skip List Data Structure

Motivation:

Sorted array

Sorted linked list

Find pred. of \( k \)

\( \log_n \) comp.

\( k \leq f? \) no \( k \leq g? \) no

pred of \( k \leq d? \) yes

\( k \) goes after this

head \( \rightarrow a \) \( \rightarrow c \) \( \rightarrow d \) \( \cdots \) \( \rightarrow e \) tail
List 0 holds all elements
List 1 sparse version of List 0
List 2 sparse version of List 1

Search f
Search g

head

= [a, b, b, c, d, a, g, g, k, m, n, p, p, s, l]

peak

peak

put f here

put g here
call a tower $t_j$

t.$next[1]$

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

abstract view

$\{ L_2 : \}

L_1 : \quad \begin{array}{cccc} b & r & r & t \\ \end{array}

L_0 : \quad \begin{array}{cccccccccccc} a & a & a & b & c & r & s & t & t & t \\ \end{array}$
Relationship between Skip List + $B^+$-tree

$L_0$: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

$L_1$: 4 $\rightarrow$ 8 $\rightarrow$ 12

$B^+$-tree
Search 10 in $B^+$-tree

Skip List
Search 10 in Skip List
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]`
How do you search for an element?

level

head

peak

level = X

1

2

"∞" a a b c r s t t "∞"

would fall here
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;
}
How can you find predecessor/successor of an element (e.g. cl)?

Use level 0 to navigate in sorted order.
Insertion

Let $p$ be the probability of a tail (continue).
Ex. $p = \frac{1}{4}$

Find position to insert using search.
Then randomly pick the height of new tower as the number of 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 $+$ then splice out of each level it is in (just like in a linked list).

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

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

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

$i = 0 \quad n \quad n$

$i = 1 \quad n \cdot p \quad n/4$

$i = 2 \quad n \cdot p^2 \quad n/16$

\[ \vdots \]

Can prove by induction.
When is \( n \cdot p^i = 1/p \)?

Solve for \( i \):

\[
N = \left(\frac{1}{p}\right)^{i+1} \Rightarrow i+1 = \log_{1/p} N
\]

\[
i = \log_{1/p} N - 1
\]

When \( p = 1/4 \) only 4 items expected at level \( i \).

Level when expect \( 1/p \) elements to remain

\[
E[\text{height}] = \log_{1/p} N + \frac{1}{1-p}
\]

Ex: \( p = 1/4 \)

\[
E[\text{height}] = \log_{1/4} N + \frac{4}{3}
\]
We'll finish analysis next time.

Skip List

Overview

<table>
<thead>
<tr>
<th></th>
<th>As a function of $p$</th>
<th>$p = 1/2$</th>
<th>$p = 1/e$</th>
<th>$p = 1/4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>space usage</td>
<td>$2n/(1-p) + 4 \log_{1/p} n$</td>
<td>$4n + 4 \log_2 n$</td>
<td>$\approx 3.2n + 2.8 \log_2 n$</td>
<td>$\approx 2.67n + 2 \log_2 n$</td>
</tr>
<tr>
<td>search cost</td>
<td>$(\log_{1/p} n)/p$</td>
<td>$2 \log_2 n$</td>
<td>$\approx 1.88 \log_2 n$</td>
<td>$2 \log_2 n$</td>
</tr>
</tbody>
</table>