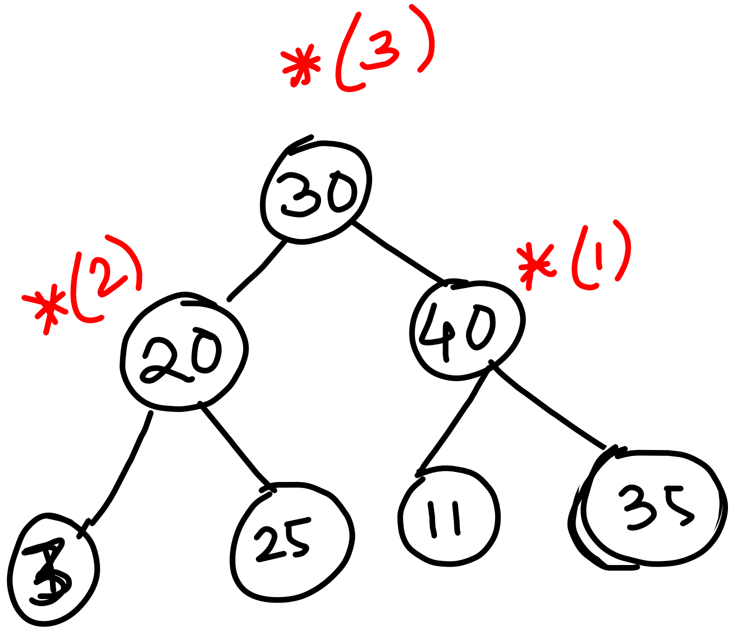
Heap Sort

Heapify (A, i) → Heapify on i^{th} node. → $O(\log n)$

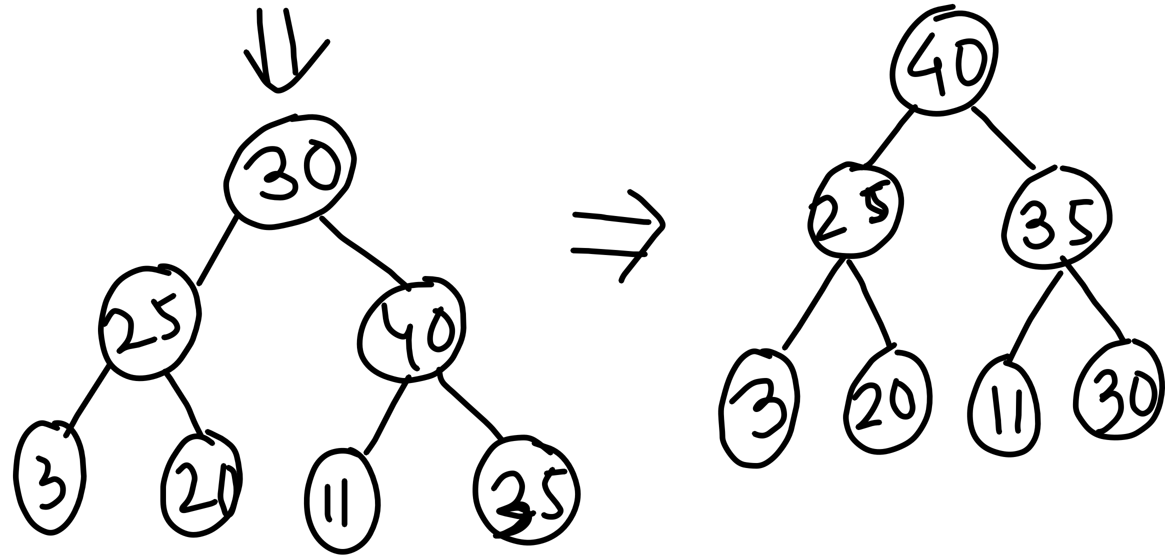
Build Heap (A) → Make A a max-heap.

For $i = \lfloor n/2 \rfloor$ to 1
  Heapify (A, i)

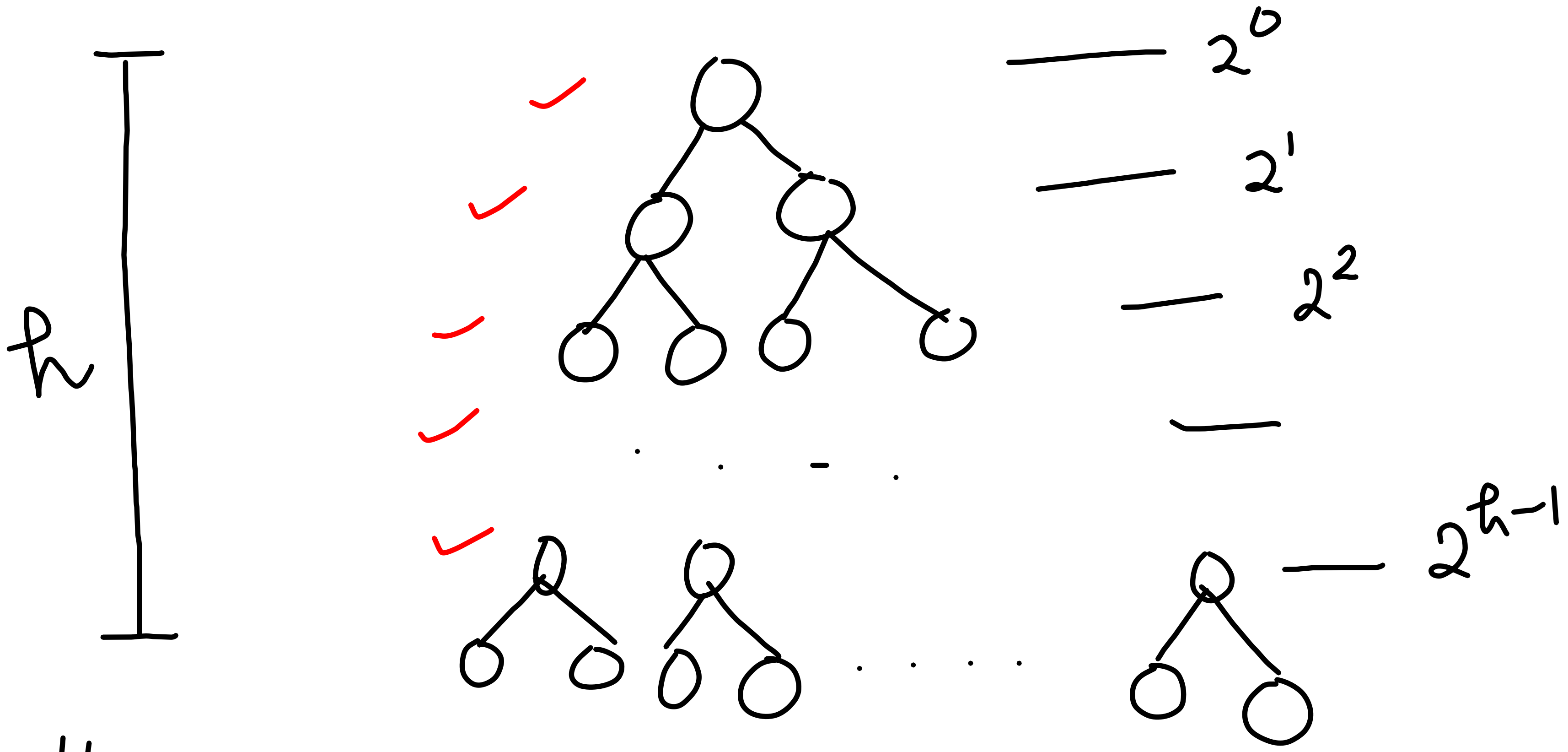


$1+1+2$

height(40) $= 1$



40



How many time Heapify(·) executed?

\# nodes $= n$

$h = \log n$

$\frac{i+1}{2^i} - \frac{1}{2^i}$

$2^{h-1} \cdot 1 + 2^{h-2} \cdot 2 + \cdots + 2^1 \cdot (h-1) + 2^0 \cdot h$

$$= \sum_{i=1}^{h} 2^{h-i} \cdot i$$

$$= 2^h \sum_{i=1}^{h} \frac{i}{2^i}$$

$$= n \sum_{i=1}^{\log n} \frac{i}{2^i}$$

$$= O(n)$$

Heap Sort (A)

$$
\left\{
\begin{array}{ll}
\text{Build-Heap}(A) & \text{———} \; O(n) \\
\text{For } i = n \text{ to } 1 & \\
\quad \text{Swap } A[1] \text{ with } A[i] & \text{—} \; O(n) \\
\quad B[i] = A[i] & \text{—} \; O(n) \\
\quad A.size = A.size - 1 & \text{—} \; O(n) \\
\quad \text{Heapify}(A, 1) & \text{—} \; O(n \log n)
\end{array}
\right.
$$

$$\overline{}$$

$$O(n \log n)$$

Priority Queue

Design a data Structure to perform the following operations efficiently (on a sequence A)

1) max(A) → returns the max element
2) extract_max(A) → remove & extract the max element
3) insert(A, x) → insert element x into A.
4) increase_Key(A, i, k) → increase the value of i^{th} element to k.

$A \to$ Array

- max : $O(n)$
- extract max : $O(n)$
- insert : $O(1)$
- increase key : $O(1)$

$A \to$ link-list

- max : $O(n)$
- extract max : $O(n)$
- insert :

$A \rightarrow$ Array

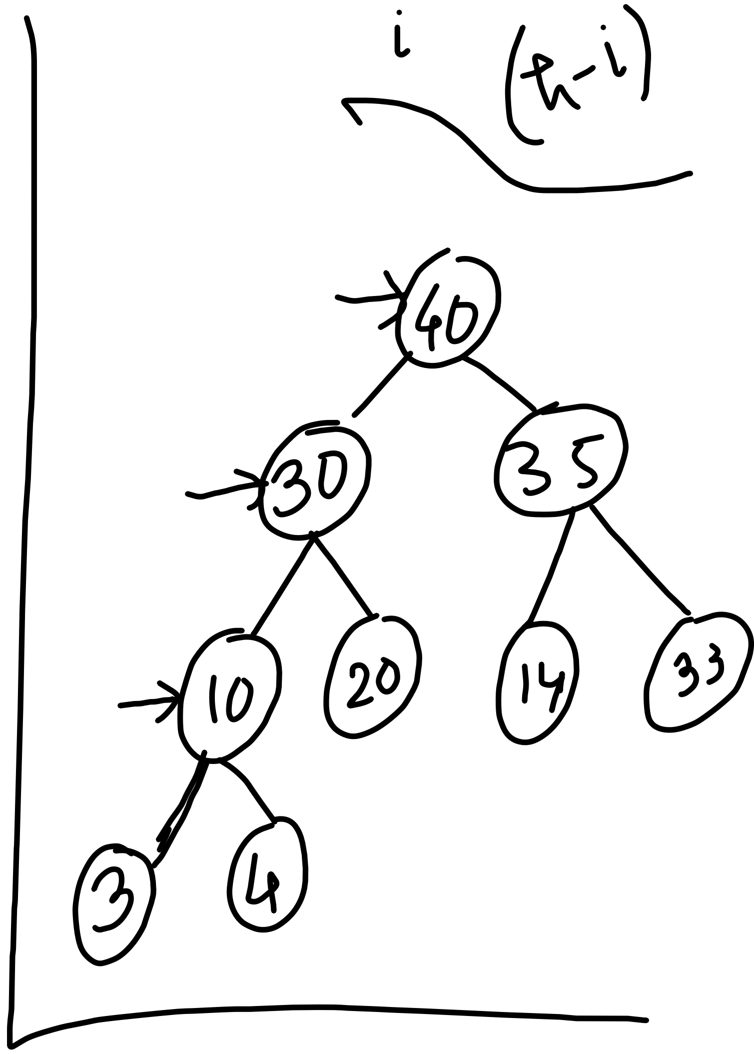
- max : $O(n)$
- extract max : $O(n)$
- insert : $O(n)$ [static], $O(1)$ [dynamic]
  amortize
- increase key : $O(1)$

$A \rightarrow$ link-list

- max : $O(n)$
- extract max : $O(n)$
- insert : $O(1)$
- increase key : $O(n)$

$A \rightarrow$ Heap

- max : $O(1)$
- extract_max : $O(\log n)$ [As used in Heap Sort]
- insert : $O(\log n)$
- increase key : $O(\log n)$

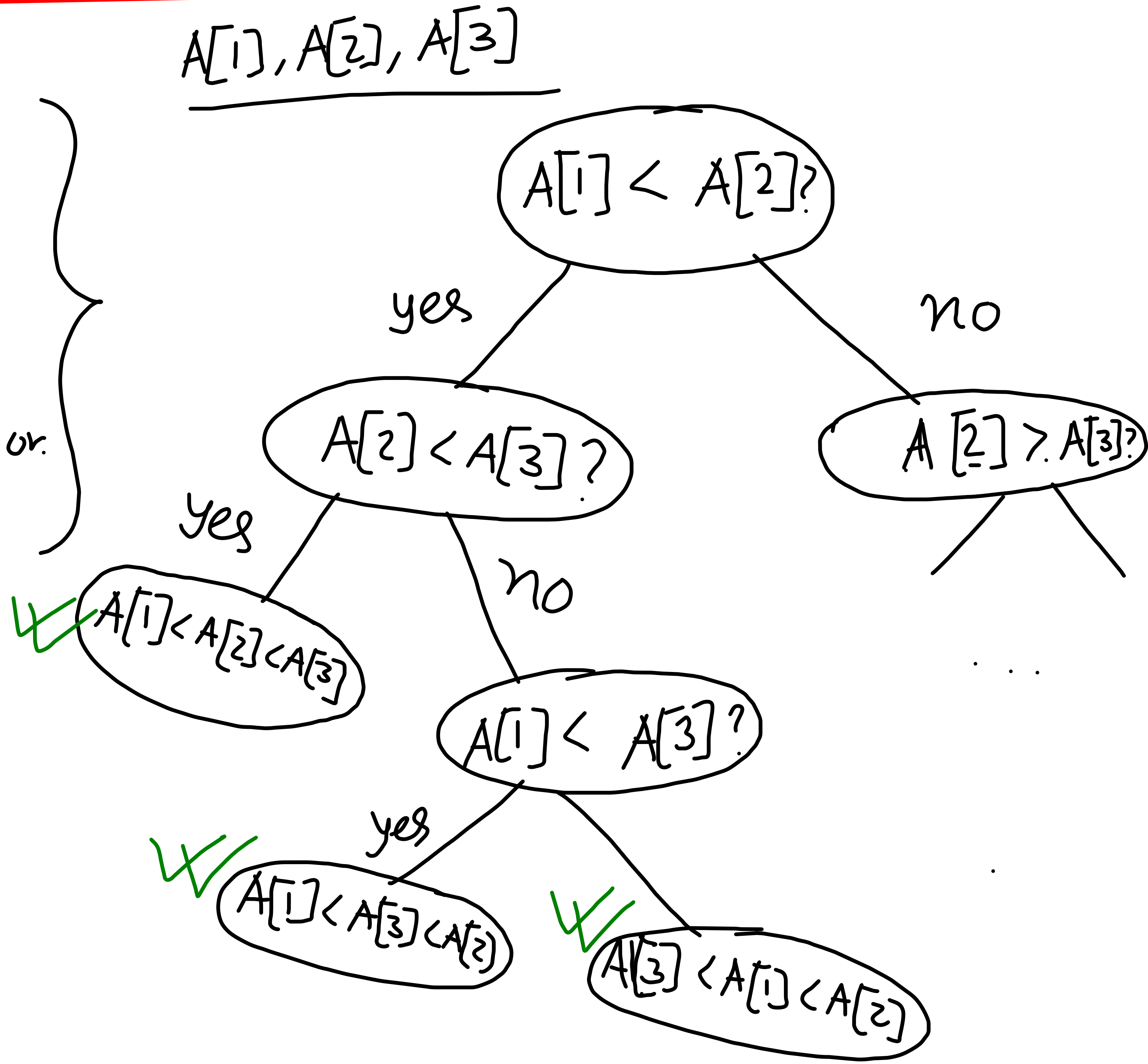


Comparison based Sorting — A lower bound

- Comparison Model
- For any two elements $A[i]$ & $A[j]$ we compare whether $A[i] < A[j]$ or.

Decision-Tree

Each leaf node gives you one Ordering.



An array of n elements which we would like to Sort.

- What would be # ordering $= n!$
- The height of your decision tree $= h$

$$2^h \geq n!$$

$$\boxed{h = \Omega(n \log n)}$$

$\log \frac{n}{2} + \log(\frac{n+1}{2}) + \cdots \log \ldots$

$\geq \frac{n}{2} \cdots$

$$h \geq \log n!$$

$$= \log(n \cdot (n-1) \cdots 2 \cdot 1)$$

$$= \log n + \log(n-1) + \cdots + \log 1$$

$$= \sum_{i=1}^{n} \log i \geq \sum_{i=n/2}^{n} \log \frac{n}{2}$$

$$\geq \frac{n}{2} \log \frac{n}{2}$$