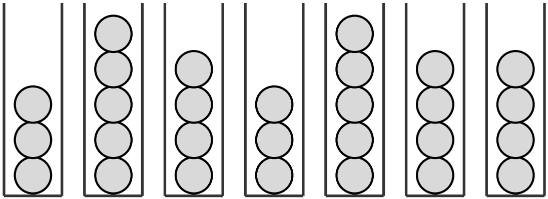
History-Independent Load Balancing



Michael A. Bender — Stony Brook University

William Kuszmaul — CMU

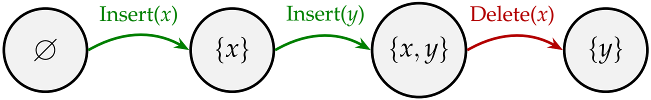
Elaine Shi — CMU

Rose Silver — CMU

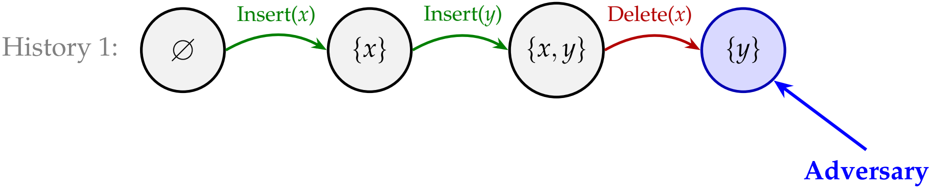
History-Independent Data Structures

History-Independent Data Structures

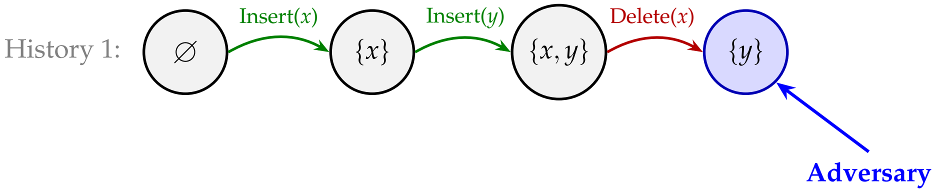
History 1:



History-Independent Data Structures



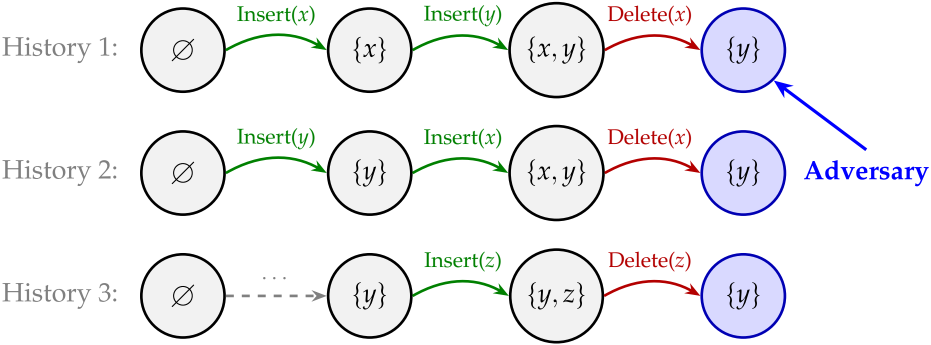
History-Independent Data Structures



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

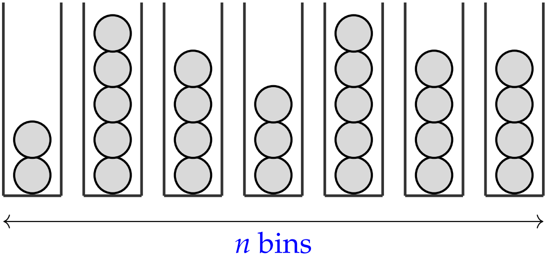
Hash tables, trees, memory allocation, PMAs, graph algorithms, cache-oblivious data structures, and more.

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

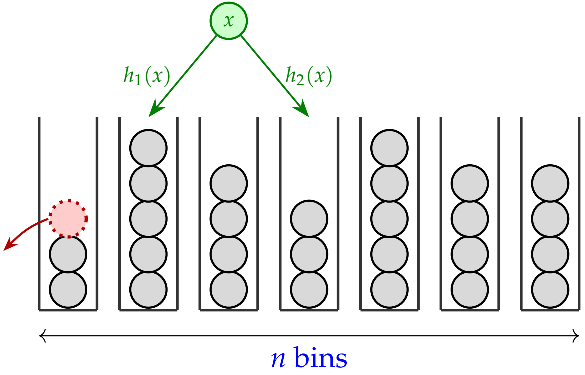
Yet some basic questions remain open.

This work: History-Independent Load Balancing

Two-Choice Load Balancing

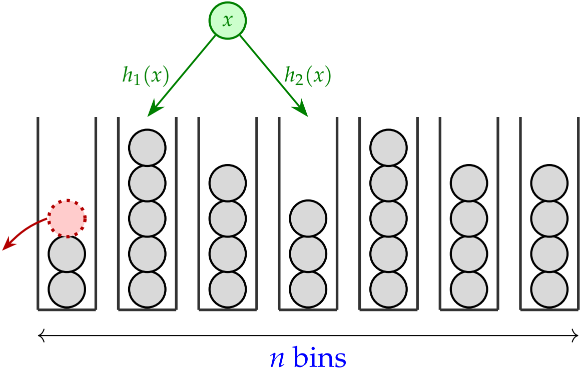


Two-Choice Load Balancing



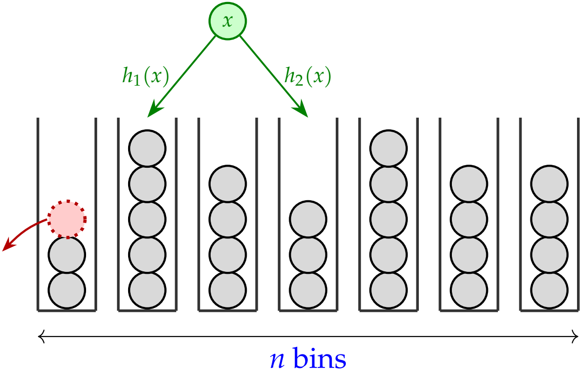
- Balls are inserted/deleted, with up to m present at a time.

Two-Choice Load Balancing



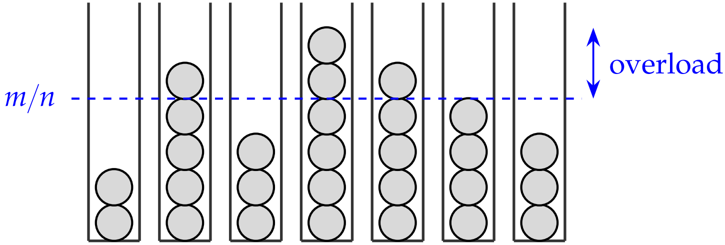
- Balls are inserted/deleted, with up to m present at a time.
- Each ball has two random bins where it can go.

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.

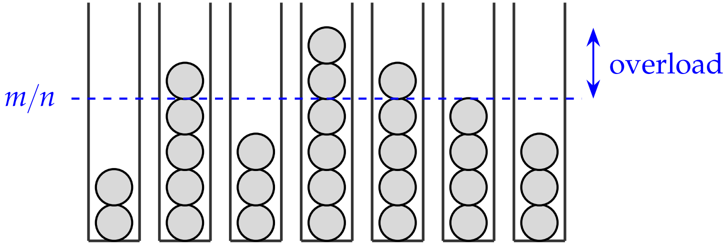
Two Goals



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:

- For all sets S of balls: If the current set is S, then the assignment is always A_S.

Putting it All Together

History-Independent Load Balancing:

- For all sets S of balls: If the current set is S, then the assignment is always A_S.

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

Putting it All Together

History-Independent Load Balancing:

- For all sets S of balls: If the current set is S, then the assignment is always 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

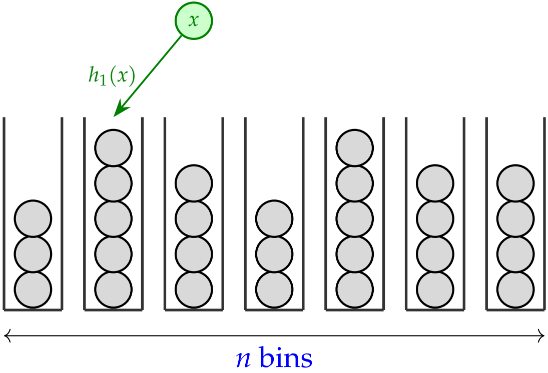
A Simple Warmup

Theorem: There exists a history-independent solution with:

- High-probability overload $\cancel{O(1)}$ $O(\log \log n)$.
- Expected recourse $\cancel{O(\log \log(m/n))}$ $O(m/n)$.

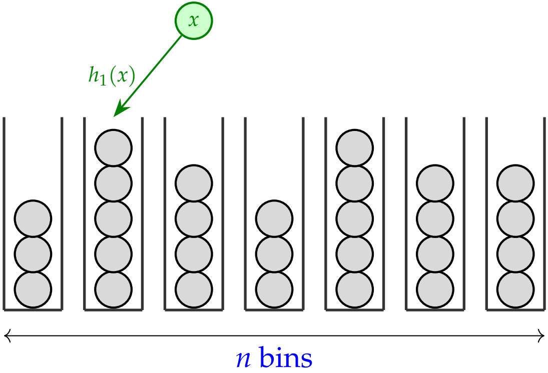
Baseline 1: The Single-Choice Strategy

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



Baseline 1: The Single-Choice Strategy

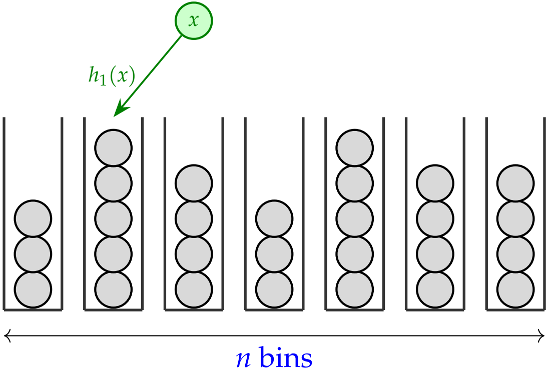
To insert a ball x, just put it in bin $h_1(x)$:



- This is history-independent ✓

Baseline 1: The Single-Choice Strategy

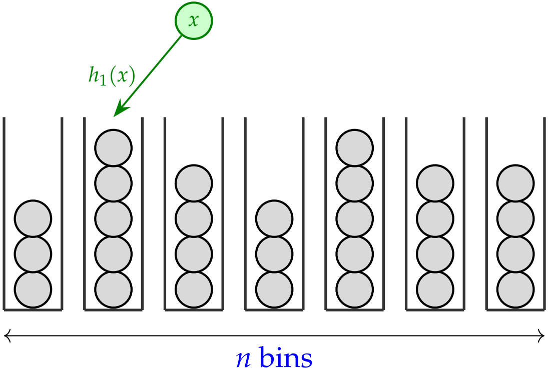
To insert a ball x, just put it in bin $h_1(x)$:



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

Baseline 1: The Single-Choice Strategy

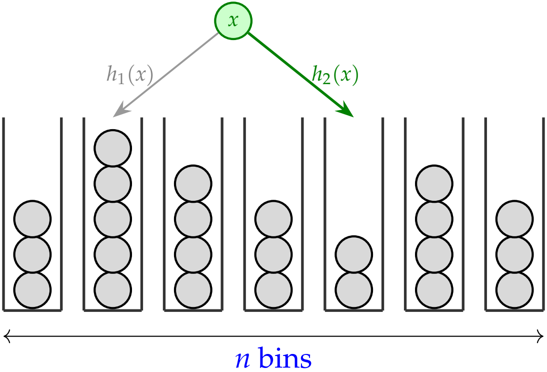
To insert a ball x, just put it in bin $h_1(x)$:



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

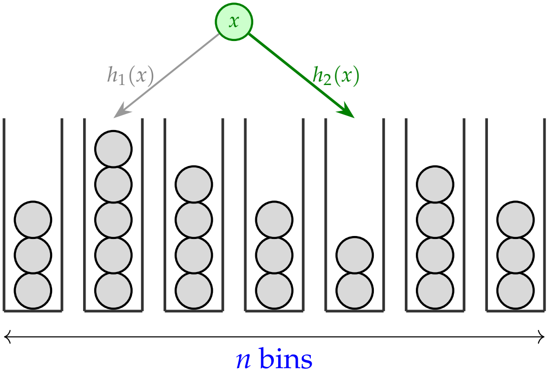
Baseline 2: Greedy Insertions

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



Baseline 2: Greedy Insertions

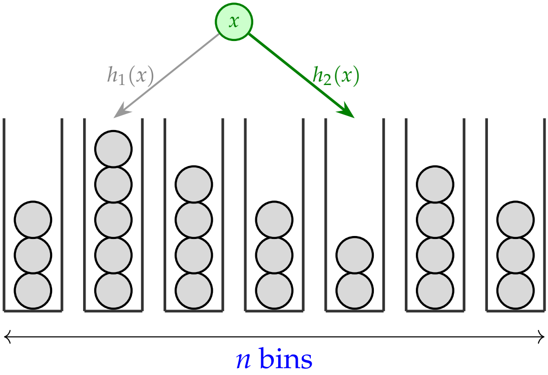
To insert a ball x, put it in the emptier of its choices:



- This is **not** history-independent ✗

Baseline 2: Greedy Insertions

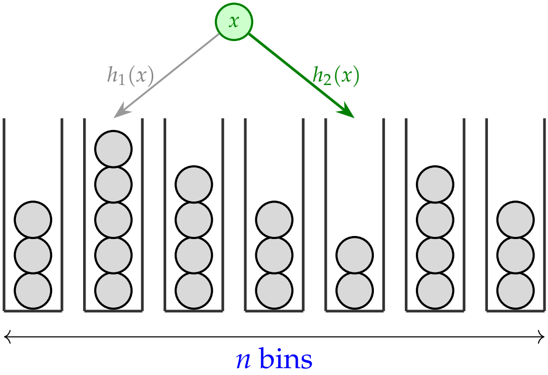
To insert a ball x, put it in the emptier of its choices:



- 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:

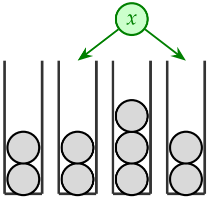


- This is **not** history-independent ✗
- The recourse is 0 ✓
- In the insertion-only case, the overload is $O(\log \log n)$ ✓

[Berenbrink, Czumaj, Steger, and Vöcking '00]

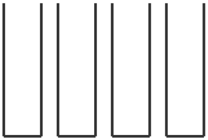
Warmup: History-Independent Greedy

WARMUP: HISTORY-INDEPENDENT GREEDY



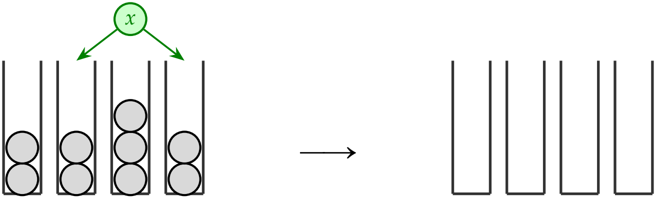
Current state A_S

$\longrightarrow$



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

Warmup: History-Independent Greedy

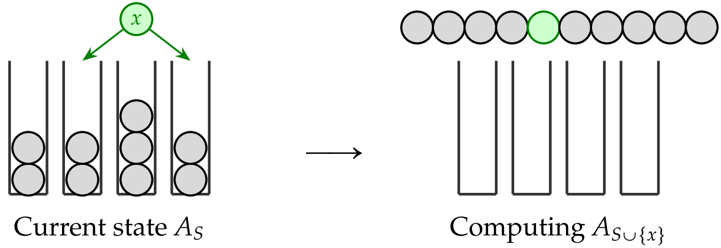


Current state A_S Computing $A_{S \cup \{x\}}$

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

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

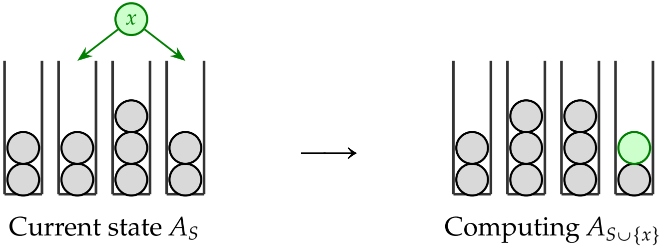
Warmup: History-Independent Greedy



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

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

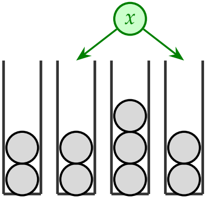
Warmup: History-Independent Greedy



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

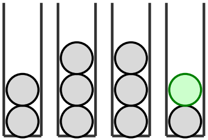
1. Empty out the bins.
2. Sort the balls in $S \cup \{x\}$ to get $x_1, x_2, \ldots$
3. Insert the balls in sorted order using greedy.

Analyzing History-Independent Greedy



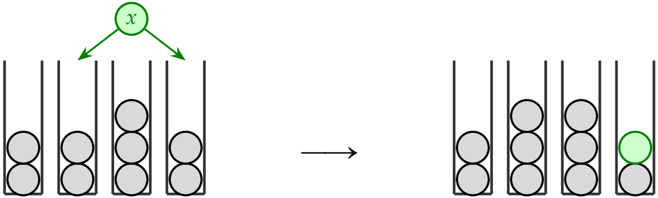
Current state A_S

$\longrightarrow$



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

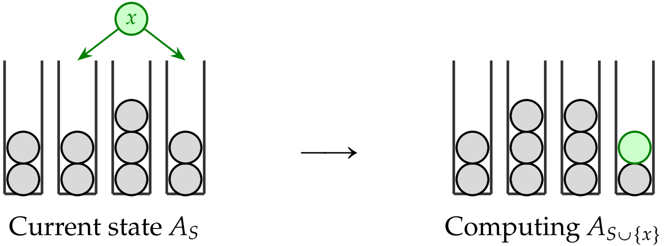
Analyzing History-Independent Greedy



Current state A_S $\longrightarrow$ Computing $A_{S \cup \{x\}}$

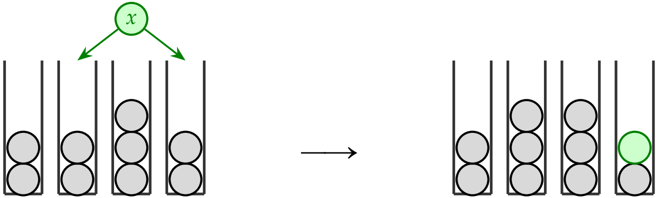
- The algorithm is history independent ✓

Analyzing History-Independent Greedy



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

Analyzing History-Independent Greedy

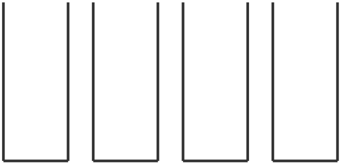


Current state A_S $\longrightarrow$ Computing $A_{S \cup \{x\}}$

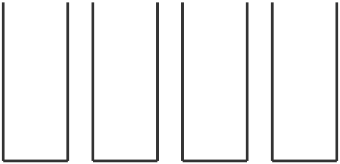
- The algorithm is history independent ✓
- The overload is $O(\log \log n)$ ✓
- What is the recourse?

Analyzing the Recourse

Recourse $= 0$



Computing A_S

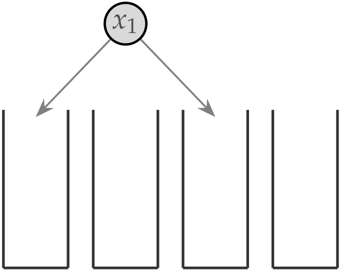


Computing $A_{S \cup \{x\}}$

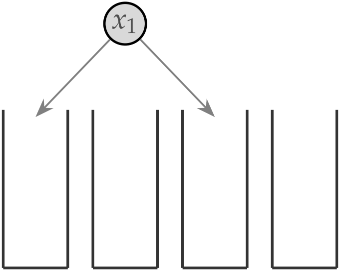
How many balls change assignments between A_S and $A_{S \cup \{x\}}$?

Analyzing the Recourse

Recourse = 0



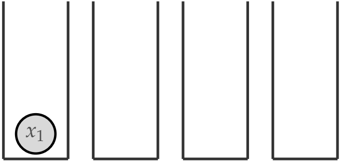
Computing A_S



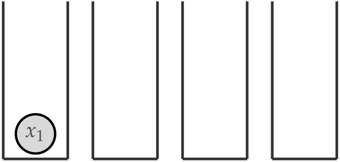
Computing $A_{S \cup \{x\}}$

Analyzing the Recourse

Recourse $= 0$



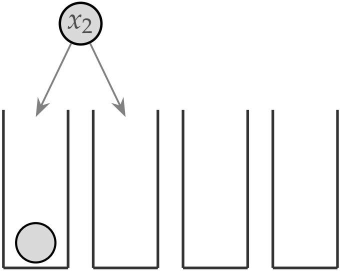
Computing A_S



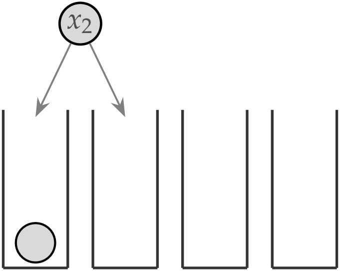
Computing $A_{S \cup \{x\}}$

Analyzing the Recourse

Recourse = 0



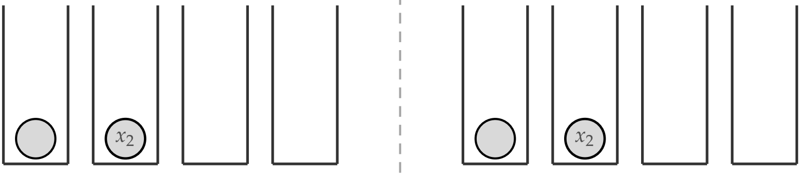
Computing A_S



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

Analyzing the Recourse

Recourse = 0

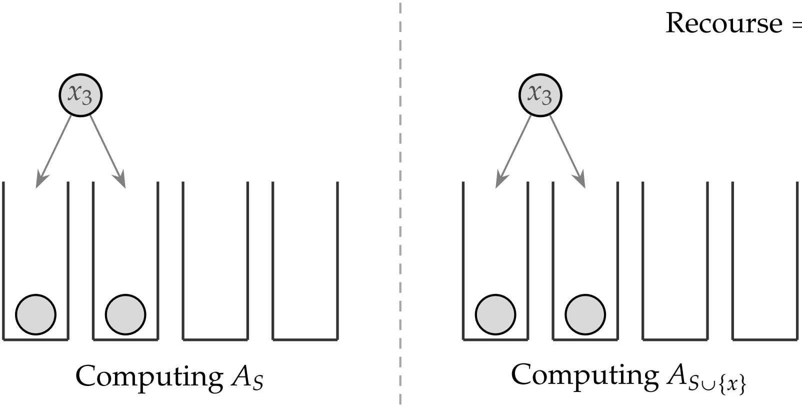


Computing A_S

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

Analyzing the Recourse

Recourse = 0

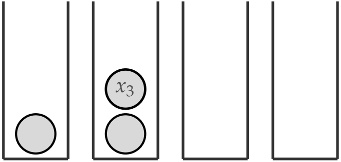


Computing A_S

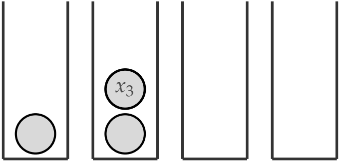
Computing $A_{S \cup \{x\}}$

Analyzing the Recourse

Recourse = 0



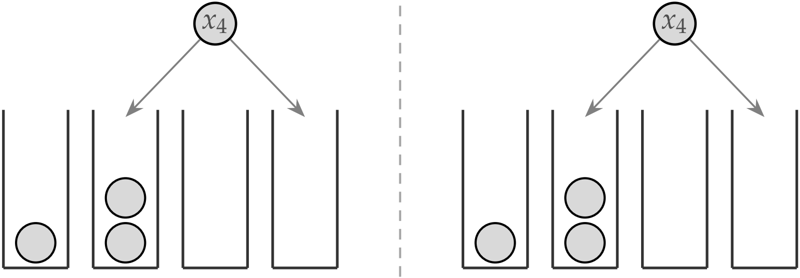
Computing A_S



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

Analyzing the Recourse

Recourse = 0

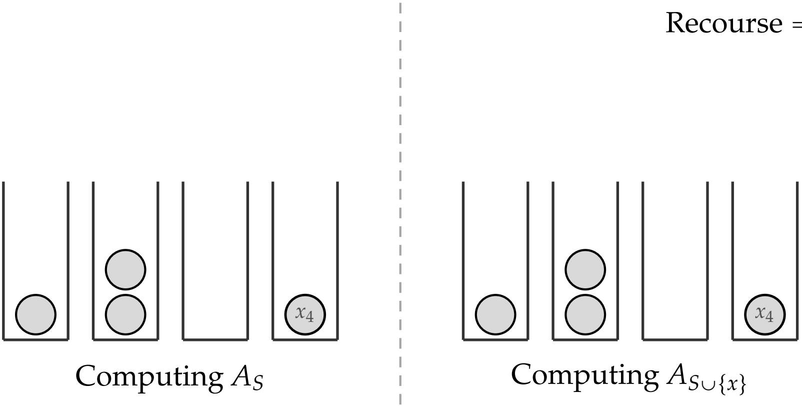


Computing A_S

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

ANALYZING THE RECOURSE

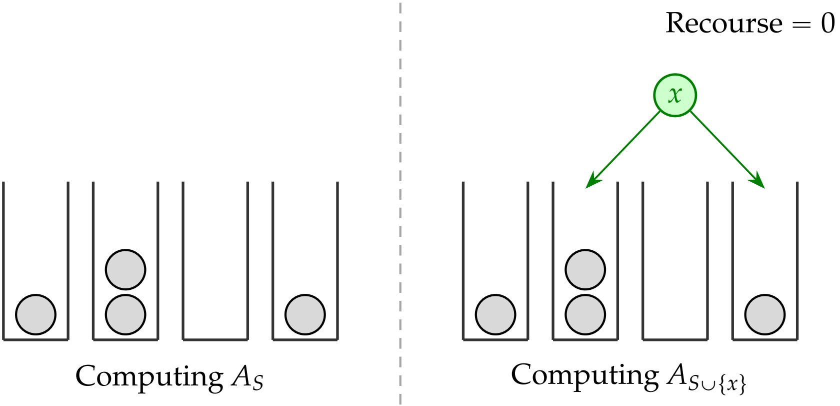
Recourse = 0



Computing A_S

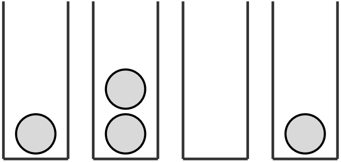
Computing $A_{S \cup \{x\}}$

Analyzing the Recourse

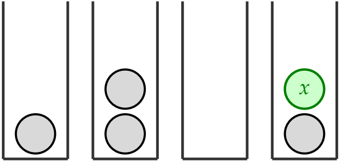


Analyzing the Recourse

Recourse $= 0$



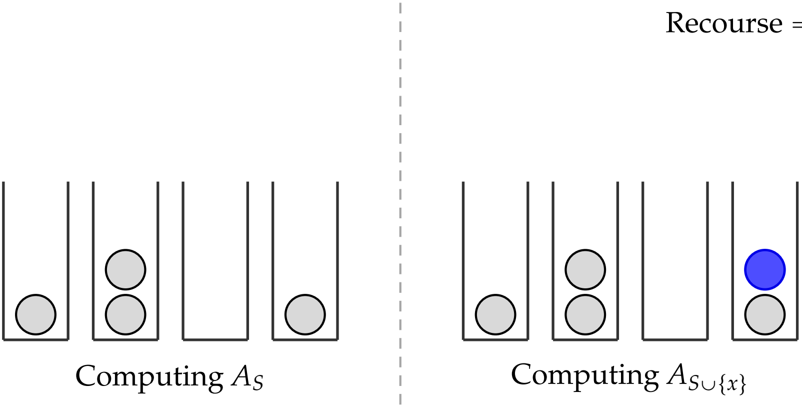
Computing A_S



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

Analyzing the Recourse

Recourse = 0

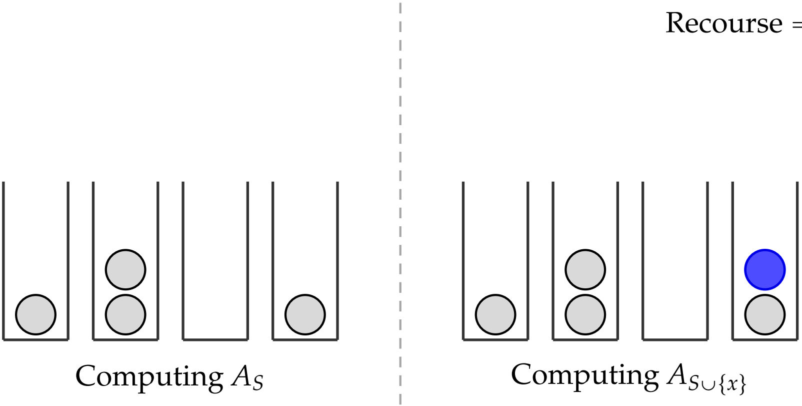


Computing A_S | Computing $A_{S \cup \{x\}}$

Subsequent balls will experience either:

Analyzing the Recourse

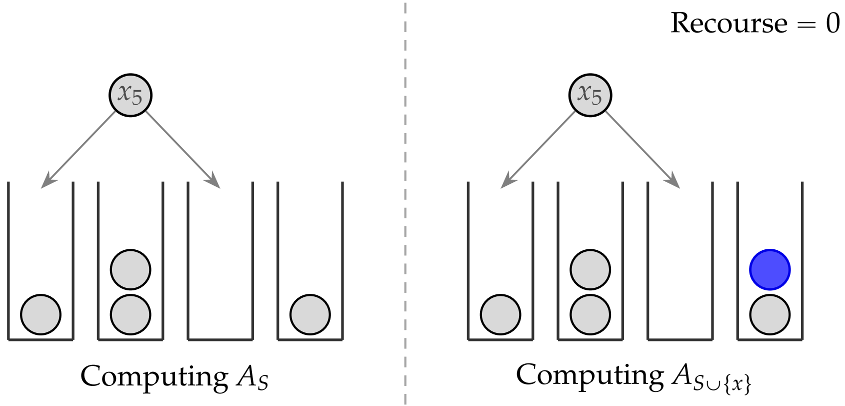
Recourse = 0



Subsequent balls will experience either:

1. No recourse

Analyzing the Recourse

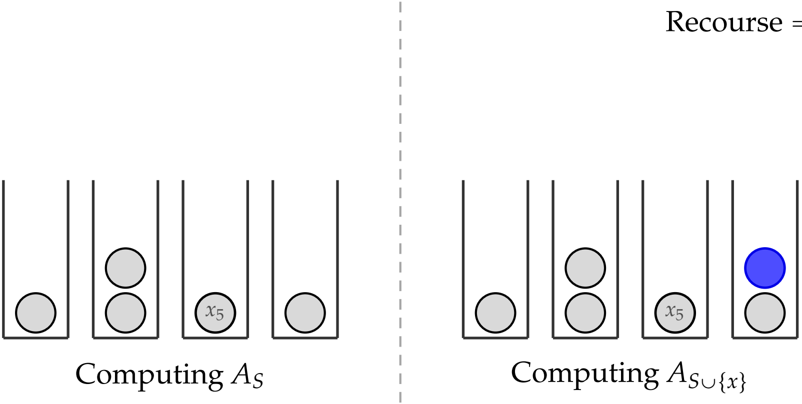


Future insertions will experience either:

1. No recourse

Analyzing the Recourse

Recourse $= 0$



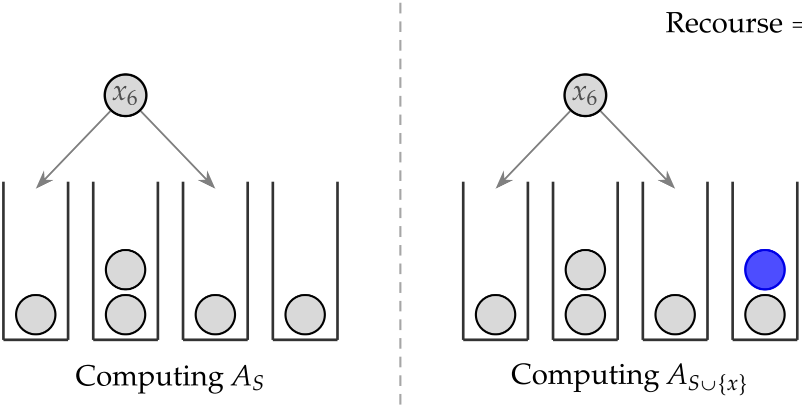
Computing A_S | Computing $A_{S \cup \{x\}}$

Subsequent balls will experience either:

1. No recourse

Analyzing the Recourse

Recourse = 0

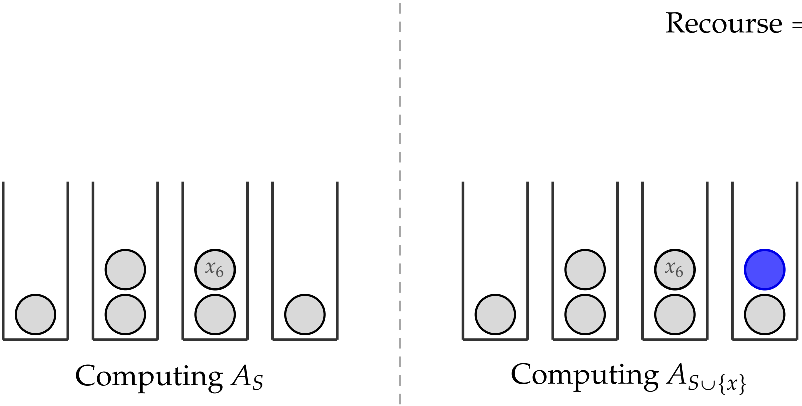


Subsequent balls will experience either:

1. No recourse

Analyzing the Recourse

Recourse = 0

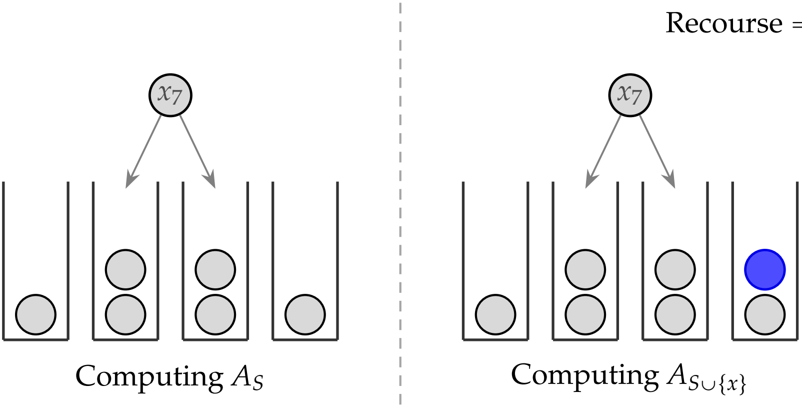


Subsequent balls will experience either:

1. No recourse

Analyzing the Recourse

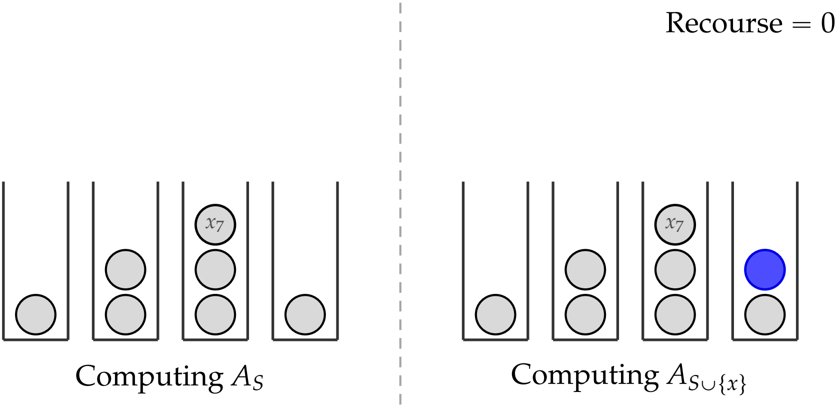
Recourse = 0



Subsequent balls will experience either:

1. No recourse

Analyzing the Recourse

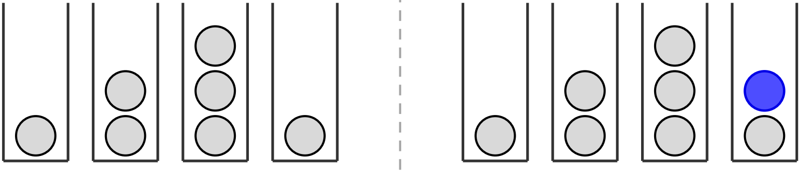


Subsequent balls will experience either:

1. No recourse

Analyzing the Recourse

Recourse = 0



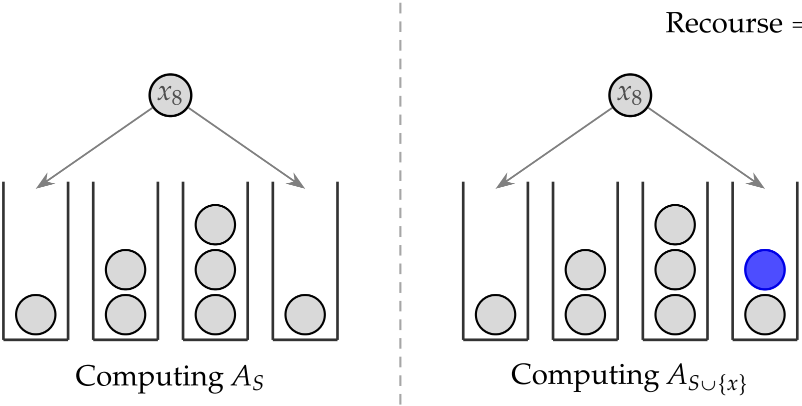
Computing A_S — Computing $A_{S \cup \{x\}}$

Subsequent balls will experience either:

1. No recourse
2. Recourse

Analyzing the Recourse

Recourse = 0

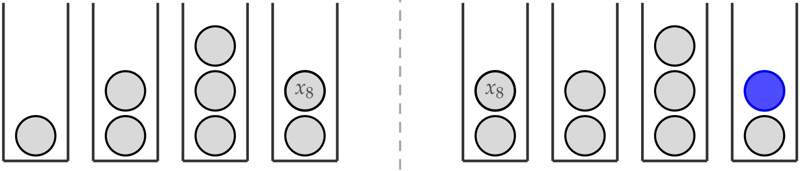


Subsequent balls will experience either:

1. No recourse
2. Recourse

Analyzing the Recourse

Recourse = 1



Computing A_S

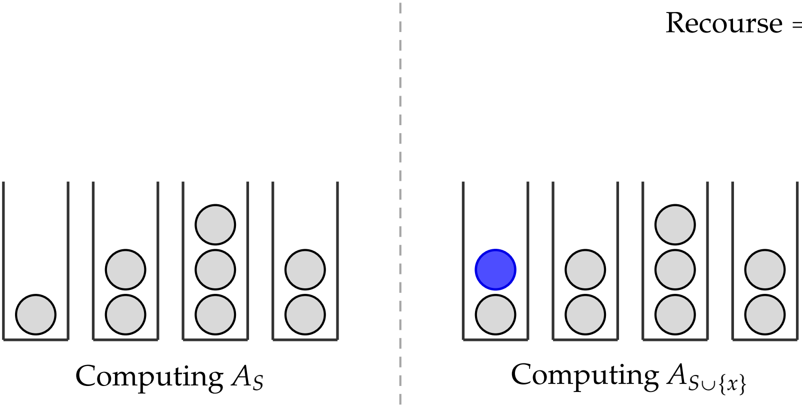
Computing $A_{S \cup \{x\}}$

Subsequent balls will experience either:

1. No recourse
2. Recourse

Analyzing the Recourse

Recourse = 1

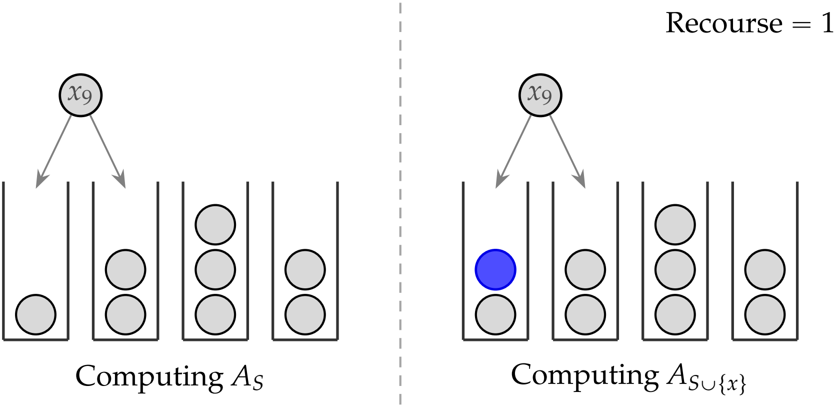


Computing A_S Computing $A_{S \cup \{x\}}$

Subsequent balls will experience either:

1. No recourse
2. Recourse

Analyzing the Recourse

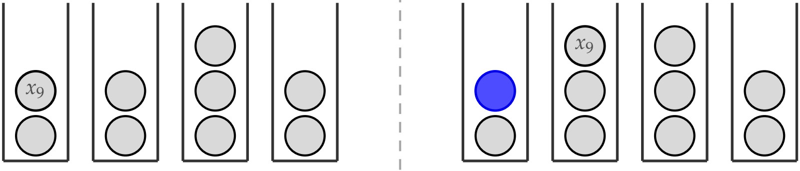


Subsequent balls will experience either:

1. No recourse
2. Recourse

Analyzing the Recourse

Recourse = 2



Computing A_S

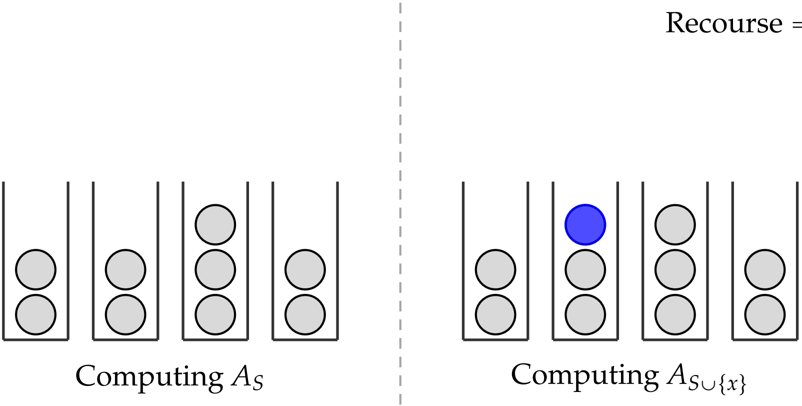
Computing $A_{S\cup\{x\}}$

Subsequent balls will experience either:

1. No recourse
2. Recourse

Analyzing the Recourse

Recourse = 2

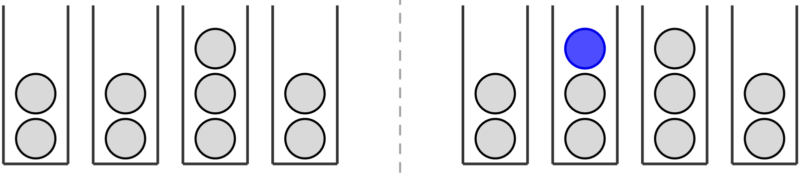


Subsequent balls will experience either:

1. No recourse
2. Recourse

Analyzing the Recourse

Recourse = 2

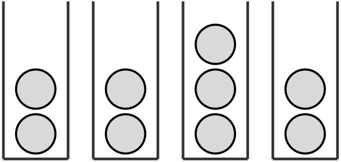


Computing A_S Computing $A_{S \cup \{x\}}$

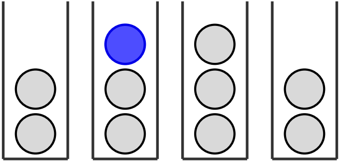
Two key observations:

Analyzing the Recourse

Recourse = 2



Computing A_S

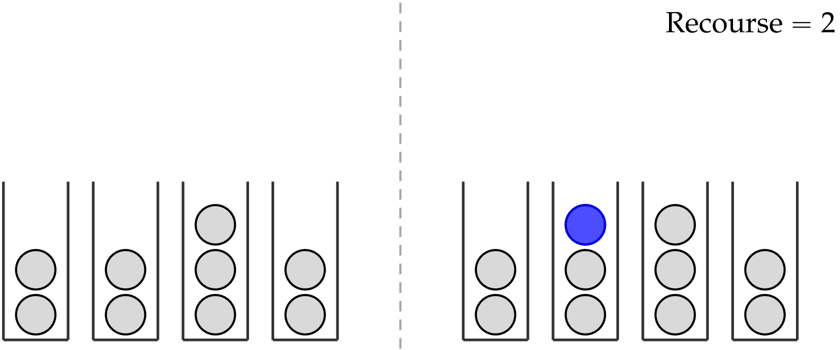


Computing $A_{S \cup \{x\}}$

Two key observations:

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

Analyzing the Recourse



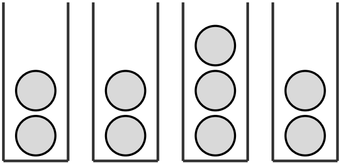
Computing A_S | Computing $A_{S \cup \{x\}}$

Two key observations:

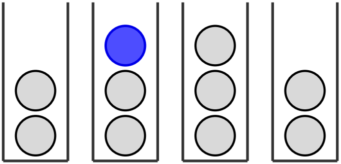
1. There's always one special bin with an extra ball
2. If a ball incurs recourse, one of its choices is the special bin

Analyzing the Recourse

Recourse = 2



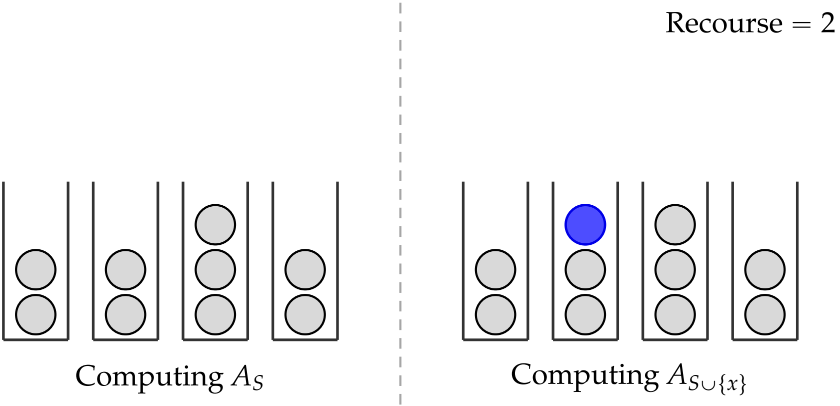
Computing A_S



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

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

Analyzing the Recourse



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

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

A Simple Warmup

Theorem: History-Independent Greedy achieves:

- High-probability overload $\cancel{O(1)}$ $O(\log\log n)$.
- Expected recourse $\cancel{O(\log\log(m/n))}$ $O(m/n)$.

A Simple Warmup

Theorem: History-Independent Greedy achieves:

- High-probability overload $\cancel{O(1)}$ $O(\log \log n)$.
- Expected recourse $\cancel{O(\log \log(m/n))}$ $O(m/n)$.

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

Rest of Talk

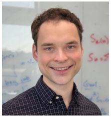
1. A Simple Warmup ✓
2. The Full Algorithm

Part 2: The Full Algorithm

Baking a Cake

Michael

Elaine

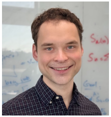


Bill

Baking a Cake

Michael

Elaine



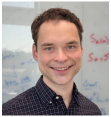
Bill

Baking a Cake

Slice off the
excess flour!

Michael

Elaine



Bill

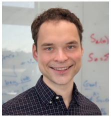
Baking a Cake

Slice off the excess flour!

Michael

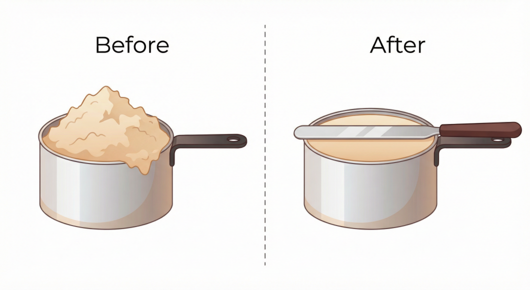
Spread it evenly!

Elaine



Bill

Baking a Cake

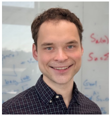


Slice off the excess flour!

Spread it evenly!

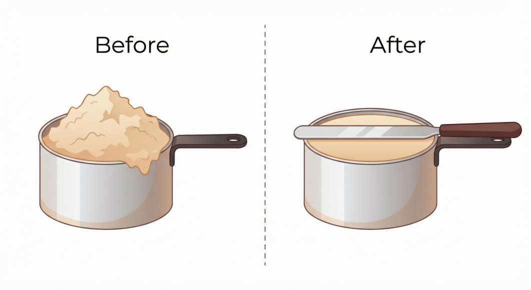
Michael

Elaine



Bill

Baking a Cake



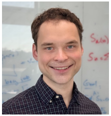
Slice off the excess flour!

Michael

Spread it evenly!

Elaine

You two rehearsed this, didn't you?

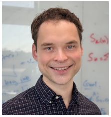


Bill

Baking a Cake

Michael

Elaine



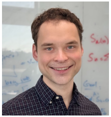
Bill

Baking a Cake

Slice off the
excess frosting!

Michael

Elaine



Bill

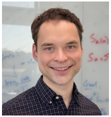
Baking a Cake

Slice off the excess frosting!

Spread it smooth!

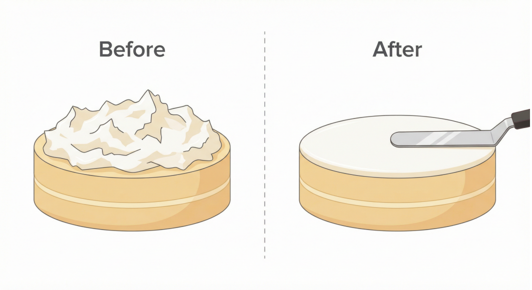
Michael

Elaine



Bill

Baking a Cake

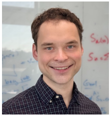


Slice off the excess frosting!

Spread it smooth!

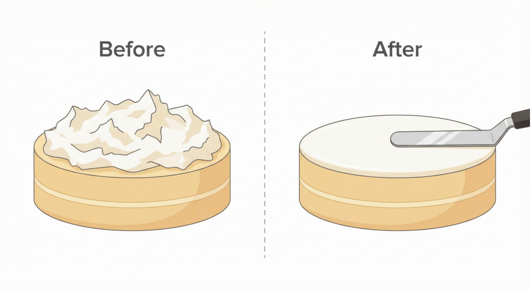
Michael

Elaine



Bill

Baking a Cake



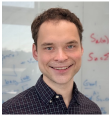
Slice off the excess frosting!

Spread it smooth!

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

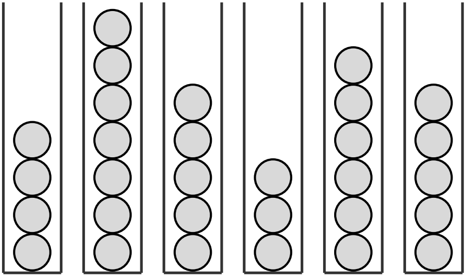
Michael

Elaine

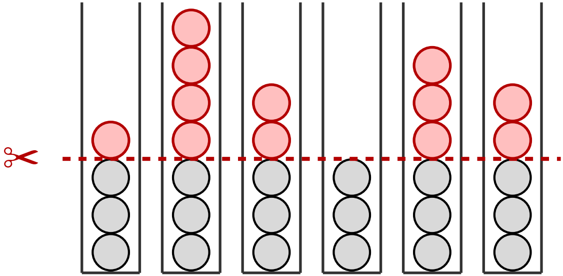


Bill

Slice and Spread

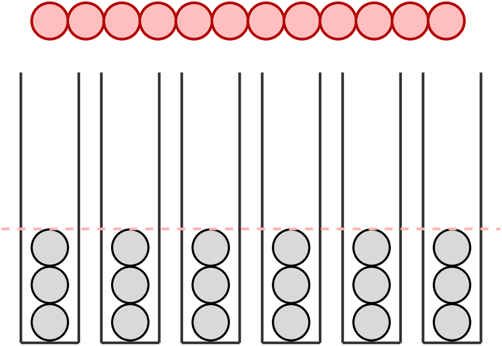


Slice and Spread



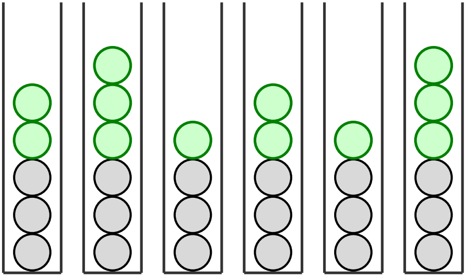
1. **Slice** off the jagged surface

Slice and Spread



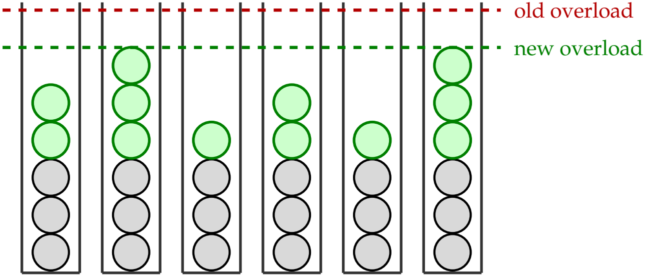
1. **Slice** off the jagged surface

Slice and Spread



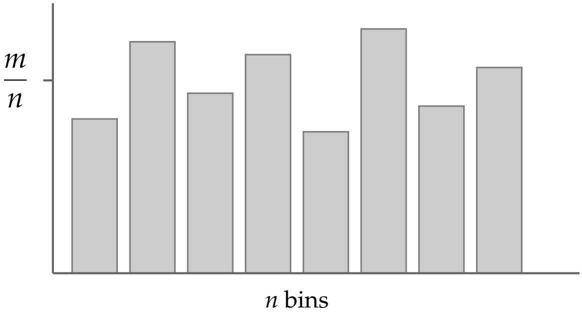
1. **Slice** off the jagged surface
2. **Spread** balls to their second-choice bins

Slice and Spread

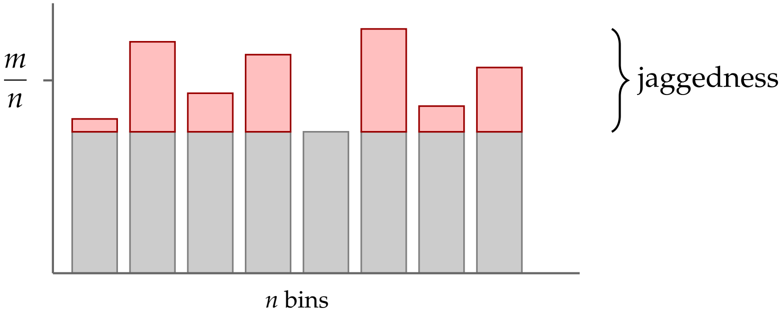


1. **Slice** off the jagged surface
2. **Spread** balls to their second-choice bins

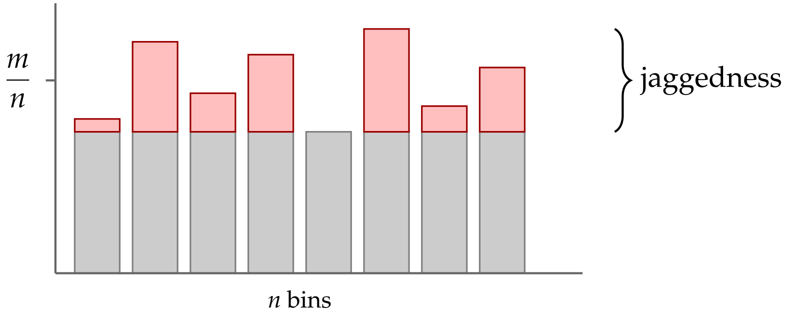
Slice and Spread Reduces Overload



Slice and Spread Reduces Overload

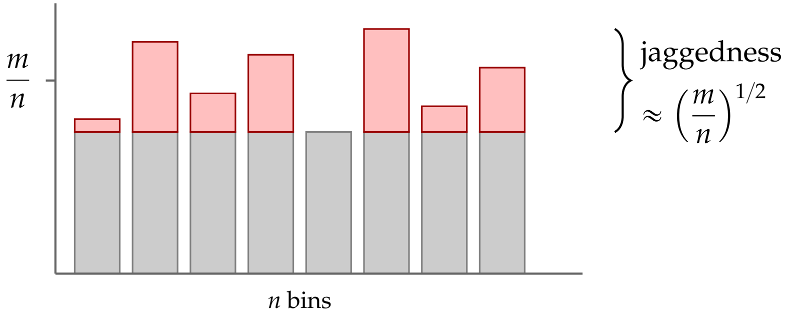


Slice and Spread Reduces Overload



Q: How much is the jaggedness?

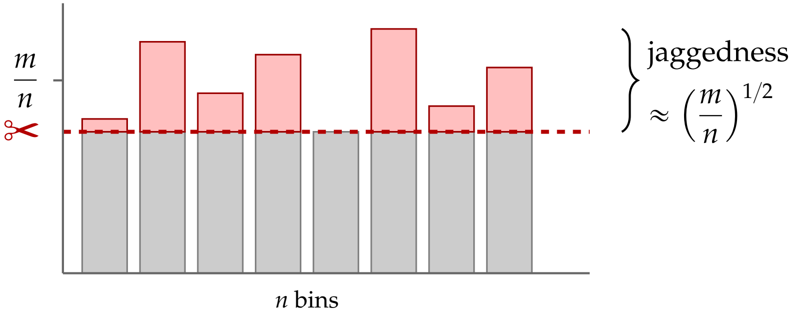
SLICE AND SPREAD REDUCES OVERLOAD



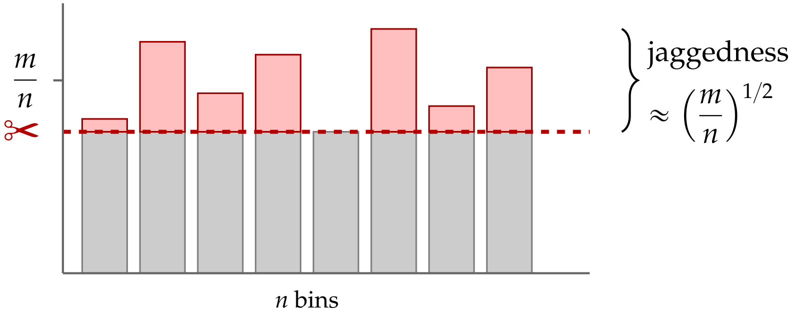
Q: How much is the jaggedness?

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

SLICE AND SPREAD REDUCES OVERLOAD

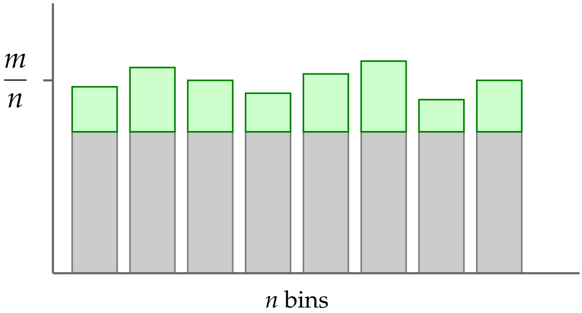


Slice and Spread Reduces Overload



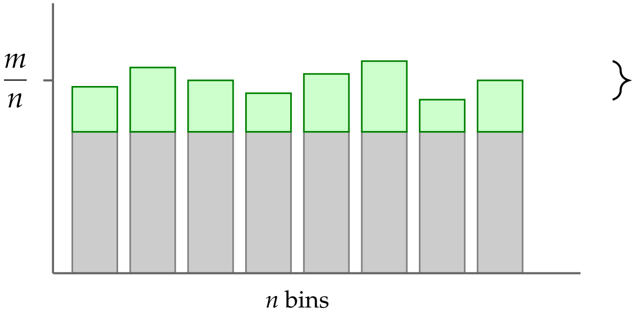
Balls sliced away: $m' \approx n \cdot (m/n)^{1/2} = (mn)^{1/2}$

Slice and Spread Reduces Overload



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

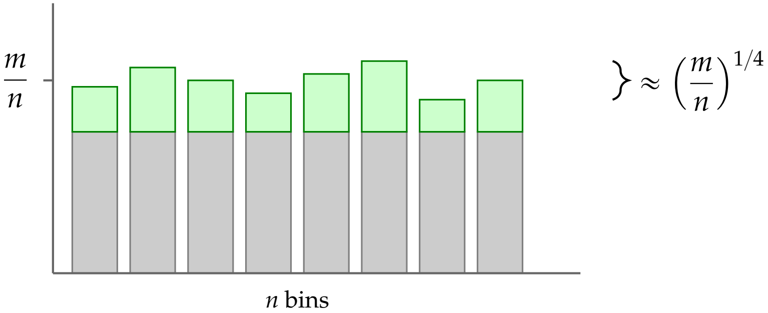
Slice and Spread Reduces Overload



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

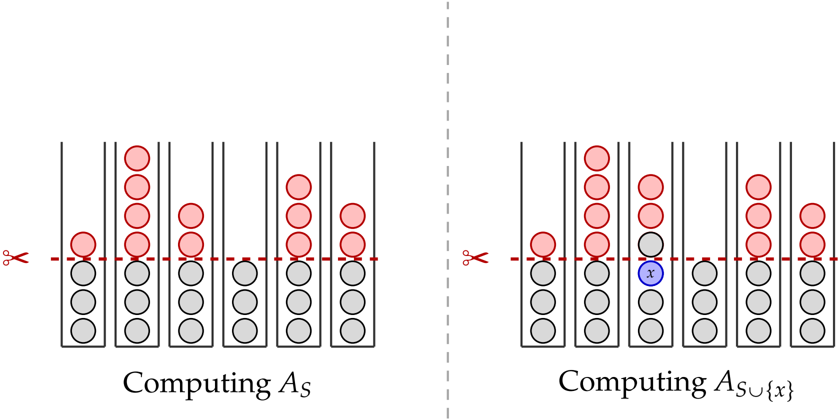
- Overload: $(m/n)^{1/2} \;\rightarrow\; (m/n)^{1/4}$
- History independent? Yes!
- Recourse:

Slice and Spread

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

Slice and Spread

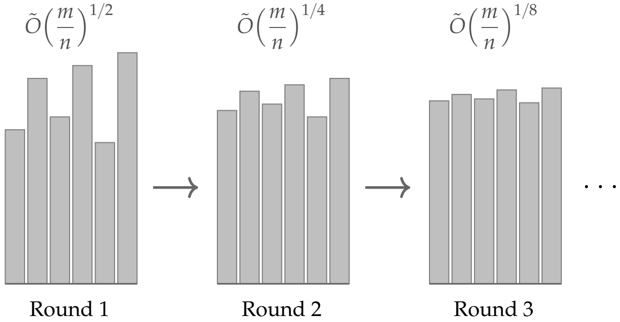
- Overload: $(m/n)^{1/2} \rightarrow (m/n)^{1/4}$
- History independent? Yes!
- Recourse: 1



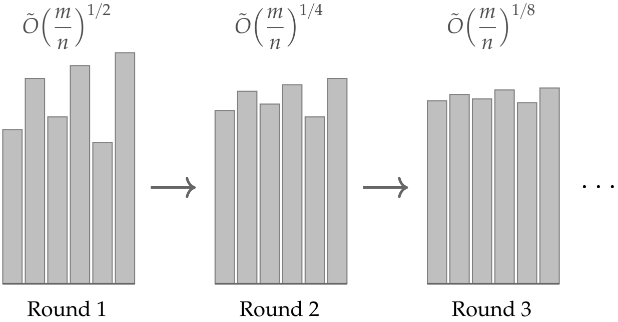
Computing A_S

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

Repeatedly Slicing and Spreading



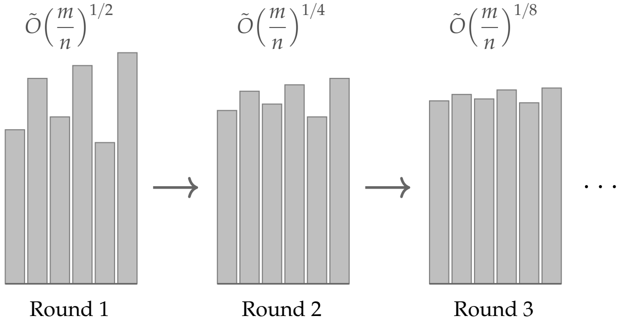
Repeatedly Slicing and Spreading



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

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

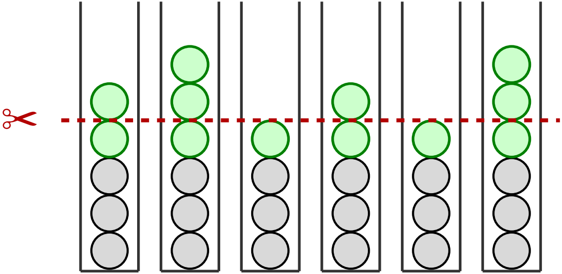
Repeatedly Slicing and Spreading



~~After $O(\log\log(m/n))$ rounds…~~

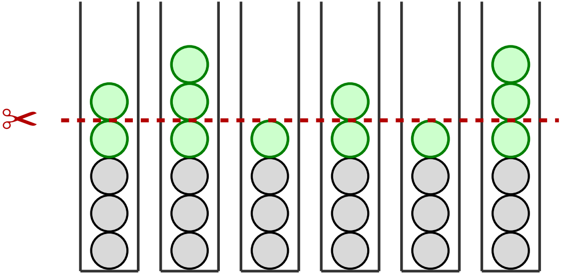
- ~~Overload = $O(1)$?~~
- ~~Recourse = $O(\log\log(m/n))$?~~

Algorithmic Question



Question: Which balls do we slice in round k?

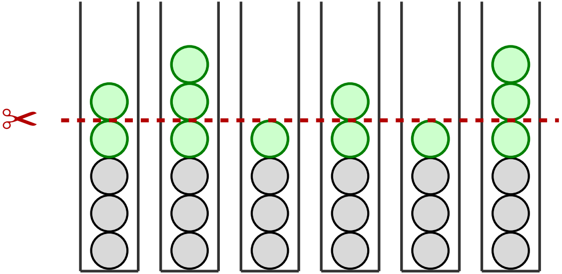
Algorithmic Question



Question: Which balls do we slice in round k?

- **Option 1:** Scrape off the top

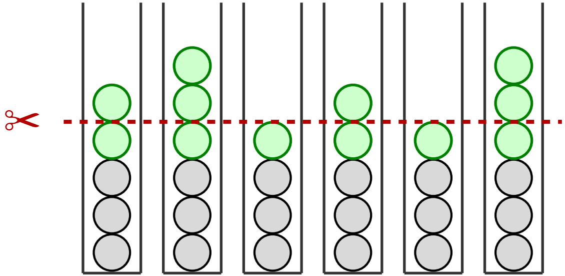
Algorithmic Question



Question: Which balls do we slice in round k?

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

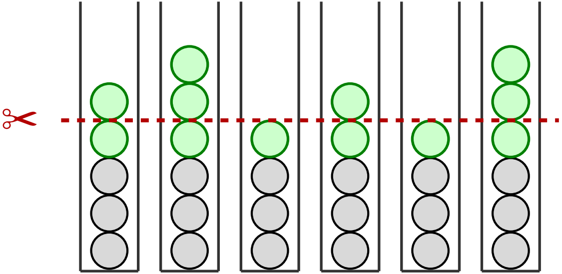
Algorithmic Question



Question: Which balls do we slice in round k?

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

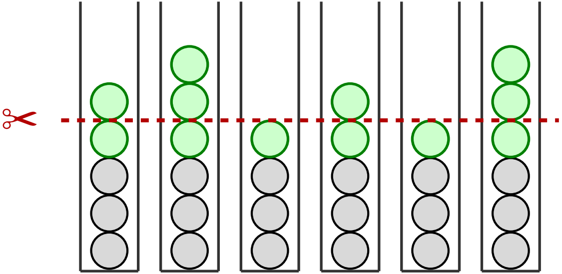
Algorithmic Question



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

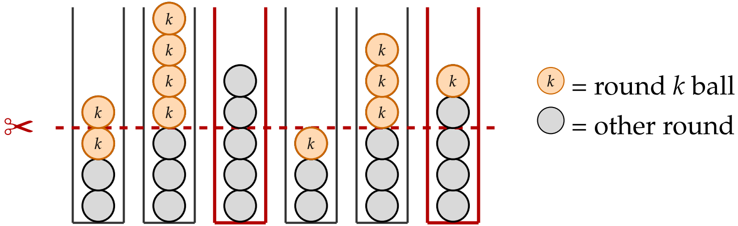
ALGORITHMIC QUESTION



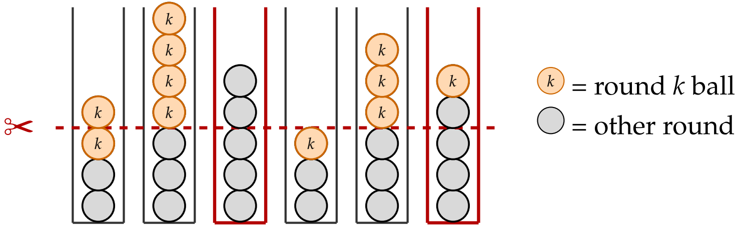
Question: Which balls do we slice in round k?

- **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

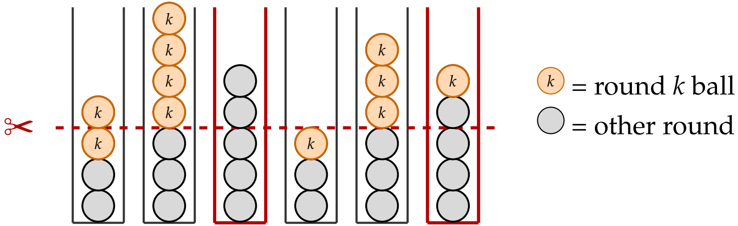


Challenge 1: Slicing Failures



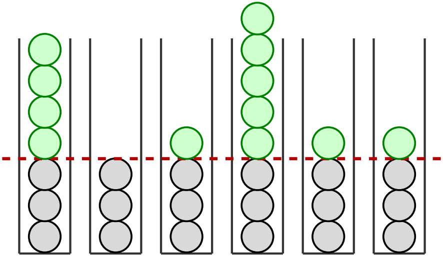
Challenge: Some bins may not have enough round-k balls to support slicing.

Challenge 1: Slicing Failures

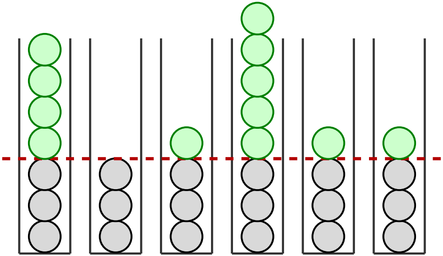


Challenge: Some bins may not have enough round-k balls to support slicing.
Result: We can’t slice evenly — the jaggedness remains in those bins.

Challenge 2: Spreading Failures

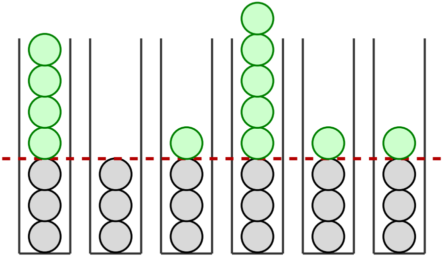


Challenge 2: Spreading Failures



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

Challenge 2: Spreading Failures

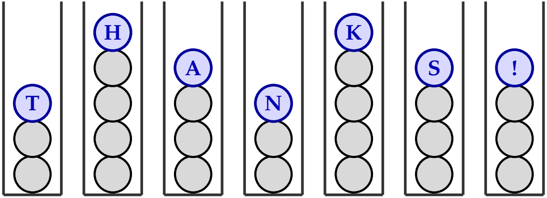


Challenge: The spreading step may distribute balls unevenly — creating new jaggedness.

Dilemma:

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

History-Independent Load Balancing



Michael A. Bender
Stony Brook University

William Kuszmaul
CMU

Elaine Shi
CMU

Rose Silver
CMU