4. Heaps and Heap Sort

Link:https://www.youtube.com/watch?v=B7hVxCmfPtM&list=PLSX2U_ZE4Huk19DPn34oZlygPbsig380X&index=7

Priority Queue

- Implements a set S of elewments, each of elements associated with a key

Properties of PQ

- Insert(s,x) insert element x into set S
- max(s) : return element of S with the largest key
- extract_max(s): and remove it from S
- increase_key(s,x,k): increases the val of x's key to new value k

Heap

- An implementation of a priority queue
- An array visualized as a nearly complete binary tree

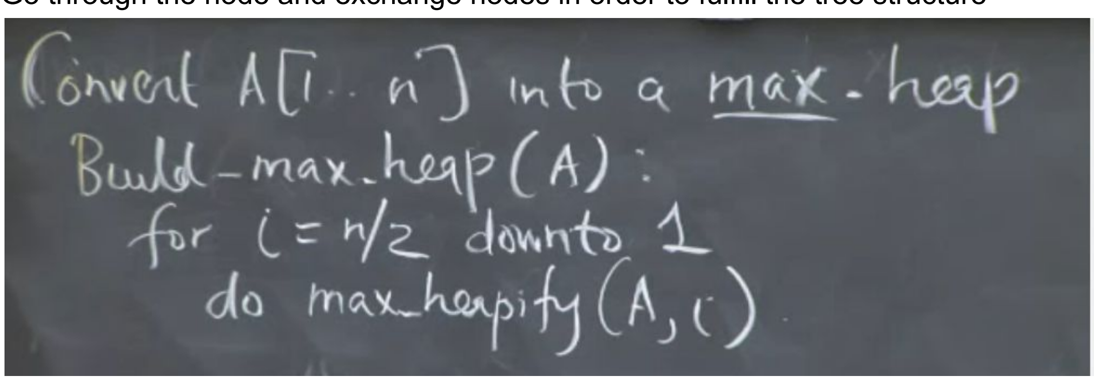
Heap as a Tree

- Root of tree: first element (i=1)
- parent(i) = i/2
- left(i) = 2i right(i) = 2i+1

Max/Min-Heap property:

- Max Heap - the key of a node is >= the keys of its children
- Min Heap - the opposite

Building Max Heaps:

- Build_max_heap: produces a max heap from an unordered array
- Max-heapify: correct a single violation of the heap property in a subtree's root
    - Assume that the trees rooted at left(i) and right(i) are max heaps
    - You must define the heap size first
    - Go through the node and exchange nodes in order to fulfill the tree structure
    - 

- Work from bottom up
- Elements A[n/2 +1 ..n] are all leaves
- O(nlogn) simple analysis
- Observation: Max Heapify takes O(1) for nodes that one level above the leaves and i general O(l) time of nodes that are 1 levels above the leaves
  - n/4 nodes with level 1, n/8 with level 2, … 1 node lgn level
  - Total amount of work in the for loop n/4(1 c) + n/8 (2 c) + n/16(3 c) + …. 1(lgn c)
    - (1 c) is a constant factor that is later removed
  - Set n/4 = 2^k
  - This entire expression is bounded by a constant (this is the key observation)
  - Since 2k = n/4, it really amounts to O(n) after removal of constants

1) Build max heap from unordered array O(n)
2) Find max element A[i] O(1)
3) Snap elements A[n] with A[i] O(1)
   a) Now max element is the end of the array
4) Discard note n from heap - decrementing heap size
5) New root may violate max heap, but children are max heaps - max-heapify O(lgn)