CS630 Graduate Algorithms

October 31, 2024

by Dora Erdos and Jeffrey Considine

- Hash tables
- Power of two choices

slides by Dora Erdos and Jeffrey Considine, based on Kevin Wayne, Adam Smith

Hash Tables

Interface:

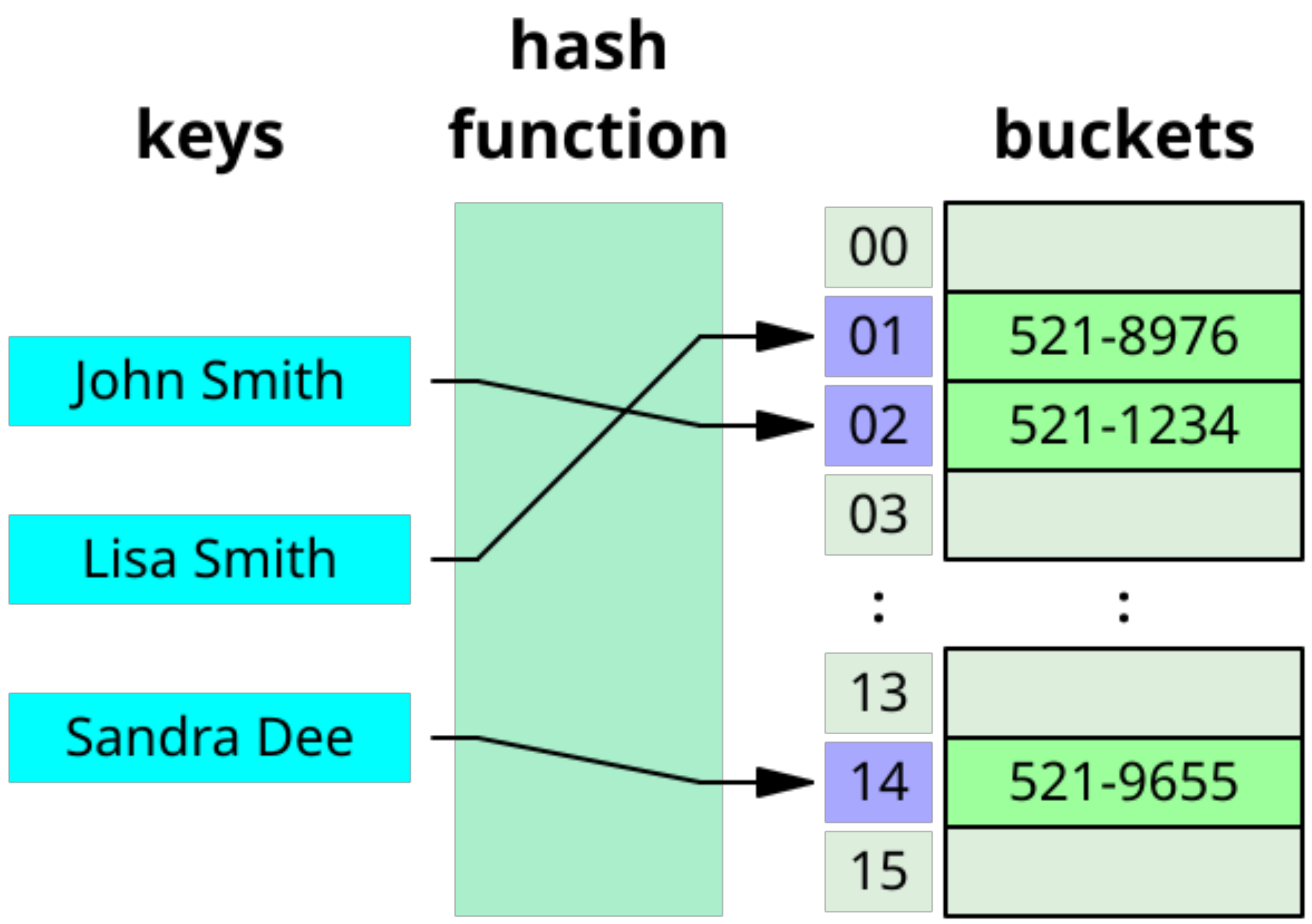
- A hash table is a data structure implementing a key-value mapping.
  - Insert
  - Lookup / search
  - Delete
- A common variant: skip values to implement a set interface.

Internals:

- Uses a “hash” function to evenly map keys across an array…
- Array entries are often called buckets.
- Bucket implementation details vary a lot to get different behaviors.

What are some examples of hash tables that you’ve used when programming?

Hash Tables



Values in hash table are phone numbers.

Image source: https://en.wikipedia.org/wiki/Hash_table

Hashing with Separate Chaining

Handle collisions with a linked list from each bucket.

- Usually analysis assumes adding to the end.
- Easiest variant to analyze
- Less space efficient from linked list overhead and memory management

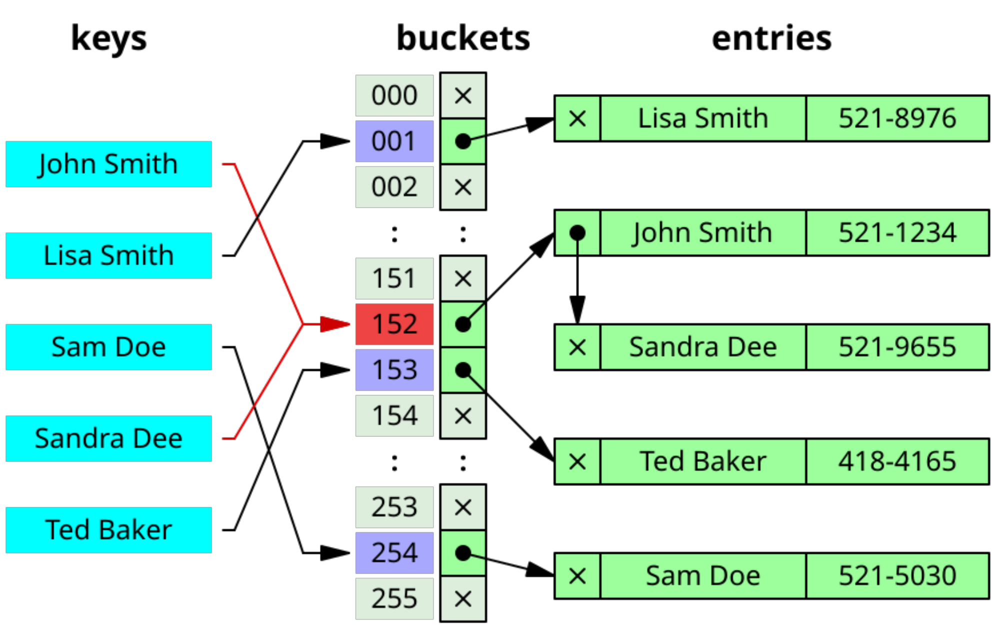


Image source: https://en.wikipedia.org/wiki/Hash_table

Linear Probing

Insertion:

- If collision on $h(x)$, try $h(x)+1,\ h(x)+2,\ h(x)+3,\ \dots$ until an empty bucket is found.
  - Memory locality is great.
- What happens if the hash table is full?
  - If the hash table is full, then this will loop forever.
  - Will get really slow when almost full.
- Primary clustering problem
  - Collisions create short runs of full buckets.
  - But if those short runs catch up to another run of full buckets, they combine.
- Leads to insertion times of $\Theta\left(\dfrac{1}{(1-\alpha)^2}\right)$
- Searches for items not in hash table take the same time.
- Searches for items in hash table somewhat faster $\Theta\left(\dfrac{1}{1-\alpha}\right)$

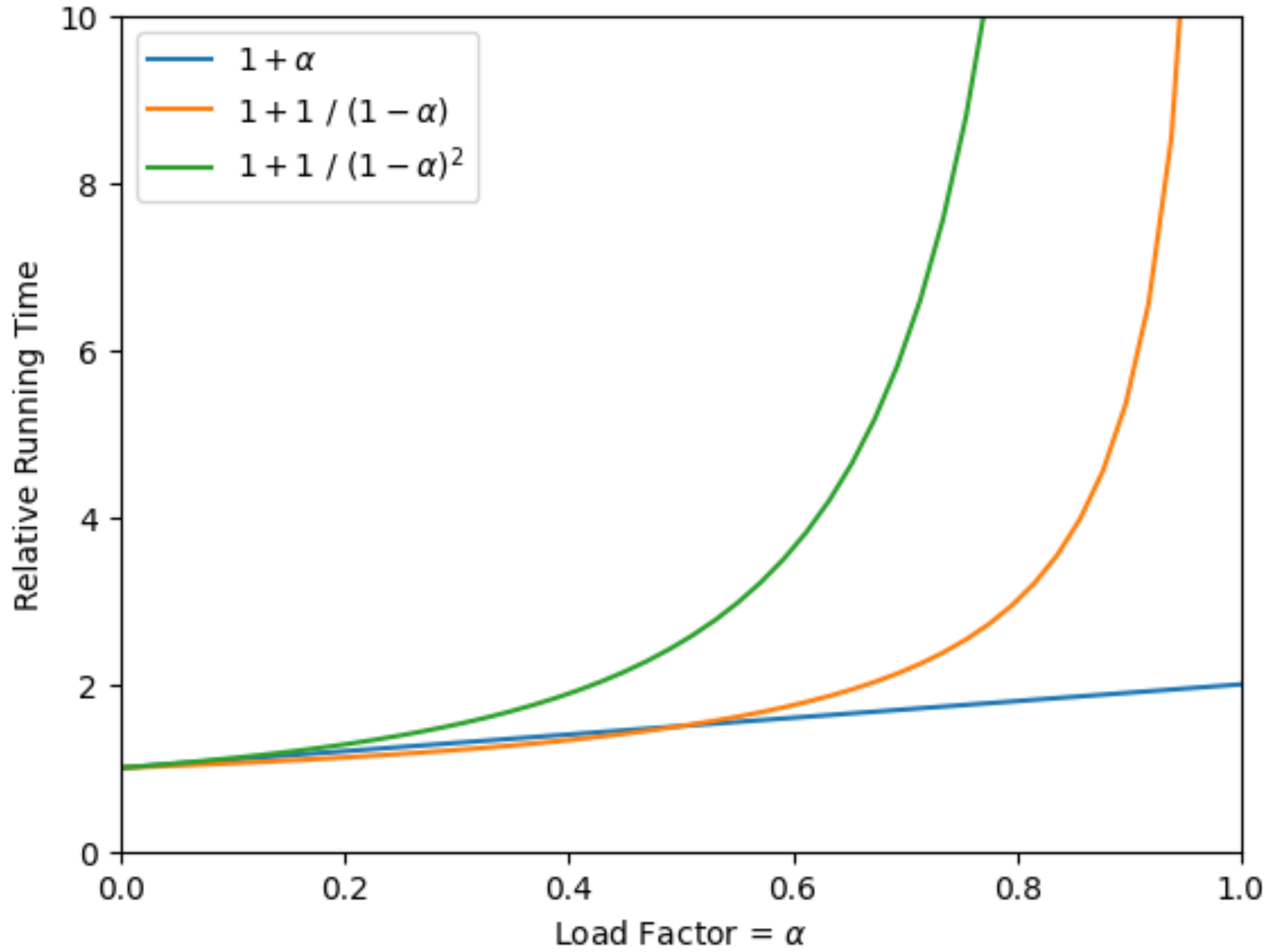
Average Case Performance So Far

For a hash table with m items inserted into n buckets, so load factor $\alpha = m/n$…

	Separate Chaining	Linear Probing
Insert / Delete	$1 + \alpha$	$1 + \dfrac{1}{(1-\alpha)^2}$
Search (successful)	$1 + \alpha$	$1 + \dfrac{1}{1-\alpha}$
Search (unsuccessful)	$1 + \alpha$	$1 + \dfrac{1}{(1-\alpha)^2}$

Average Case Performance So Far

For a hash table with m items inserted into n buckets, so load factor $\alpha = m/n$…



Quadratic Probing

Insertion

- If collision on $h(x)$, try $h(x) + 1^2,\ h(x) + 2^2,\ h(x) + 3^2,\ \ldots$ until an empty bucket is found.

Quadratic Probing

Insertion

- If collision on $h(x)$, try $h(x) + 1^2,\ h(x) + 2^2,\ h(x) + 3^2,\ \ldots$ until an empty bucket is found.
- Avoid primary clustering problem of linear probing.

Quadratic Probing

Insertion

- If collision on $h(x)$, try $h(x) + 1^2,\ h(x) + 2^2,\ h(x) + 3^2,\ \ldots$ until an empty bucket is found.
- Avoid primary clustering problem of linear probing.
- TRIVIA: If alternating signs used, this even gives a permutation through all buckets.

Quadratic Probing

Insertion

- If collision on $h(x)$, try $h(x) + 1^2,\ h(x) + 2^2,\ h(x) + 3^2,\ \ldots$ until an empty bucket is found.
- Avoid primary clustering problem of linear probing.
- TRIVIA: If alternating signs used, this even gives a permutation through all buckets.
- Memory locality not as good as linear probing because of increasing gaps.

Quadratic Probing

Insertion

- If collision on $h(x)$, try $h(x) + 1^2,\ h(x) + 2^2,\ h(x) + 3^2,\ \ldots$ until an empty bucket is found.
- Avoid primary clustering problem of linear probing.
- TRIVIA: If alternating signs used, this even gives a permutation through all buckets.
- Memory locality not as good as linear probing because of increasing gaps.
- Complexity???

Double Hashing

Insertion

- If collision on $h(x)$, try $h(x) + h_2(x),\ h(x) + 2h_2(x),\ h(x) + 3h_2(x),\ \ldots$ until an empty bucket is found.
- Avoids primary clustering problem with one extra hash function evaluation.
- Both searches and insertions take

$\Theta\left(1 + \frac{1}{1-\alpha}\right)$ expected time (2007)

Average Case Performance So Far

For a hash table with m items inserted into n buckets, so load factor $\alpha = m/n$…

	Separate Chaining	Linear Probing	Quadratic Probing	Double Hashing
Insert / Delete	$1+\alpha$	$1+\dfrac{1}{(1-\alpha)^2}$	???	$1+\dfrac{1}{1-\alpha}$
Search (successful)	$1+\alpha$	$1+\dfrac{1}{1-\alpha}$	???	$1+\dfrac{1}{1-\alpha}$
Search (unsuccessful)	$1+\alpha$	$1+\dfrac{1}{(1-\alpha)^2}$	???	$1+\dfrac{1}{1-\alpha}$

Resizing Hash Tables

Usual answer to high load factors is resizing the hash table.

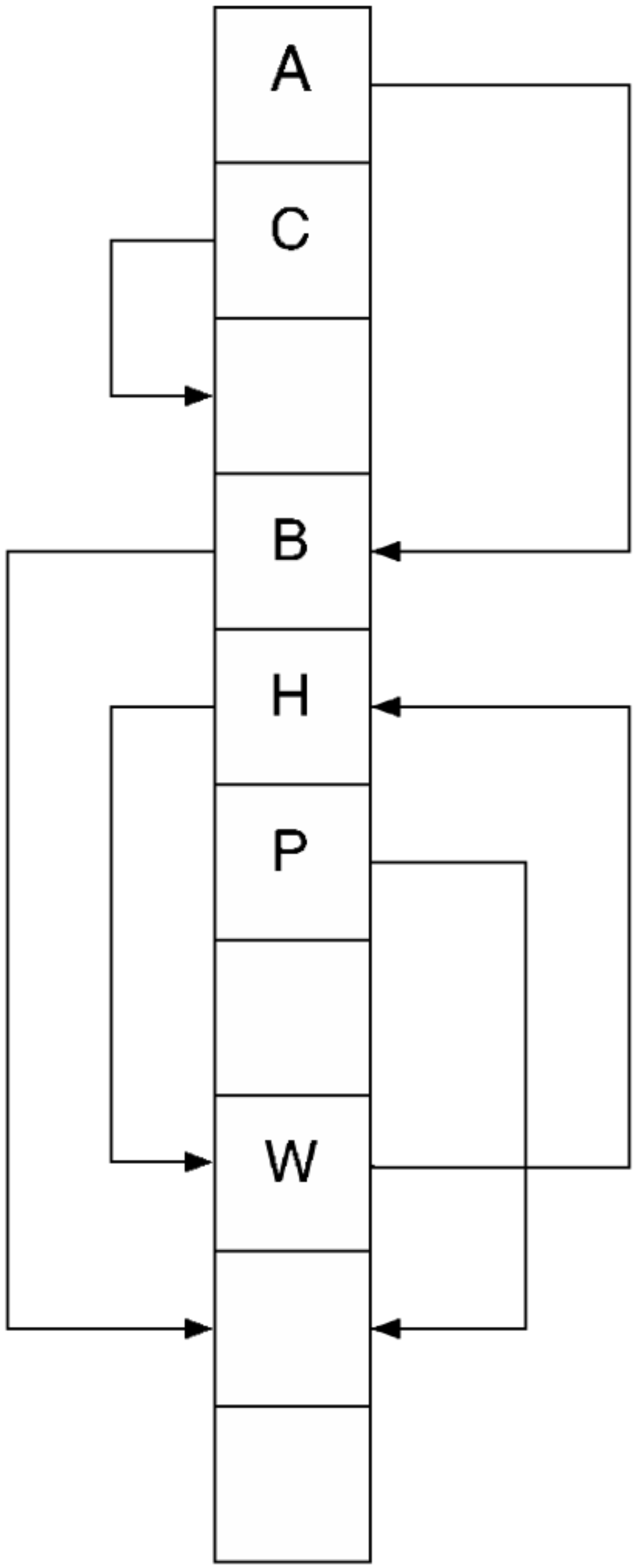
- Works if not too memory constrained.
- And can wait for rebuild.
- Or can handle memory overhead of building a second copy in parallel.
- Better if you can get the size right the first time.

Cuckoo Hashing

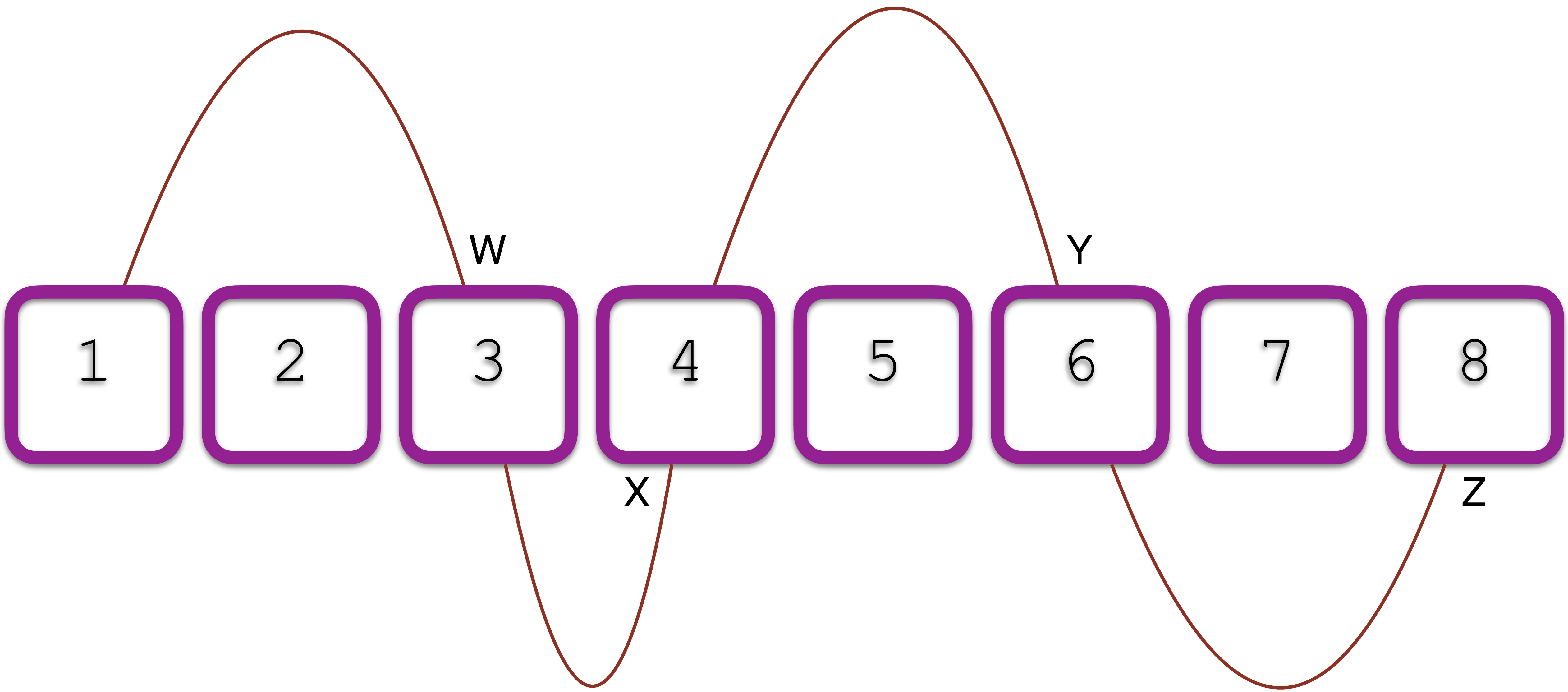
Insertion

- Check two buckets - $h_1(x)$ and $h_2(x)$.
- If $h_1(x)$ is empty,
  - Place item at $h_1(x)$ and stop.
- If $h_2(x)$ is empty,
  - Place item at $h_2(x)$ and stop.
- Otherwise,
  - Remove old item x' from $h_2(x)$.
  - Place new item at $h_2(x)$.
  - Recursively place old item x'.

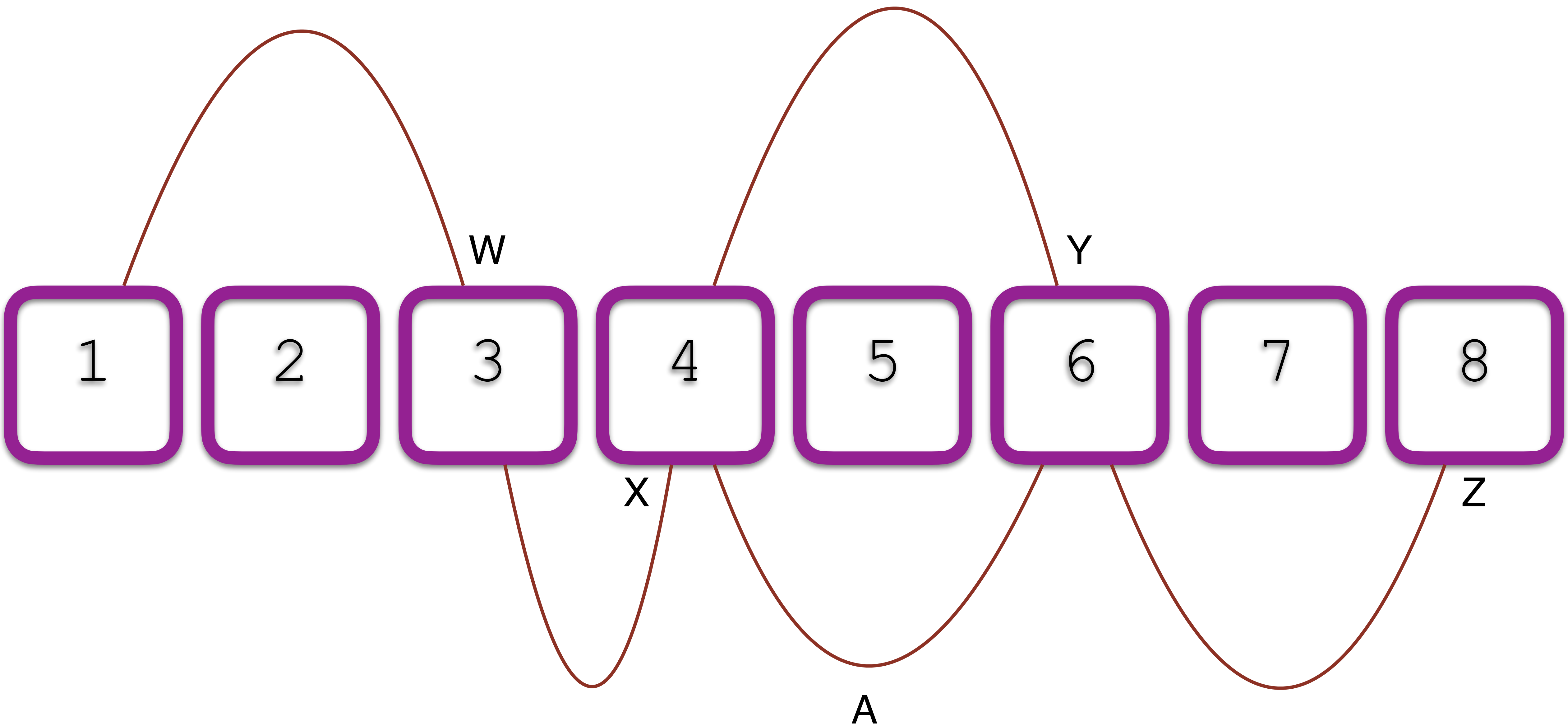
Recursive placements need to push old items to the position indicated by the other hash function.



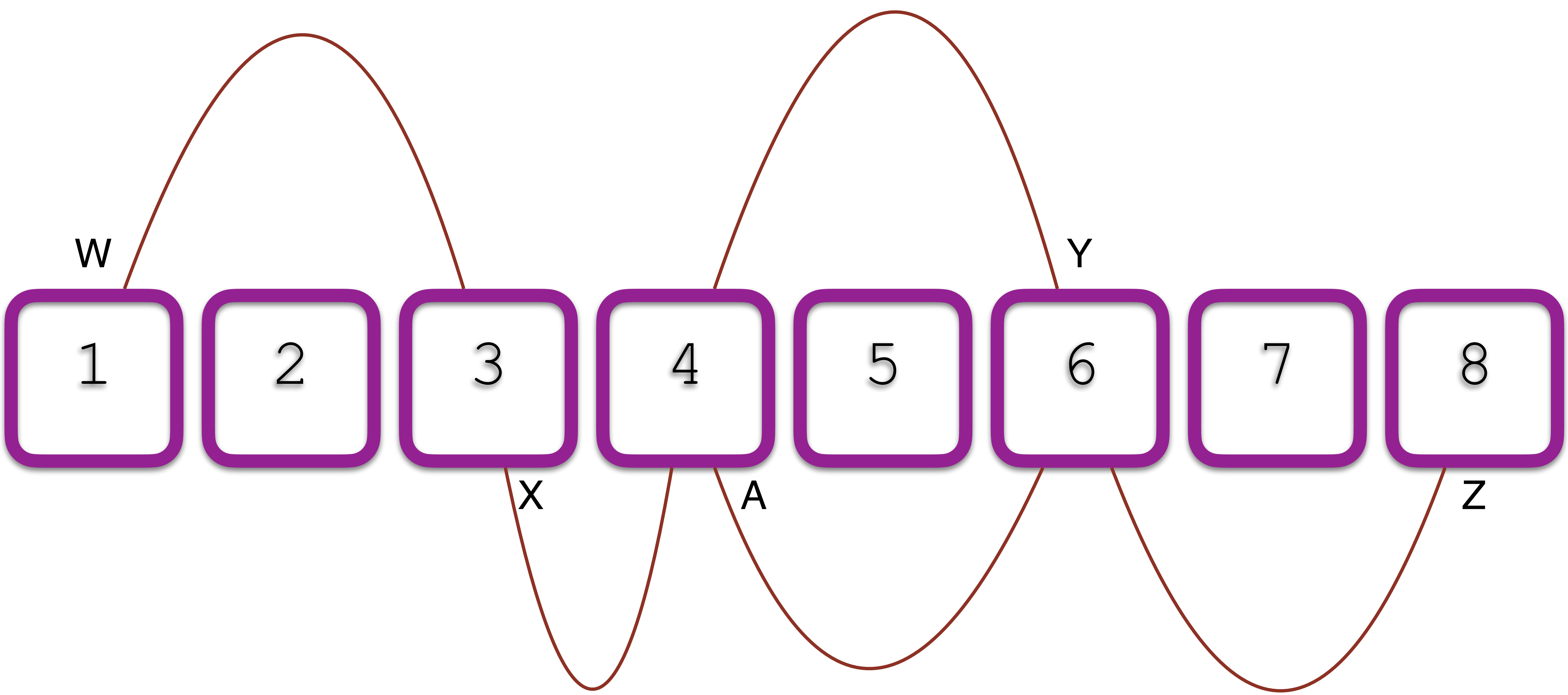
Cuckoo Hashing as a Random Graph



Cuckoo Hashing as a Random Graph



Cuckoo Hashing as a Random Graph



Cuckoo Hashing with Two Tables

The original version used two tables… the analysis is essentially the same.

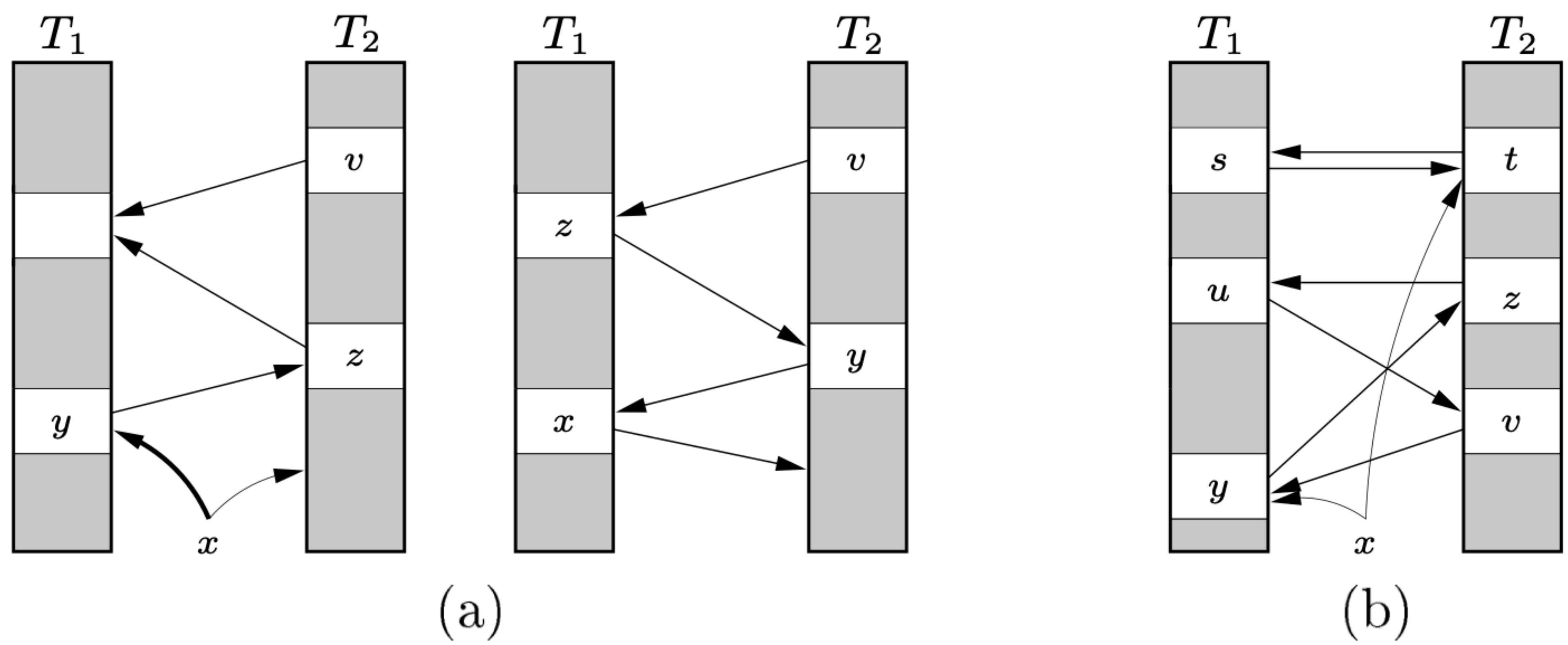


Figure 1: (a) Key x is successfully inserted by moving keys y and z to the other table. (b) Key x cannot be accomodated and a rehash is necessary.

Image source: Cuckoo Hashing by Pagh and Rodler (2001)

Cuckoo Hashing

Insertion

- Assume $n/2 \geq m(1+\epsilon)$ (so $\alpha < 1/2$)

- No loop with probability $1 - 1/n$

- Assuming no loop, $O(1 + 1/\epsilon)$ expected time.

So expected constant time if you pick a maximum α below 1/2.

- Must resize to maintain this!

Top Hat

Which of the following statements are true about cuckoo hashing?

1. Search takes $\Theta(1 + 1/\epsilon)$ expected time.
2. Search takes $\Theta(1)$ expected time.
3. Search takes $\Theta(1)$ worst case time.

Cuckoo Hashing

Search:

- Check $h_1(x)$ and $h_2(x)$.

Cuckoo Hashing

Delete:

- Check $h_1(x)$ and $h_2(x)$ and delete if found.

Average Case Performance So Far

For a hash table with m items inserted into n buckets, so load factor $\alpha = m/n$…

	Separate Chaining	Linear Probing	Quadratic Probing	Double Hashing	Cuckoo Hashing
Insert / Delete	$1+\alpha$	$1+\frac{1}{(1-\alpha)^2}$	???	$1+\frac{1}{1-\alpha}$	$1+\frac{1}{\epsilon}$
Search (successful)	$1+\alpha$	$1+\frac{1}{1-\alpha}$	???	$1+\frac{1}{1-\alpha}$	1
Search (unsuccessful)	$1+\alpha$	$1+\frac{1}{(1-\alpha)^2}$	???	$1+\frac{1}{1-\alpha}$	1

Balls and Bins Problems

Given n balls inserted into n bins, assigning balls to bins uniformly at random, what is the worst case load?

Balls and Bins Problems

Given n balls inserted into n bins, assigning balls to bins uniformly at random, what is the worst case load?

- $\Theta(\log n / \log \log n)$
- Same as randomized load balancing and hashing with separate chaining.

Power of Two Choices

Given n balls inserted into n bins as follows -

- For each ball sequentially,
  - Pick two bins uniformly.
  - Put the ball in the least loaded bin breaking ties arbitrarily.

What is the maximum case load?

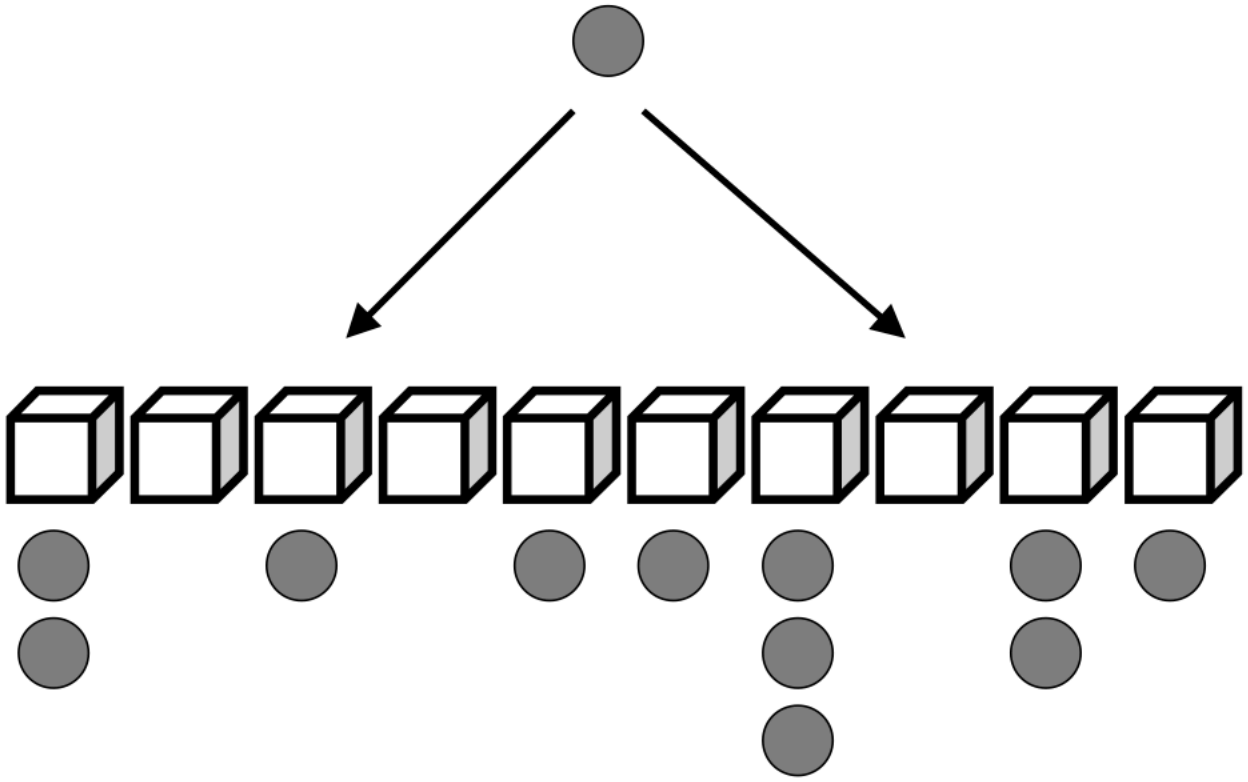


Image source: The Power of Two Random Choices - A Survey of Techniques and Results by Mitzenmacher, Richa, Sitaraman (2001)

Power of Two Choices

Given n balls inserted into n bins as follows -

- For each ball sequentially,
  - Pick two bins uniformly.
  - Put the ball in the least loaded bin breaking ties arbitrarily.

What is the maximum case load?

- With probability at least $1 - 1/n$, the maximum load is $\ln \ln n/2 + O(1)$.

Power of Two Choices

Given n balls inserted into n bins as follows -

- For each ball sequentially,
  - Pick two bins uniformly.
  - Put the ball in the least loaded bin breaking ties arbitrarily.

What is the maximum case load?

- With probability at least $1 - 1/n$, the maximum load is $\ln \ln n / \ln 2 + O(1)$.

If d choices instead of 2, then

- With probability at least $1 - 1/n$, the maximum load is $\ln \ln n / \ln d + O(1)$.

Power of Two Choices Sketch

For $i \geq 1$, let β_i be a bound on the number of bins with load at least i w.h.p.

- w.h.p. = "With high probability" = probability at least $1 - 1/n$

Power of Two Choices Sketch

For $i \geq 1$, let β_i be a bound on the number of bins with load at least i w.h.p.

- w.h.p. = "With high probability" = probability at least $1 - 1/n$

For example,

- $\beta_1 \leq 0.9n$ (compare to (1-1/e)n expected bins with at least one ball)

Power of Two Choices Sketch

For $i \geq 1$, let β_i be a bound on the number of bins with load at least i w.h.p.

- w.h.p. = "With high probability" = probability at least $1 - 1/n$

For example,

- $\beta_1 \leq 0.9n$ (compare to (1-1/e)n expected bins with at least one ball)

How can we bound β_{i+1} given β_i?

- Expected number of bins with at least $i+1$ balls is at most $n\left(\frac{\beta_i}{n}\right)^d$

  since the probability of all d bins having at least i balls is at most $\left(\frac{\beta_i}{n}\right)^d$.

Power of Two Choices Sketch

For $i \geq 1$, let β_i be a bound on the number of bins with load at least i w.h.p.

- w.h.p. = "With high probability" = probability at least $1 - 1/n$

For example,

- $\beta_1 \leq 0.9n$ (compare to (1-1/e)n expected bins with at least one ball)

How can we bound β_{i+1} given β_i?

- Expected number of bins with at least $i + 1$ balls is at most $n \left(\frac{\beta_i}{n}\right)^d$

  since the probability of all d bins having at least i balls is at most $\left(\frac{\beta_i}{n}\right)^d$.

- So we can pick c such that $\beta_{i+1} = cn \left(\frac{\beta_i}{n}\right)^d$ and β_{i+1} holds w.h.p.

Power of Two Choices Sketch

For $i \geq 1$, let β_i be a bound on the number of bins with load at least i w.h.p.

- w.h.p. = "With high probability" = probability at least $1 - 1/n$

For example,

- $\beta_1 \leq 0.9n$ (compare to (1-1/e)n expected bins with at least one ball)

How can we bound β_{i+1} given β_i?

- Expected number of bins with at least $i+1$ balls is at most $n\left(\frac{\beta_i}{n}\right)^d$

  since the probability of all d bins having at least i balls is at most $\left(\frac{\beta_i}{n}\right)^d$.

- So we can pick c such that $\beta_{i+1} = cn\left(\frac{\beta_i}{n}\right)^d$ and β_{i+1} holds w.h.p.

- If we look at the sequence $\beta_1, \beta_2, \beta_3, \ldots$ each step roughly squares the previous value for $d = 2$ and it drops faster for higher d.

Power of Two Choices Sketch

For $i \geq 1$, let β_i be a bound on the number of bins with load at least i w.h.p.

- w.h.p. = "With high probability" = probability at least $1 - 1/n$

For example,

- $\beta_1 \leq 0.9n$ (compare to (1-1/e)n expected bins with at least one ball)

How can we bound β_{i+1} given β_i?

- Expected number of bins with at least $i+1$ balls is at most $n\left(\frac{\beta_i}{n}\right)^d$

  since the probability of all d bins having at least i balls is at most $\left(\frac{\beta_i}{n}\right)^d$.

- So we can pick c such that $\beta_{i+1} = cn\left(\frac{\beta_i}{n}\right)^d$ and β_{i+1} holds w.h.p.

- If we look at the sequence $\beta_1, \beta_2, \beta_3, \ldots$ each step roughly squares the previous value for $d = 2$ and it drops faster for higher d.
- $\beta_j < 1$ for $j = O(\log\log n)$ so max load $< j$.

Power of Two Choices and Randomized Load Balancing

Original version:

- max load is $\Theta(\log n / \log \log n)$ w.h.p.

With d choices,

- max load is $\ln \ln n / \ln d + O(1)$ w.h.p.

Exponentially / logarithmically smaller.

- No hidden multiplicative constant for $d > 1$ choices
- Just $d = 2$ is a big gain.

Power of Two Choices with Left Bias

Given n balls inserted into n bins as follows -

- Partition the bins into d sets of n/d and order them from left to right.
- For each ball sequentially,
  - Pick d bins, one from each partition and uniformly within each partition.
  - Put the ball in the least loaded bin breaking ties to the left most tied bin.

What is the maximum case load?

Power of Two Choices with Left Bias

Given n balls inserted into n bins as follows -

- Partition the bins into d sets of n/d and order them from left to right.
- For each ball sequentially,
  - Pick d bins, one from each partition and uniformly within each partition.
  - Put the ball in the least loaded bin breaking ties to the left most tied bin.

What is the maximum case load?

- The maximum load is $\ln \ln n/(d \ln \phi_d) + O(1)$ where ϕ_d is the exponent for growth of a

  generalized Fibonacci sequence…

Power of Two Choices Summary

What is the maximum load with high probability…

- For uniform random load balancing?
  - $\Theta(\log n / \log \log n)$

- For d uniform random choices?
  - $\ln \ln n / \ln d + O(1)$

- For d left-biased partitioned random choices?
  - $\ln \ln n / (d \ln \phi_d) + O(1)$

Power of Two Choices Applications

- Hash tables (usually separate chaining)
- Server load balancing (requires load check)
- Processor cache design (more efficient cache lines)
- Circuit routing
- Virtual connection routing
- Peer-to-peer networks (storage and lookups, uneven bins)

- Almost any load balancing problem