CS 361, Lecture 26

Jared Saia
University of New Mexico

Outline

² Skip Lists

Project

² Any questions on the group project? (hw6)

² Project will be due May 8th in class
² Late projects will not be accepted
² You can get partial credit for an unfinished project turned in on time but will get no credit for a finished project turned in late

HW

² There will also be a hw due on May 8th in class
² This will be a “final review” hw

Administrative

² Lab Section evaluation this week
² This week, Kanglin will take attendance at sections, if you're there, you'll get an extra check for participation
² Sections are Thursday 3:30-4:20 and Friday 1:00-1:50
² Good chance to review material for final

Project

² Any questions on the group project? (hw6)
Final will be Tuesday May 13th, 7:30-9:30am in our regular classroom.
Closed book, but two pieces of paper are allowed (for cheat sheets).
No calculators.

High Level Analysis

Comparison of various BSTs

- RB-Trees: + guarantee $O(\log n)$ time for each operation, easy to augment, – high constants
- AVL-Trees: + guarantee $O(\log n)$ time for each operation, – high constants
- B-Trees: + works well for trees that won’t fit in memory, guarantee $O(\log n)$ time for each operation, – inserts and deletes are more complicated
- Splay Tress: + small constants, – amortized guarantees only
- Skip Lists: + easy to implement, – runtime guarantees are probabilistic only

Which Data Structure to use?

- Splay trees work very well in practice, the “hidden constants” are small
- Unfortunately, they can not guarantee that every operation takes $O(\log n)$
- When this guarantee is required, B-Trees are best when the entire tree will not be stored in memory
- If the entire tree will be stored in memory, RB-Trees, AVL-Trees, and Skip Lists are good

Skip List

- Technically, not a BST, but they implement all of the same operations
- Very elegant randomized data structure, simple to code but analysis is subtle
- They guarantee that, with high probability, all the major operations take $O(\log n)$ time

- A skip list is basically a collection of doubly-linked lists, $L_1, L_2, \ldots, L_x$, for some integer $x$
- Each list has a special head and tail node, the keys of these nodes are assumed to be $-\text{MAXNUM}$ and $+\text{MAXNUM}$ respectively
- The keys in each list are in sorted order (non-decreasing)

- Every key is in the list $L_1$.
- For all $i > 2$, if a key $k$ is in the list $L_i$, it is also in $L_{i-1}$. Further there are up and down pointers between the $k$ in $L_i$ and the $k$ in $L_{i-1}$.
- All the head(tail) nodes from neighboring lists are interconnected
Search(k){
    pLeft = L_x.head;
    for (i=x;i>=0;i--){
        Search from pLeft in L_i to get the rightmost elem, r,
        with value <= k;
        pLeft = pointer to r in L_(i-1);
    }
    if (pLeft==k)
        return pLeft
    else
        return nil
}

Insert(k){
    First call Search(k), let pLeft be the leftmost elem <= k in L_1
    Insert k in L_1, to the right of pLeft
    i = 2;
    while (rand())< p){
        insert k in the appropriate place in L_i;
    }
}

Deletion

```
• Deletion is very simple
• First do a search for the key to be deleted
• Then delete that key from all the lists it appears in from
  the bottom up, making sure to "zip up" the lists after the
deletion
```

In-Class Exercise Trick

A trick for computing expectations of discrete positive random variables:

```
• Let X be a discrete r.v., that takes on values from 1 to n
  \[ E(X) = \sum_{i=1}^{n} P(X \geq i) \]
```

Why?

```
\[ \sum_{i=1}^{n} P(X \geq i) = P(X = 1) + P(X = 2) + P(X = 3) + \ldots \\
+ P(X = 2) + P(X = 3) + P(X = 4) + \ldots \\
+ P(X = 3) + P(X = 4) + P(X = 5) + \ldots \\
+ \ldots \\
= 1 \cdot P(X = 1) + 2 \cdot P(X = 2) + 3 \cdot P(X = 3) + \ldots \\
= E(X) 
```
In-Class Exercise

Q: How much memory do we expect a skip list to use up?

- Let \( X_i \) be the number of lists that element \( i \) is inserted in.
- Q: What is \( P(X_i \geq 1) \), \( P(X_i \geq 2) \), \( P(X_i \geq 3) \)?
- Q: What is \( E(X_i) \)?
- Q: Let \( X = \sum_{i=1}^{n} X_i \). What is \( E(X) \)?

Height of Skip List

- Assume there are \( n \) nodes in the list
- Q: What is the probability that a particular key \( i \) achieves height exceeding \( k \log n \) for some constant \( k \)?
- A: If \( p = 1/2 \), \( P(X_i \geq k \log n) = \frac{1}{n^k} \)

Search Time

- Note that the expected number of “siblings” of a node, \( x \), at any level \( i \) is 2
- Why? Because for a node to be a sibling of \( x \) at level \( i \), it must have failed to advance to the next level
- The first node that advances to the next level ends the possibility of further siblings.
- This is the same as asking expected number of times we need to flip a coin to get a heads.

Flipping to get Heads

- How many times in expectation do we need to flip a coin to get heads, if the coin is heads with probability \( p \)?
- Let \( X \) be a random variable giving the number of times the coin is flipped until we get heads, then \( E(X) \) is the expected number of times needed to flip to get heads
- Then \( E(X) = 1 + (1-p)E(X) \) since we take 1 flip, plus in the case where the coin is tails (which happens with probability \((1-p)\)), we then take “the expected number of times needed to flip to get heads” (i.e. we’re no better off than when we started)
- Solving for \( E(X) \) gives \( E(X) = \frac{1}{p} \). If \( p = 1/2 \), then \( E(X) = 2 \)
• The expected number of "siblings" of a node, $x$, at any level $i$ is $2$
• The number of levels is $O(\log n)$ with high probability
• From these two facts, we can prove that the expected search time is $O(\log n)$ (the proof is omitted)
• (Warning: The argument is not as simple as multiplying these two values. We can't do this since the two random variables are not independent.)