2-2 The Efficiency of Algorithms

魏恒峰

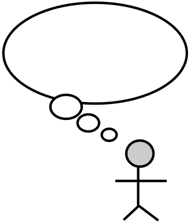
hfwei@nju.edu.cn

2018 年 04 月 02 日

(1) Diameter of Convex Polygon: $\Theta(n)$

(2) Lower Bound for Sorting: $\Omega(n \lg n)$

(3) Traversal over Trees: DFS/BFS ($\Theta(n)$)



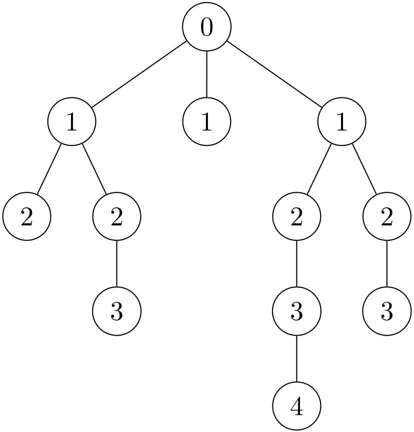
(1) Diameter of Convex Polygon: $\Theta(n)$

(2) Lower Bound for Sorting: $\Omega(n \lg n)$

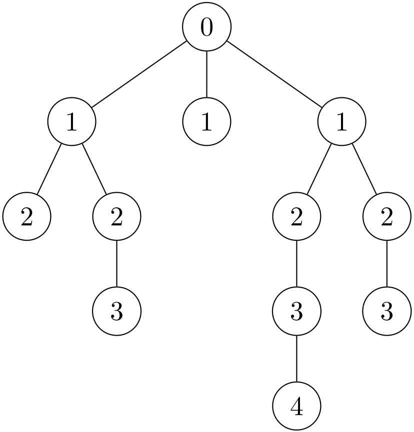
(3) Traversal over Trees: DFS/BFS ($\Theta(n)$)

I have thought that $\cdots$

DH 4.2 (a): Sum of Depths

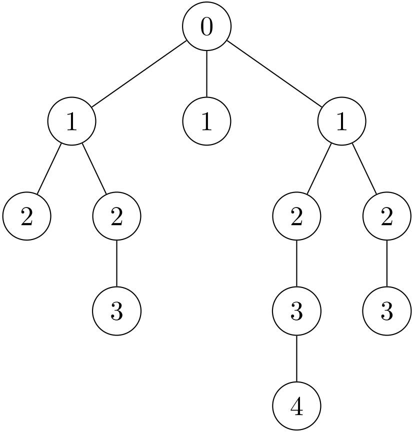


DH 4.2 (a): Sum of Depths



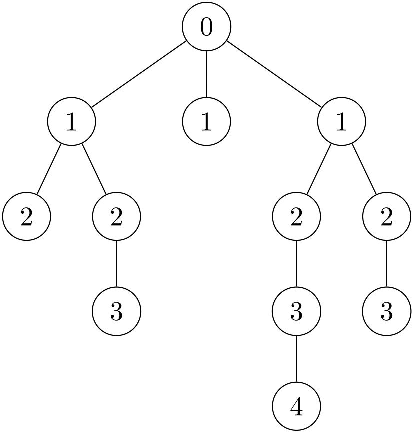
$$\text{sum-of-depths}(r) = \begin{cases} \\ \displaystyle\sum_{v\text{:child of } r} \text{sum-of-depths}(v) + \text{depth of } r, \end{cases}$$

DH 4.2 (a): Sum of Depths



$$\text{sum-of-depths}(r) = \begin{cases} \text{depth of } r, & r \text{ is a leaf} \\ \displaystyle\sum_{v:\text{child of } r} \text{sum-of-depths}(v) + \text{depth of } r, & \text{o.w.} \end{cases}$$

DH 4.2 (a): Sum of Depths



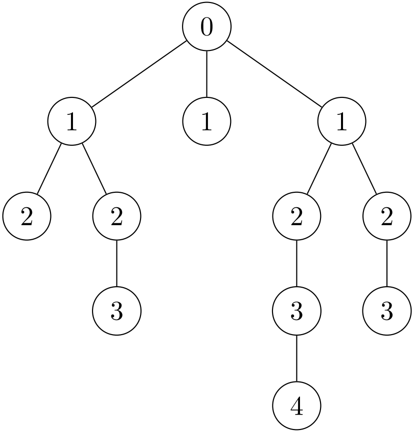
$$\text{sum-of-depths}(r, d) = \begin{cases} d, & r \text{ is a leaf} \\ \displaystyle\sum_{v:\text{child of } r} \text{sum-of-depths}(v, d+1) + d, & \text{o.w.} \end{cases}$$

Algorithm 1 Calculate the sum of depths of all nodes of a tree T.

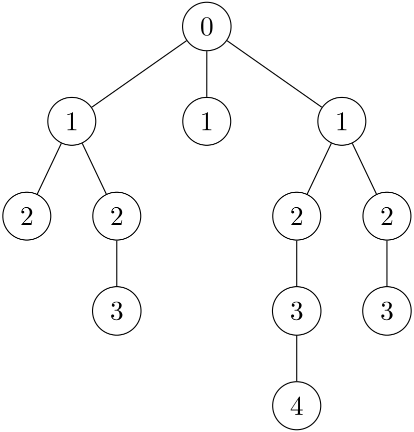
procedure $\textsc{Sum-of-Depths}()$
  return $\textsc{Sum-of-Depths}(T, 0)$

procedure $\textsc{Sum-of-Depths}(r, \mathit{depth})$ ▷ r: root of a tree
  if T is a leaf **then**
    return depth
  for all child vertex v of r **do**
    $\mathit{depth} \leftarrow \mathit{depth} + \textsc{Sum-of-Depths}(v, \mathit{depth} + 1)$
  return depth

DH 4.2 (b): Number of Nodes at Depth K

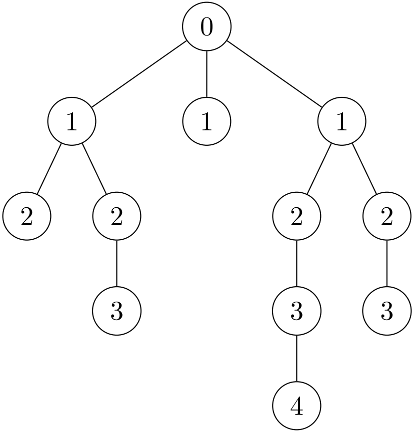


DH 4.2 (b): Number of Nodes at Depth K



$$\text{nodes-at-depth}(r, k) = \begin{cases} \\ \displaystyle\sum_{v:\text{child of } r} \text{nodes-at-depth}(v, k-1), \end{cases}$$

DH 4.2 (b): Number of Nodes at Depth K



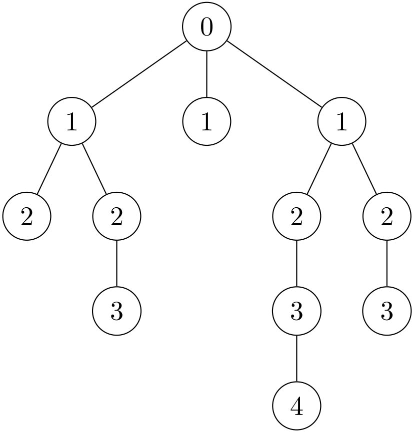
$$\text{nodes-at-depth}(r, k) = \begin{cases} 1, & k = 0 \\ 0, & k > 0 \land r \text{ is a leaf} \\ \displaystyle\sum_{v\text{:child of } r} \text{nodes-at-depth}(v, k-1), & \text{o.w.} \end{cases}$$

Algorithm 2 Count the number of nodes in T at depth K.

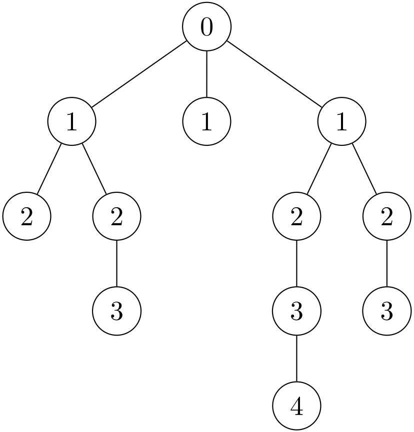
procedure $\textsc{Nodes-at-Depth}()$
    return $\textsc{Nodes-at-Depth}(T, K)$

procedure $\textsc{Nodes-at-Depth}(r, k)$ $\qquad\triangleright$ r: root of a tree
    if $k = 0$ **then**
        return 1
    if r is a leaf **then**
        return 0
    $num \gets 0$
    for all child vertex v of r **do**
        $num \gets num + \textsc{Nodes-at-Depth}(v, k-1)$
    return num

DH 4.2 (c): Any Leaf at an Even Depth?

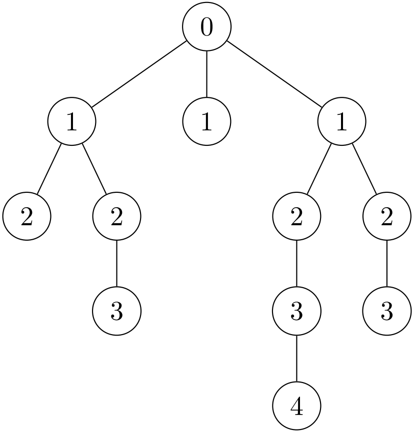


DH 4.2 (c): Any Leaf at an Even Depth?



$$\text{leaf-at-depth}(r, {\color{red}parity}) = \begin{cases} \\ \sum\limits_{v:\text{child of } r} (v, {\color{red}1 - parity}), \end{cases}$$

DH 4.2 (c): Any Leaf at an Even Depth?



$$\text{leaf-at-depth}(r, \color{red}{parity}) = \begin{cases} 1 - parity, & r \text{ is a leaf} \\ \displaystyle\sum_{v:\text{child of } r} (v, \color{red}{1 - parity}), & \text{o.w.} \end{cases}$$

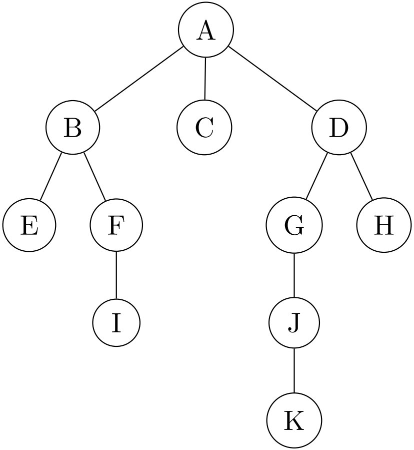
Algorithm 3 Check whether a tree T has any leaf at an even depth.

procedure $\textsc{Leaf-at-Even-Depth}()$
    return $\textsc{Leaf-at-Depth}(T, even = 0)$

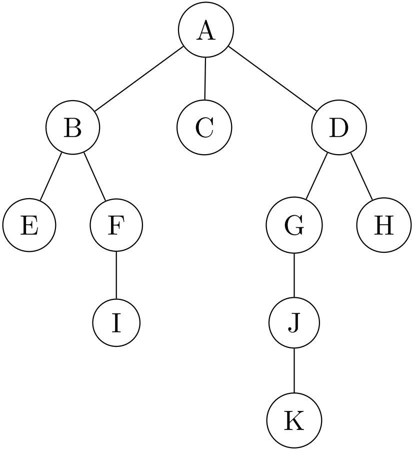
procedure $\textsc{Leaf-at-Depth}(r, parity)$ ▷ r: root of a tree
    if r is a leaf **then**
        return $1 - parity$

    $result \leftarrow 0$
    for all child vertex v of r **do**
        $result \leftarrow result \lor \textsc{Leaf-at-Depth}(v, 1 - parity)$
    return $result$

DH 4.3 (a): Sum of Contents at Each Depth



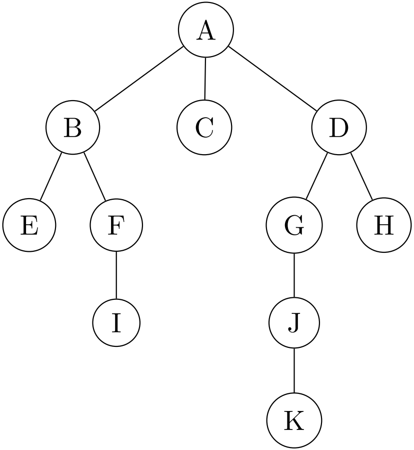
DH 4.3 (a): Sum of Contents at Each Depth



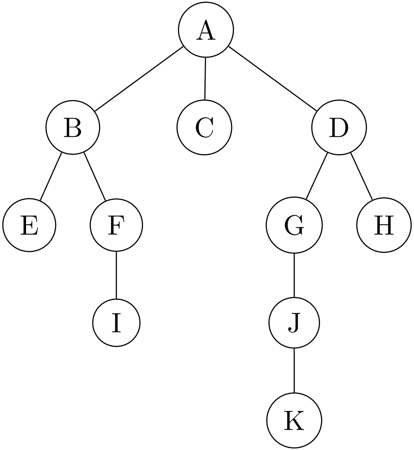
Algorithm 4 Calculate the sum of contents of nodes of a tree T at each depth.

```
procedure Sum-at-Depth(r)                      ▷ r: root of the tree T
    r.depth ← 0
    Q ← ∅
    Enqueue(Q, r)
    while Q ≠ ∅ do
        u ← Dequeue(Q)
        sumAtDepth[u.depth] += u.content
        for all child vertex v of u do
            v.depth ← u.depth + 1
            Enqueue(Q, v)
```

DH 4.3 (b): Depth K with the Maximum Number of Nodes



DH 4.3 (b): Depth K with the Maximum Number of Nodes



Algorithm 5 Count the number of nodes of a tree T at each depth.

```
procedure NODES-AT-DEPTH(r)                    ▷ r: root of the tree T
    r.depth ← 0
    Q ← ∅
    ENQUEUE(Q, r)
    while Q ≠ ∅ do
        u ← DEQUEUE(Q)
        nodesAtDepth[u.depth] += 1
        for all child vertex v of u do
            v.depth ← u.depth + 1
            ENQUEUE(Q, v)
```

Lower Bound for Comparion-based Sorting

Lower Bound for Comparion-based Sorting

Lower Bound for Comparison-based Sorting (DH 6.13)

Prove a lower bound of $O(n \lg n)$ on the time complexity of any comparison-based sorting algorithm.

Lower Bound for Comparison-based Sorting (DH 6.13)

Prove a lower bound of $O(n \lg n)$ on the time complexity of any comparison-based sorting algorithm.

Lower Bound for Comparison-based Sorting (DH 6.13)

Prove a lower bound of $O(n \lg n)$ on the time complexity of any comparison-based sorting algorithm.

Lower Bound for Comparison-based Sorting (DH 6.13)

Prove a lower bound of $\Omega(n \lg n)$ on the time complexity of any comparison-based sorting algorithm.

Lower Bound for Comparison-based Sorting (DH 6.13)

Prove a lower bound of $O(n \lg n)$ on the time complexity of any comparison-based sorting algorithm.

Lower Bound for Comparison-based Sorting (DH 6.13)

Prove a lower bound of $\Omega(n \lg n)$ on the time complexity of any comparison-based sorting algorithm on inputs of size n.

Lower Bound for Comparison-based Sorting (DH 6.13)

Prove a lower bound of $\Omega(n \lg n)$ on the time complexity of any **comparison-based** sorting algorithm on inputs of size n.

Lower Bound for Comparison-based Sorting (DH 6.13)

Prove a lower bound of $\Omega(n \lg n)$ on the time complexity of any **comparison-based** sorting algorithm on inputs of size n.

Computational Model:

the only way to gain order info.

Lower Bound for Comparison-based Sorting (DH 6.13)

Prove a lower bound of $\Omega(n \lg n)$ on the time complexity of any **comparison-based** sorting algorithm on inputs of size n.

Computational Model:

the only way to gain order info.

$x \in [1 \cdots 99]$

$x/10$

Lower Bound for Comparison-based Sorting (DH 6.13)

Prove a lower bound of $\Omega(n \lg n)$ on the time complexity of any **comparison-based** sorting algorithm on inputs of size n.

Cost Model:

the critical operations to count

Computational Model:

the only way to gain order info.

$$x \in [1 \cdots 99]$$

$$x/10$$

Lower Bound for Comparison-based Sorting (DH 6.13)

Prove a lower bound of $\Omega(n \lg n)$ on the time complexity of any **comparison-based** sorting algorithm on inputs of size n.

Cost Model:

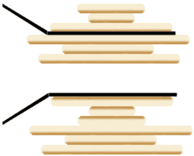
the critical operations to count

Computational Model:

the only way to gain order info.

$x \in [1 \cdots 99]$

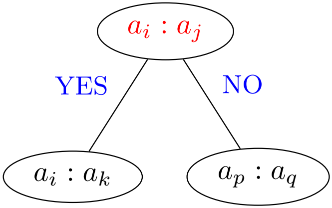
$x/10$



"Bounds For Sorting By Prefix Reversal", 1979

Decision Tree Model

Decision Tree Model



Nodes: comparisions $a_i : a_j$

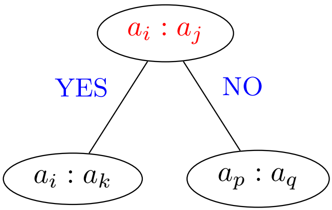
$$a_i < a_j,\ a_i \le a_j,\ a_i = a_j$$

$$a_i \ge a_j,\ a_i > a_j$$

Edges: two-way decisions

Leaves: possible permutations

Decision Tree Model



Nodes: comparisions $a_i : a_j$

$$a_i < a_j,\ a_i \leq a_j,\ a_i = a_j$$

$$a_i \geq a_j,\ a_i > a_j$$

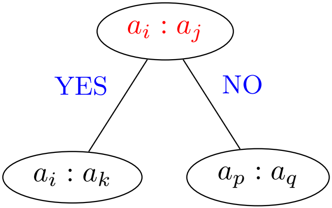
Edges: two-way decisions

Leaves: possible permutations

Assumption (By aware of any assumptions !!!):

All the input elements are **distinct**.

Decision Tree Model



Nodes: comparisions $a_i : a_j$

$$a_i < a_j, \; a_i \leq a_j, \; a_i = a_j$$

$$a_i \geq a_j, \; a_i > a_j$$

Edges: two-way decisions

Leaves: possible permutations

Assumption (By aware of any assumptions !!!):

All the input elements are **distinct**.

$$a_i < a_j$$

Decision Tree Model

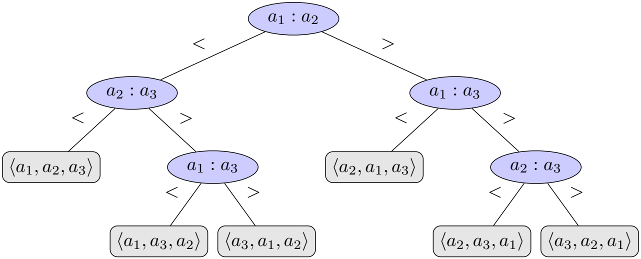
$$\boxed{\text{Any Comparison-based Sorting Algorithm} \xRightarrow{\text{modeled by}} \text{A Decision Tree}}$$

Decision Tree Model

$$\boxed{\text{Any Comparison-based Sorting Algorithm} \xRightarrow{\text{modeled by}} \text{A Decision Tree}}$$

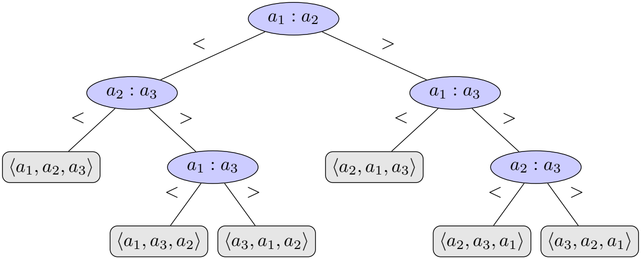
Decision Tree Model

Any Comparison-based Sorting Algorithm $\xRightarrow{\text{modeled by}}$ A Decision Tree



Decision Tree Model

Any Comparison-based Sorting Algorithm $\xRightarrow{\text{modeled by}}$ A Decision Tree



The decision tree for insertion sort on three elements.

Decision Tree Model

$$\boxed{\text{Any Comparison-based Sorting Algorithm} \xRightarrow{\text{modeled by}} \text{A Decision Tree}}$$

Decision Tree Model

Any Comparison-based Sorting Algorithm $\xRightarrow{\text{modeled by}}$ A Decision Tree

```
procedure                -Sort(A, n)
    for i ← 1 to n − 1 do
        for j ← i + 1 to n do
            if A[j] < A[i] then
                Swap(A[j], A[i])
```

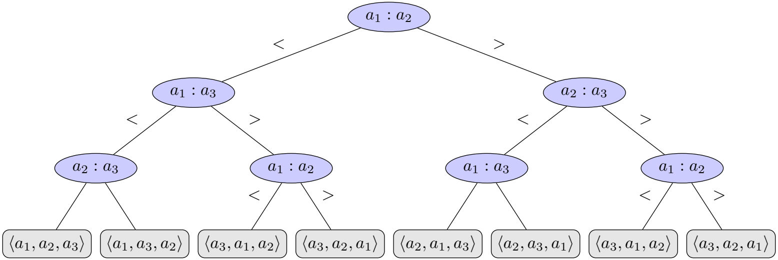
Decision Tree Model

$$\text{Any Comparison-based Sorting Algorithm} \xRightarrow{\text{modeled by}} \text{A Decision Tree}$$

```
procedure SELECTION-SORT(A, n)
    for i ← 1 to n − 1 do
        for j ← i + 1 to n do
            if A[j] < A[i] then
                SWAP(A[j], A[i])
```

Decision Tree Model

Any Comparison-based Sorting Algorithm $\xRightarrow{\text{modeled by}}$ A Decision Tree



The decision tree for selection sort on three elements.

Decision Tree Model

$$\boxed{\text{Any Comparison-based Sorting Algorithm } \mathcal{A} \xRightarrow{\text{modeled by}} \text{A Decision Tree } \mathcal{T}}$$

Decision Tree Model

$$\boxed{\text{Any Comparison-based Sorting Algorithm } \mathcal{A} \xrightarrow{\text{modeled by}} \text{A Decision Tree } \mathcal{T}}$$

$$\text{Algorithm } \mathcal{A} \text{ on a specific input of size } n \xrightarrow{\text{modeled by}} \text{A path through } \mathcal{T}$$

Decision Tree Model

$$\boxed{\text{Any Comparison-based Sorting Algorithm } \mathcal{A} \xRightarrow{\text{modeled by}} \text{A Decision Tree } \mathcal{T}}$$

$$\text{Algorithm } \mathcal{A} \text{ on a specific input of size } n \xRightarrow{\text{modeled by}} \text{A path through } \mathcal{T}$$

$$\text{Worst-case time complexity of } \mathcal{A} \xRightarrow{\text{modeled by}} \text{The height of } \mathcal{T}$$

Decision Tree Model

$$\boxed{\text{Any Comparison-based Sorting Algorithm } \mathcal{A} \xRightarrow{\text{modeled by}} \text{A Decision Tree } \mathcal{T}}$$

Algorithm $\mathcal{A}$ on a specific input of size $n \xRightarrow{\text{modeled by}}$ A path through $\mathcal{T}$

Worst-case time complexity of $\mathcal{A} \xRightarrow{\text{modeled by}}$ The height of $\mathcal{T}$

$$\boxed{\begin{array}{c}\text{Worst-case Lower Bound of Comparison-based Sorting on inputs of size } n \\ \xRightarrow{\text{modeled by}} \\ \text{The Minimum Height of All } \mathcal{T}\text{s}\end{array}}$$

Decision Tree Model

Worst-case Lower Bound of Comparison-based Sorting on inputs of size n

$$\xRightarrow{\text{modeled by}}$$

The Minimum Height of All $\mathcal{T}$s

Decision Tree Model

Worst-case Lower Bound of Comparison-based Sorting on inputs of size n

$$\xRightarrow{\text{modeled by}}$$

The Minimum Height of All $\mathcal{T}$s

To be a full binary tree:

$$\# \text{ of leaves } \leq 2^h$$

Decision Tree Model

Worst-case Lower Bound of Comparison-based Sorting on inputs of size n

$$\xRightarrow{\text{modeled by}}$$

The Minimum Height of All $\mathcal{T}$s

To be a full binary tree:

$$\# \text{ of leaves } \leq 2^h$$

To be a correct sorting algorithm:

$$\# \text{ of leaves } \geq n!$$

Lower Bound for Comparison-based Sorting

$$\boxed{n! \leq \# \text{ of leaves} \leq 2^h}$$

Lower Bound for Comparison-based Sorting

$$\boxed{n! \leq \#\ \text{of leaves} \leq 2^h}$$

$$h \geq \lg n! = \Omega(n \lg n)$$

Lower Bound for Comparison-based Sorting

$$\boxed{n! \le \#\ \text{of leaves} \le 2^h}$$

$$h \ge \lg n! = \Omega(n \lg n)$$

Stirling Formula (by *James Stirling*):

$$n! = \Theta(\sqrt{2\pi n}\left(\frac{n}{e}\right)^n)$$

Proof. □

k-sorted

Repeated Elements Problem


Thank You!