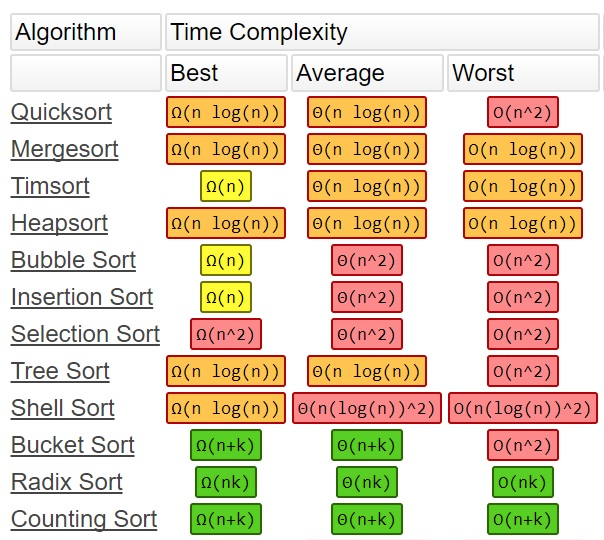
**Analysis/Big-O**

O – upper bound, Ω – lower bound, Ө – both



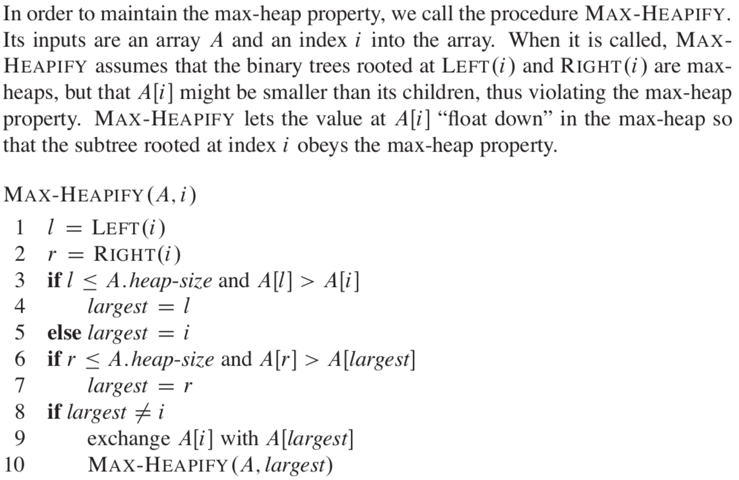
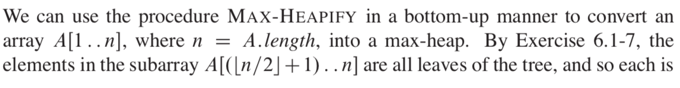
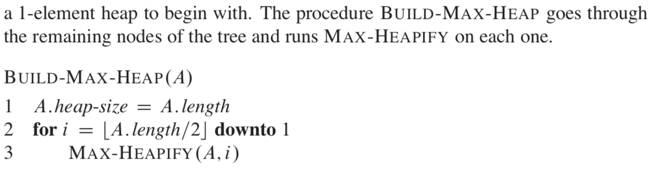
Stacks: LIFO. Push, Pop

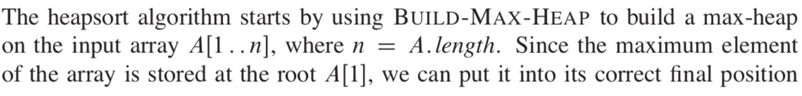
Queues: FIFO. Enqueue, Dequeue

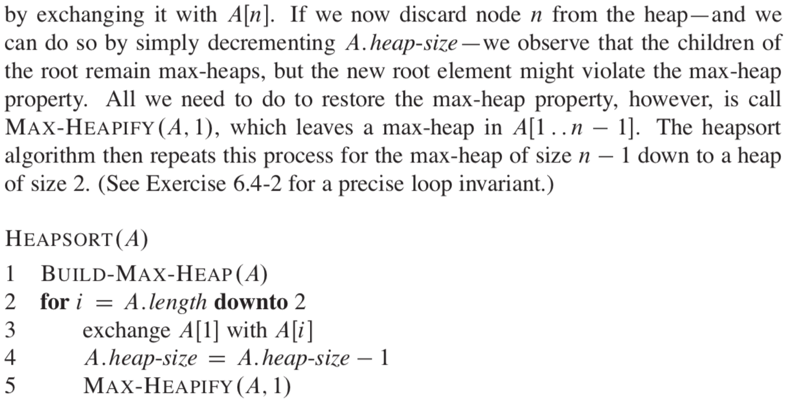
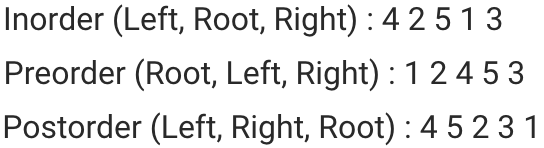
Linked Lists: pointer to next element. Doubly Linked Lists: net and prev. pointers

Binary Trees: lchild and rchild pointers

Binary Heaps: Height = O(nlgn). Parent in floor(i/2), lchild in 2i, rchild in 2i + 1

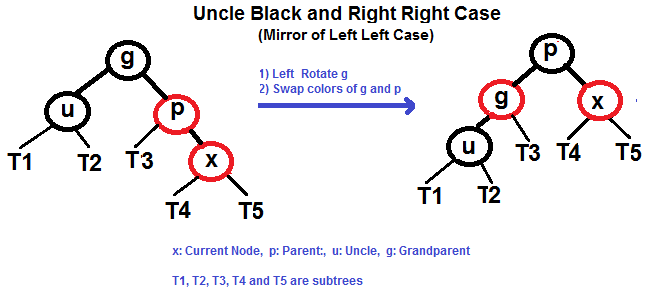
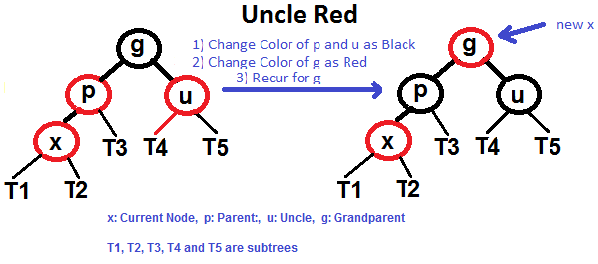


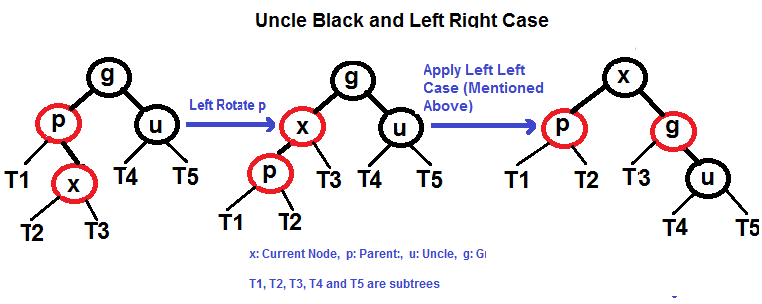
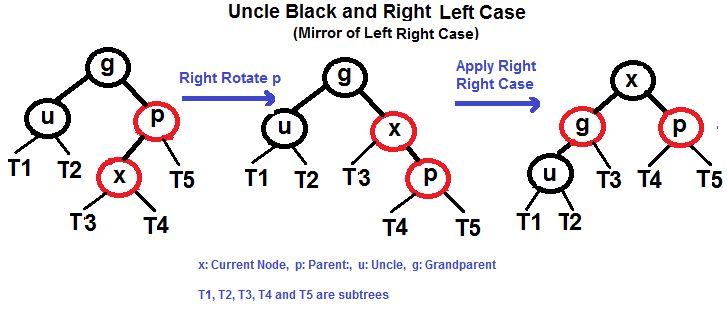
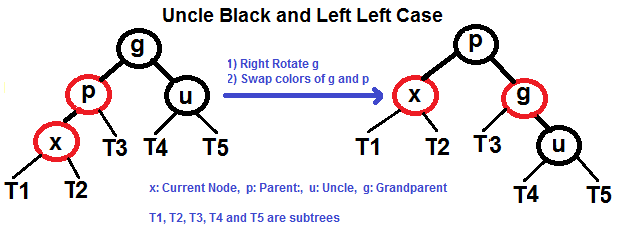




**BST** Delete: if leaf, remove. If one child, move up. If two, replace with successor (r, then far l)

**Red-Black Trees**





**B Trees**: if t = 3, max 5 keys, 6 children per node. If keys > 5, split at middle, move middle up

**quicksort**(low, high) //starting index, ending index

if(low<high)

pi = partition(low,high) //pi is partitioning index (good pi is median/avg)

quicksort(low,pi-1) //before pi

quicksort(pi+1, high) //after pi

Comparison-Based Sorts are worst-case (lower bounds) Ω(nlgn)

First-Order Statistic: i=1, min. nth Order: i = n, max.

Median: if n odd, i = (n+1)/2. If even, lower med = floor((n+1)/2), upper = ceiling((n+1)/2)

**Randomized-Select**: O(n). like quicksort, but only analyze one side of the partition