6.046 Recitation 5: Binary Search Trees Bill Thies, Fall 2004

My contact information:

Bill Thies

thies@mit.edu

Office hours: Sat 1-3pm, 36-153

Recitation website: http://cag.lcs.mit.edu/~thies/6.046/

Announcements:

- PS 3 up, due Mon 10/18
- Quiz graded, back today

Today:

- BST vs. Quicksort
- Expected height of BST

Binary Search Tree vs. Quicksort

A: 3, 1, 8, 2, 6, 7, 5

Binary Search Tree (BST):

[bold represents actual nodes;
others were compared at the given level]

3
1 (2) **8** (6 7 5)
2 **6** (7 5)
5 **7**

Quicksort:

[bold represents pivot]:

3
[**1** 2] [**8** 6 7 5]
2 [**6** 7 5]
5 **7**

Can use BST and Quicksort to reason about each other.

E[total # of comparisons made] = Θ(n lg n) [know from quicksort]
E[# comparisons / node] = Θ(n lg n) / n = Θ(lg n)

Depth of node in BST = # comparisons made for that node
E[depth / node] = Θ(lg n) [due to E[# comp / node] for quicksort, above]

Time: 12:37

Expected Height of BST

Height is max depth over nodes.

Does the fact that average node depth is $\Theta(\lg n)$ imply that height is $\Theta(\lg n)$?

→ **No.**

Counterexample:

```
     x
   x   x
 x  x  x  x
          x
            x
```

Most of tree has depth $\leq \lg n$. But one spine has a length of $\sqrt{n}$

Height = $\sqrt{n}$

Average depth of node = $(\sum_{\text{nodes}} \text{depth of node}) / n$

$\leq (n * \lg(n) + \sqrt{n}\sqrt{n}) / n$

$= \lg (n) + 1$

So this shows a case where the average node depth is $\Theta(\lg n)$, but the height of the tree is order $\sqrt{n}$ (that is, polynomial in n).

It turns out that the expected height of the tree is $\Theta(\lg n)$, but we need a considerably more sophisticated analysis to show it.

Clock: 16:37

Calculating Expected Height of BST

Rand. var H = height of tree = max depth of node in tree

$E[H] = \sum_{d=0}^{\infty} d * Pr[H = d]$

Where do we start evaluating this? It turns out that it is pretty hard to get a closed formula for the exact probability that H is a given depth. (In part because H is a maximum over many nodes; also because calculating the depth of a given node is hard.)

So we use a general technique: **Guess that height $\leq$ t** [see next page]

Then we can bound the expectation, splitting it up as follows:

$E[H] \leq Pr[H < t] * t + Pr[H > t] * n$

$\leq t + Pr[H>t] * n$ [upper bound for any probability is 1]

Now we have to calculate Pr[H > t]. We bound this value by considering a single node, instead of the whole tree at a time:

Let h_x = height of node x

Calculate q = Pr [h_x > t] [see next page]

Then $Pr[H > t] \leq n * q$ [because n nodes in tree]

Thus: $E[H] \leq t + (nq) * n$

$E[H] \leq t + n^2 q$

Note that q is a function of t and n. If we can show that q grows arbitrarily small with n, then we can eliminate the n^2 term and be left with our bound of t.

At end of lecture, come back having shown that for $t = c \log_{10/9} n$, $q \leq 1/n^{\varepsilon (c-1)}$ for $\varepsilon > 0$

Then $E[H] \leq c \log_{10/9} n + n^2 / n^{\varepsilon (c-1)}$ for $\varepsilon > 0$

For c sufficiently large, the $n^{\varepsilon (c-1)}$ term dominates the n^2, and we have:

$E[H] \leq c \log_{10/9} n + \Theta(1)$

$E[H] = \Theta(\lg n)$ ⟵ *Major result of this whole recitation*

Calculating Probability that a Node Exceeds Logarithmic Height

Guess $t = \log_{10/9} n$
Calculate $q = \Pr[h_x > t]$

Recall that height of node in BST is number of times that quicksort had to partition to isolate a single element of array.

Lucky partition: ratio of 1:9 [Piotr used 1:3 in lecture, I think 1:9 turns out to be simpler]

10% lucky 90%

|-----|--------------------------------------|----|

Let p = probability of unlucky partition

$p = 2 * 10\% = 1/5$

lucky partitions before element becomes a leaf = $\log_{10/9} n$
- Because dividing by 10/9 each time

If $h_x > t$ (that is, $h_x > c \log_{10/9} n$) then:
- in first t partitions, # lucky partitions $\leq \log_{10/9} n$
  (otherwise we would have hit a leaf before making it to depth t)
- in first t partitions, # unlucky partitions $\geq$ t - # lucky partitions
  $\geq c \log_{10/9} n - c \log_{10/9} n$
  $\geq (c-1) \log_{10/9} n$

  Call this quantity k: $k = (c-1) \log_{10/9} n$

$\Pr[h_x > t]$ = Pr [at least k out of t partitions are unlucky]
p = prob. of being unlucky = 1/5 TODO →

Using results from pages 6 and 7, come back to conclude:

$\Pr[h_x > t] \leq \binom{t}{k} p^k$ [see page 6]

$\leq (et/k)^k p^k$ [see page 7]

$= (ep\,(t/k))^k$ [the rest is algebra]

$= (\,(e/5)\,(c/(c-1))\,)^{(c-1)\log_{10/9} n}$

$\leq (\,0.7^{\log_{10/9} n}\,)^{c-1}$ (for c>10 or so)

$= (n^{\log_{10/9} 0.7})^{c-1}$

$= n^{-\varepsilon\,(c-1)}$ (for some $\varepsilon > 0$)

$\Pr[h_x > t] \leq 1 / n^{\varepsilon\,(c-1)}$ (can choose c to make this arbitrarily small)

Theorem (CLRS p. 1118) -- Tails of a Binomial Distribution

Consider n trials, each with probability of success p
Rand var X = number of successful trials

Probability of at least k successes:

$\Pr[X \geq k] \leq \binom{n}{k} p^k$ [see next slide for reminder of n-choose-k notation]

Intuition:

Let's choose k trials. Number of ways of doing this is n-choose k. Then the probability that each of k trials is successful is p^k. So overall probability is $\leq$ (n-choose-k) p^k. The bound is not tight because we are double-counting some overlapping trials.

Proof:

Let $S \subseteq \{1, 2, \ldots n\}$ be a subset of the trials s.t. $|S| = k$
Let A_s = event that i'th trial succeeds $\forall\, i \in S$

Then, for given S, $\Pr[A_S] = p^k$

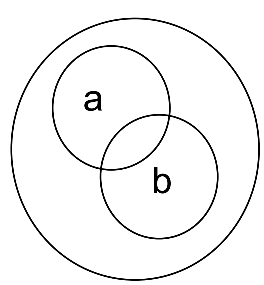
$\Pr[X \geq k]$ = Pr [there exists S as above and A_S holds]
$= \Pr[\bigcup_{S \text{ of size } k} A_S]$
$\leq \sum_{S \text{ of size } k} \Pr[A_S]$ [Boole's inequality, see below]
$= \sum_{S \text{ of size } k} p^k$
$= \binom{n}{k} p^k$ [$\binom{n}{k}$ ways of choosing k elements from set of size n]

Thus $\Pr[X \geq k] \leq \binom{n}{k} p^k$

Boole's inequality:

$\Pr[a \cup b] \leq \Pr[a] + \Pr[b]$

Intuition: overlapping region counted twice



Potpourri

We referred to the following at different points:

n-choose-k

$\binom{n}{k}$ = number of k-combinations from n-element set

$$= \frac{\text{\# permutations of size k}}{\text{\# permutations of size k per combination of size k}}$$

$$= \frac{n\,(n-1)\,(n-2)\,\ldots\,(n-k+1)}{k\,(k-1)\,(k-2)\,\ldots\,1} \qquad \text{[Eq 1]}$$

$= n! \,/\, (\,k!\,(n-k)!\,)$

Using Eq 1, we can bound n-choose-k as follows:

$$\binom{n}{k} \leq n^k / k!$$

Using Stirling's approximation that $k! \geq (k/e)^k$ (see below), this becomes:

$$\binom{n}{k} \leq (en/k)^k \qquad \text{[Eq 2.]}$$ **[** used on page 5]**

Stirling's approximation (CLRS p. 55)

Stirling's approximation states that:

$$n! = \text{root}(2\,\pi\,n)\,(n/e)^n\,(1+\Theta(1/n))$$

From this we can deduce a number of useful bounds. One that has been used in class already follows from taking the log of both sides:

$\lg(n!) = (n \lg n)$

Another comes in useful above: $n! \geq (n/e)^n$ **[** used above, Eq. 2]**

Implication for Quicksort

We have shown, as stated at the bottom of page 4, that the expected height of a binary tree is $\Theta(\lg n)$. This expectation is taken over all random permutations of the input array.

The same analysis tells us something new about Quicksort. Before we showed that the *average* number of partition operations that a given element participates in is $\Theta(\lg n)$. However, the present analysis shows that the *maximum* number of partitions that any element participates in is $\Theta(\lg n)$.

Another way of stating this is that the maximum recursion depth of the QuickSort procedure, when processing from the root to any leaf, is expected to be $\Theta(\lg n)$.