Big Oh, et cetera

Definitions

Big Oh

A function `f(n)` is `O( g(n) )` if there exists $c > 0$ and n_0 such that $f(n) \le c\, g(n)$ for all $n \ge n_0$.

A common way to express `f(n)` is `O( g(n) )` is to say `f(n) = O( g(n) )`.

STOP: Draw a picture of what this means.

STOP: What is the domain of c? What is the domain of n0?

STOP: Suppose there are two functions g1(n) and g2(n) that both satisfy the definition of Big Oh for f(n). (I.e., f(n) is O(g1(n)) and f(n) is O(g2(n)). Draw a picture. What does that mean? Is there a way to choose a best g(n)?

Big Omega

A function `f(n)` is $\Omega($ `g(n)` $)$ if there exists $c > 0$ and n_0 such that $f(n) \ge c\, g(n)$ for all $n \ge n_0$.

Big Theta

A function `f(n)` is $\Theta($ `g(n)` $)$ if and only if `f(n) = O( g(n) )` and $f(n) = \Omega(g(n))$.

STOP: Think about the intent of the definition of Ω(). Do the c and the n_0 have to be the same for the Big Oh definition and for the Ω definition? If you have two different c's and/or two different n_0's, how can you resolve them?

Common `g(n)` functions in sequential algorithms and their names:

g(n)	name
`1`	constant
`log n`	logarithmic
$\log^2 n$	log-squared
`n`	linear
`n log n`	log-linear
n^2	quadratic
n^3	cubic
n^k	polynomial
2^n	exponential
`2!`	factorial

STOP: What does is mean when one says a function is O(m n) ?

Rules for "combining" growth rates

*If `T1(n) = O( g1(n) )` and `T2(n) = O( g2(n)` then

a. `T1(n) + T2(n) = O ( g1(n) + g2(n) )`
also, `T1(n) + T2(n) = max ( O (g1(n) ), O( g2(n) ) )`

b. `T1(n) * T2(n) = O( g1(n) * g2(n) )`

* If `T(n)` is a polynomial of degree `k`, then `T(n)` is $O(n^k)$.

Also, if `T(n)` is a polynomial of degree `k`, then `T(n)` is $\Omega(n^k)$.

STOP: Do the rules make sense?

STOP: Draw a picture of the last "rule" stated above.

→ *go to the white paper on Big Oh and work out computing c and n_0.*

Solve the same problem to compute Ω *()*

→ *go to the white paper on recursion, make sure everyone is ok with recursion.*

→ *go to the white paper on recurrence, work out*
Merge sort.
Binary search

Review tree and terms:

Height(n): length of longest path node n to leaf
Depth(n): length of path from root to node n
Internal (inner) node: any node that is not a leaf.
External (outer) node: any node that is a leaf.
Perfect tree: a full tree (all levels are at the same depth)
Complete tree: every level, EXCEPT POSSIBLY THE LAST, is completely filled AND all nodes are as far to the left as possible. A heap is a complete binary tree.

Properties of binary trees (copied from wikipedia)

* The number of nodes `n` in a perfect binary tree can be found using this formula: $n = 2^{h+1} - 1$ where `h` is the height of the tree.

* The number of nodes n in a complete binary tree is minimum: $n = 2^h$ and maximum: $n = 2^{h + 1} - 1$ where h is the height of the tree.

* The number of leaf nodes L in a perfect binary tree can be found using this formula: $L = 2^h$ where h is the height of the tree.

* The number of nodes n in a perfect binary tree can also be found using this formula: $n = 2L - 1$ where L is the number of leaf nodes in the tree.

* The number of NULL links in a Complete Binary Tree of n-node is $(n+1)$.

* The number of internal nodes in a Complete Binary Tree of n-node is $\lceil (n/2) \rceil$.

* For any non-empty binary tree with n_0 leaf nodes and n_2 nodes of degree 2, $n_0 = n_2 + 1$.

STOP: What does the depth of a complete binary search tree say about insertion time?

Review rotations

Review AVL tree

STOP: HOMEWORK FOR THE WEEKEND (not to be turned in)
(1) What is a threaded binary search tree?
(2) Show the threaded BST after inserting the following in order: 15, 20, 10, 5, 19, 11, 19, 13, 18.
(3) What is the insertion algorithm for a threaded BST and what is its running time?

New tree: kd tree

New tree: splay tree

New tree: Fibonacci tree