

Skip List Data Structure

Note Title

10/31/2007

Motivation:

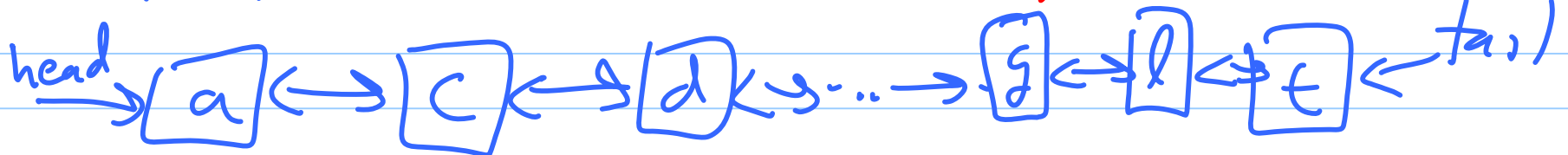
Sorted array



Find pred.
of k

$\lceil \log_2 n \rceil$
comp.

Sorted linked list

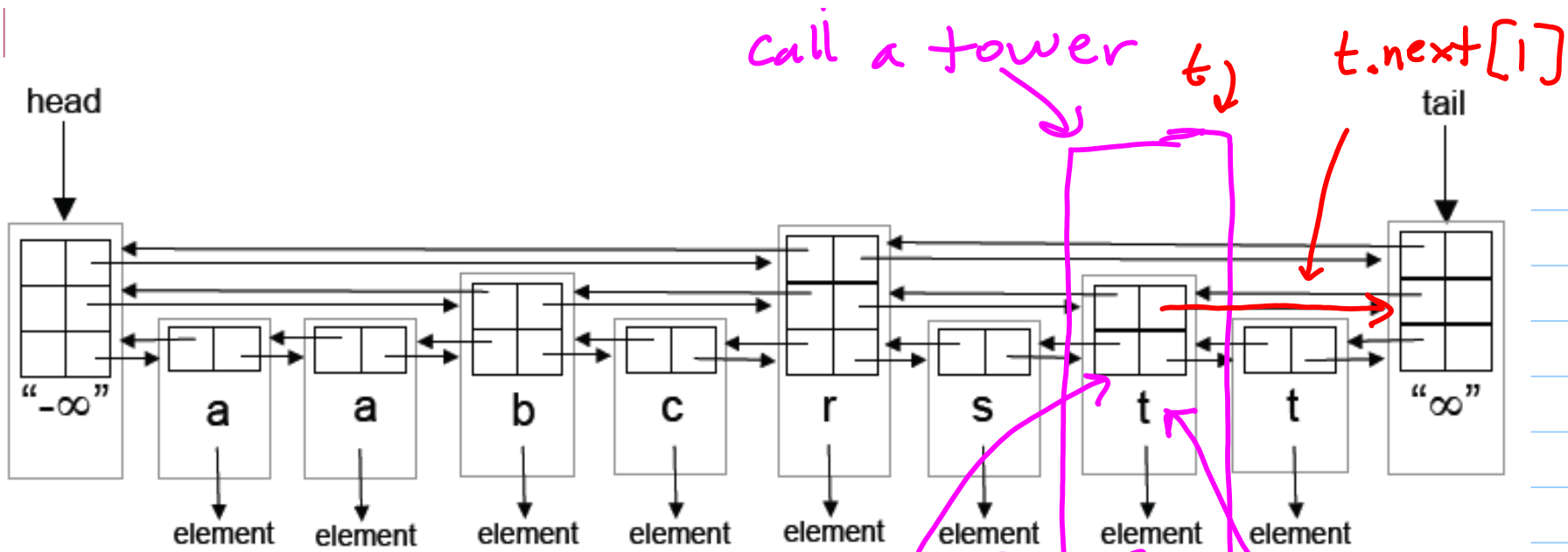




List 2 sparse version of List 1

List 1 sparse version of List 0

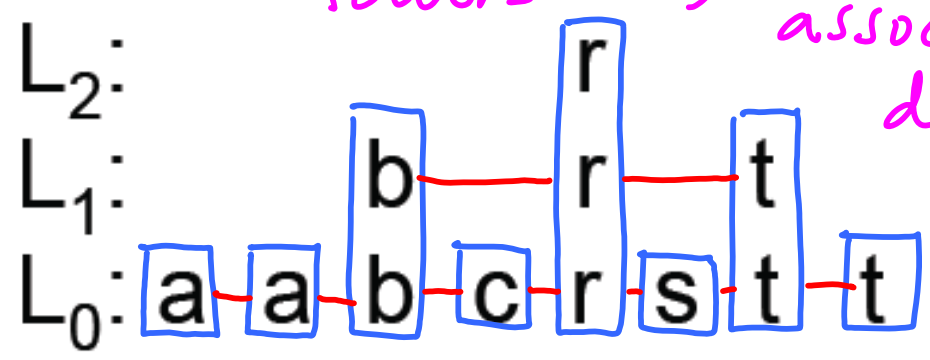
List 0 holds all elements



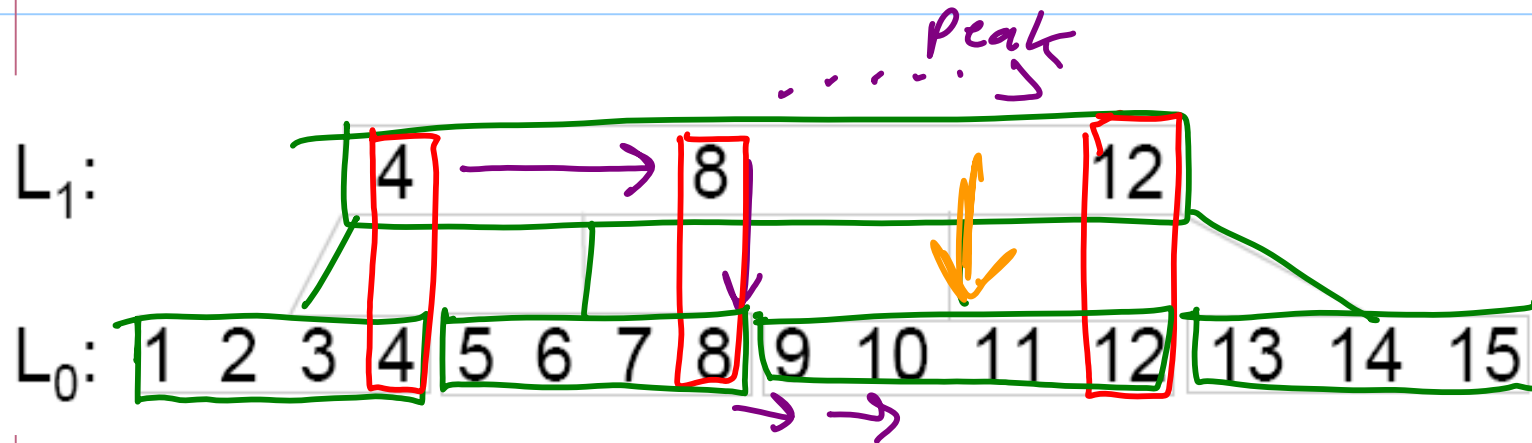
call a tower t_i $t_i.next[i]$

tower holds: array of refs to next/prev towers, ref to tag associated data,

abstract view



Relationship between Skip List + B⁺-tree



B⁺-tree

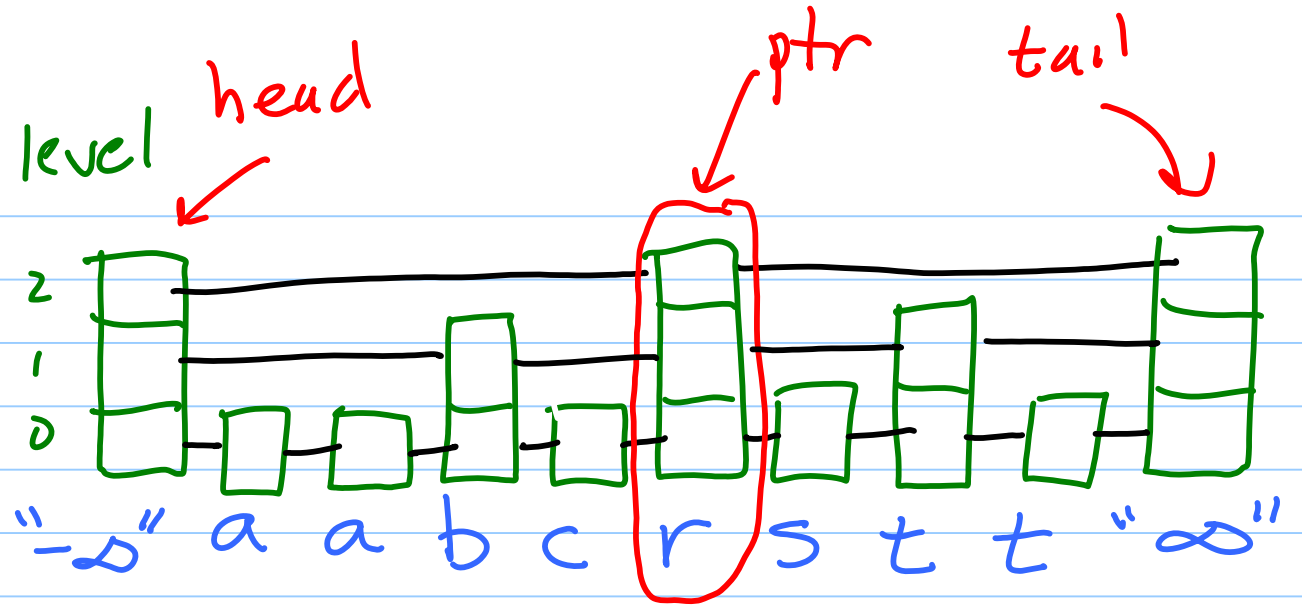
Search 10 in B⁺-tree

Skip List

Search 10 in Skip List

L₂: r
 L₁: b r t
 L₀: a a b c r s t t

Show
 as →

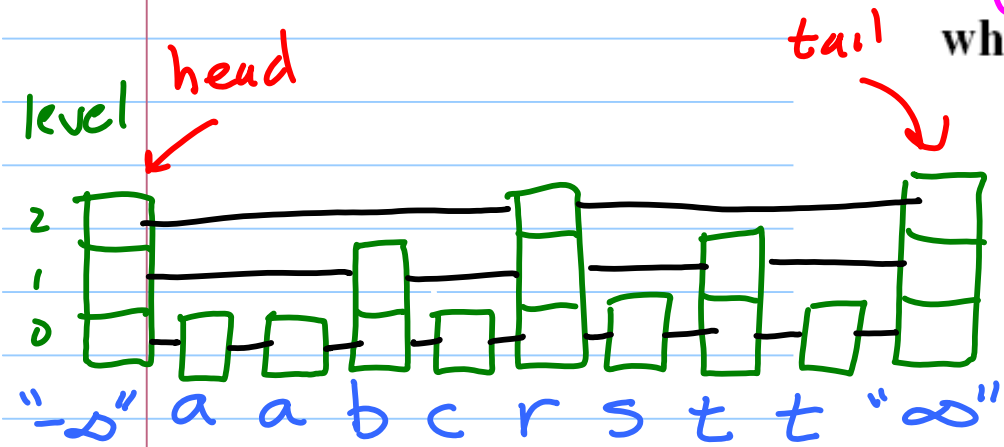


How do you find min? `head.next[0]`

max? `tail.prev[0]`

How do you move to next/prev element from a given tower? `ptr.next[0]` `ptr.prev[0]`

L₂: r
 L₁: b r t
 L₀: a a b c r s t t



```

Tower<E> findFirstOccurrence(E target) {
    Tower<E> left = head;
    Tower<E> right = tail;
    int level = height - 1;
    while (level ≥ 0) {
        Tower<E> next = left.next(level);
        if (right == next)
            level--;
        else {
            if (comp.compare(target, next.element) > 0)
                left = next;
            else
                right = next;
            level--;
        }
    }
    return right;
}

```

look in for

peak ahead

drop down

target > next

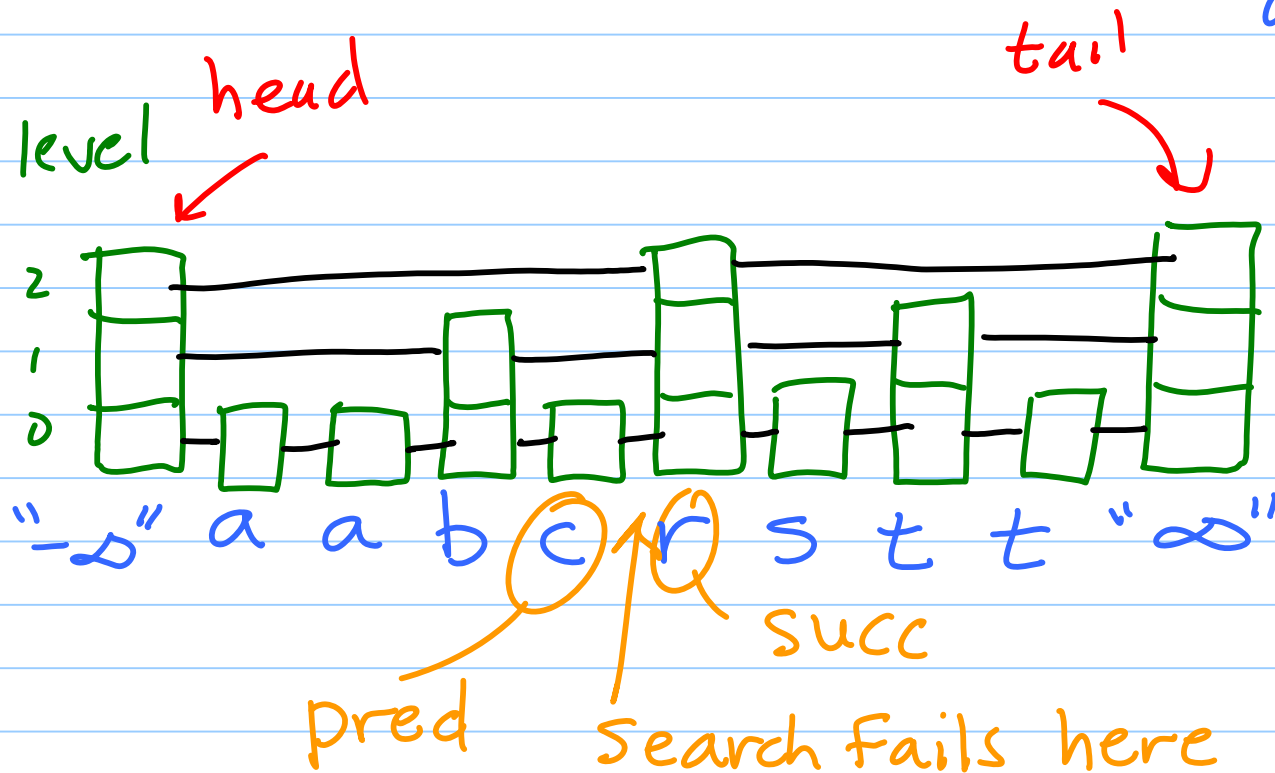
move forward

drop down + reset right

first occurrence or where to insert

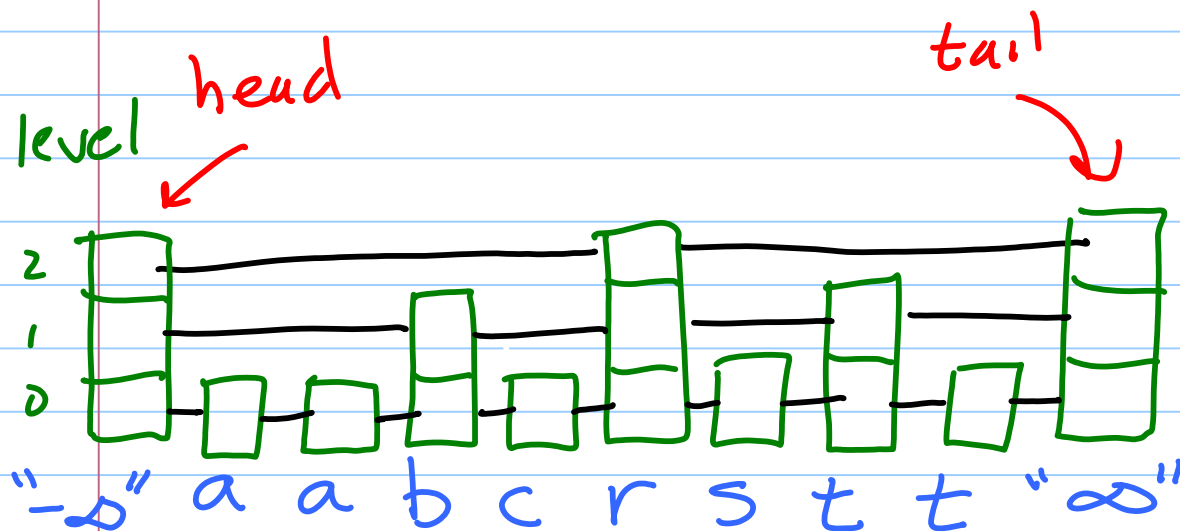
Search for t
 Search for d

How can you find predecessor/successor of an element (e.g. d)?



Use level 0 to navigate in sorted order

Insertion



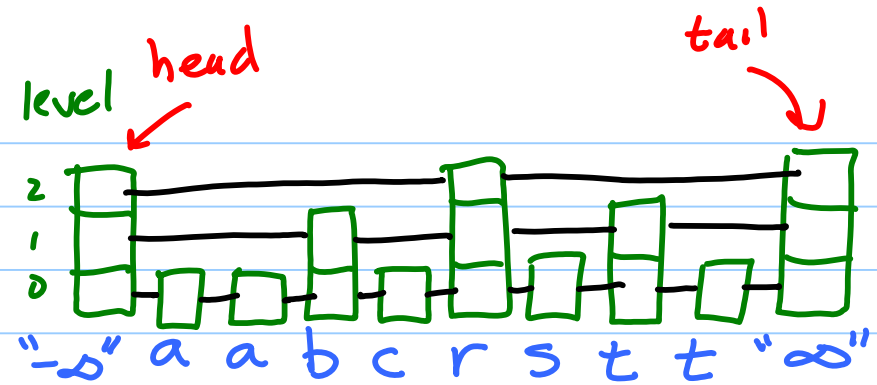
Let p be
prob of a
tail (continue).
Ex. $p = 1/4$

Find position to insert using search
Then randomly pick height of new tower
as # biased coin flips until head obtained

Once the position is found and height is selected, just splice new tower into list for each level (already know where from search).

Also update height (+ occasionally must resize head/tail).

Deletion



Search to find \pm then splice out of each level it is in (just like in a linked list).

Also update height (\pm occasionally must resize head/tail).

Analysis

expected

(What is height (# of levels in tallest tower)?

X (How many times per level do we expect to move (or peak) forward)?

Let n be # elements (towers)

Claim expect $n \cdot p^i$ elements
in level i list

$$i=0 \quad n$$

$$i=1 \quad n \cdot p$$

$$i=2 \quad n \cdot p^2$$

⋮

$$n$$

$$n/4$$

$$n/16$$

⋮

↙ For $p = 1/4$

Can prove by induction.

When is $n \cdot p^i = \frac{1}{p}$

when $p = \frac{1}{4}$
only 4
items
expected
at level i

Solve for i

$$n = \left(\frac{1}{p}\right)^{i+1} \Rightarrow i+1 = \log_{\frac{1}{p}} n$$

tallest
tower

$$i = \log_{\frac{1}{p}} n - 1$$

Level when
expect $\frac{1}{p}$
elements to
remain

$$E[\text{height}] = \log_{\frac{1}{p}} n + \frac{1}{1-p}$$

Ex $p = \frac{1}{4}$

$$E[\text{height}] = \log_4 n + \frac{4}{3}$$

We'll finish analysis next time.

Skip List Overview

	As a function of p	$p = 1/2$	$p = 1/e$	$p = 1/4$
space usage	$2n/(1-p) + 4 \log_{1/p} n$	$4n + 4 \log_2 n$	$\approx 3.2n + 2.8 \log_2 n$	$\approx 2.67n + 2 \log_2 n$
search cost	$(\log_{1/p} n)/p$	$2 \log_2 n$	$\approx 1.88 \log_2 n$	$2 \log_2 n$