Zip Trees

CS 165 - Project in Algorithms and Data Structures
Spring 2021

Presenter: Haleh Havvaei

Some slides adapted from slides from:
Robert E. Tarjan
Princeton University and Intertrust Technologies
A zip tree is a binary search tree in which each node has a numeric rank and the tree is (max)-heap-ordered with respect to ranks, with rank ties broken in favor of smaller keys.

(image from Street Art on Pinterest)
Zip Trees

Idea: On insertion, choose a height for an item and insert it at the given height, or close to it. Choose heights like those in a best-case BST: \( \frac{1}{2} \) the nodes at height 0, \( \frac{1}{4} \) at height 1, \( \frac{1}{8} \) at height 2...

Choose the heights randomly.
We cannot choose heights exactly.

Instead, for each node to be inserted we choose a rank, as follows: flip a fair coin and count the number of heads before the first tail. The rank of a node does not change while it is in the tree.

The rank of a node has a geometric distribution: a node has rank $k$ with probability $1/2^{k+1}$.
**Zip* Tree**

A binary search tree in which each node has a rank chosen randomly on insertion, with nodes symmetrically ordered by key and heap ordered by rank, breaking rank ties in favor of smaller key:

\[
x.\text{left.key} < x.\text{key} < x.\text{right.key}
\]

\[
x.\text{left.rank} < x.\text{rank}
\]

\[
x.\text{right.rank} \leq x.\text{rank}
\]

*Zip: “to move very fast”*
A Zip Tree
Zip tree insertion: Insert key “K”

Choose the rank of a node randomly. Here, let the rank to be 3.
- Search for the node with key ’K’ in the tree

Image from the paper by Tarjan et. al.
Zip tree insertion: Insert key “K”

Choose the rank of a node randomly. Here, let the rank to be 3.

- Search for the node with key 'K' in the tree

Image from the paper by Tarjan et. al.
Zip tree insertion: Insert key “K” (cont.)

Find a node \( y \) with \( y.\text{rank} \leq x.\text{rank} \)
with strict inequality if \( y.\text{key} < x.\text{key} \)

\[ y.\text{rank} < 3 \quad \Rightarrow \quad y = \text{node } H \]

Insert \( K \)

keys less than \( x.\text{key} \)
keys greater than \( x.\text{key} \)
Zip Tree Deletion: Delete key “K”

- Search for the node with key ’K’ in the tree

Delete K
Zip Tree Deletion: Delete key “K”

- Search for the node with key ’K’ in the tree

Form a single path R by merging them from top to bottom in non-increasing rank order, breaking a tie in favor of the smaller key.
Static properties of zip trees

The structure of a zip tree is uniquely determined by the keys and ranks of its nodes.

Expected root rank : $\lg n + O(1)$

Root rank is $O(\log n)$ with high probability

$E(\text{node depth}) = 1.5\lg n + O(1)$ (Prodinger 1996)

Tree depth is $O(\log n)$ with high probability

For the proof read the paper referenced on the last slide
Dynamic properties of zip trees

Expected restructuring time to insert or delete a node of rank $k = O(k)$.

Expected restructuring time is $O(1)$

Probability of restructuring taking $O(k)$ time is exponentially small in $k$

Insertion/deletion can be done purely top-down