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

Table of Contents

Motivation

- Uses of Binary Search Trees (BSTs)?
  - Priority queues, lookup tables, link-cut, dynamic sets, ...
- Why?
  - Insert : $\mathcal{O}(\log n)\mathcal{O}(h)$
  - Delete: $\mathcal{O}(\log n)\mathcal{O}(h)$
  - Search: $\mathcal{O}(\log n)\mathcal{O}(h)$
  - Space : $\mathcal{O}(n)$
- How to balance efficiently?
- How much is balancing going to cost?

History

- circa 1960 – BSTs discovered [2]
- 1962 – AVL tree [1] - complicated
  - $1.44 \log n$ height worst case
  - Time cost: amortized constant per insertion
  - Space cost: 2 bits per node $\mathcal{O}(1)$
- 1989 – Treap [3] - space inefficient
  - $1.39 \log n$ expected average depth w.h.p.
  - Time cost: expected constant w.h.p.
  - Space cost: $\mathcal{O}(\log n)$ bits per node
- 2018 – Zip tree [4] - unbalanced
  - $1.5 \log n$ expected average depth w.h.p.
  - Time cost: expected constant w.h.p.
  - Space cost: $\log \log n$ bits per node

Our BST

- 2018 – Zip tree [4] - unbalanced
  - 1.5 $\log n$ expected average depth
  - Time cost: expected constant w.h.p.
  - Space cost: $\log \log n$ bits per node

- We would like:
  - ☐ Lower expected average depth - 1.39 $\log n$
  - ☐ Same, low time cost
  - ☐ At least as good space cost - either $\log \log n$ or $\mathcal{O}(1)$ w.h.p.
  - ☐ Maintain history independence? - may be strongly history independent
  - ☐ Persistent? - partially persistent
  - ☐ Biased? - keys can be biased in a natural way

- 2023 – Zip-zip trees

Table of Contents

Skip List

- Sorted vector: Insert / Delete $\mathcal{O}(n)$, Find $\mathcal{O}(\log n)$
- Sorted linked-list: Find: $\mathcal{O}(n)$, Insert / Delete $\mathcal{O}(1)$ after find
- 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

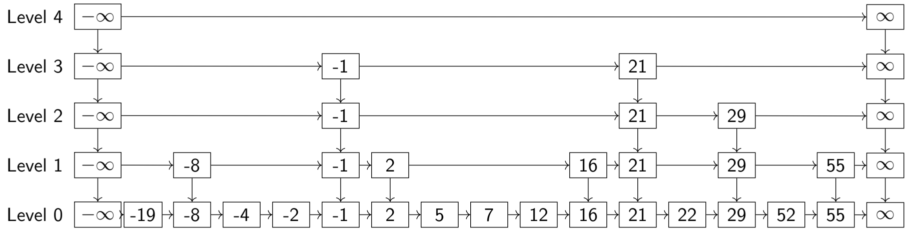


Figure 1: A sorted linked list A skip list with one coin flip A skip list with two coin flips A skip list with three coin flips A skip list with four coin flips

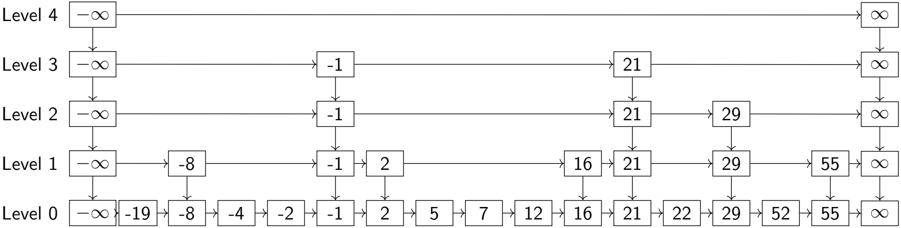
Skip List Height

Theorem

The height of a skip list is less than $\log n + f(n)$ *w/ probability* $1 - 2^{-f(n)}$

Proof.

- Each node has height of geometric random variable w/ $p = 1/2$, X_i
- $\Pr(X_i > \log n + f(n)) < 2^{-(\log n + f(n))} = 2^{-f(n)}/n$
- Let $X = \max\{X_1, X_2, ..., X_n\}$
- By union bound, $\Pr(X > \log n + f(n)) < 2^{-f(n)}$ □



Skip List Size

Theorem

The size of a skip list is expected to be $2n$

Proof.

- $\mathbb{E}(X_i) = \sum_{k=1}^{\infty} k \times 2^{-k} = 2$
- From linearity of expectations, $\mathbb{E}(\sum_i X_i) = 2n$ □

- This is not good!

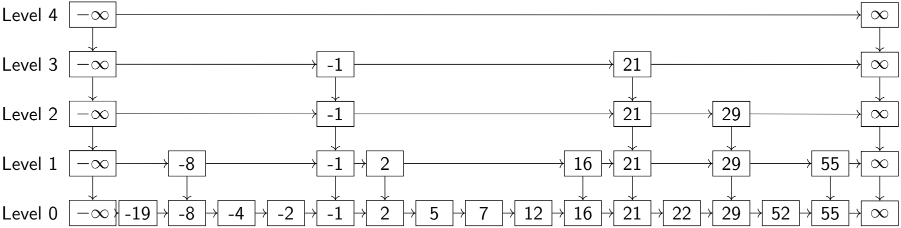
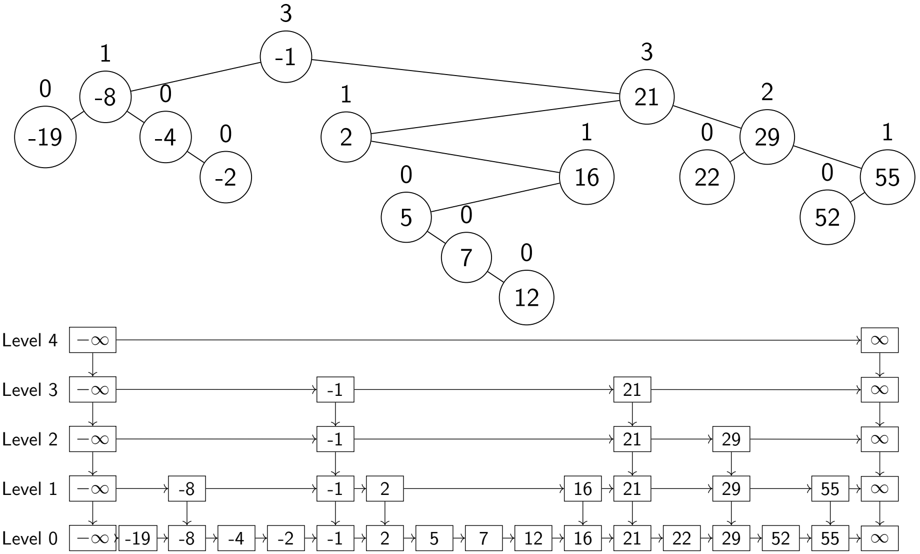


Table of Contents

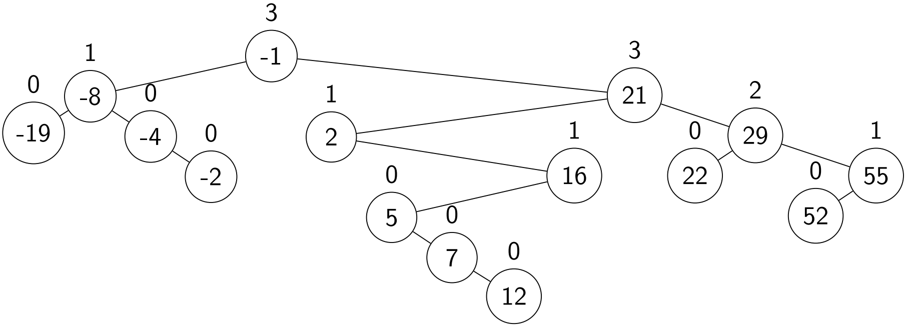
Zip Tree

- Idea: Construct a BST from a skip-list - flat-out better!



Zip Tree Structure I

- What is the structure?
- It's a BST, so $x.left.key < x.key < x.right.key$
- Each node has a geometrically distributed 'rank'
- $x.rank > x.left.rank$, $x.rank \geq x.right.rank$
- Not symmetric!



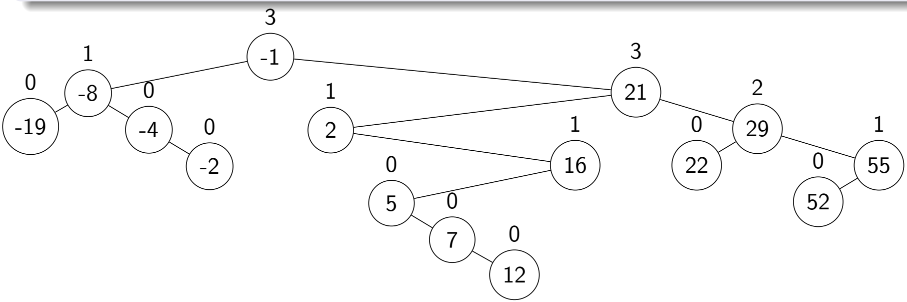
Zip Tree Structure II

Lemma

If root has rank k, then the expected depth of the max key is at most k

Proof.

- Nodes on path above minimum value have increasing rank
- Difference between them is geometrically distributed
- Average increase is 1, length is expected $k/1 = k$ if max has rank 0 □



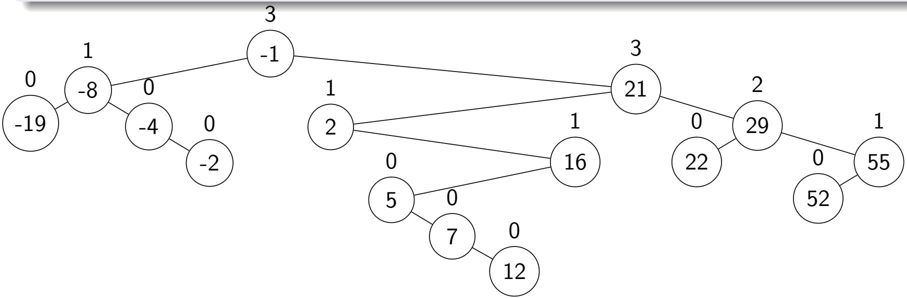
Zip Tree Structure III

Lemma

If root has rank k, then the expected depth of the min key is at most $k/2$

Proof.

- Nodes on path above minimum value have *strictly* increasing rank
- Difference between them is geometrically distributed
- Average increase is 2, length is expected $k/2$ if min has rank 0 □



Zip Tree Structure IIII

- Recall:
  - Max key expected depth k
  - Min key expected depth $k/2$[1]
- Asymmetric!
- Recall: Height of skip list is $< \log n + f(n)$ w/ probability $1 - 2^{-f(n)}$

Theorem

The average depth of a node in a zip tree is $1.5 \log n$

Proof.

- Average rank of the root is $\log n + O(1)$
- Average rank of arbitrary node is $1 \rightarrow$ Average k is $\log n + O(1)$
- Arbitrary node expected depth $= k + k/2 = 1.5 \log n$ □

[1] Let k be the rank difference to root

Zip Tree Problem

- Zip tree expected depth: $1.5 \log n$
- Treap expected depth: $1.39 \log n$

- What is a treap?
  - Uniformly distributed ranks
  - If collision, rebuild $\rightarrow$ no collisions!

- Why difference? Collisions $\rightarrow$ Unbalance

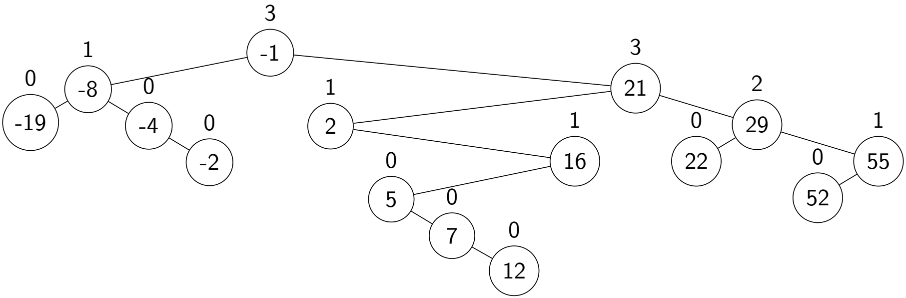


Table of Contents

Uniform Zip Trees

- Problem with zip tree? Collisions $\rightarrow$ Unbalance
- What if few collisions?
- Idea:
    - Pick ranks from large enough range uniformly, $[1, n^c]$
    - When collision, break ties like zip tree
- This works but...
- Metadata space is $c \log n$
- We want (at least) $O(\log \log n)$!

Table of Contents

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

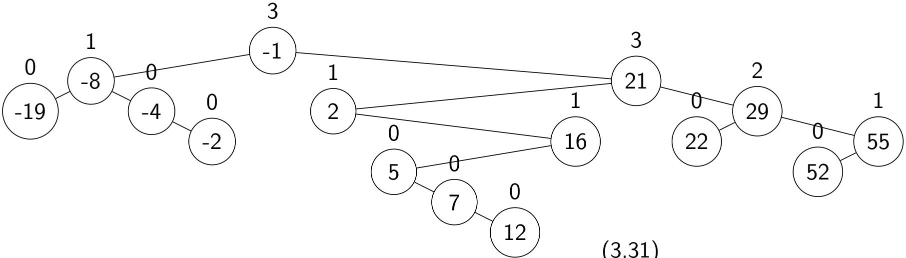


Figure 2: A zip tree

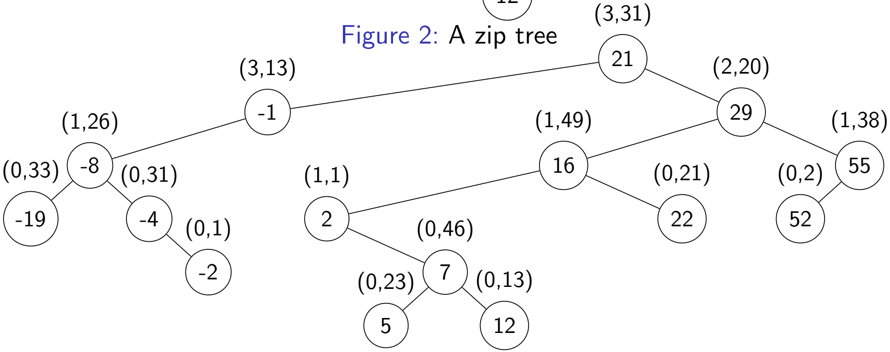
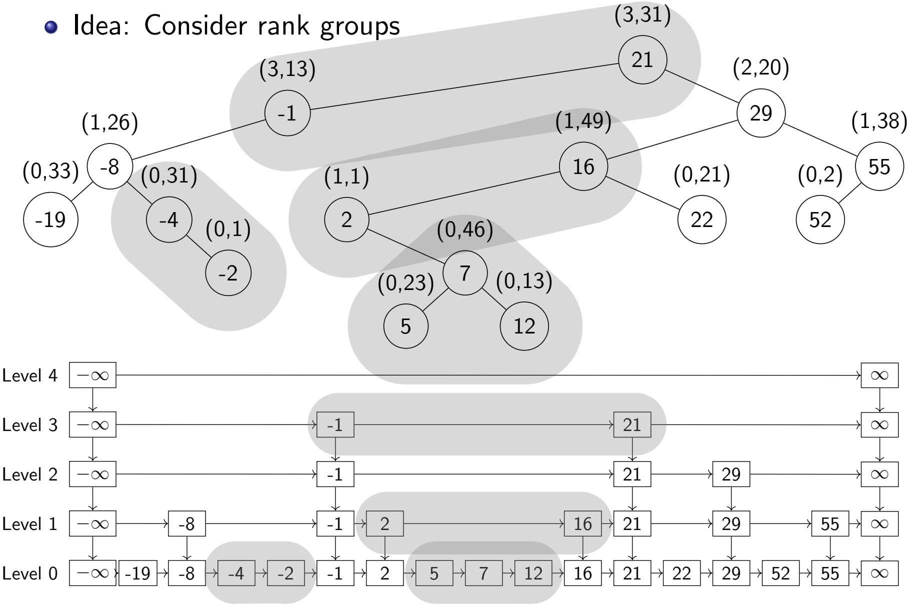


Figure 3: A random zip-zip tree generated from the above zip tree

Zip-zip Trees Analysis I

- Idea: Consider rank groups



Zip-zip Trees Analysis II

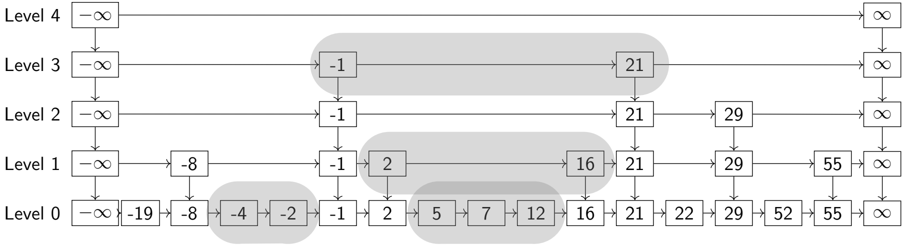
- How big are rank groups?

Lemma

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

Proof.

- Size is (at most) geometrically distributed □



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.

- Rank of the root $< 3 \log n$ w.h.p
- Probability there are r_2 rank collisions is negligible w.h.p.
- Bound follows assuming low-ranked root & no collisions □

Corollary

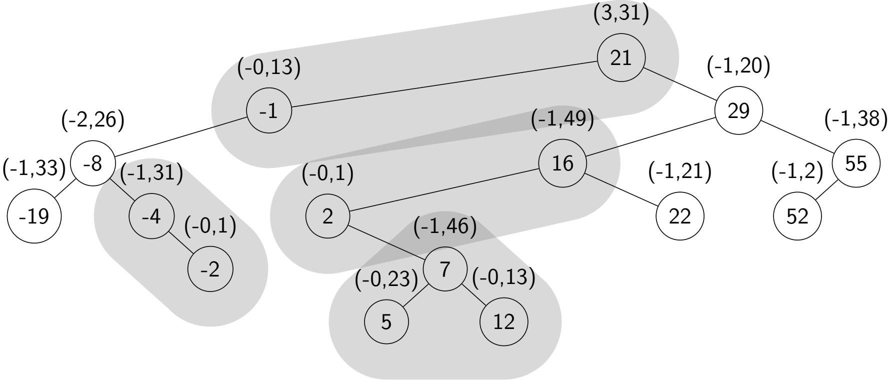
The expected depth of the min and max keys is $0.69 \log n + \gamma + o(1)$

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
  - Store r_1 rank differences! (Expected $O(1)$)
- Rank groups are small...
  - Generate r_2 ranks on the fly! (Expected $O(1)$)[2]



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

Table of Contents

Depth Discrepancy

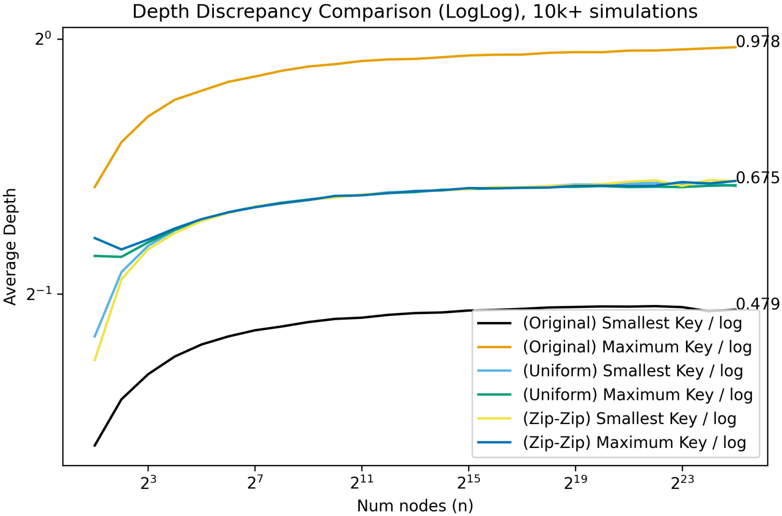


Figure 4: The depth discrepancy between the min and max keys for three variants

Average Key Depth and Tree Height

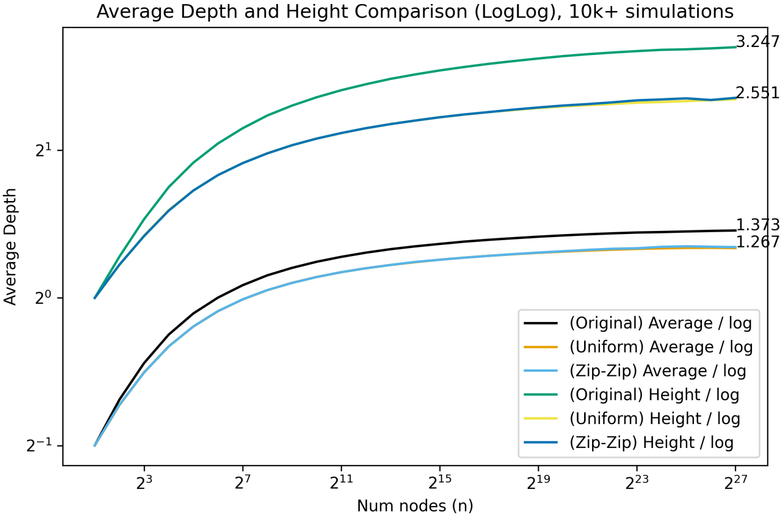


Figure 5: The average node depth and tree height for three variants

Rank Collisions

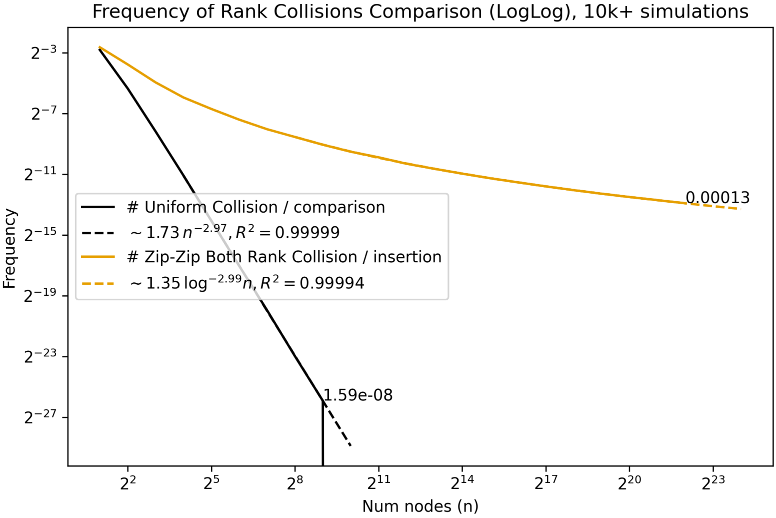


Figure 6: The frequency of encountered rank ties per rank comparison for the uniform variant and per element insertion for the zip-zip variant

Just-in-Time Zip-zip Tree Size

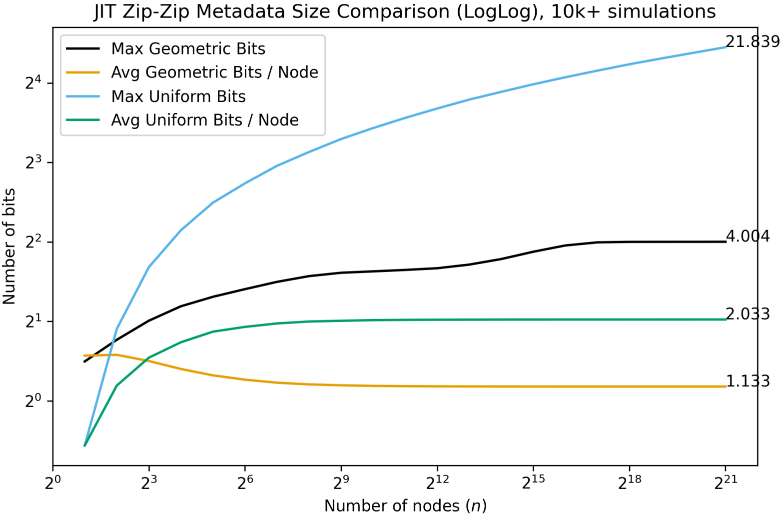


Figure 7: The metadata size for the just-in-time implementation

Table of Contents

Summary

- 2023 – Zip-zip tree
  - ☑ 1.39 $\log n$ expected average depth
  - ☑ Time cost: expected constant w.h.p.
  - ☑ 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)

Table of Contents

References I

- M. AdelsonVelskii and E. M. Landis, *An algorithm for the organization of information*, Defense Technical Information Center. OCLC: 227312191.
- J. Culberson and J. I. Munro, *Explaining the behaviour of binary search trees under prolonged updates: A model and simulations¶*, 32, pp. 68–75.
- B. Haeupler, S. Sen, and R. E. Tarjan, *Rank-balanced trees*, 11, pp. 30:1–30:26.
- R. E. Tarjan, C. C. Levy, and S. Timmel, *Zip trees*, 17, pp. 1–12.