Lecture 3 — Sorting (and searching)

Binary Search (A[b:e] and v)

sorted

$m = \frac{b+e}{2}$

if (v == A[m]) done!

if (v > A[m]) BinarySearch (A[m+1:e], v)

if (v < A[m]) BinSearch (A[b:m-1], v)

$$T(n) = T(n/2) + \Theta(1) \Rightarrow T(n) = \Theta(\log n)$$

Th No search alg works faster than $\Theta(\log n)$ by comparisons

$\Theta(\log_3 n) = \Theta(\log_2 n)$

Ternary search $T(n) = 1 + T(n/3)$

$m = \frac{2b+e}{3}$ $n = \frac{b+2e}{3}$

test v vs A[m], A[n] ⇒ go accordingly ($|A| = \frac{|A|}{3}$)

proof

alg A w/ cmp

As a tree of decisions

$\#\boxed{\text{done}} \geq \text{until } \Omega(n)$

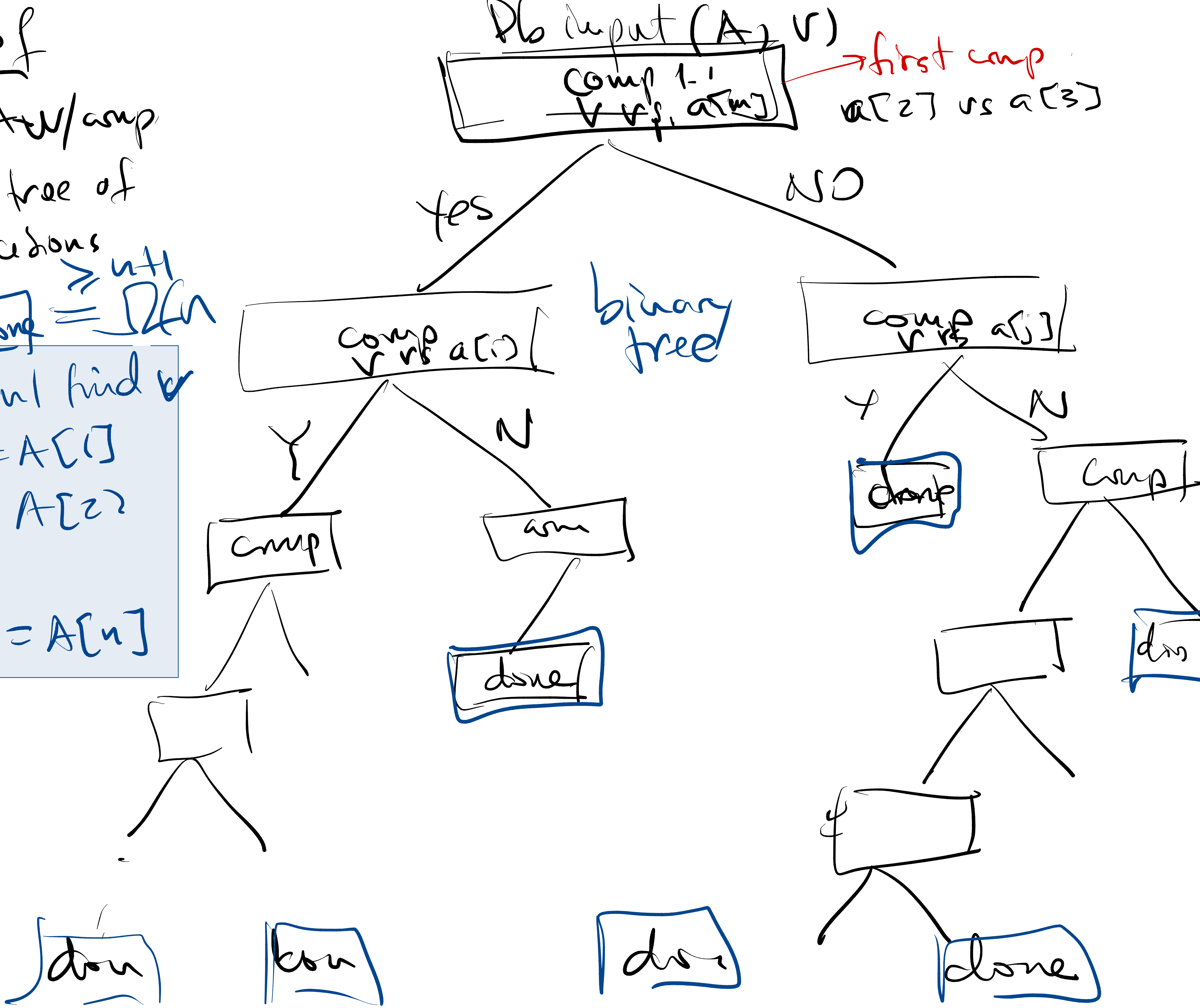
can't find v

$V = A[1]$

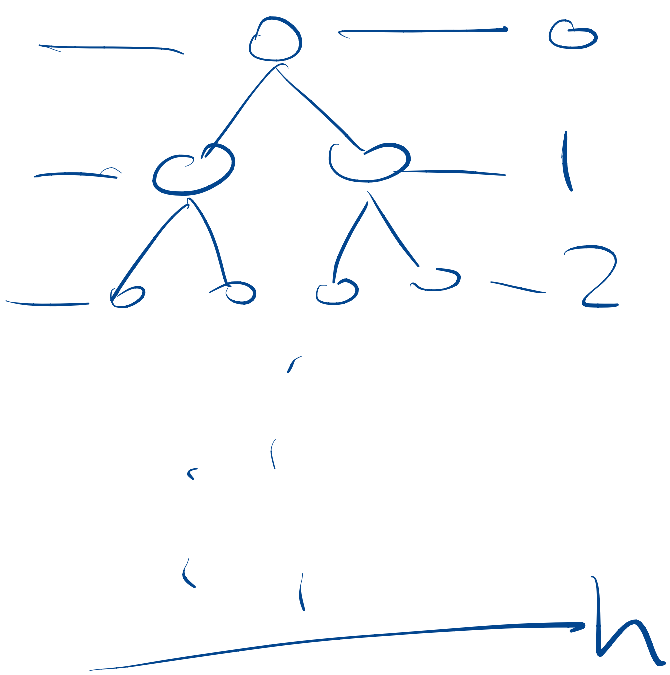
$V = A[2]$

$\vdots$

$V = A[n]$



binary tree depth h (# levels)



max # leafs $\leq 2^{h+1}$

$\vee\!\!/$

$n+1$

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

$$\boxed{h \geq \Theta(\log n)}$$

At least one branch is

$\Rightarrow \Theta(\log n)$

Sorting Recap (A not sorted)

Bubble Sort   cond = True

while (cond)

cond = False

traverse A, look for bubbles (i, i+1)

$A[i] > A[i+1]$

if find bubble (i, i+1)

cond = True

swap (A[i] A[i+1])

Th

no bubbles $\Downarrow$ A sorted

worst case $A = \boxed{n \mid n-1 \mid n-2 \mid \ldots \quad 1}$

#inversions $i<j$, $A[i] > A[j]$

#bubbles $\binom{n}{2} = \Theta(n^2)$

best case : $A = \boxed{1 \ 2 \ 3 \ldots \quad n}$

avg : $\Theta(n^2)$ $\Theta(n)$

Selection Sort

A =

10	-1	-5	-12	-1	9
	4	2	1	3	

n-1 1) select min

-12	~

n-2 2) select min -5
(skip the used ones)

-12	-5

n-3 3) repeat -1

-12	-5	-1

4) ⋮

-12	-5	-1	-1

1 until I finished all element (used)

$\Theta(n^2)$ all the time

Insertion Sort

• keep read-so-far SORTED (in input order)

• insert next value ⟹ SORTED

- worst case $\Theta(n^2)$ backwards
- best case $\Theta(n)$
- avg case $\Theta(n^2)$ inside ALG

1	2	3	4	5	6
1	5	8	20	49	....

next val = 9

like

strategy 1: drag "9" in

1 | 5 | 8 | 20 | 49 | 9

bubble? Y

1 | 5 | 8 | 20 | 9 | 49

Y

like.

1 | 5 | 8 | 9 | 20 | 49

N

✓

strategy 2: binary search make space

- bin-search ⟹ 9 in pos 4 $\Theta(n \log n)$
- make space move by 1 $\Theta(n^2)$

1	5	8		20	49

- put 9 in

1	5	8	9	20	49

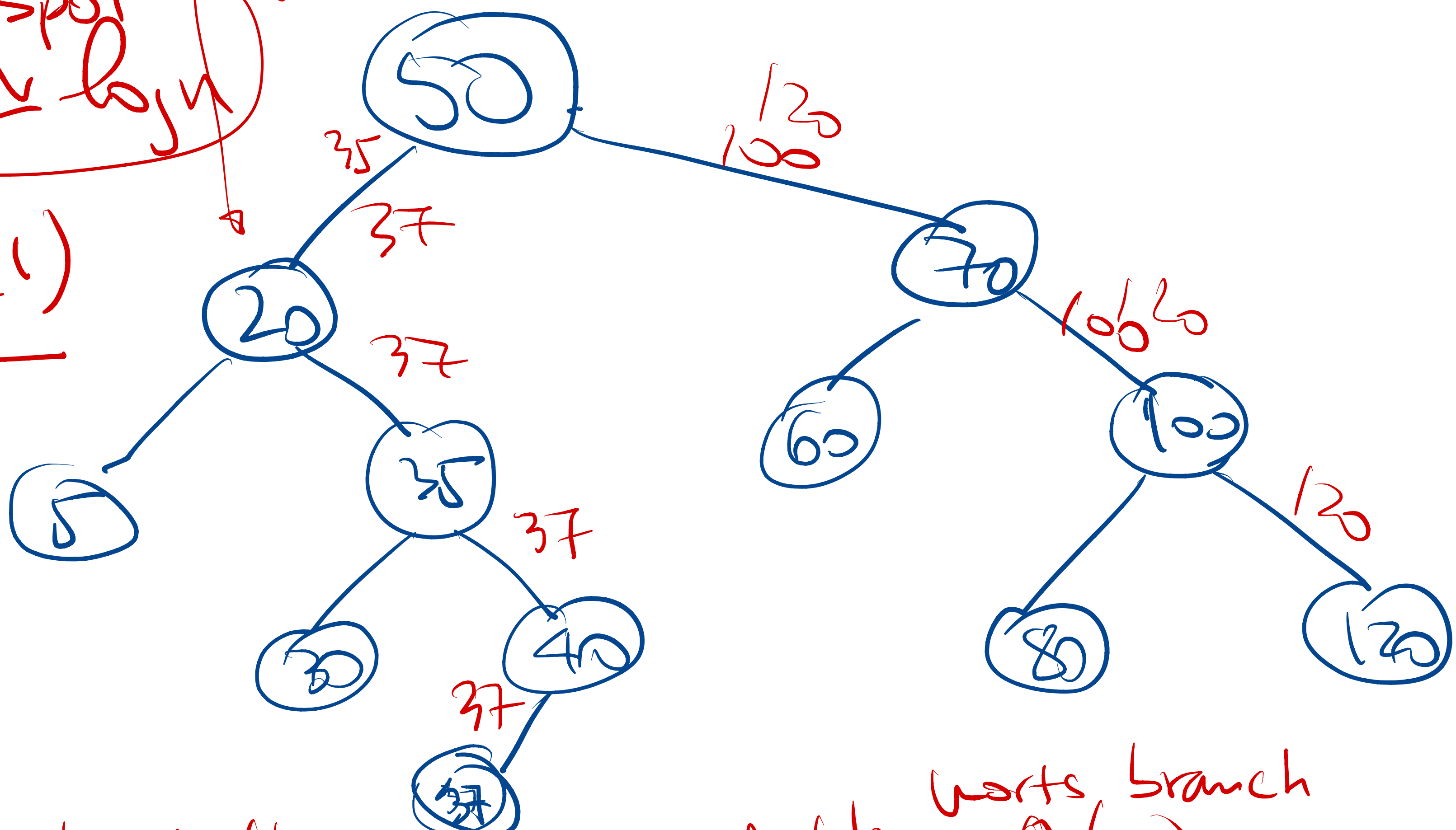
	list DS	array-DS	BT-DS
finding the spot	$\Theta(n)$	BS: $\log(n)$	$\log(n)$?
insertion make space	$\Theta(1)$	$\Theta(n)$	$\Theta(1)$

Insertion Sort with BST

Input 50, 20, 70, 35, 5, 100, 60, 80, 30, 40, 120, 37

- search for spot $\simeq$ depth $\simeq \log n$ with
- insert $\Theta(1)$

- inorder
  $\Downarrow$
  sorted



if depth $= \Theta(\log n)$
BALANCED

worst branch depth $= \Theta(n)$
UNBALANCED
$\Theta(n^2)$ sort time

Merge sort
procedure/steps

$T(n) = 2T(n/2) + n$

$\Theta(n \log n)$

- worst case

every case

$O(n \log n)$

6, 3, 7, 2, | 4, 8, 1, 5
S1

63 | 72   48 | 15
S2   S3

S4 S5 S6 S7

6 | 3   7 | 2   4 | 8   1 | 5

merging

36   27   48   15

merge

2367   1458

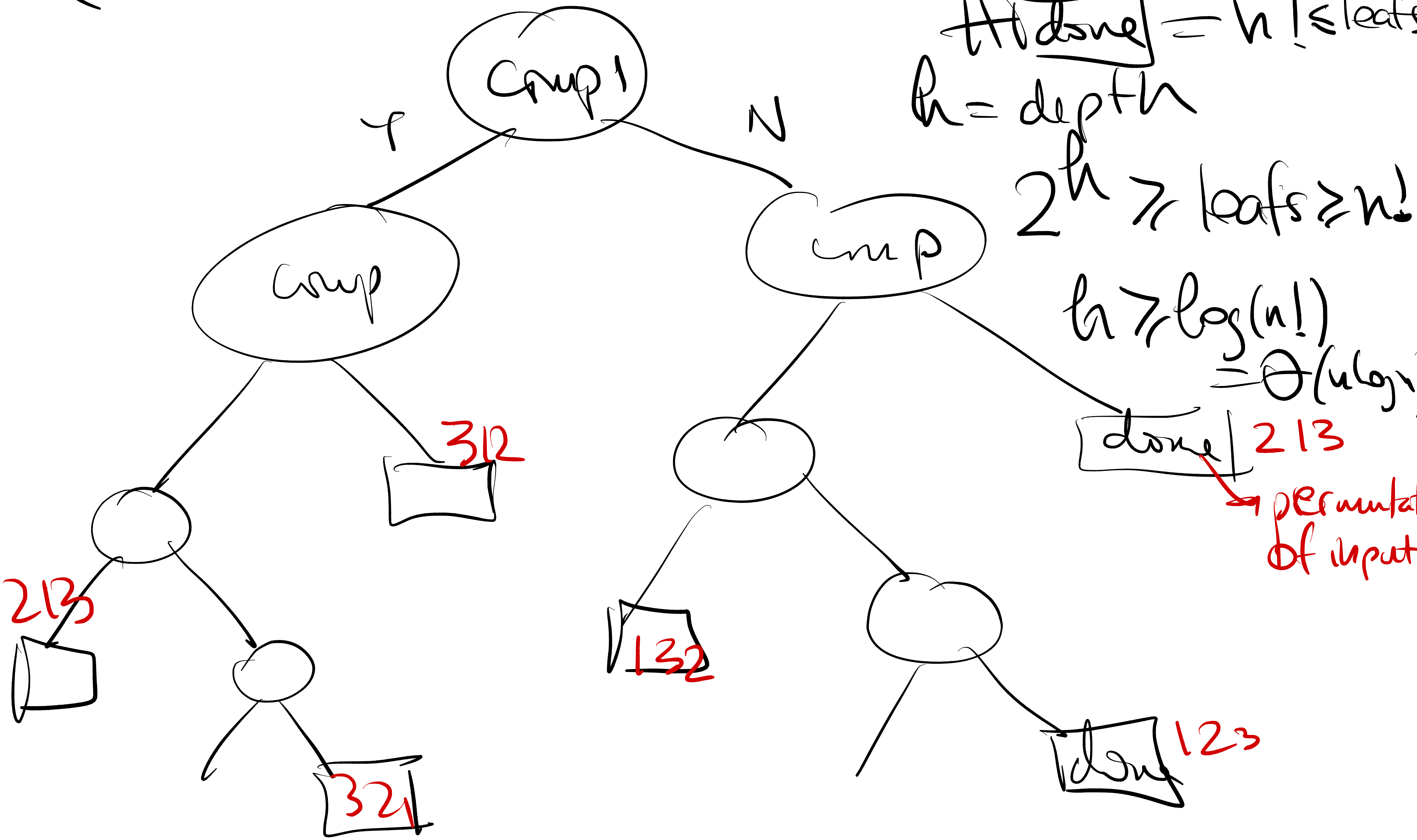
merge

12345678

Th Any comparison-based ALG (A) $|A| = n$ $\quad n=3$

asymptotically at least $\Theta(n \log n)$

(ALG sorts any array correctly)



$\#\text{done} = n! \le \text{leafs}$

$h = \text{depth}$

$2^h \geq \text{leafs} \geq n!$

$h \geq \log(n!) = \Theta(n \log n)$

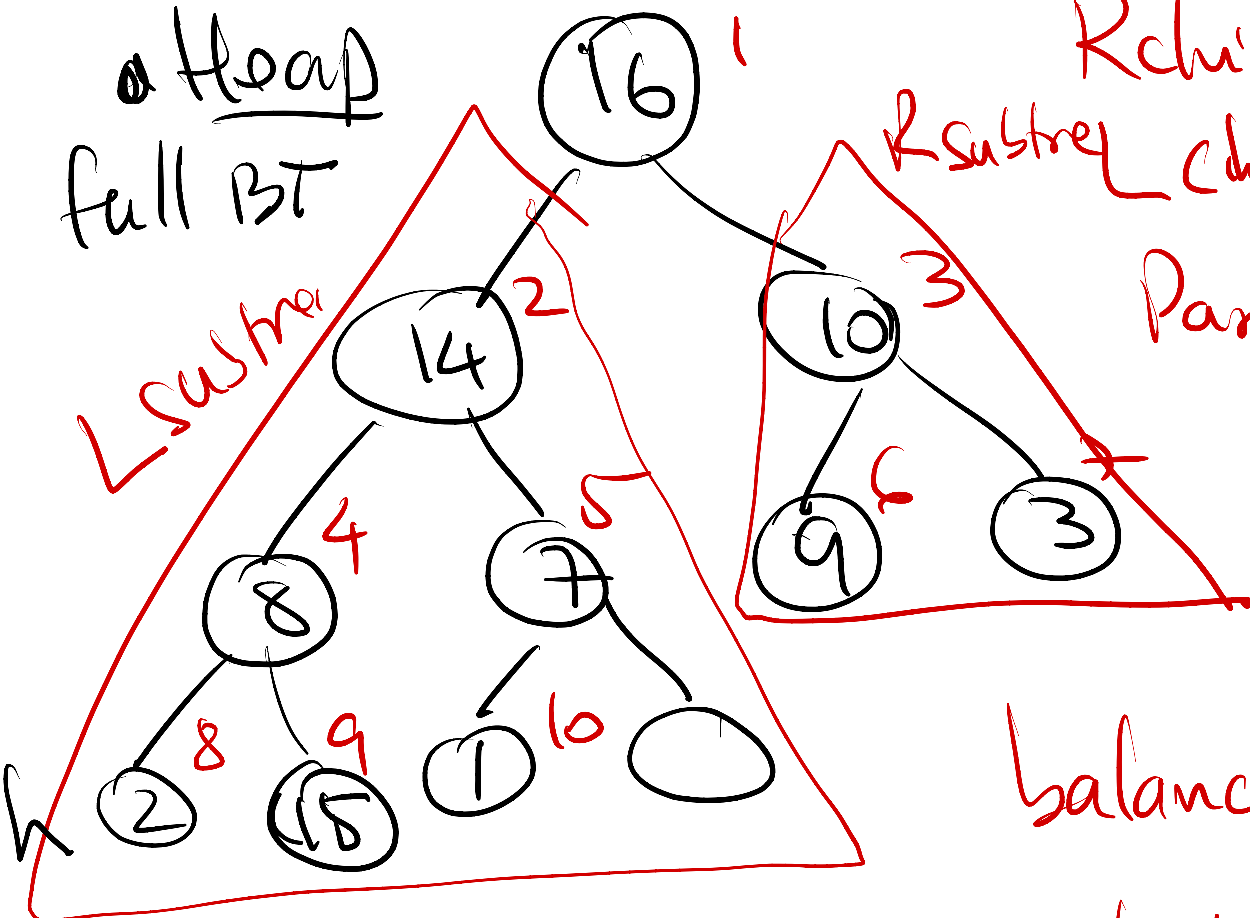
HeapSort

input Array

1	2	3	4	5	6	7	8	9	10
16	14	10	8	7	9	3	2	15	1

a Heap

full BT



$$\text{Rchild}(\text{index } k) = 2k+1$$

$$\text{Lchild}(\text{index } k) = 2k$$

$$\text{Parent}(k) = \left\lfloor \frac{k}{2} \right\rfloor$$

$$\text{depth} = \log(n)$$

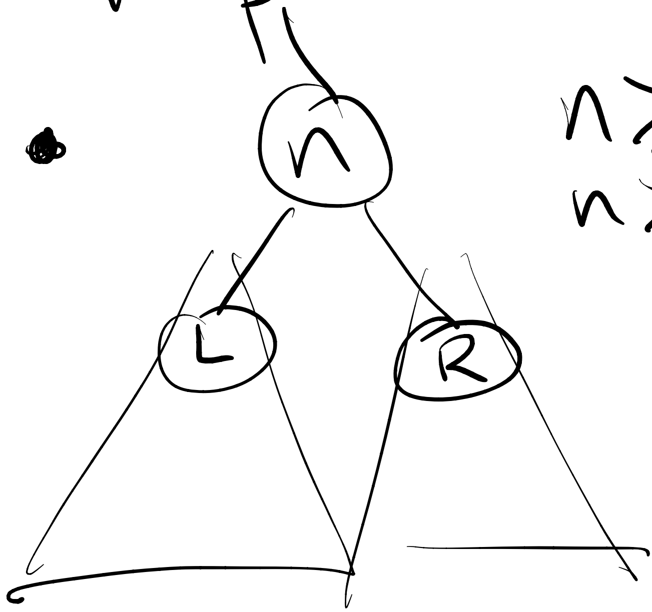
$$\text{balance} = \text{ratio} \; \frac{|L\ \text{Subtree}|}{|R\ \text{Subtree}|}$$

$$1 \leq \text{balance} \leq \underbrace{\frac{2^h - 1}{2^{h-1} - 1}}_{\text{max balance}} \simeq 2$$

layer h full-RS, empty-RS

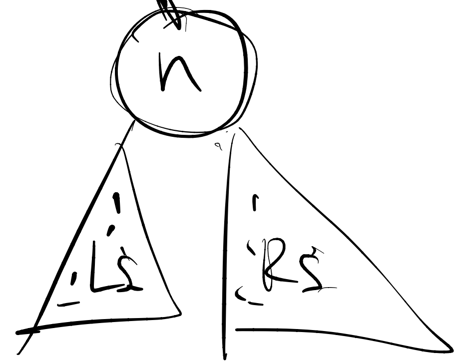
Max-heap

- heap

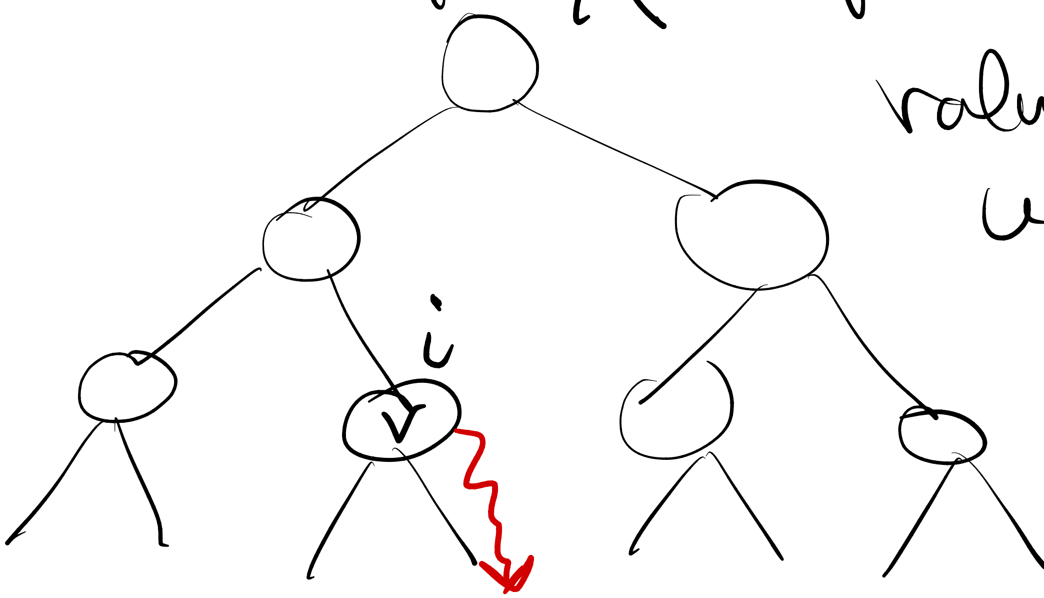


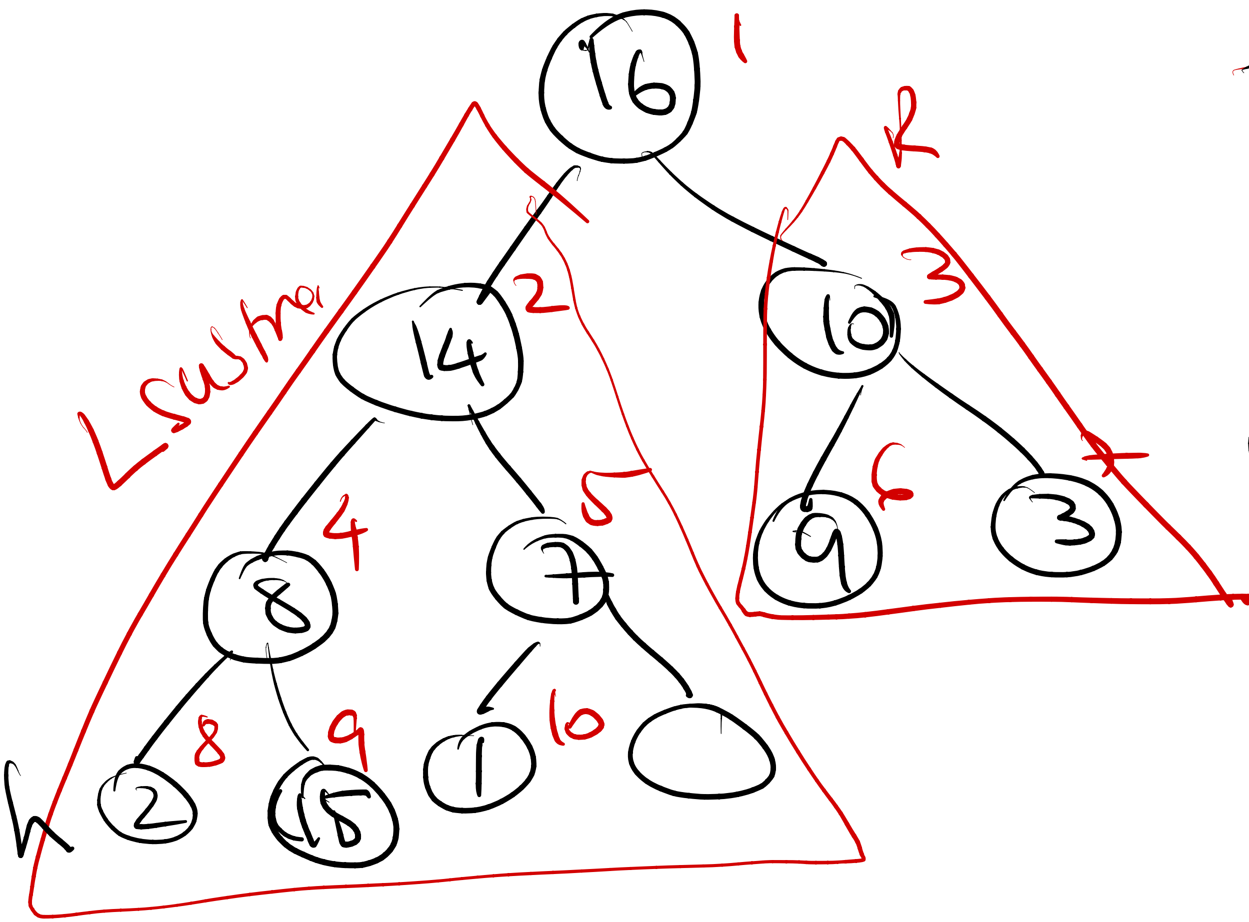
$n \geq L$
$n \geq R$

BST : $Ls \leq n \leq Rs$



Max-heapify(heap ~~H~~, index i / val v) : float that value down in the heap until its good

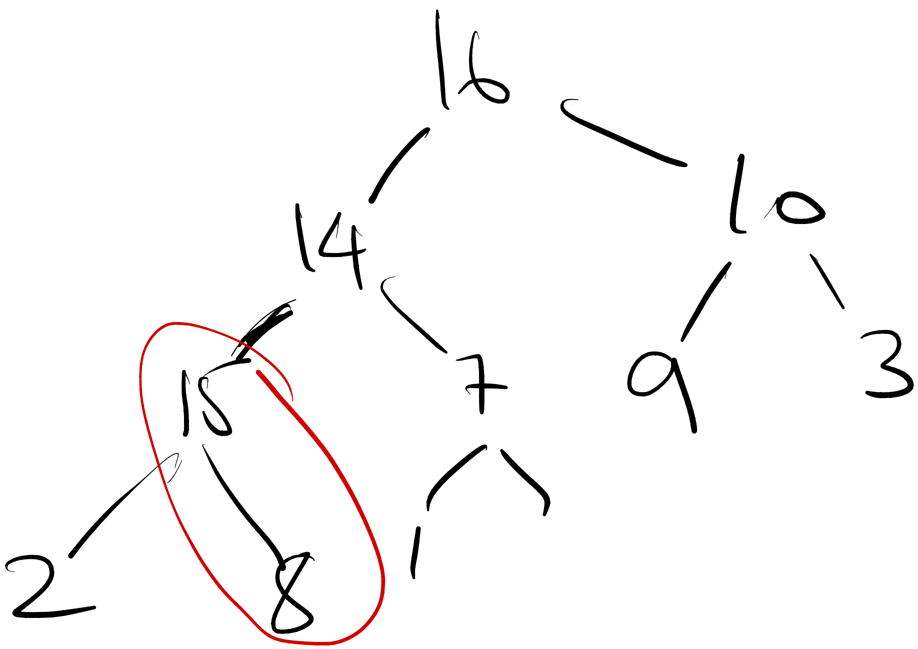




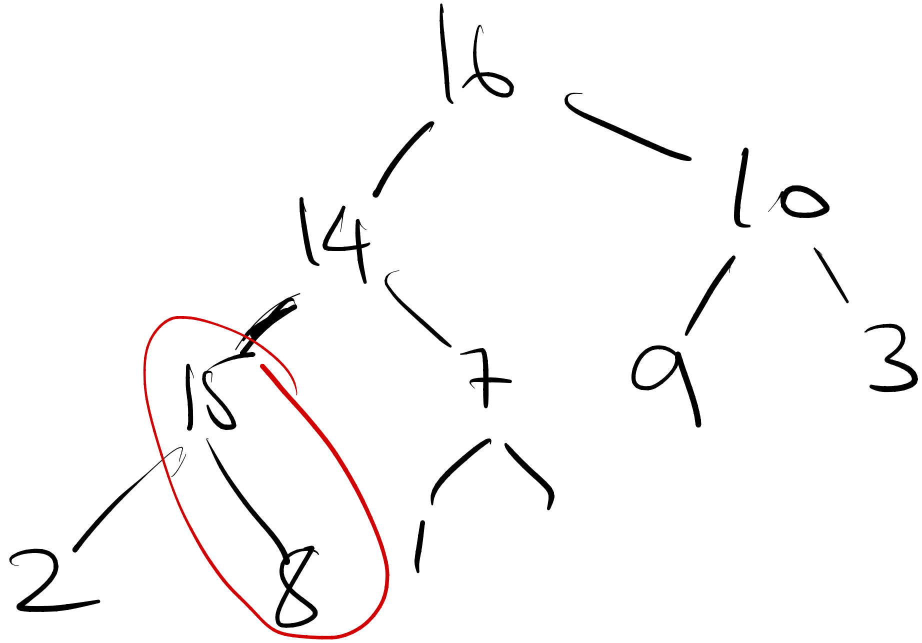
float down 8, 14
(val too small)
along the branc

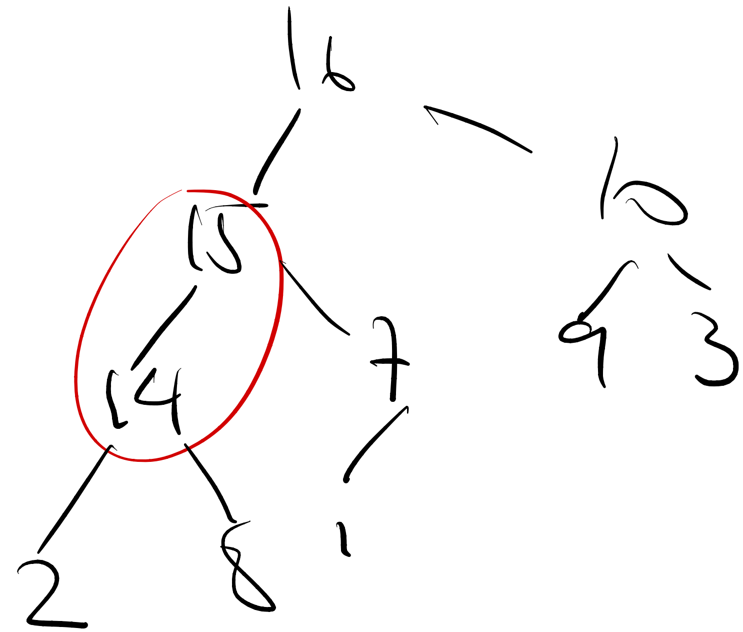
$\boxed{-8}$ float down children(8)

if 8 > (2, 15) done
else 8<15 swap
repeat untill 8 is bigger than both children



$\boxed{-14}$





14 - swap down with 15

RT ~~M~~MAX-heap

$\#\text{swaps} \times \Theta(\text{swap})$

$\Theta(\log n) \times \Theta(1)$

$\Theta(\log n)$

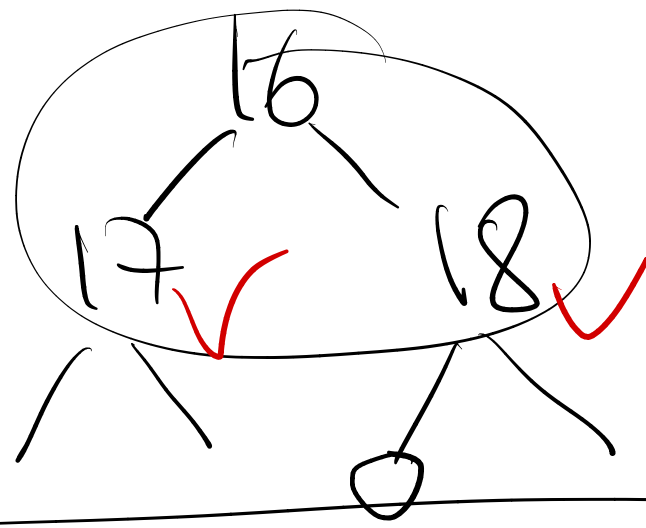
- 14 - vs (2,8) done

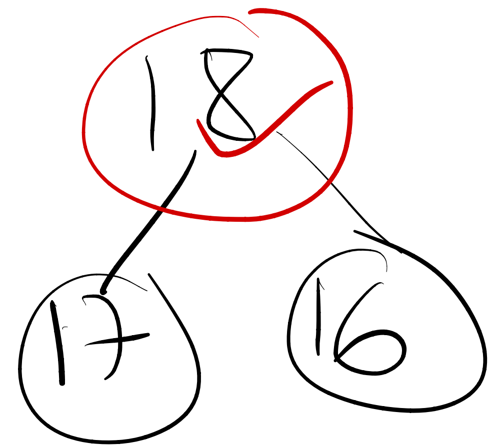
14 is in correct spot

Max heap (H)

for i = index n : downto : 1 $n/2$? (exclude leafs) $n/2$

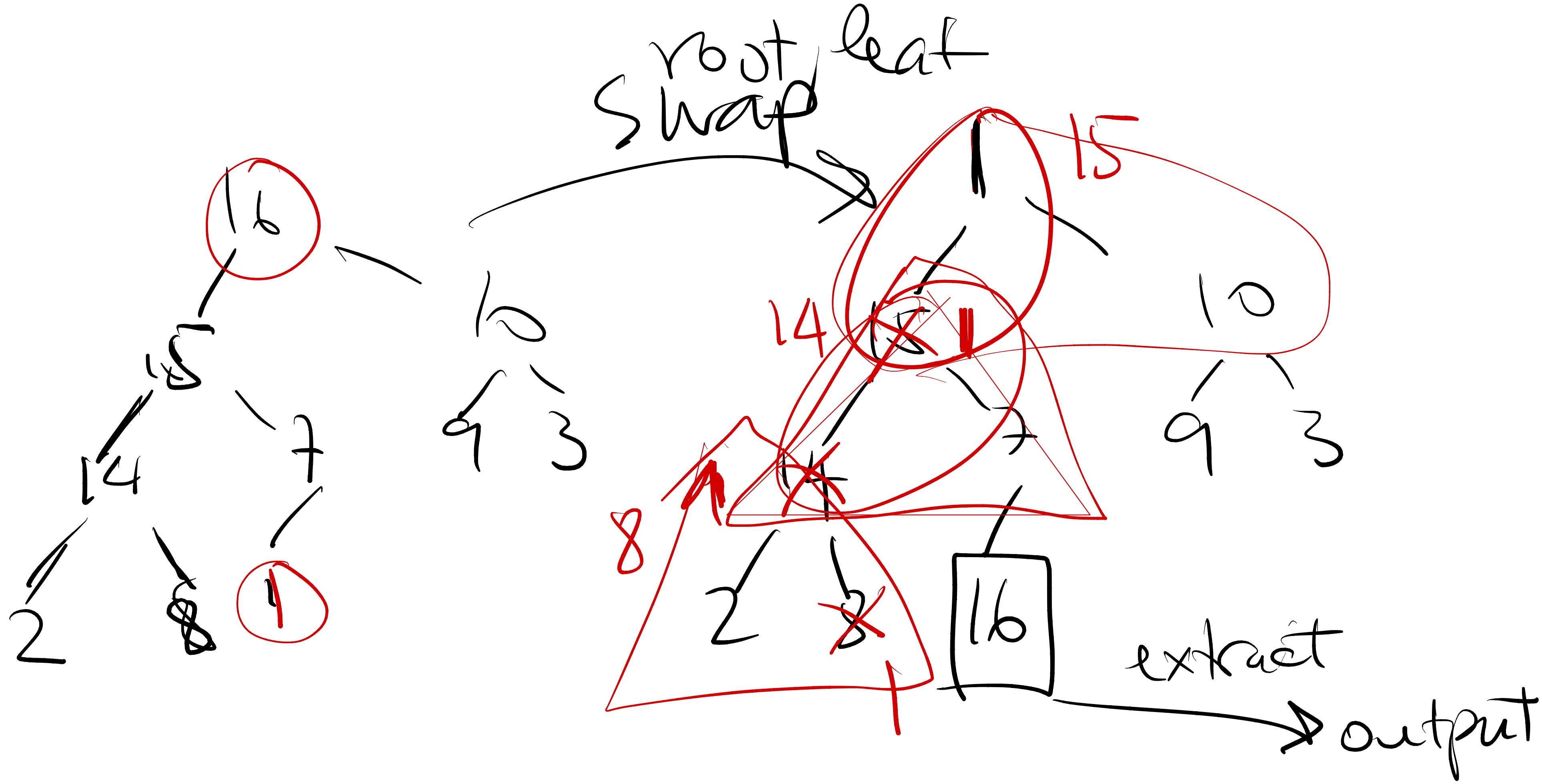
Max-heapify (H, i)





HeapSort - Decrease $\Theta(n \log n)$

- create Heap H
- Max-heapify (H)
- Loop all elem $\Theta(n)$
  - swap (root) with leaf } extract(root) $\Theta(1)$
  - Max Heapify (H, root) $\Theta(\log n)$



- fix max-heap property

Max-Heapify (H, root) floats down $\boxed{1}$ to correct spot.

Quicksort (A[b:e])

A [0, 29, 18, 7, 10, 7 - -]

1. p = pivot
   A[p] = 7

PARTITION (non-rec)

1 2 3 A

5	0	-3	7	7	29	18	8	21	10

smaller than A[p] p = 5

pivot (index)

bigger than A[p]

2. QS(A[b:p]) left side

3. QS(A[p+1 : e])

p = half

$T(n) = \Theta(\text{part}) + 2T(n/2)$

p = 1/3

$\Theta(\text{part}) + T(n/3) + T(2n/3)$

p

$\Theta(\text{part})$

$T(p) + T(n-p)$

need $\Theta(\text{part}) = \Theta(n)$

worst p = 1

$T(n) = \Theta(n) + T(1) + T(n-1)$

Partition

i=0 j=1 unprocessed

2 8 7 13 5 6 | 4 pivot value

≤ pivot val 4 | > pivot val | unprocessed

i j

1 : i i+1 : j-1 j : n-1

2<4 2 | 8 7 13 5 6 4

8>4 2 | 8 | 7 13 5 6 4

7>4 2 | 8 7 | 1 3 5 6 4

1<4 2 1 | 7 8 | 3 5 6 4 $\Theta(1)$

3<4 2 1 3 | 8 7 | 5 6 $\Theta(1)$

2 1 3 | 8 7 5 6 | 4

swap (1,8) → first in big group

i++ } j++

$\Theta(n)$

2 1 3 4 8 7 5 6

≤ 4 > 4

<u>Exercise</u>

(A) Partition with $p = \text{median}$?

(B)

$$n \cdot r\% < p \le n(1-r)\%$$

$r = 1\%$

$$\boxed{\frac{n}{100} \le p \le \frac{99n}{100}}$$

prove that $T(n) = \Theta(n \log n)$

$$T(n) = T(p) + T(n-p) + \Theta(n)$$

Th ~~On average~~ ?? $T(n) = \Theta(n \log n)$

prob(p = index) = uniform = $\frac{1}{n}$ (shuffle the array once)

left side	right side	prob
	0	1/n
n-1	1	1/n
n-2	2	1/n
n-3	⋮	1/n
⋮	⋮	
1	n-2	1/n
0	n-1	

$$T(n) = n + \frac{1}{n}\sum_{p=0}^{n-1}\left[T(p) + T(n-p-1)\right]$$