CSE 101 5-29-26

Supplemental Lecture

6.5 Priority Queues

A Priority Queue (PQ) is an ADT that maintains a set S of elements called records.

record: $x = (\underbrace{\cdot, \cdot, \cdot}_{\text{satellite data}}, \underline{\text{key}}) \in S$

state: $S = \{\cdot, \cdot, x, \cdot, \cdot\}$

Two kinds: max PQ, min PQ

A max PQ has operations

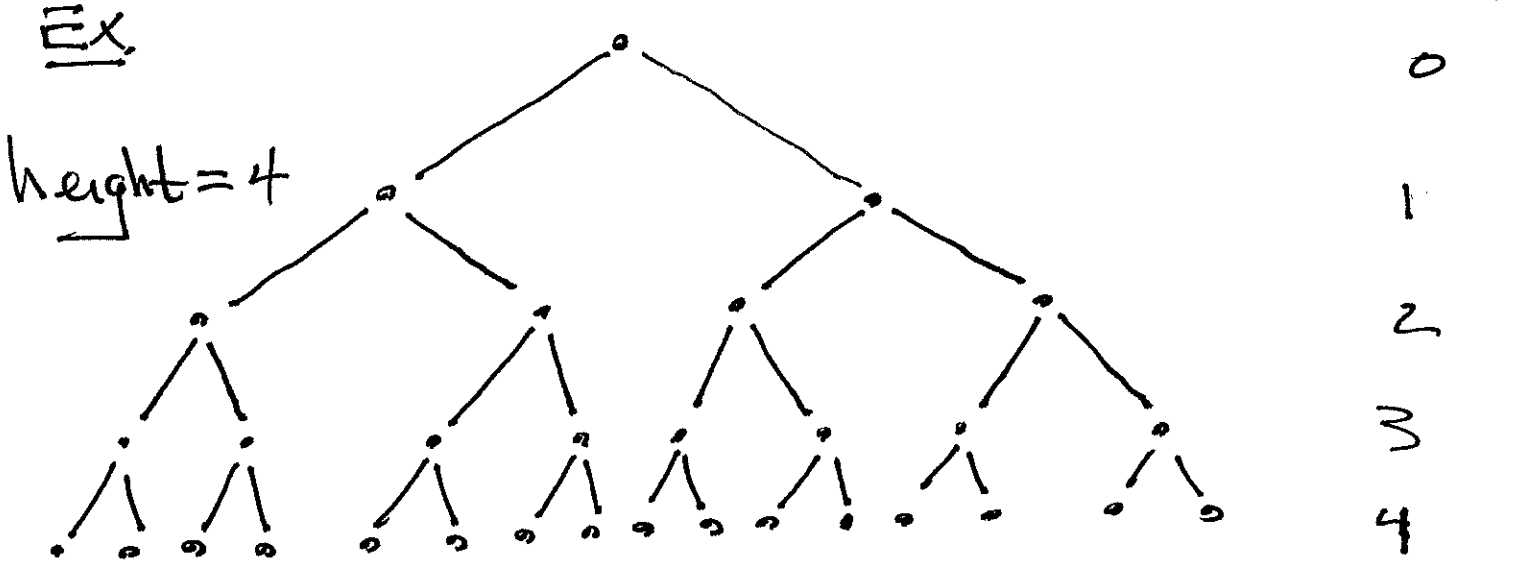
- Insert(S, x): insert x into S
- Max(S): returns record with largest key.
- ExtractMax(S): return and delete record with largest key.
- IncreaseKey(S, x, k): if x.key < k change x.key to k, else do nothing. (Book considers else to be an error condition)

chap 6 : Binary Heaps

Defn

A complete Binary Tree (CBT) is a Binary Tree in which all leaves have same depth, and all internal nodes have 2 children

Ex.



note the # nodes at depth d is $\boxed{2^d}$

so the # nodes n in a CBT of height h is

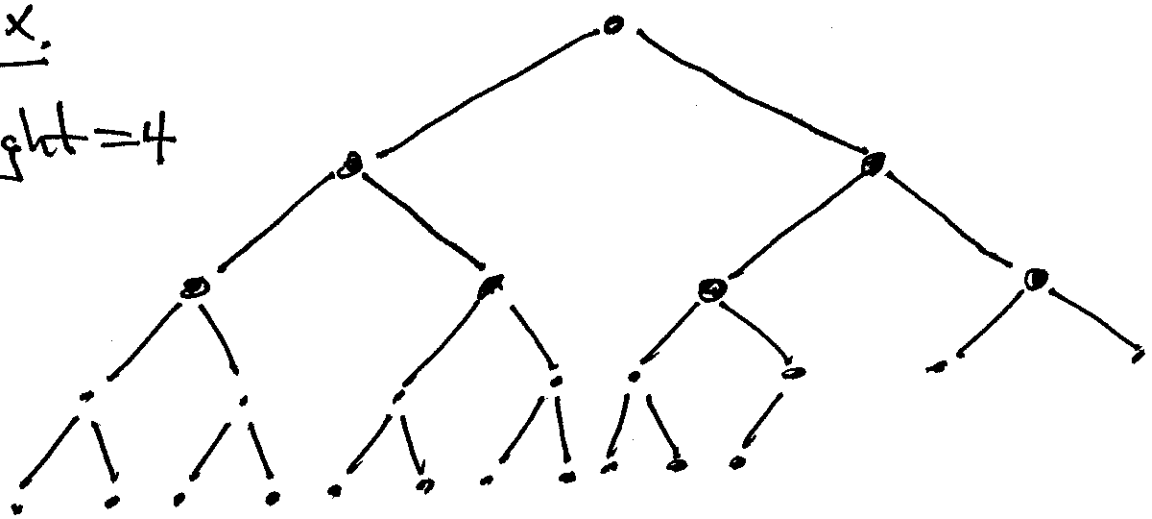
$$n = \sum_{d=0}^{h} 2^d = \frac{2^{h+1}-1}{2-1} = \boxed{2^{h+1}-1}$$

Defn

An almost complete binary tree (ACBT) is a Binary Tree that is filled at all levels, except possibly the last (deepest), which may be Partially filled from left to right.

Ex.

height = 4



Let n = # nodes in an ACBT of height h.

Then

$$2^h - 1 < n \leq 2^{h+1} - 1$$

$$\therefore \quad 2^h \leq n < 2^{h+1}$$

$$\therefore \quad h \leq \lg(n) < h+1 \quad \therefore \quad \boxed{h = \lfloor \lg(n) \rfloor}$$

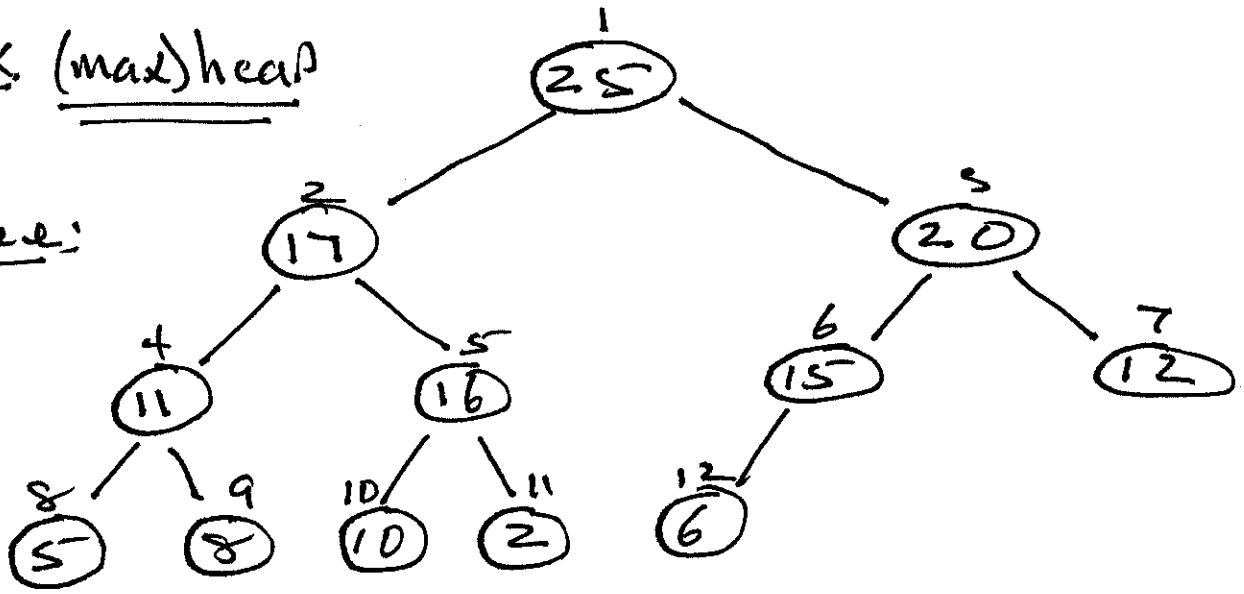
(Binary)

6.1 Heaps

A Heap is an array object that represents an ACRT.

Ex. (max) heap

Tree:



Array:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
A	25	17	20	11	16	15	12	5	8	10	2	6	•	•	•	•

Array attributes: A.length = 16

A.heapSize = 12

helper functions :

- $Parent(i) = \lfloor \frac{i}{2} \rfloor$ (for $i \geq 2$)
- $left(i) = 2i$
- $right(i) = 2i + 1$

Heap Properties

- max heap property

$$A[Parent(i)] \geq A[i]$$

- min heap property

$$A[Parent(i)] \leq A[i]$$

Heap Operations

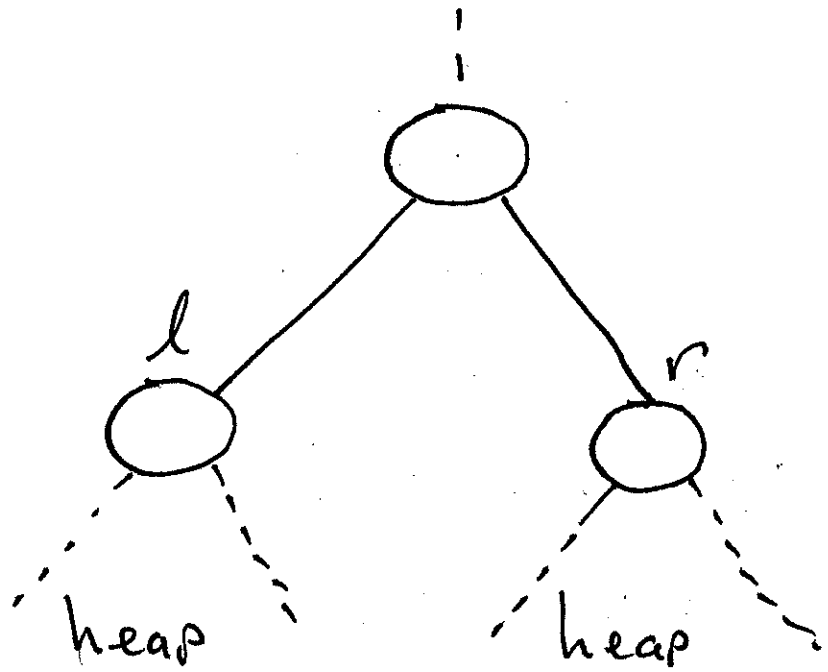
- Heapify
- BuildHeap
- HeapSort
- PQ OPS

HeapInsert
HeapMax
HeapDeleteMax
HeapExtractMax
HeapIncreaseKey

6.2 Heapify

Goal: establish (max) heap property at index i in A.

Heapify(A, i):



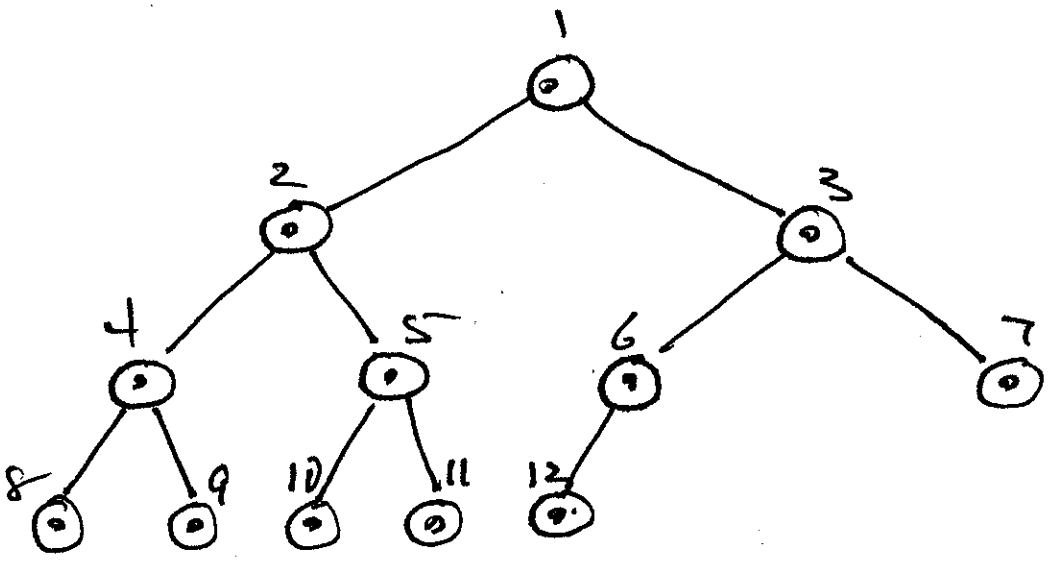
Pre: subtrees at l, r are valid heaps

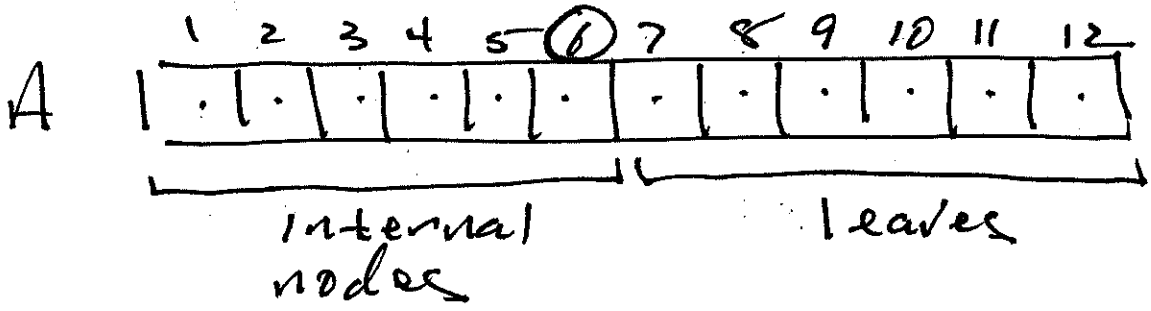
Runtime: $\Theta(\lfloor \lg n \rfloor) = \Theta(\log n)$

6.3 BuildHeap

BuildHeap(A) converts an unordered array into a valid heap.

Ex





Runtime: $\Theta(n)$

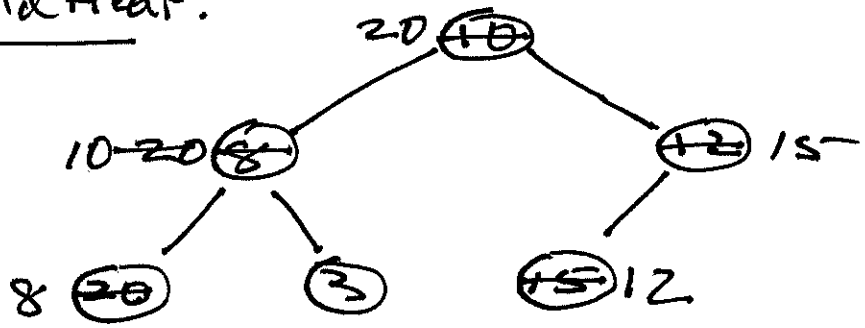
6.4 Heapsort

Heapsort(A) sorts A in increasing order.

Ex. A

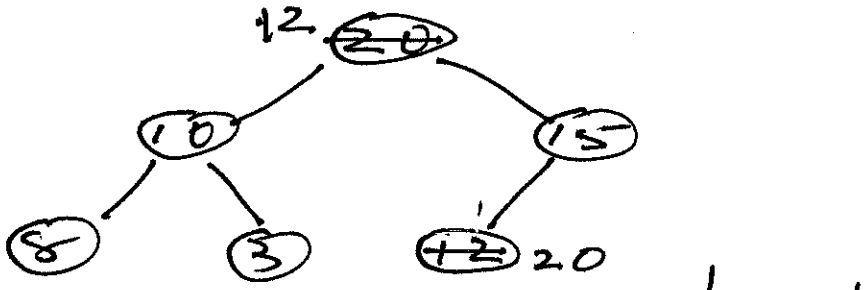
1	2	3	4	5	6
10	8	12	20	3	15

BuildHeap:

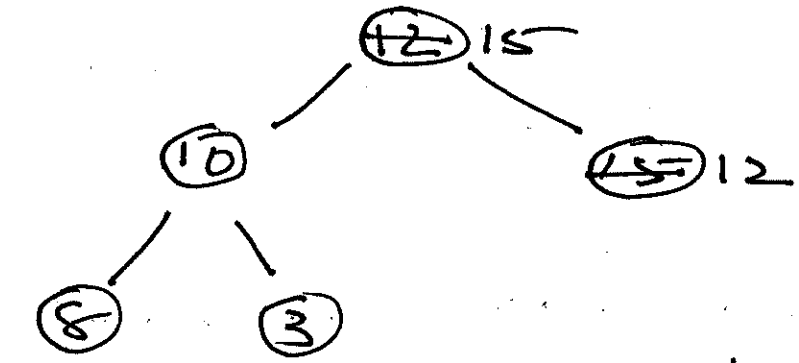


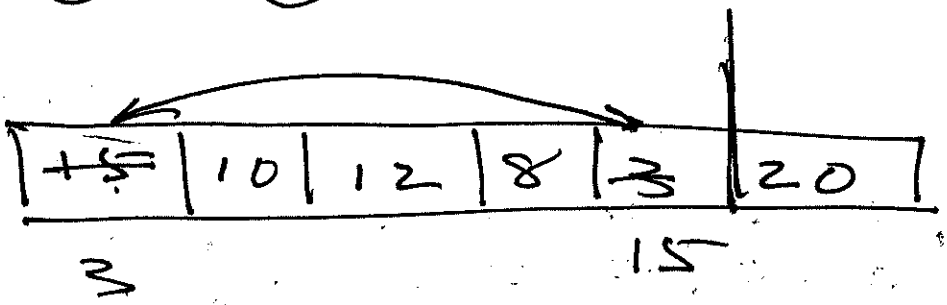
Result:

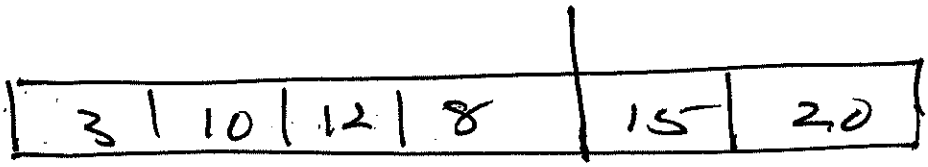
1	2	3	4	5	6
20	10	15	8	3	12

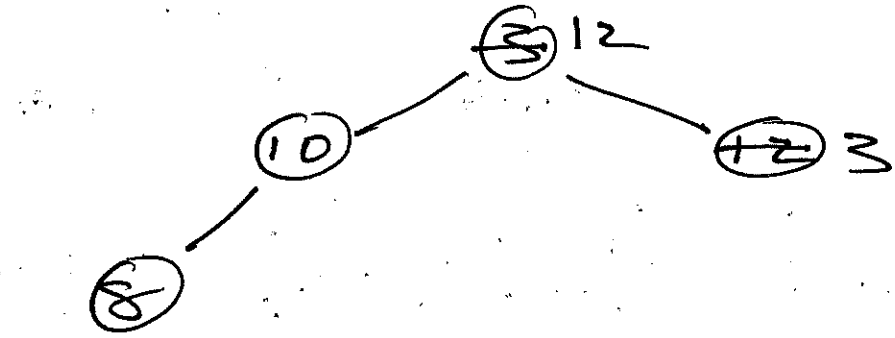


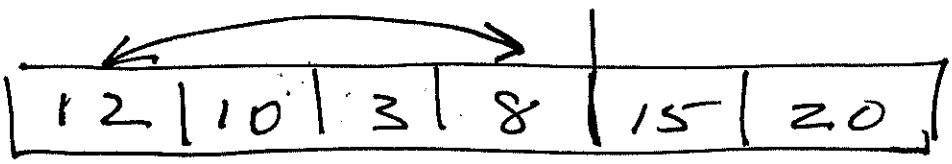
12	10	15	8	3	20

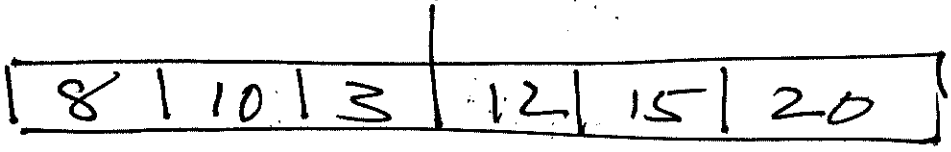

15
10
12
8
3

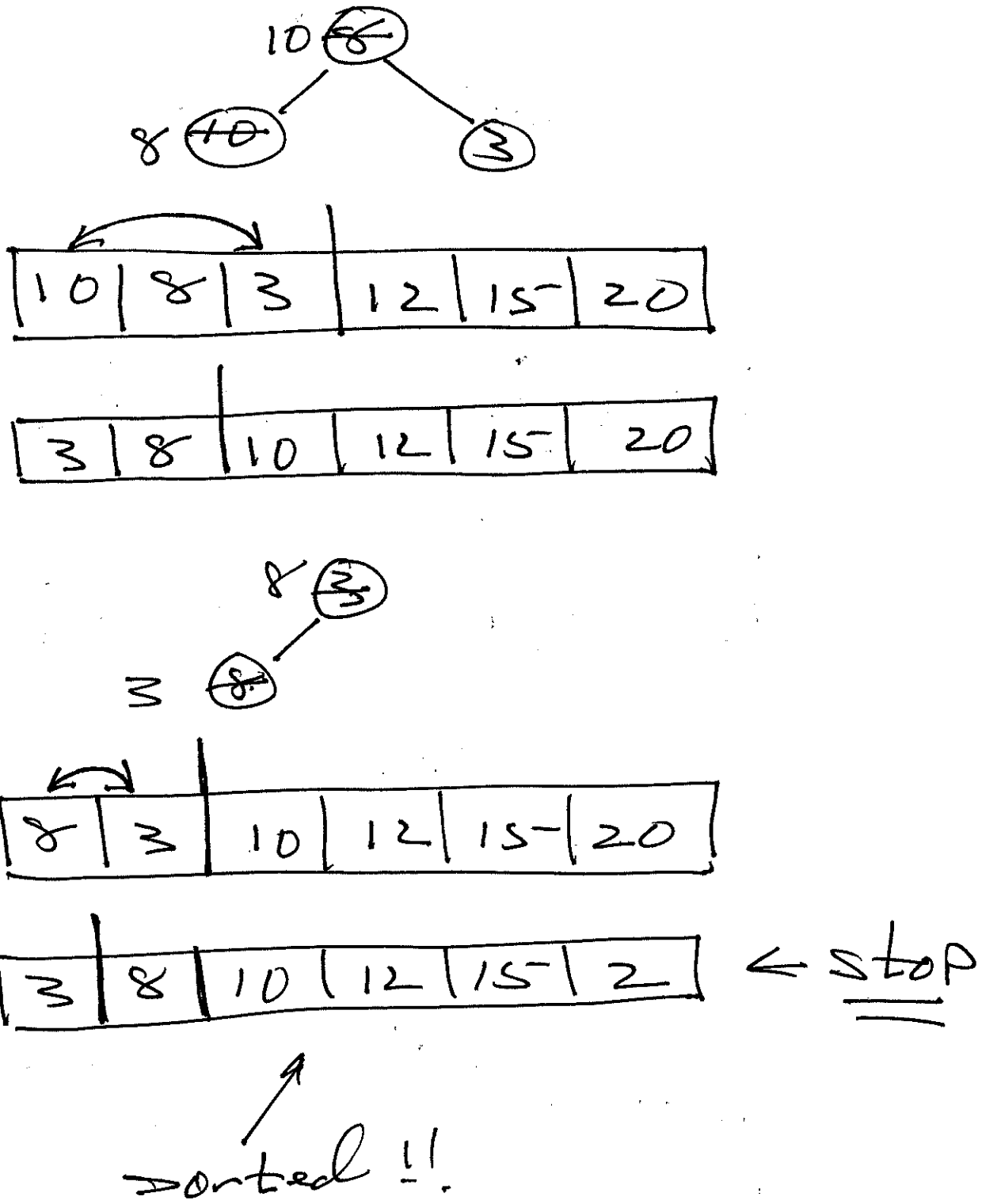

15
10
12
8
3
20
3
15


3
10
12
8
15
20


12
10
3
8


12
10
3
8
15
20


8
10
3
12
15
20



Runtime : $\Theta(n) + (n-1)\Theta(\log n)$

$\therefore T(n) = \Theta(n \log n)$

next .....

6.5 Priority Queues

PQ operations in a Heap

chap. 24

SSSP in a weighted graph

- Belman-Ford
- Dijkstra