Zip-zip Trees: Making Zip Trees More Balanced, Biased, Compact, or Persistent

Ofek Gila[1] , Michael T. Goodrich[1] , and Robert E. Tarjan[2]

[1]University of California, Irvine
[2]Princeton University

WADS, 2023

Motivation

- 2018 – Zip tree (Tarjan, Levy, Timmel)
  - $1.5 \log n$ expected average depth
    - Min key: $0.5 \log n$, Max key: $\log n$
  - Space cost: $\log \log n$ bits per node

Motivation

- 2018 – Zip tree (Tarjan, Levy, Timmel) - asymmetric
  - 1.5 $\log n$ expected average depth
    - Min key: $0.5 \log n$, Max key: $\log n$
  - Space cost: $\log \log n$ bits per node

Motivation

- 2018 – Zip tree (Tarjan, Levy, Timmel) - asymmetric
  - 1.5 $\log n$ expected average depth
    - Min key: $0.5 \log n$, Max key: $\log n$
  - Space cost: $\log \log n$ bits per node

- 2023 – Zip-zip trees:
  - ☑ 1.39 $\log n$ expected average depth

Motivation

- 2018 – Zip tree (Tarjan, Levy, Timmel) - asymmetric
  - 1.5 $\log n$ expected average depth
    - Min key: $0.5 \log n$, Max key: $\log n$
  - Space cost: $\log \log n$ bits per node

- 2023 – Zip-zip trees:
  - ☑ 1.39 $\log n$ expected average depth
  - ☑ Space cost: $\log \log n$ bits per node (or $O(1)$ bits per update w.h.p.)

Motivation

- 2018 – Zip tree (Tarjan, Levy, Timmel) - asymmetric
  - 1.5 $\log n$ expected average depth
    - Min key: $0.5 \log n$, Max key: $\log n$
  - Space cost: $\log \log n$ bits per node

- 2023 – Zip-zip trees:
  - ☑ 1.39 $\log n$ expected average depth
  - ☑ Space cost: $\log \log n$ bits per node (or $O(1)$ bits per update w.h.p.)
  - ☑ Easy to implement

Motivation

- 2018 – Zip tree (Tarjan, Levy, Timmel) - asymmetric
  - 1.5 $\log n$ expected average depth
    - Min key: $0.5 \log n$, Max key: $\log n$
  - Space cost: $\log \log n$ bits per node
- 2023 – Zip-zip trees:
  - ☑ 1.39 $\log n$ expected average depth
  - ☑ Space cost: $\log \log n$ bits per node (or $O(1)$ bits per update w.h.p.)
  - ☑ Easy to implement
  - ☑ Strongly history independent

Motivation

- 2018 – Zip tree (Tarjan, Levy, Timmel) - asymmetric
  - $1.5 \log n$ expected average depth
    - Min key: $0.5 \log n$, Max key: $\log n$
  - Space cost: $\log \log n$ bits per node

- 2023 – Zip-zip trees:
  - ☑ $1.39 \log n$ expected average depth
  - ☑ Space cost: $\log \log n$ bits per node (or $O(1)$ bits per update w.h.p.)
  - ☑ Easy to implement
  - ☑ Strongly history independent
  - ☑ May be partially persistent

Motivation

- 2018 – Zip tree (Tarjan, Levy, Timmel) - asymmetric
  - 1.5 $\log n$ expected average depth
    - Min key: $0.5 \log n$, Max key: $\log n$
  - Space cost: $\log \log n$ bits per node

- 2023 – Zip-zip trees:
  - ☑ 1.39 $\log n$ expected average depth
  - ☑ Space cost: $\log \log n$ bits per node (or $O(1)$ bits per update w.h.p.)
  - ☑ Easy to implement
  - ☑ Strongly history independent
  - ☑ May be partially persistent
  - ☑ Supports biased keys (still $O(\log \log n)$ bits per node)

Skip List

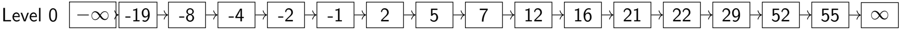


Figure 1: A sorted linked list

Skip List

- What if you add 'fast lanes'?

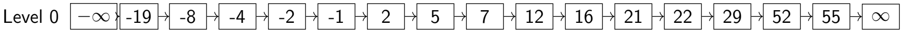


Figure 1: A sorted linked list

Skip List

- What if you add ‘fast lanes’?

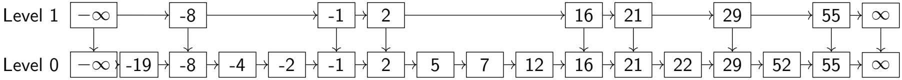


Figure 1: A skip list with one coin flip

Skip List

- What if you add ‘fast lanes’?

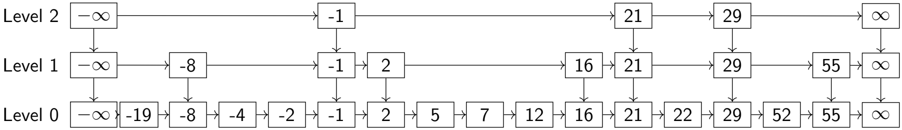


Figure 1: A skip list with two coin flips

Skip List

- What if you add 'fast lanes'?

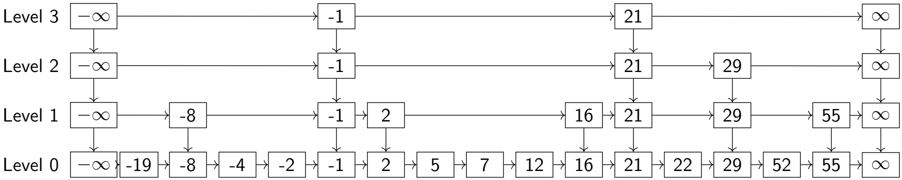


Figure 1: A skip list with three coin flips

Skip List

- What if you add 'fast lanes'?

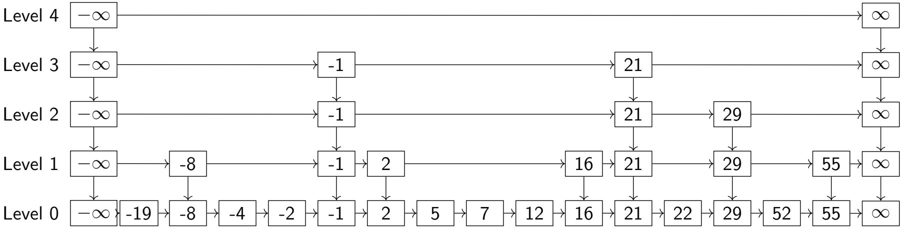


Figure 1: A skip list with four coin flips

Skip List

- What if you add 'fast lanes'?
- Idea: 1 move in level $k \approx 2$ in level $k-1 \approx 2^k$ in level 0

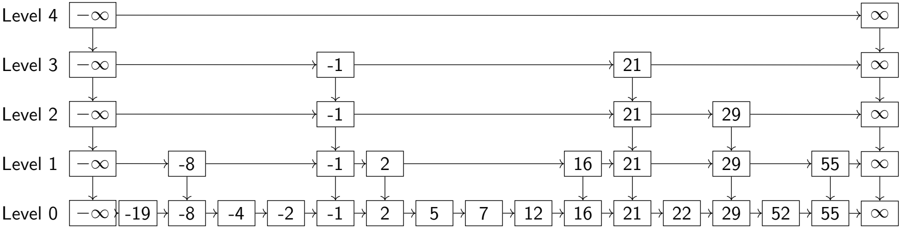


Figure 1: A skip list with four coin flips

Skip List

- What if you add 'fast lanes'?

- Idea: 1 move in level $k \approx 2$ in level $k-1 \approx 2^k$ in level 0
- $\mathcal{O}(\log n)$ expected search time

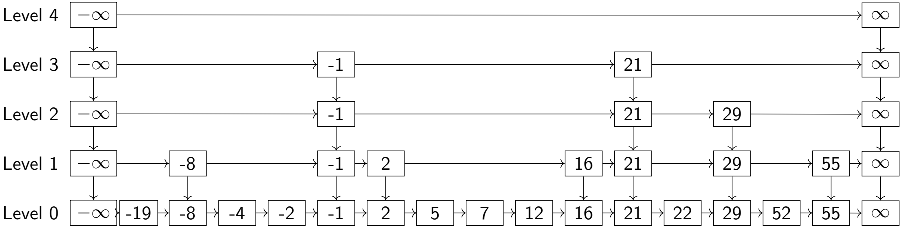


Figure 1: A skip list with four coin flips

Skip List

- What if you add 'fast lanes'?
- Idea: 1 move in level $k \approx 2$ in level $k-1 \approx 2^k$ in level 0
- $\mathcal{O}(\log n)$ expected search time
- Expected height of $\mathcal{O}(\log n)$

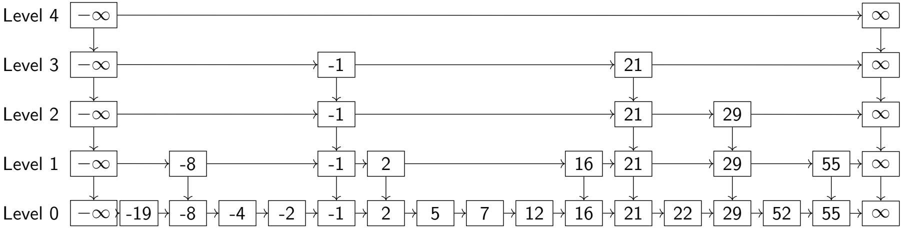


Figure 1: A skip list with four coin flips

Skip List

- What if you add ‘fast lanes’?

- Idea: 1 move in level $k \approx 2$ in level $k-1 \approx 2^k$ in level 0
- $\mathcal{O}(\log n)$ expected search time

- Expected height of $\mathcal{O}(\log n)$
- Expected $2n$ nodes

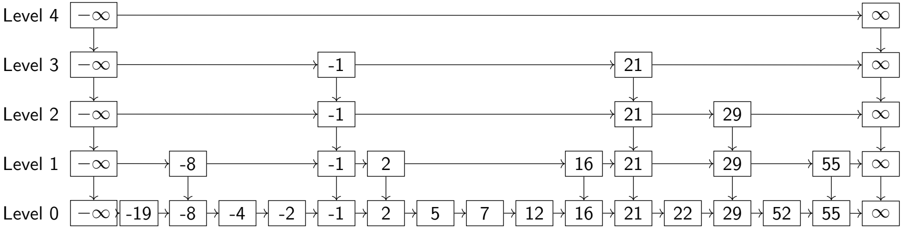


Figure 1: A skip list with four coin flips

Skip List

- What if you add ‘fast lanes’?

- Idea: 1 move in level $k \approx 2$ in level $k - 1 \approx 2^k$ in level 0
- $\mathcal{O}(\log n)$ expected search time

- Expected height of $\mathcal{O}(\log n)$
- Expected 2*n* nodes - bad

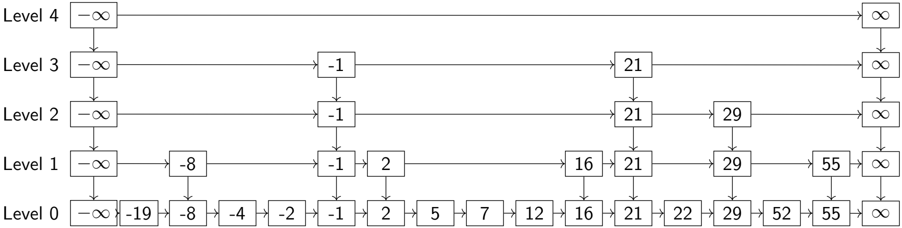
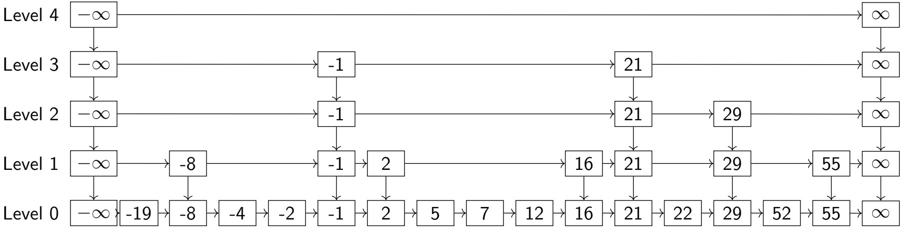


Figure 1: A skip list with four coin flips

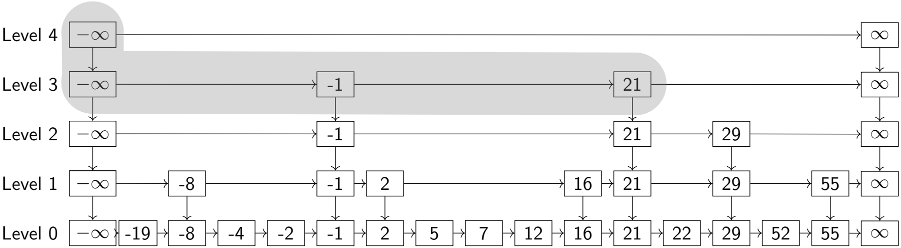
Zip Tree

- Idea: Construct a BST from a skip-list



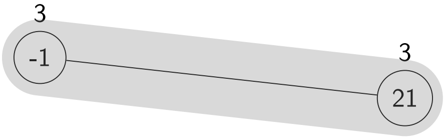
Zip Tree

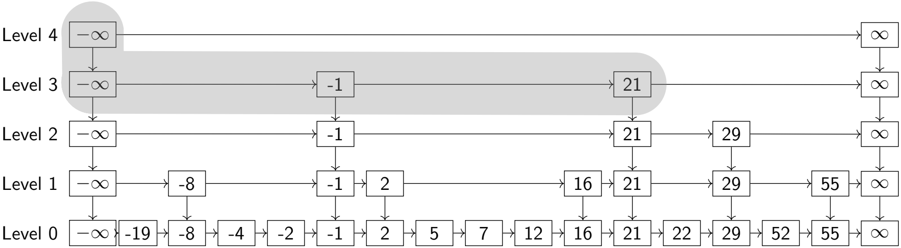
- Idea: Construct a BST from a skip-list



Zip Tree

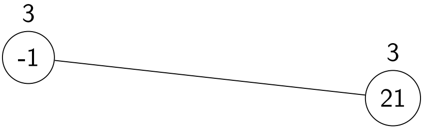
- Idea: Construct a BST from a skip-list

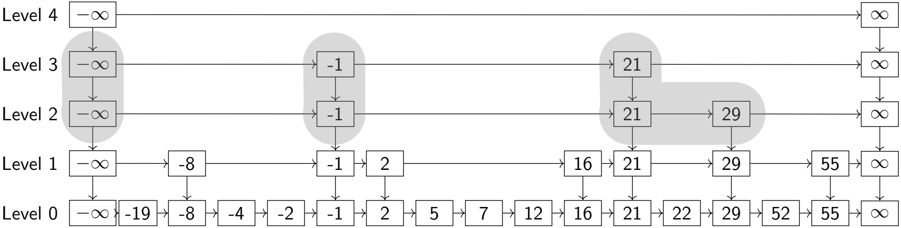




Zip Tree

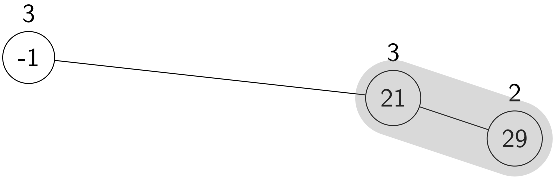
- Idea: Construct a BST from a skip-list

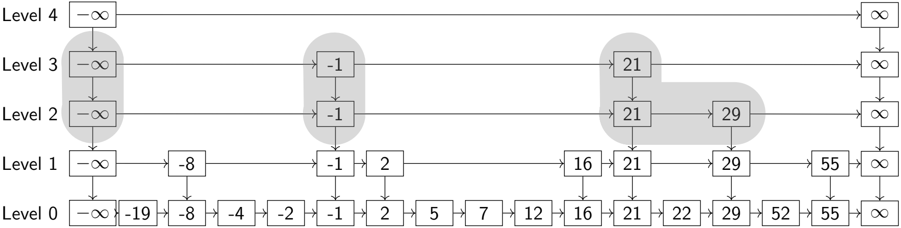




Zip Tree

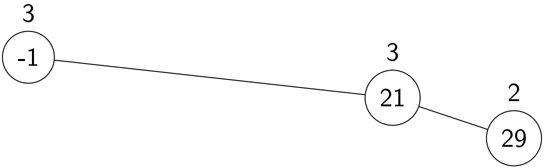
- Idea: Construct a BST from a skip-list

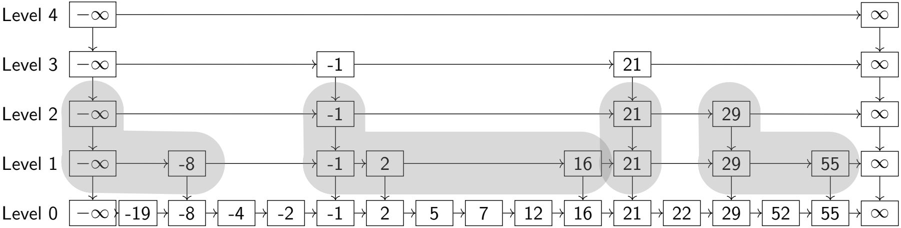




Zip Tree

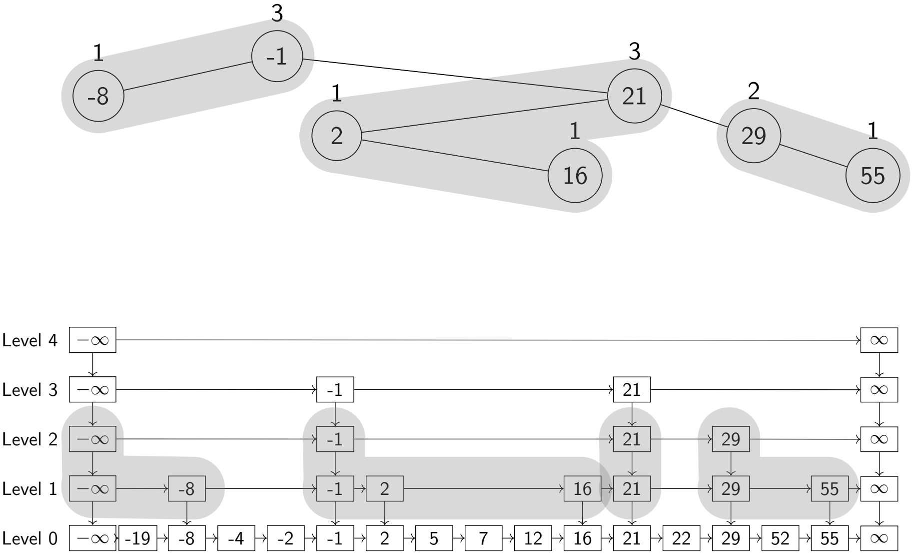
- Idea: Construct a BST from a skip-list





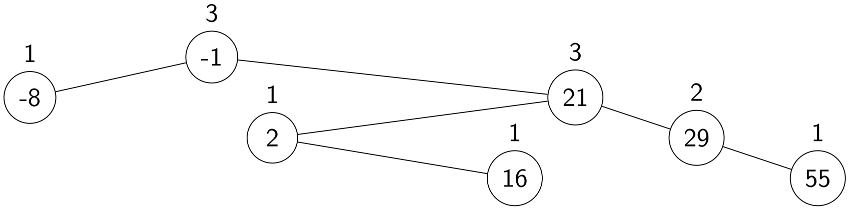
Zip Tree

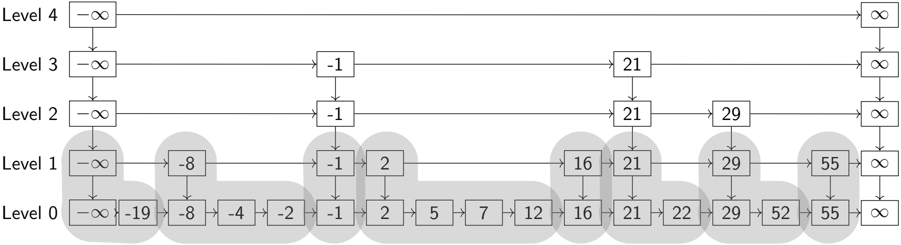
- Idea: Construct a BST from a skip-list



Zip Tree

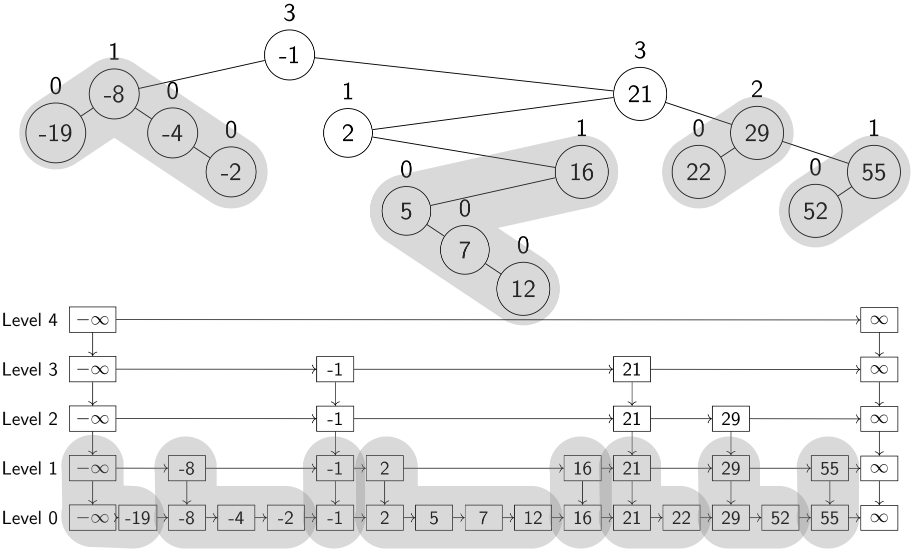
- Idea: Construct a BST from a skip-list





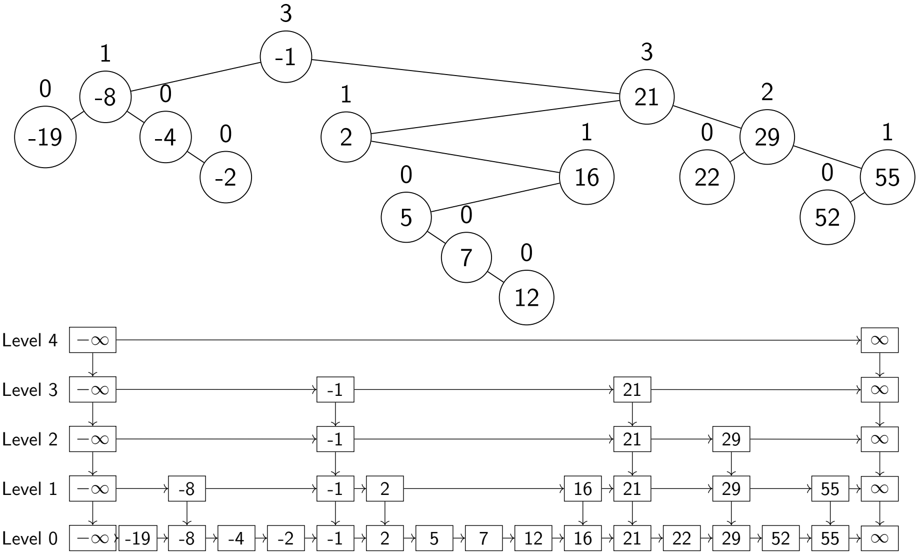
Zip Tree

- Idea: Construct a BST from a skip-list



Zip Tree

- Idea: Construct a BST from a skip-list



Zip Tree Analysis

- Max key expected depth $\log n$
- Min key expected depth $\log n/2$

- Asymmetric!

Zip Tree Analysis

- Max key expected depth $\log n$
- Min key expected depth $\log n/2$

- Asymmetric!

- Results in average node depth of $1.5 \log n$

Zip Tree Analysis

- Max key expected depth $\log n$
- Min key expected depth $\log n/2$
- Asymmetric!
- Results in average node depth of $1.5 \log n$
- Can we do better?

Zip-zip Trees

- What if rank was a tuple, (r_1, r_2)?

Zip-zip Trees

- What if rank was a tuple, (r_1, r_2)?
  - Let r_1 be geometrically distributed

Zip-zip Trees

- What if rank was a tuple, (r_1, r_2)?
  - Let r_1 be geometrically distributed
  - Let r_2 be uniformly distributed from $[1, \log^c n]$

Zip-zip Trees

- What if rank was a tuple, (r_1, r_2)?
  - Let r_1 be geometrically distributed
  - Let r_2 be uniformly distributed from $[1, \log^c n]$

- Compare ranks lexicographically

Zip-zip Trees

- What if rank was a tuple, (r_1, r_2)?
  - Let r_1 be geometrically distributed
  - Let r_2 be uniformly distributed from $[1, \log^c n]$

- Compare ranks lexicographically

☐ Metadata size $O(\log \log n)$?

Zip-zip Trees

- What if rank was a tuple, (r_1, r_2)?
  - Let r_1 be geometrically distributed
  - Let r_2 be uniformly distributed from $[1, \log^c n]$

- Compare ranks lexicographically

☑ Metadata size $O(\log \log n)$? - $O(\log \log n) + O(c \log \log n)$

Zip-zip Trees

- What if rank was a tuple, (r_1, r_2)?
  - Let r_1 be geometrically distributed
  - Let r_2 be uniformly distributed from $[1, \log^c n]$
- Compare ranks lexicographically
- ☑ Metadata size $O(\log\log n)$? - $O(\log\log n) + O(c \log\log n)$
- Hope: fewer collisions, better depth?

Zip-zip Trees Example

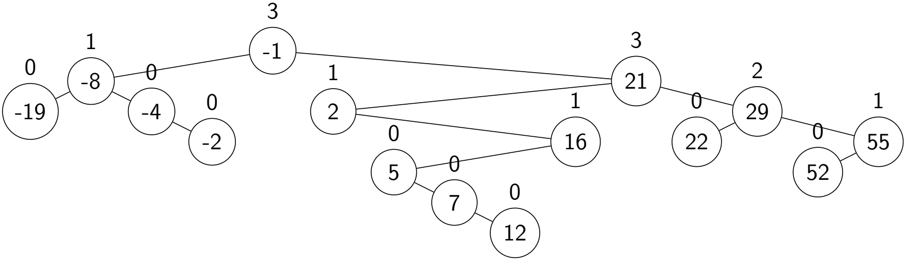


Figure 2: A zip tree

Zip-zip Trees Example

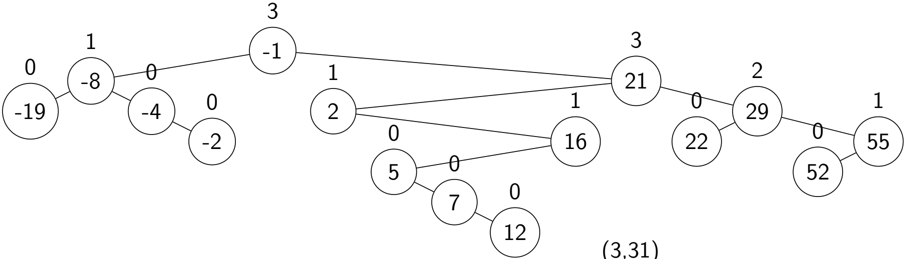


Figure 2: A zip tree

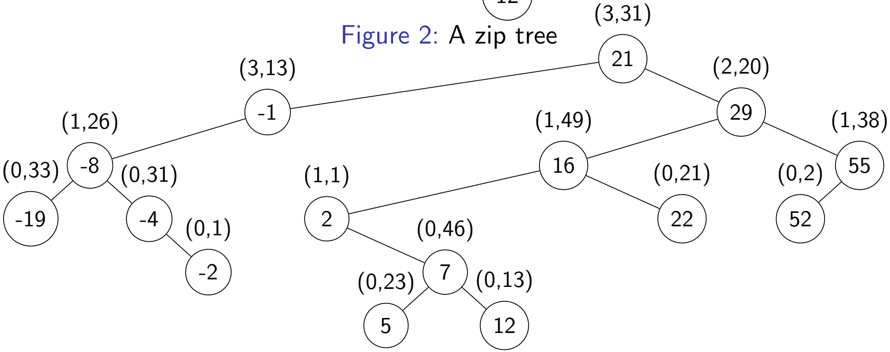
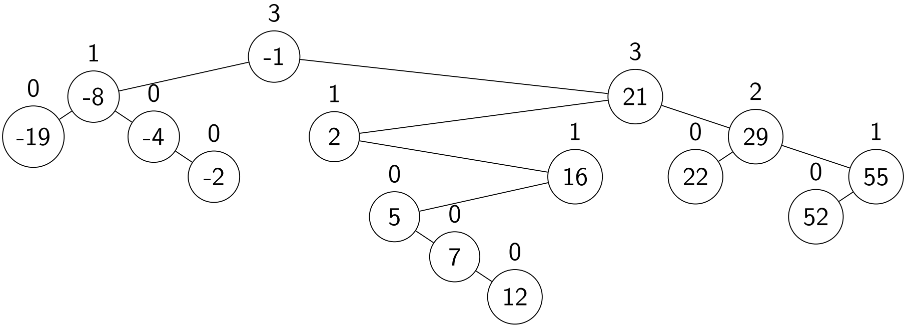
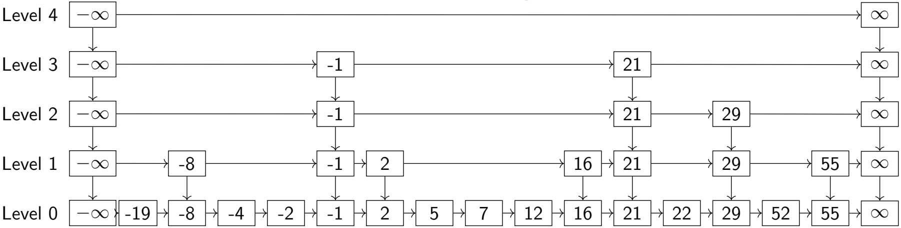


Figure 3: A random zip-zip tree generated from the above zip tree

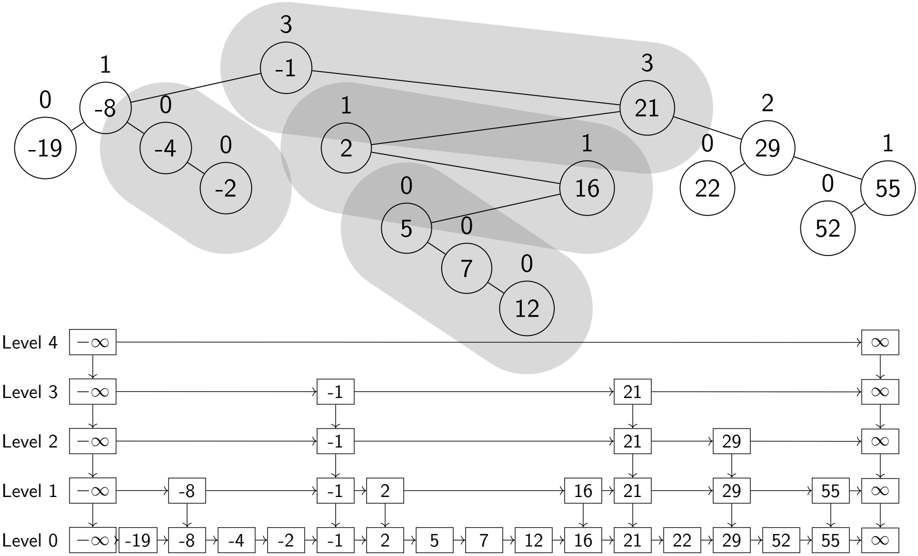
Zip-zip Trees Analysis I

- Idea: Consider rank groups



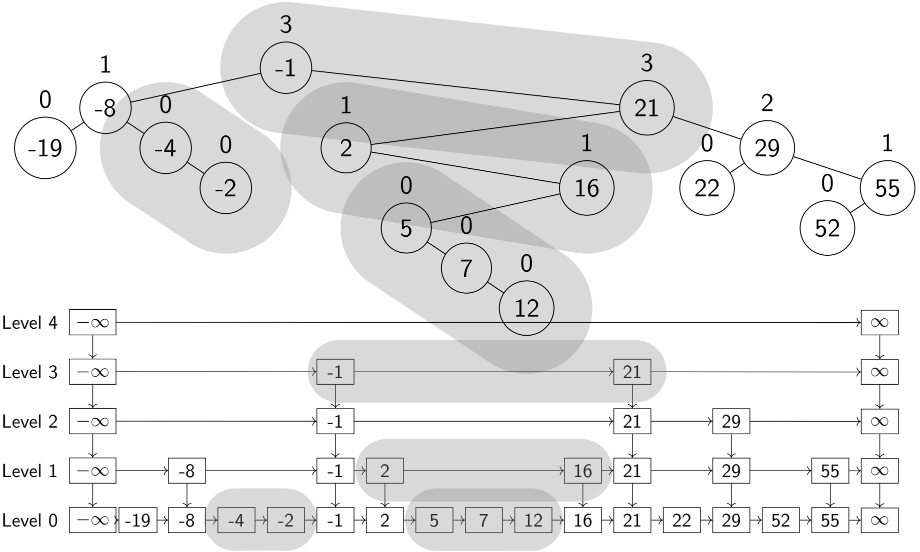


- Idea: Consider rank groups



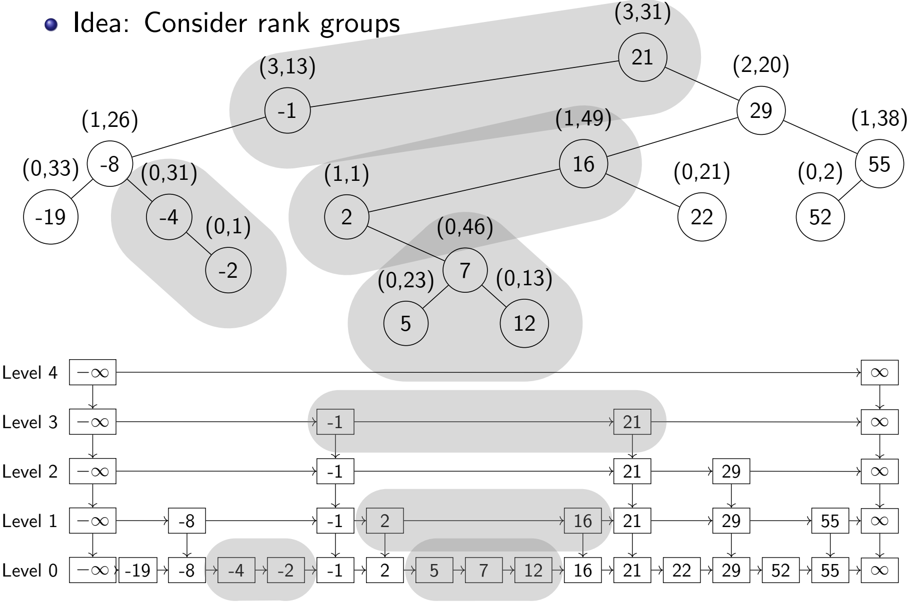
Zip-zip Trees Analysis I

- Idea: Consider rank groups



Zip-zip Trees Analysis I

- Idea: Consider rank groups



Zip-zip Trees Analysis II

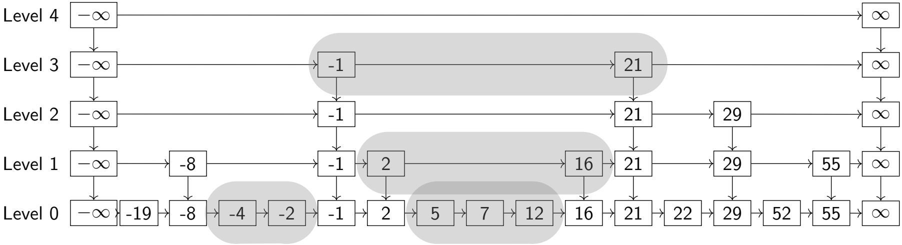
- How big are rank groups?

Lemma

The size of an r_1 rank group has expected value 2 and is $< 2\log n$ w.h.p.

Proof Sketch.

- Size is (at most) geometrically distributed $\square$



Zip-zip Trees Analysis III

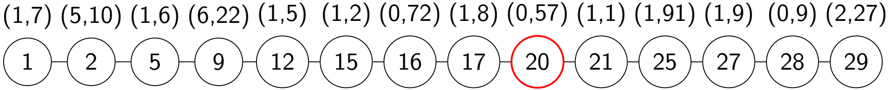
Theorem

The expected depth, δ_j, of the j-th smallest key in a zip-zip tree is $H_j + H_{n-j+1} - 1 + o(1)$

Zip-zip Trees Analysis III

Theorem

The expected depth, δ_j, of the j-th smallest key in a zip-zip tree is $H_j + H_{n-j+1} - 1 + o(1)$



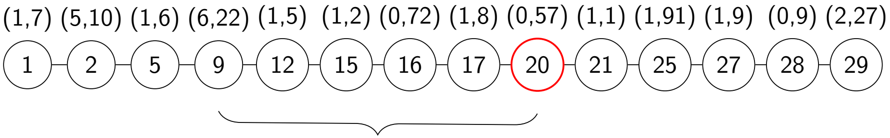
Zip-zip Trees Analysis III

Theorem

The expected depth, δ_j, of the j-th smallest key in a zip-zip tree is $H_j + H_{n-j+1} - 1 + o(1)$

Proof Sketch.

- A node x_i is an ancestor of node x_j iff x_i has maximum rank in $[i, j]$



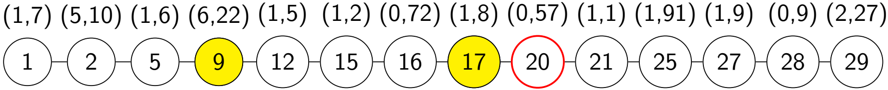
Zip-zip Trees Analysis III

Theorem

The expected depth, δ_j, of the j-th smallest key in a zip-zip tree is $H_j + H_{n-j+1} - 1 + o(1)$

Proof Sketch.

- A node x_i is an ancestor of node x_j iff x_i has maximum rank in $[i, j]$



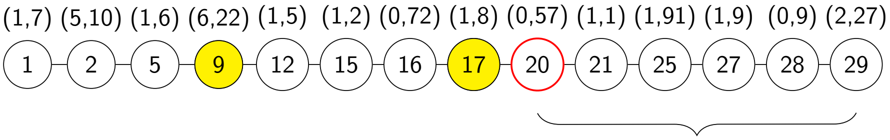
Zip-zip Trees Analysis III

Theorem

The expected depth, δ_j, of the j-th smallest key in a zip-zip tree is $H_j + H_{n-j+1} - 1 + o(1)$

Proof Sketch.

- A node x_i is an ancestor of node x_j iff x_i has maximum rank in $[i, j]$



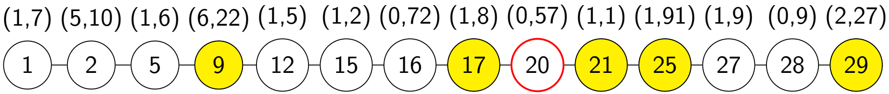
Zip-zip Trees Analysis III

Theorem

The expected depth, δ_j, of the j-th smallest key in a zip-zip tree is $H_j + H_{n-j+1} - 1 + o(1)$

Proof Sketch.

- A node x_i is an ancestor of node x_j iff x_i has maximum rank in $[i, j]$



Zip-zip Trees Analysis III

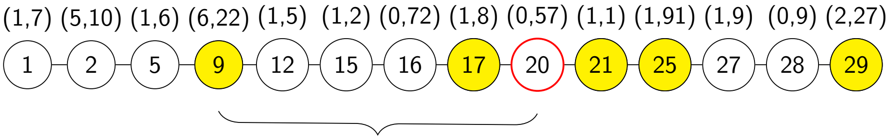
Theorem

The expected depth, δ_j, of the j-th smallest key in a zip-zip tree is $H_j + H_{n-j+1} - 1 + o(1)$

Proof Sketch.

- A node x_i is an ancestor of node x_j iff x_i has maximum rank in $[i, j]$

$$P(x_i \prec x_j) = \frac{1}{|i-j|+1}$$



Zip-zip Trees Analysis III

Theorem

The expected depth, δ_j, of the j-th smallest key in a zip-zip tree is $H_j + H_{n-j+1} - 1 + o(1)$

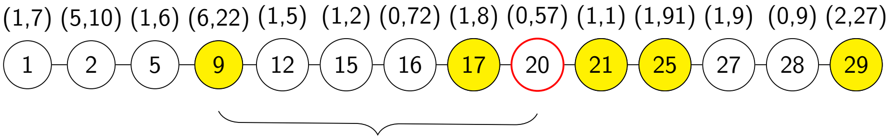
Proof Sketch.

- A node x_i is an ancestor of node x_j iff x_i has maximum rank in $[i, j]$

$$P(x_i \prec x_j) = \frac{1}{|i-j|+1}$$

$$\delta_j = \sum_{i=1}^{j} \frac{1}{j-i+1} + \sum_{i=j+1}^{n} \frac{1}{i-j+1} = H_j + H_{n-j+1} - 1$$

$\square$



Zip-zip Trees Analysis IIII

Theorem

The expected depth, δ_j, of the j-th smallest key in a zip-zip tree is $H_j + H_{n-j+1} - 1 + o(1)$

Corollary

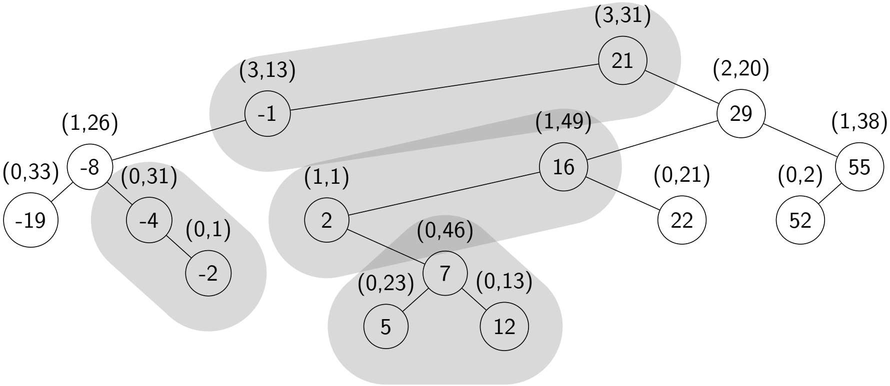
The expected depth of the min and max keys is $0.69 \log n + \gamma + o(1)$

Corollary

The expected depth of any key is at most $1.39 \log n - 1 + o(1)$

Just-in-Time (JIT) Zip-zip Trees

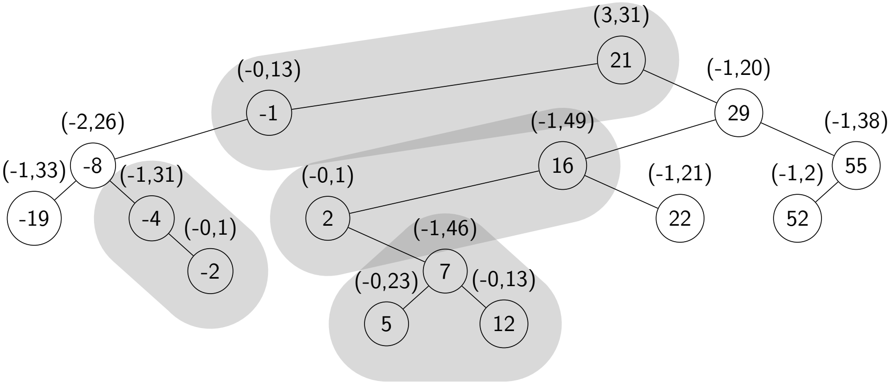
- Ranks can be up to $O(\log n)$, but don't differ much



1

Just-in-Time (JIT) Zip-zip Trees

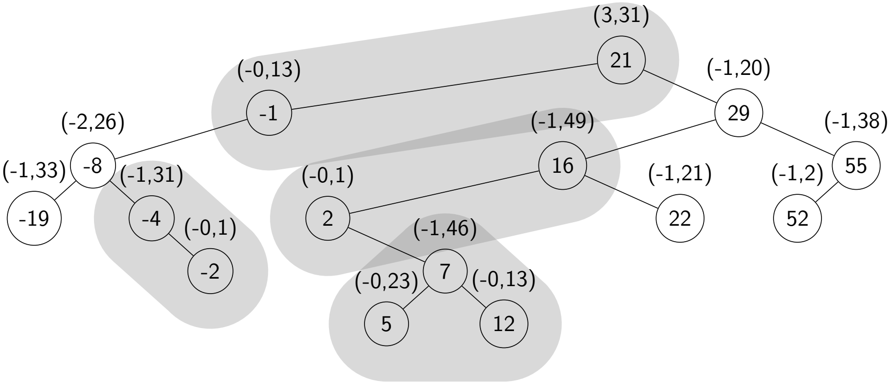
- Ranks can be up to $O(\log n)$, but don't differ much
  - Store r_1 rank differences! (Expected $O(1)$)



1

Just-in-Time (JIT) Zip-zip Trees

- Ranks can be up to $O(\log n)$, but don't differ much
  - Store r_1 rank differences! (Expected $O(1)$)
- Rank groups are small...

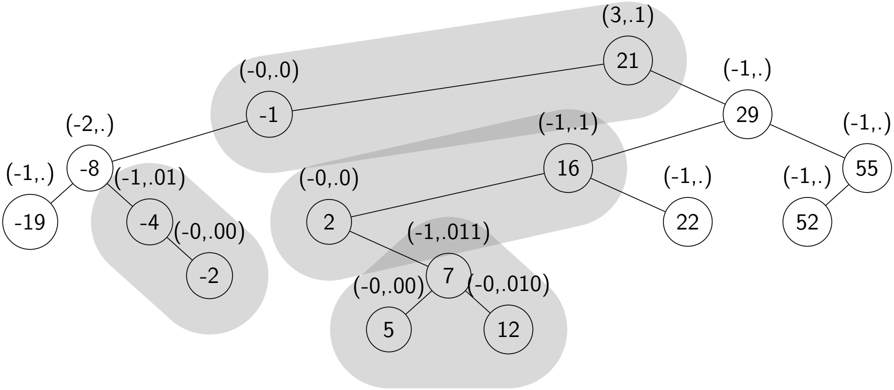


1

Just-in-Time (JIT) Zip-zip Trees

- Ranks can be up to $O(\log n)$, but don't differ much
  - Store r_1 rank differences! (Expected $O(1)$)

- Rank groups are small...
  - Generate r_2 ranks on the fly! (Expected $O(1)$)[1]



[1] r_1 differences are $O(1)$ per node, r_2 are $O(1)$ per operation

Depth Discrepancy

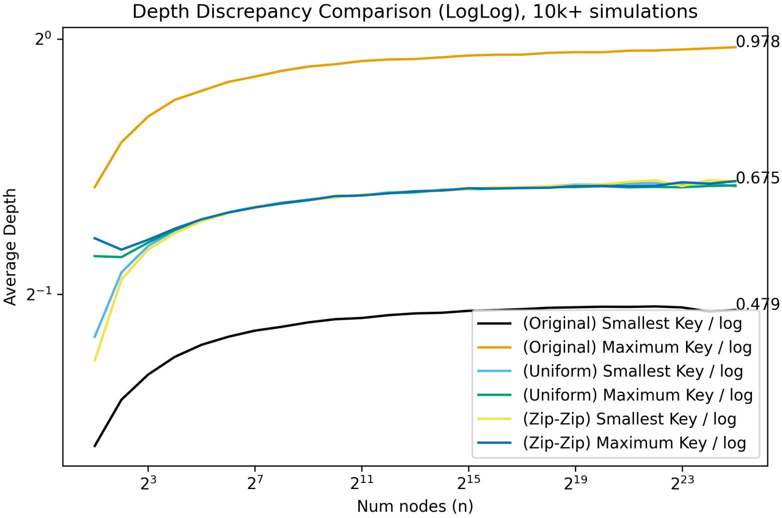


Figure 4: The depth discrepancy between the min and max keys for three variants

Average Key Depth and Tree Height

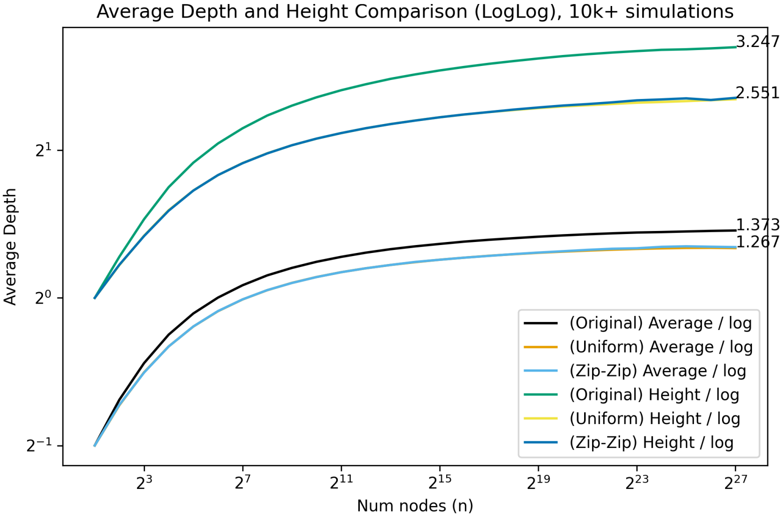


Figure 5: The average node depth and tree height for three variants

Rank Collisions

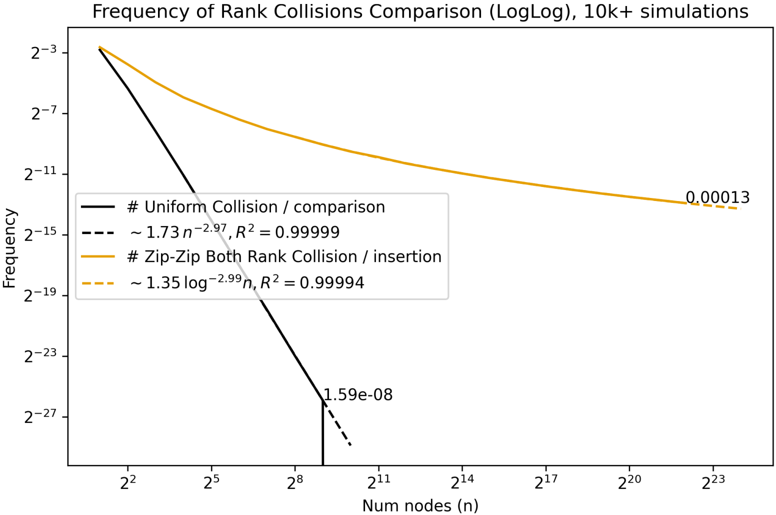


Figure 6: The frequency of encountered rank ties per rank comparison for the uniform variant and per element insertion for the zip-zip variant

Just-in-Time Zip-zip Tree Size

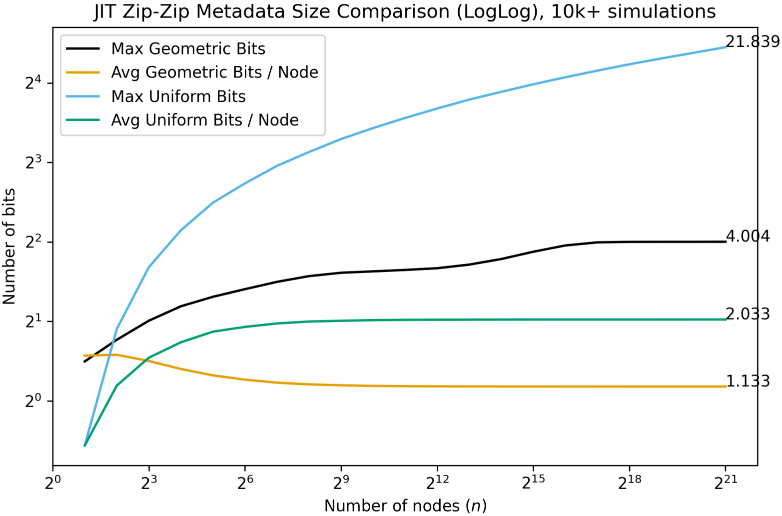


Figure 7: The metadata size for the just-in-time implementation

Summary

- 2023 – Zip-zip tree
  - ☑ $1.39 \log n$ expected average depth
  - ☑ Space cost: $\log \log n$ bits per node (or $O(1)$ bits per update w.h.p.)
  - ☑ Easy to implement
  - ☑ Strongly history independent (except JIT)
  - ☑ May be partially persistent
  - ☑ Supports biased keys (still $O(\log \log n)$ bits per node)

Summary

- 2023 – Zip-zip tree
  - ☑ $1.39 \log n$ expected average depth
  - ☑ Space cost: $\log \log n$ bits per node (or $O(1)$ bits per update w.h.p.)
  - ☑ Easy to implement
  - ☑ Strongly history independent (except JIT)
  - ☑ May be partially persistent
  - ☑ Supports biased keys (still $O(\log \log n)$ bits per node)

- Any questions?