Class Notes for October 28, 2021

Agenda

- Skiplist Analysis
- Hash Tables

Announcements

- midterm pushed back a week (week of 11/9)

SkipList Analysis

- Last class we found that the expected height of a node is 1, and the average references per node is 2
- Today we'll analyze the expected tallest node, and the expected search time
- Note that we have no control over the order in which the user adds elements

Question: What is the expected max height of skiplist with n nodes?

- Back of envelope calculation:
  - if I have n nodes, how many have height ≥ 0?
    - all n
  - how many have height ≥ 1?
    - $n/2$
  - how many have height ≥ 2?
    - $n/4$
  - how many have height $\geq k$?
    - $n/2^k$
  - when is the number of nodes with height k less than 1?

> > - $n/2^k < 1 \iff n < 2^k \iff log(n) < k$
>
> - **Thus, the expected max height is $O(\log(n))$**

Formalizing this Argument

- Let H be a random variable that represents the max height in a list of length n
- For each $k = 0, 1, 2..$., $L_k =$ length of list at height $k =$ number of nodes with height $\geq k$
- Observe the expected value of L_k

$$E(L_k) = \frac{n}{2^k}$$

- Therefore if $k = \log(n) + j, (j \geq 0)$ (ie. the number of nodes with height greater than $\log(n)$)

$$E(L_{\log(n)+j}) = \frac{n}{2^{\log(n)+j}} = \frac{n}{2^{\log(n)} * 2^j} = \frac{1}{2^j}$$

- To analyze $E(H)$, write it as the sum of simpler variables.

$$J_k = \begin{cases} 1 & if H \geq k \\ 0 & otherwise \end{cases}$$

Therefore,

$$H = J_1 + J_2 + J_3 + ...$$
$$E(H) = E(J_1) + E(J_2) + E(J_3) + ...$$

Two facts about J_k :

1. $J_k \leq 1$ always
2. $J_k \leq L_k$ (ie, if $L_k = 0$, then $J_k = 0$)

$$E(H) = E(J_1) + E(J_2) + E(J_3) + ...$$
$$E(H) = E(J_1) + E(J_2) + ... + E(J_{\log(n)}) + E(J_{\log(n)+1}) + ...$$
$$E(H) = 1 + 1 + ... + 1 + 1/2 + 1/4 + 1/8 + ...$$
$$\leq log(n) + 1$$

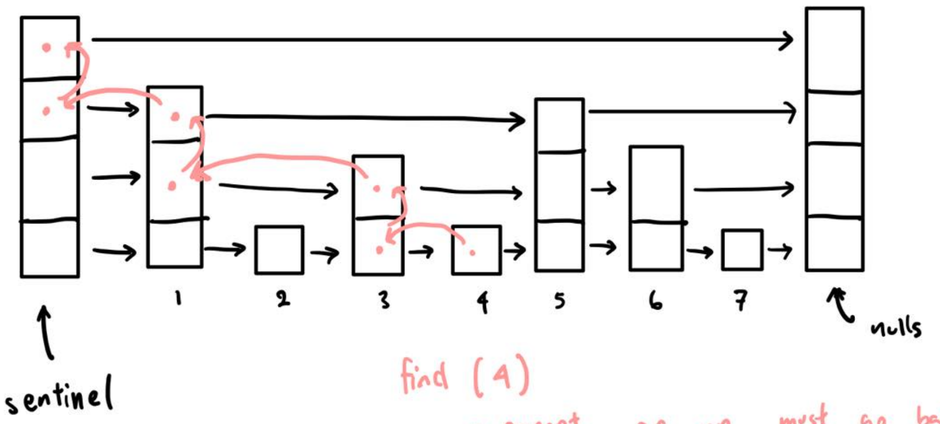
- This is because for J_k, $k \leq \log(n)$, $L_k \geq 1$, and thereafter we have $1/2^j$

  This is good because we see that the max height of our skiplist does not grow

too quickly with size

Question: How long is search on average?

Trick : Analyze search in reverse

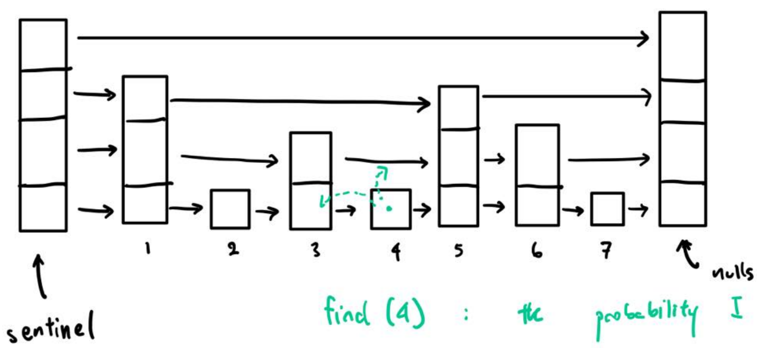


find (4)
- cannot go up, must go backwards
- node 3 is taller than node 4, so I must have come down

- In the reverse direction, if I can go up, go up, if I can't, go left.
- If each step takes $O(1)$, the total running time of the find procedure is

$$total\#ofsteps = \#up + \#left$$

- We know that $\#up$ is always the max height of the list



find (4) : the probability I can go up is 1/2, and thus the probability I go left is 1/2.

→ It can be shown that E (lefts) = log (n) + C , following a similar analysis.

- At the same time, we know that at each node, the probability of going up is $1/2$, and the probability of going left is $1/2$

- This structure mimics the analysis of going up, thus it can be shown that $E(\#left) = \log(n) + C$

  > - *See ODS 4.4 for details*

- So the total expected time to find is

$$E(H) + E(\#left) = 2 * \log(n) + O(1)$$

- this is the same speed as an AVL tree!

- *For empirical runtimes see Rosenbaum's website* *https://willrosenbaum.com/teaching/2021f-cosc-211/slides/lec17/*