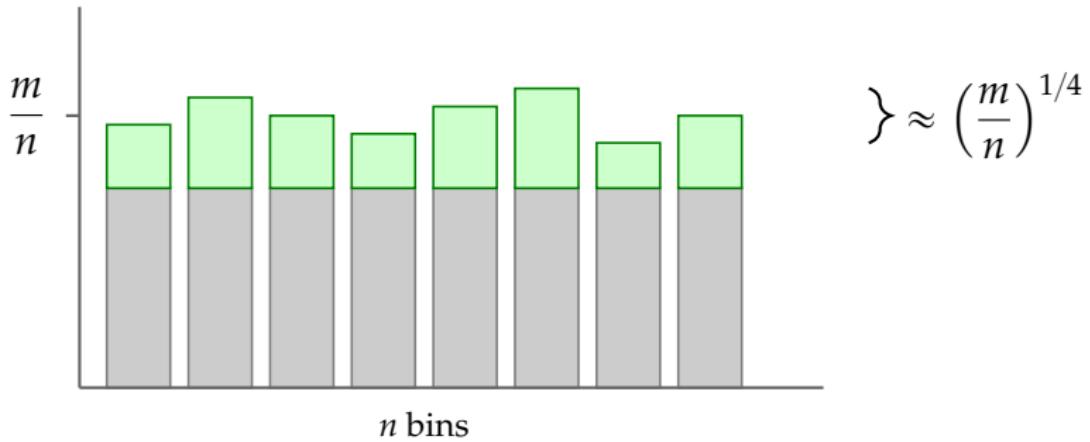
SLICE AND SPREAD REDUCES OVERLOAD



Balls sliced away: $m' \approx n \cdot (m/n)^{1/2} = (mn)^{1/2}$

Q: What is the new jaggedness?

SLICE AND SPREAD REDUCES OVERLOAD



Balls sliced away: $m' \approx n \cdot (m/n)^{1/2} = (mn)^{1/2}$

Q: What is the new jaggedness?

A: For $m' \gg n$, the new jaggedness is $(m'/n)^{1/2} = (m/n)^{1/4}$

SLICE AND SPREAD

- Overload: $(m/n)^{1/2} \rightarrow (m/n)^{1/4}$

SLICE AND SPREAD

- ▶ Overload: $(m/n)^{1/2} \rightarrow (m/n)^{1/4}$
- ▶ History independent?

SLICE AND SPREAD

- ▶ Overload: $(m/n)^{1/2} \rightarrow (m/n)^{1/4}$
- ▶ History independent? Yes!

SLICE AND SPREAD

- ▶ Overload: $(m/n)^{1/2} \rightarrow (m/n)^{1/4}$
- ▶ History independent? Yes!
- ▶ Recourse:

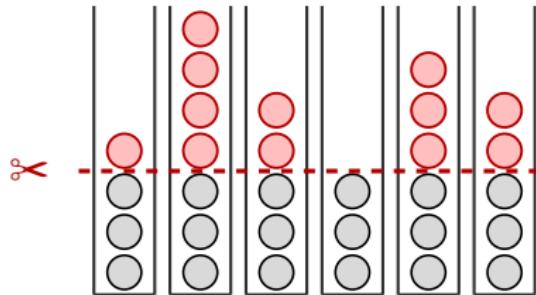
SLICE AND SPREAD

- ▶ Overload: $(m/n)^{1/2} \rightarrow (m/n)^{1/4}$
- ▶ History independent? Yes!
- ▶ Recourse: 1

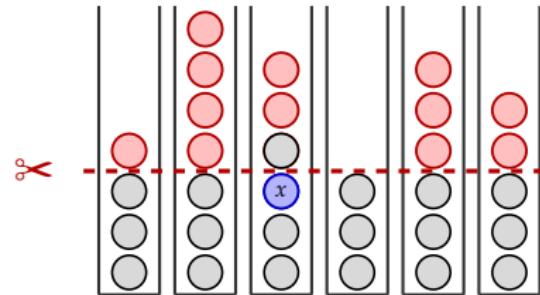
SLICE AND SPREAD

- ▶ Overload: $(m/n)^{1/2} \rightarrow (m/n)^{1/4}$
- ▶ History independent? Yes!
- ▶ Recourse: 1

Example



Computing A_S

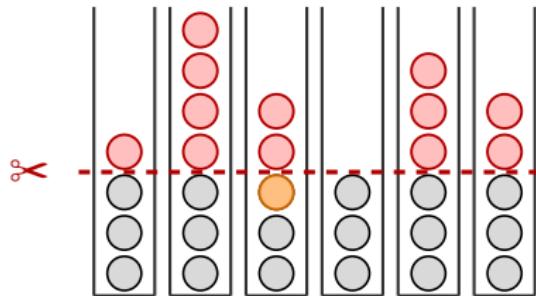


Computing $A_{S \cup \{x\}}$

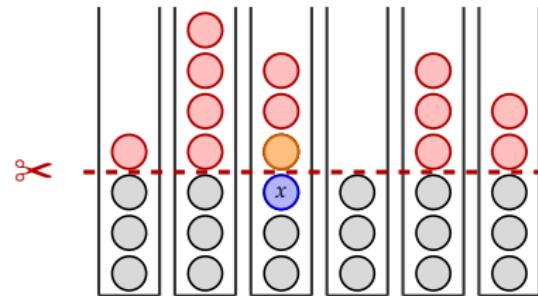
SLICE AND SPREAD

- ▶ Overload: $(m/n)^{1/2} \rightarrow (m/n)^{1/4}$
- ▶ History independent? Yes!
- ▶ Recourse: 1

Example

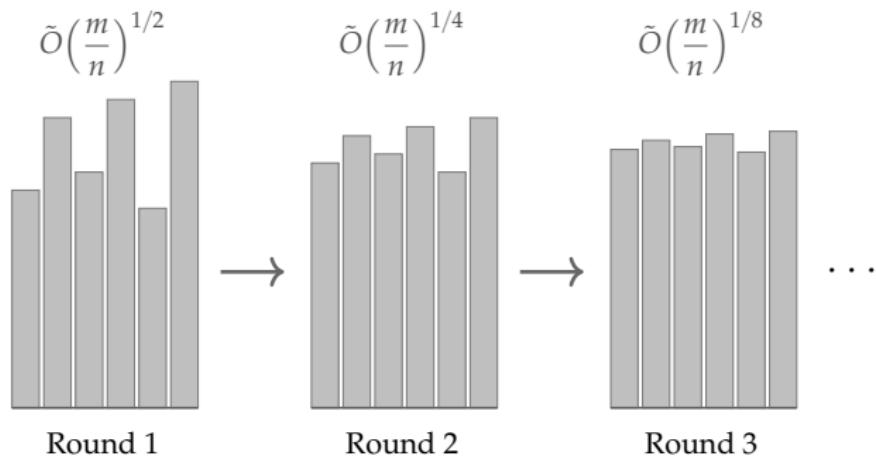


Computing A_S

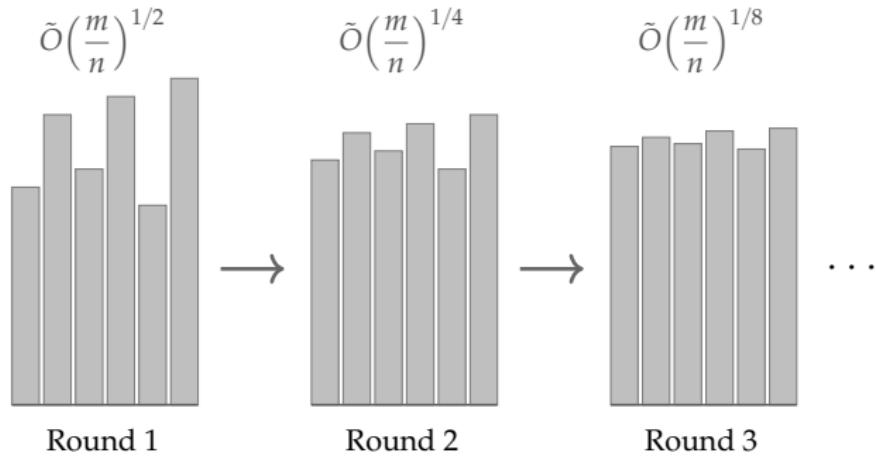


Computing $A_{S \cup \{x\}}$

REPEATEDLY SLICING AND SPREADING



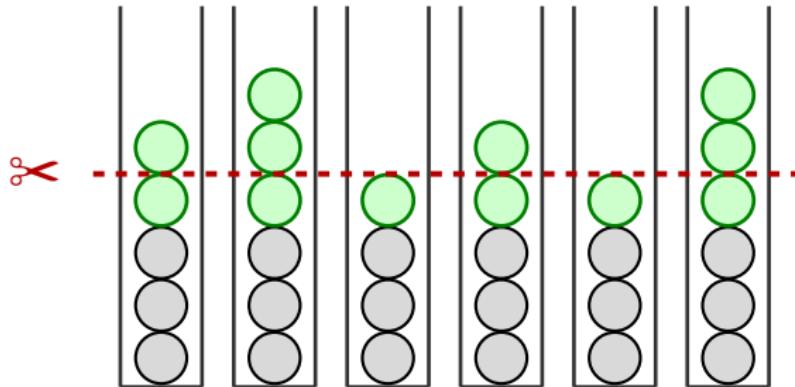
REPEATEDLY SLICING AND SPREADING



After $O(\log \log(m/n))$ rounds...

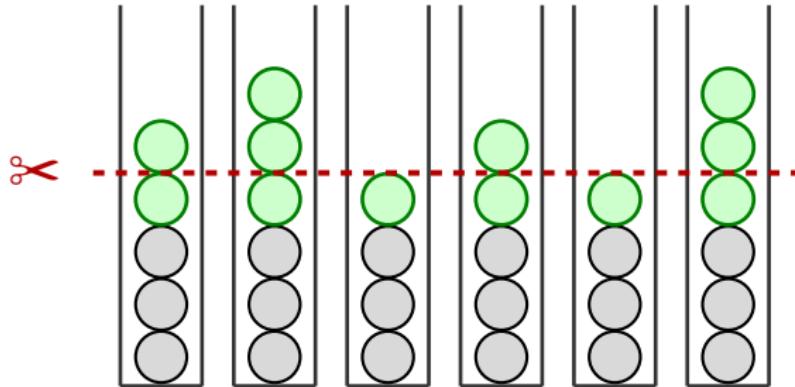
- ▶ Overload = $O(1)$?
- ▶ Recourse = $O(\log \log(m/n))$?

ALGORITHMIC QUESTION



Question: Which balls do we slice in round k ?

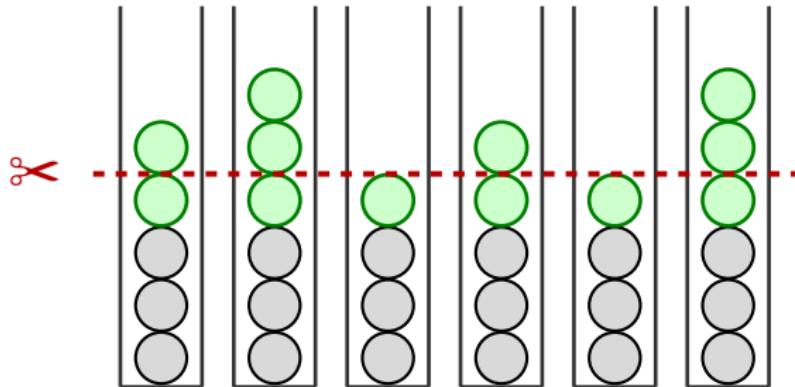
ALGORITHMIC QUESTION



Question: Which balls do we slice in round k ?

- ▶ **Option 1:** Scrape off the top

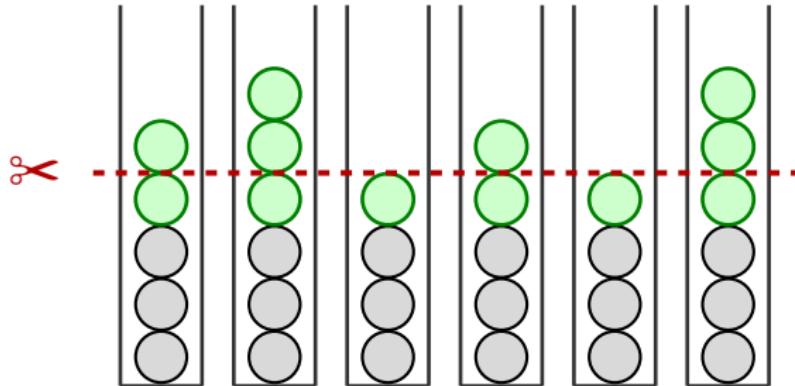
ALGORITHMIC QUESTION



Question: Which balls do we slice in round k ?

- ▶ **Option 1:** Scrape off the top ✗ Reuses stale randomness

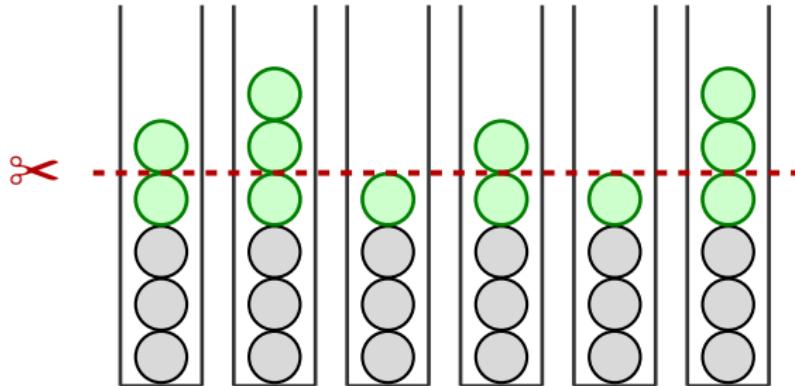
ALGORITHMIC QUESTION



Question: Which balls do we slice in round k ?

- ▶ **Option 1:** Scrape off the top ✗ Reuses stale randomness
- ▶ **Option 2:** Priority queue per bin

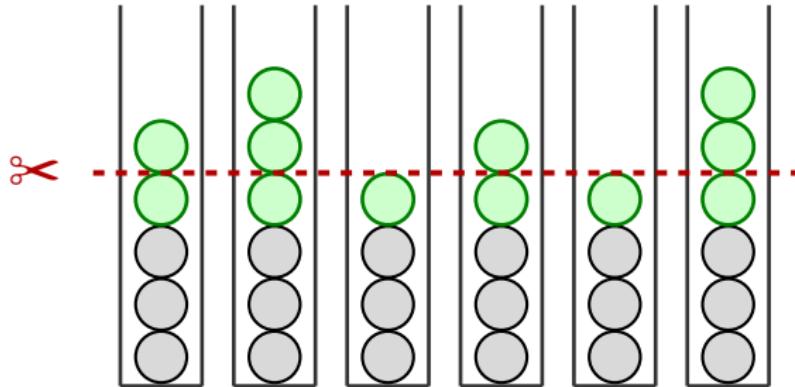
ALGORITHMIC QUESTION



Question: Which balls do we slice in round k ?

- ▶ **Option 1:** Scrape off the top ✗ Reuses stale randomness
- ▶ **Option 2:** Priority queue per bin ✗ Exploding recourse

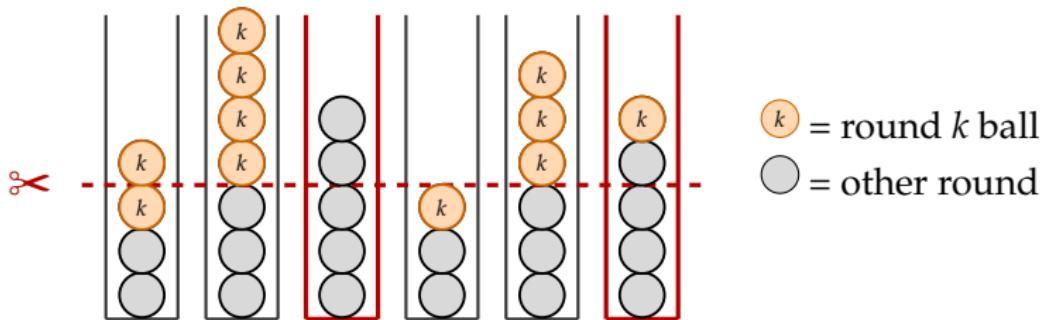
ALGORITHMIC QUESTION



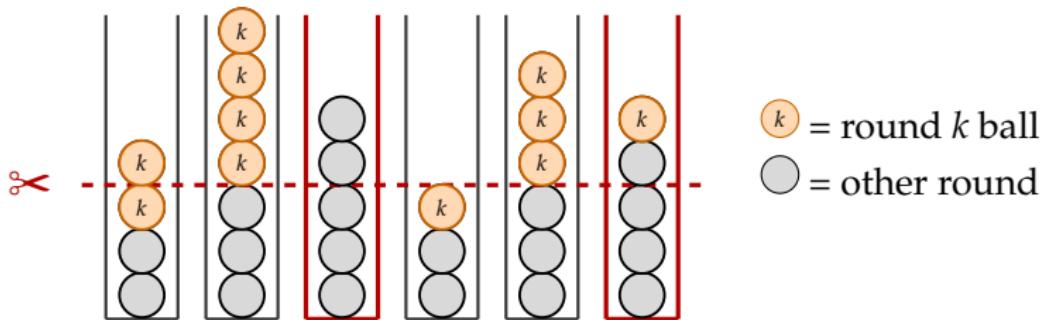
Question: Which balls do we slice in round k ?

- ▶ **Option 1:** Scrape off the top ✗ Reuses stale randomness
- ▶ **Option 2:** Priority queue per bin ✗ Exploding recourse
- ▶ **Our approach:** Assign every ball a random **round number**, only choose from the balls with round number k

CHALLENGE 1: SLICING FAILURES



CHALLENGE 1: SLICING FAILURES



$\text{---} \text{---}$

$\text{---} \text{---}$

$\text{---} \text{---}$

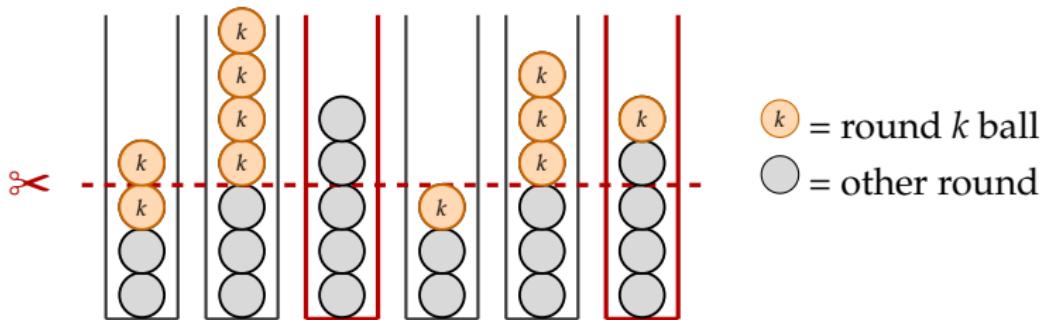
$\text{---} \text{---}$

$\text{---} \text{---}$

$\text{---} \text{---}$

Challenge: Some bins may not have enough round- k balls to support slicing.

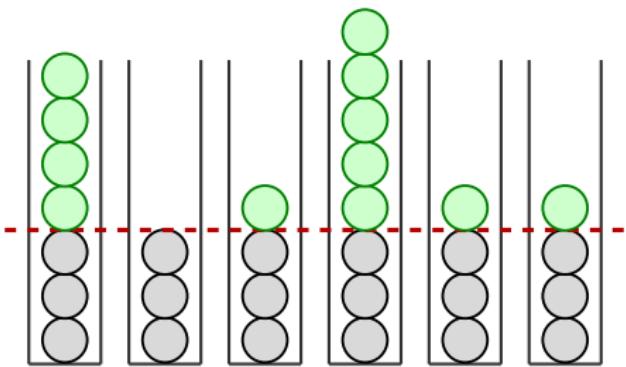
CHALLENGE 1: SLICING FAILURES



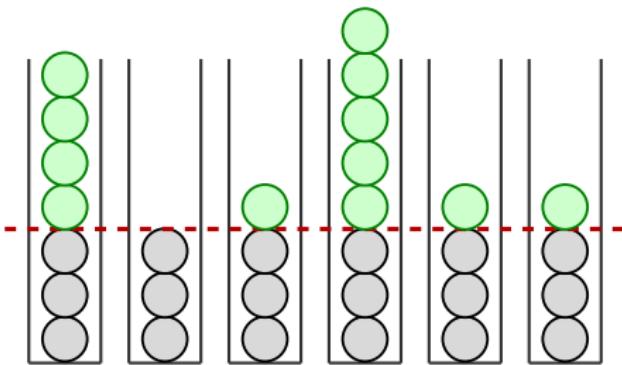
Challenge: Some bins may not have enough round- k balls to support slicing.

Result: We can't slice evenly — the jaggedness remains in those bins.

CHALLENGE 2: SPREADING FAILURES

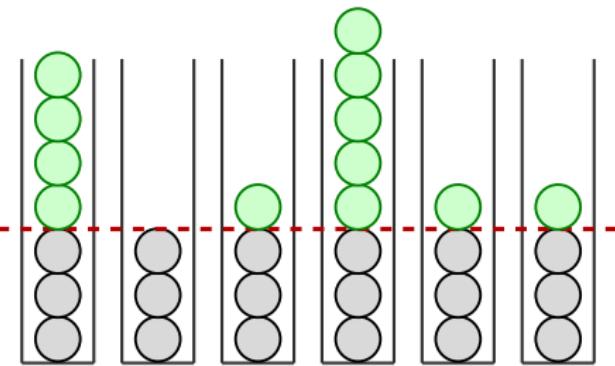


CHALLENGE 2: SPREADING FAILURES



Challenge: The spreading step may distribute balls unevenly — creating new jaggedness.

CHALLENGE 2: SPREADING FAILURES

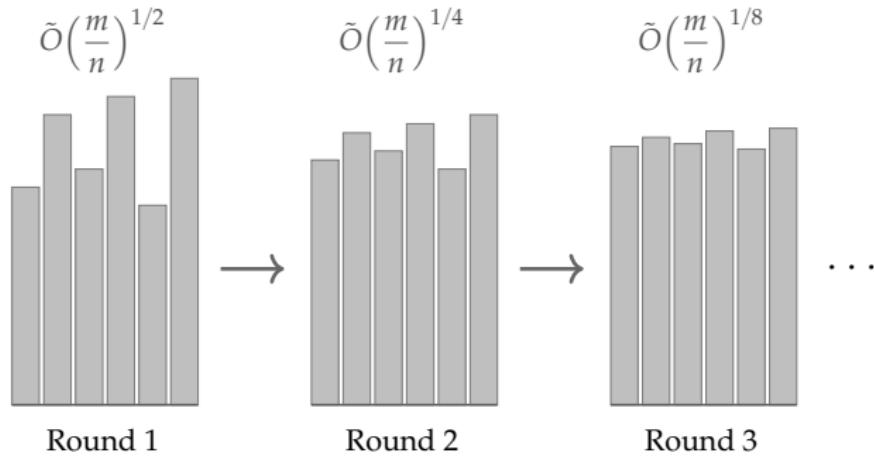


Challenge: The spreading step may distribute balls unevenly — creating new jaggedness.

Dilemma:

- ▶ Slice **more** balls \implies overload may not decrease
- ▶ Slice **fewer** balls \implies jaggedness may not decrease

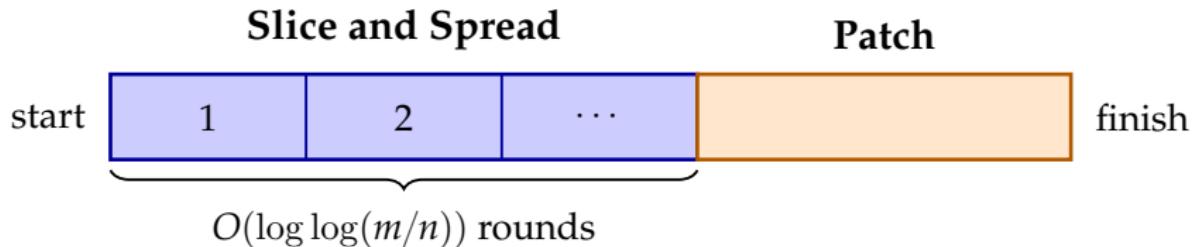
REPEATEDLY SLICING AND SPREADING



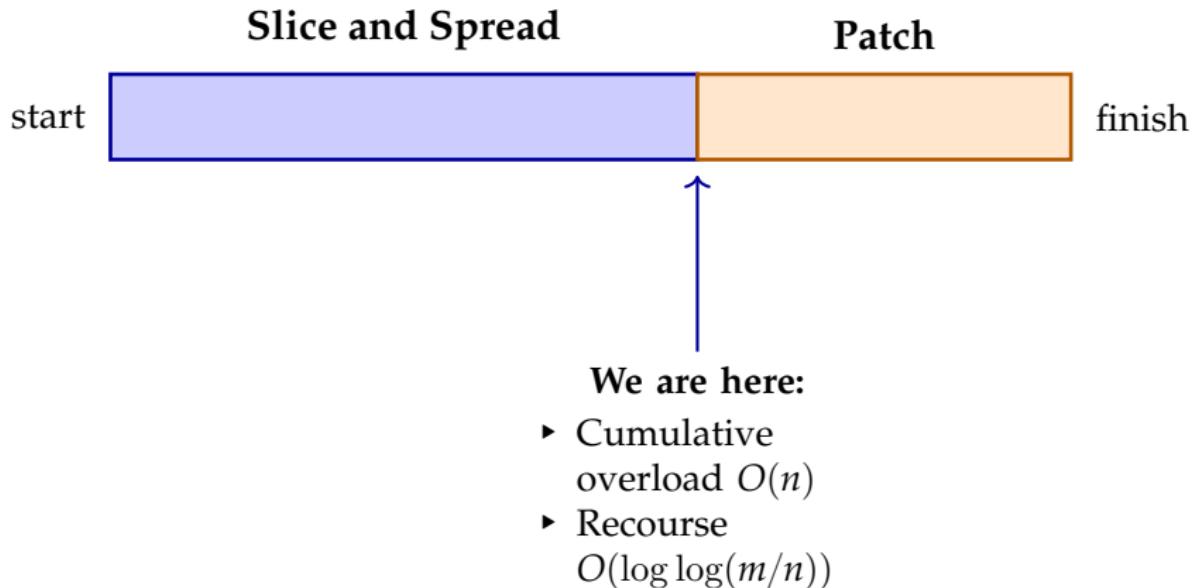
After $O(\log \log(m/n))$ rounds...

- ▶ Overload = $O(1)$ Cumulative overload = $O(n)$
- ▶ Recourse = $O(\log \log(m/n))$

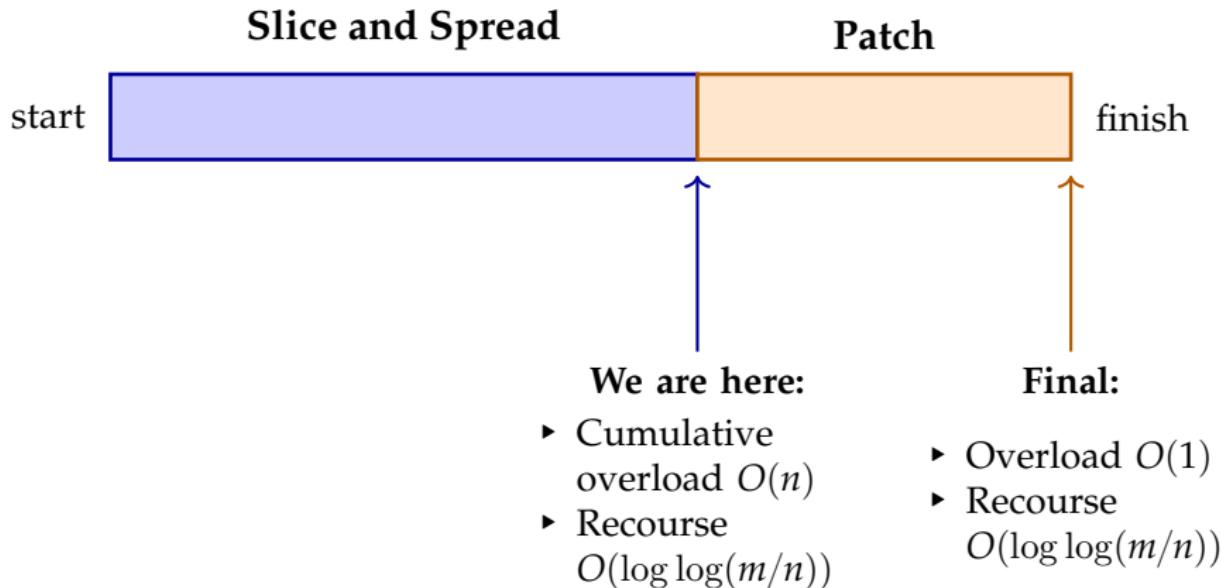
FULL ALGORITHM



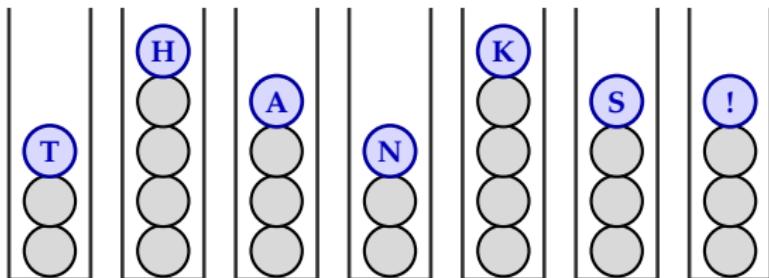
FULL ALGORITHM



FULL ALGORITHM



History-Independent Load Balancing



Michael A. Bender
Stony Brook University

William Kuszmaul
CMU

Elaine Shi
CMU

Rose Silver
CMU