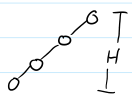
Midterm study guide

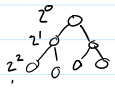
October 22, 2016 20:45

All Logs are base 2!!!

Sections
1) Overview
2) Binary Trees
3) Heaps



$H = \log_2 N$



- No Min/Max heap

d-Heap

$O(H)$	$O(\log_e N)$	- Insert
$O(H)$	$O(\log_e N)$	- DeleteMin
$O(1)$		- FindMin
$O(H)$	$O(\log_d N)$	- increaseKey - also has $d\log_d N$
	$O(\log_d N)$	- decreaseKey
	$O(\log_d N)$	- delete

Binary Search Tree

$O(H) =$
$O(H) =$
$O(H) =$
} either $O(\log N)$ if balanced or $O(N)$ degenerated

for Insert, Delete, Find
Print $O(2^H) = O(\log_2 N)$, balanced $O(H) = O(N)$ degenerated

Min heap

- Insert $O(H) = O(\log N)$
- DeleteMin $O(H) = O(\log N)$

} always balanced

2 methods to insure tree is balanced
AVL and splay trees

B-Trees

Find - $\log M$ H times
Add -
Remove -

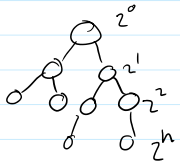
Worst Case $\log_{\frac{M}{2}}\left(\frac{N}{1/2}\right)$
Best Case $\log_M\left(\frac{N}{L}\right)$

- Big-Oh ignores leading constants

$F_1 = 2 \quad F_n = (F_{n-1})^2 \quad n \geq 2$

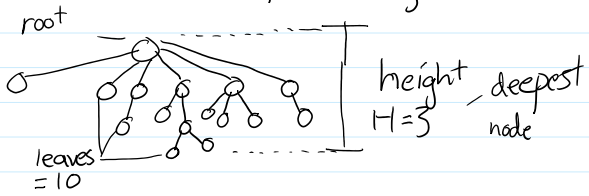
$F_n = (F_{n-1}) \cdot (F_{n-1})$

$O(2^n)$



Any recursive algorithim that solves two $\left(\frac{1}{2}\right)$ problems, does linear non-recursive work to combine/split solutions will take $O(N \log N)$

- Algorithim is $O(\log N)$ if takes constant time to reduce the problam by constant fraction



leaf = node, no children

Binary Tree
- either empt
- contains a root and N binary subtrees
$(0 \leq N \leq 2)$

Inorder
Postorder
PreOrder