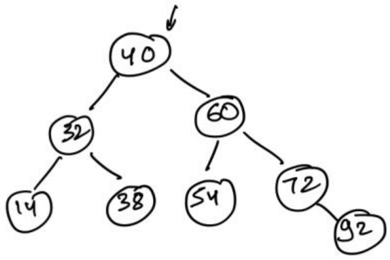
Heaps

Thursday, 7 September 2023 10:03 AM

AVL Trees



Inorder traversal ?

LRR

14 32 38 40 54 60 72 92,
sorted

AVL sort → O(n log n)

—— x ——

Heaps

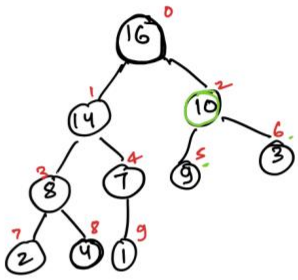
O(1) → min/max
median

- Implement using arrays

max heap

0	1	2	3	4	5	6	7	8	9							
16	14	10	8	7	9	3	2	4	1							

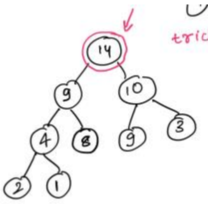
Length = 10



child (p) → 2p+1, 2p+2

parent (c) → $\left\lfloor \frac{c-1}{2} \right\rfloor$, c → index of child

① Extract_max( ) ✓



trickle it down

log(n)

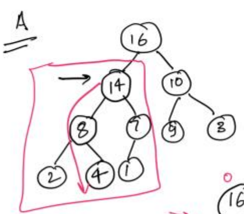
② Insert ⑨

1. Insert at next available position
2. Trickle it up

—— x ——

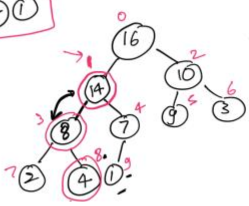
max-heapify( ) → 1. called for a node ✓
2. It assumes that the left subtree and right subtree are valid heaps ✓

A



max_heapify (A, 1, 10)

is it a valid call? ✓



✓① → both children are less than the node
✓② → children index ≤ n (length)

n = 9

✓

4	1	3	2	16	9	10	14	8	7

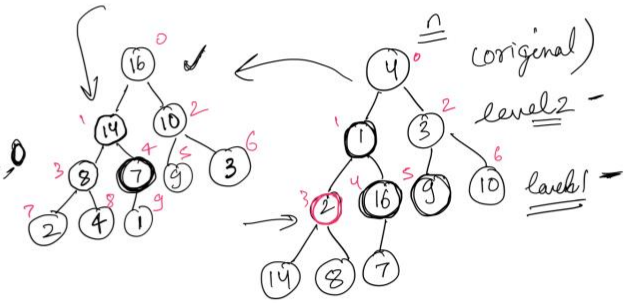
- Building a heap

→ Build_max_heap (A)

A.heapsize = A.length;

→ from $i = \left\lfloor \frac{A.length}{2} \right\rfloor - 1$ down to 0

max_heapify (A, i);



$\frac{n}{4}, \frac{n}{8}$

$\frac{n}{4}(1.c) + \frac{n}{8}(2.c) + \frac{n}{16}(3.c) + \dots$

set $\frac{n}{4} = 2^k$

$= 2^k(1.c) + \frac{2^k}{2}(2.c) + \frac{2^k}{4}(3.c) \dots$

$= c \cdot 2^k \left[\frac{1}{2^0} + \frac{2}{2^1} + \frac{3}{2^2} + \frac{4}{2^3} + \frac{5}{2^4} \dots \right]$

$\approx c \cdot 2^k [4]$

$\approx c \cdot 2^k (4) \approx c \cdot n \cdot (4)$ $O(n)$

—— x ——

Heap construction → $O(n)$

—— x ——

Heap sort

1. Buid max heap
2. Find max (extract_max()) A[0]
3. Swap A(n-1) with A[0]
   ↳ now the max element is at the end of the array
4. Discard (n) → reduce length by 1
5. Step 3 swap may lead to a violation
   max_heapify (A, 0)

length = 10