

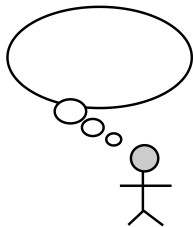
2-2 The Efficiency of Algorithms

魏恒峰

hfwei@nju.edu.cn

2018 年 04 月 02 日

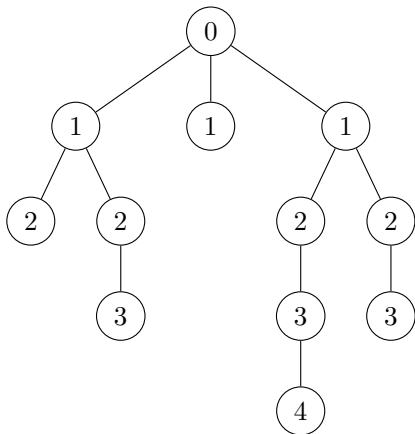




- (1) Diameter of Convex Polygon: $\Theta(n)$
- (2) Lower Bound for Sorting: $\Omega(n \lg n)$
- (3) Traversal over Trees: DFS/BFS ($\Theta(n)$)

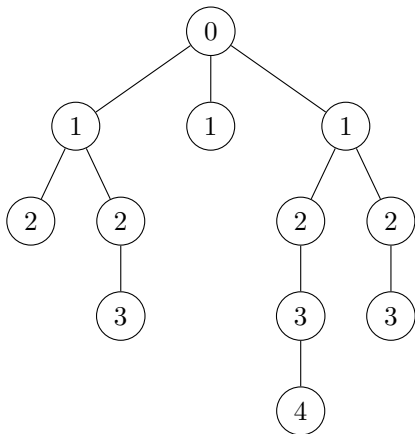
I have thought that ...

DH 4.2 (a): Sum of Depths



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

DH 4.2 (a): Sum of Depths



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

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

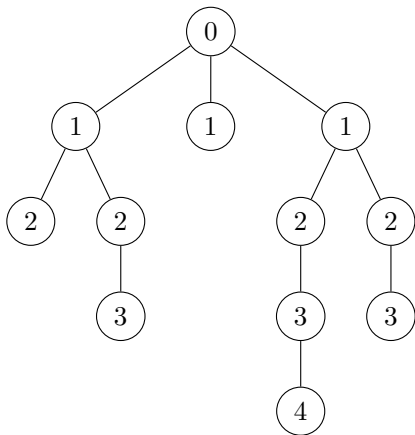
```
1: procedure SUM-OF-DEPTHS()  
2:   return SUM-OF-DEPTHS( $T, 0$ )  
  
3: procedure SUM-OF-DEPTHS( $r, depth$ )                                ▷  $r$ : root of a tree  
4:   if  $r$  is a leaf then  
5:     return  $depth$   
  
6:    $sum \leftarrow depth$   
7:   for all child vertex  $v$  of  $r$  do  
8:      $sum \leftarrow sum + \text{SUM-OF-DEPTHS}(v, depth + 1)$   
9:   return  $sum$ 
```

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

Master Theorem?

$$\Theta(m + n) = \Theta(n)$$

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

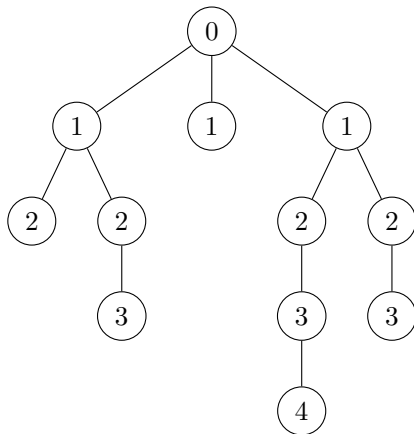


$$\text{nodes-at-depth}(r, k) = \begin{cases} 1, & k = 0 \\ 0, & k > 0 \wedge r \text{ is a leaf} \\ \sum_{v: \text{child of } r} \text{nodes-at-depth}(v, k-1), & \text{o.w.} \end{cases}$$

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

```
1: procedure NODES-AT-DEPTH()  
2:   return NODES-AT-DEPTH( $T, K$ )  
  
3: procedure NODES-AT-DEPTH( $r, k$ )                                ▷  $r$ : root of a tree  
4:   if  $k = 0$  then  
5:     return 1  
6:   if  $r$  is a leaf then  
7:     return 0  
8:    $num \leftarrow 0$   
9:   for all child vertex  $v$  of  $r$  do  
10:     $num \leftarrow num + \text{NODES-AT-DEPTH}(v, k - 1)$   
11:   return  $num$ 
```

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

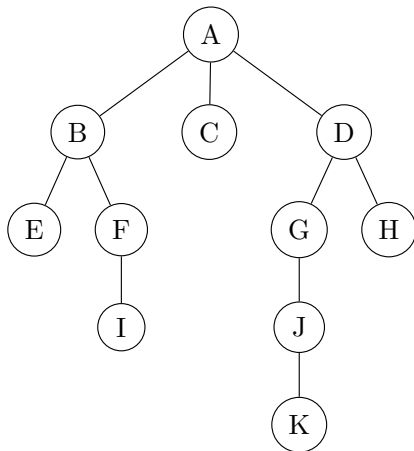


$$\text{leaf-at-depth}(r, \text{parity}) = \begin{cases} 1 - \text{parity}, & r \text{ is a leaf} \\ \bigvee_{v: \text{child of } r} (v, 1 - \text{parity}), & \text{o.w.} \end{cases}$$

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

```
1: procedure LEAF-AT-EVEN-DEPTH()  
2:   return LEAF-AT-DEPTH( $T, even = 0$ )  
  
3: procedure LEAF-AT-DEPTH( $r, parity$ )                                ▷  $r$ : root of a tree  
4:   if  $r$  is a leaf then  
5:     return  $1 - parity$   
  
6:    $result \leftarrow 0$   
7:   for all child vertex  $v$  of  $r$  do  
8:      $result \leftarrow result \vee \text{LEAF-AT-DEPTH}(v, 1 - parity)$   
9:   return  $result$ 
```

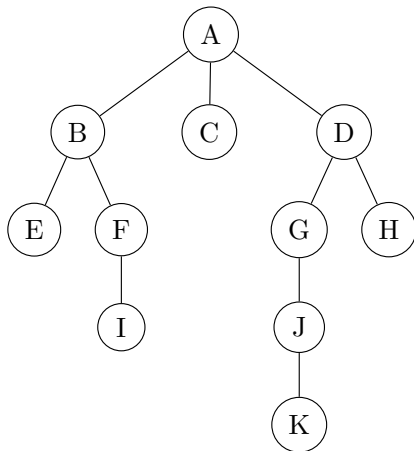
DH 4.3 (a): Sum of Contents at Each Depth



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

```
1: procedure SUM-AT-DEPTH( $r$ )                                ▷  $r$ : root of the tree  $T$ 
2:    $r.depth \leftarrow 0$ 
3:    $Q \leftarrow \emptyset$ 
4:   ENQUEUE( $Q, r$ )
5:   while  $Q \neq \emptyset$  do
6:      $u \leftarrow$  DEQUEUE( $Q$ )
7:      $sumAtDepth[u.depth] += u.content$ 
8:     for all child vertex  $v$  of  $u$  do
9:        $v.depth \leftarrow u.depth + 1$ 
10:    ENQUEUE( $Q, v$ )
```

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.

```
1: procedure NODES-AT-DEPTH( $r$ )                                ▷  $r$ : root of the tree  $T$ 
2:    $r.depth \leftarrow 0$ 
3:    $Q \leftarrow \emptyset$ 
4:   ENQUEUE( $Q, r$ )
5:   while  $Q \neq \emptyset$  do
6:      $u \leftarrow$  DEQUEUE( $Q$ )
7:      $nodesAtDepth[u.depth] += 1$ 
8:     for all child vertex  $v$  of  $u$  do
9:        $v.depth \leftarrow u.depth + 1$ 
10:      ENQUEUE( $Q, v$ )
11:  return  $\operatorname{argmax}_K nodesAtDepth[k]$ 
```



v.depth
sumAtDepth[]

$Q : \text{Space } \Theta(n) \rightarrow \Theta(1)$

Lower Bound for Comparison-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.



WRONG

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 .

Cost Model:

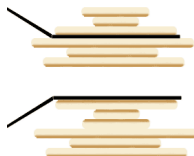
the critical operations to count

Computational Model:

the only way to gain order info.

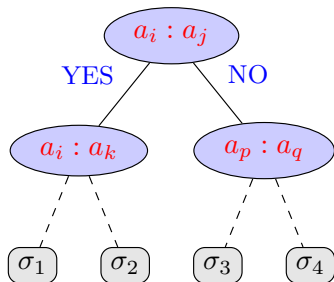
$$x \in [1 \cdots 99]$$

$$x/10$$



“Bounds For Sorting By Prefix Reversal”, 1979

Decision Tree Model



Nodes: comparisons $a_i : a_j$

$<, \leq, =, \geq, >$

Edges: two-way decisions (Y/N)

Leaves: possible permutations

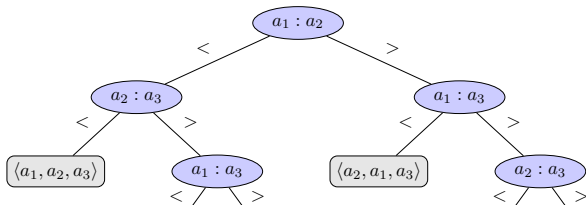
Assumption (By aware of any assumptions !!!):

All the input elements are **distinct**.

$$a_i < a_j$$

Decision Tree Model

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



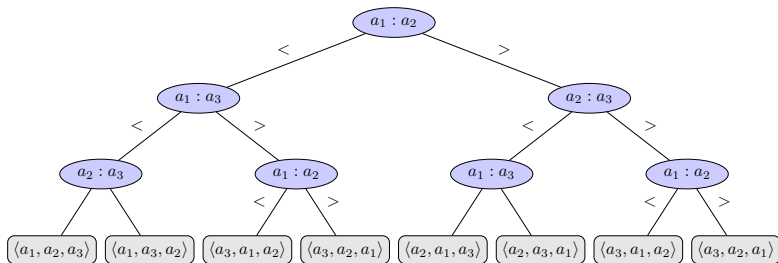
Decision Tree Model

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

```
1: procedure SELECTION-SORT( $A, n$ )  
2:   for  $i \leftarrow 1$  to  $n - 1$  do  
3:     for  $j \leftarrow i + 1$  to  $n$  do  
4:       if  $A[j] < A[i]$  then  
5:         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

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

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)

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

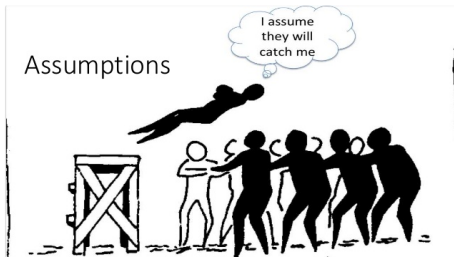
$$n! \leq \# \text{ of leaves} \leq 2^h$$

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

Stirling Formula (by *James Stirling*):

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





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

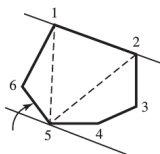
- (a) **Comparison-based** sorting algorithms.
- (b) All the input elements are **distinct**.
- (c) Completely sort all possible inputs.

The k -sorted Problem

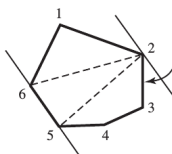
An array $A[1 \cdots n]$ is **k -sorted** if it can be divided into k blocks, each of size n/k (we assume that $n/k \in \mathbb{N}$), such that the elements in each block are larger than the elements in earlier blocks and smaller than elements in later blocks. The elements within each block need **not** be sorted.

- (a) Describe an algorithm that **k -sorts an arbitrary array** in $O(n \log k)$ time.
- (b) Prove that any comparison-based k -sorting algorithm requires $\Omega(n \log k)$ comparisons in the worst case.
- (c) Describe an algorithm that **completely sorts an already k -sorted array** in $O(n \log(n/k))$ time.
- (d) Prove that any comparison-based algorithm to completely sort a k -sorted array requires $\Omega(n \log(n/k))$ comparisons in the worst case.

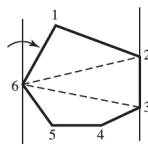
Convex Polygon Diameter



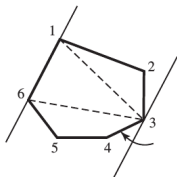
(a)



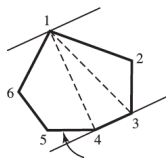
(b)



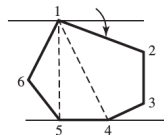
(c)



(d)



(e)



(f)

Convex Polygon Diameter (DH 6.8)

Show that the “Convex Polygon Diameter” algorithm is of **linear-time** complexity.

Q : Linear-time of **WHAT**?

A : Linear-time of **the size of input**

Q : What is the input?

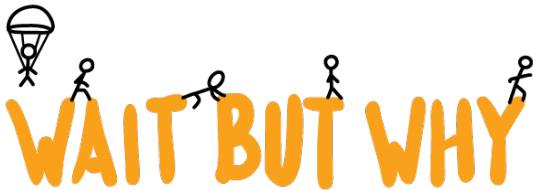
A : A convex polygon **represented by n vertices**

Q : What are the critical operations?

A : $d(p_1, p_2) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$

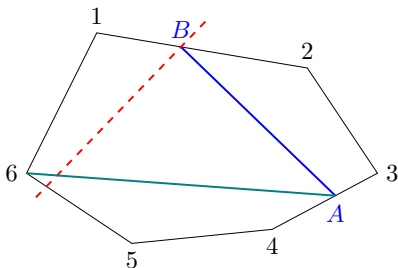
$$\Theta(c \cdot n) = \Theta(n)$$

Correctness



Theorem

For a convex polygon, a pair of vertices determine the diameter.



BUT, we have *not* enumerated *all* pairs of vertices.

We have enumerated *all* pairs of vertices
that *admits parallel supporting lines*.

Definition (Line of Support)

A line L is a *line of support* of a convex polygon P if

$$L \cap P = \text{a vertex/an edge of } P.$$

$$L \cap P \neq \emptyset \text{ and } P \text{ lies entirely on one side of } L.$$

Definition (Antipodal)

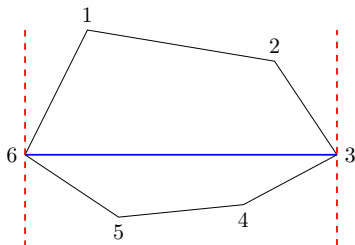
An *antipodal* is a pair of points that admits parallel supporting lines.

We have enumerated *all* antipodals.

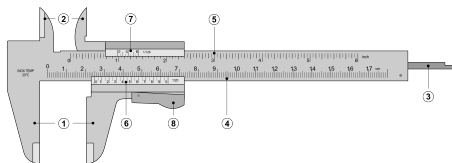
Theorem

If AB is a diameter of a convex polygon P , then AB is an antipodal.

Proof.



Rotating Caliper

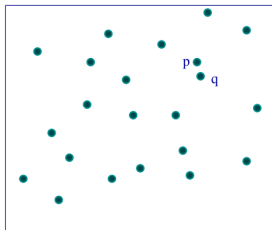


“Computational Geometry”
Ph.D Thesis, Michael Shamos, 1978



“Solving Geometric Problems with
the Rotating Calipers”, 1983

Finding the Closest Pair of Points



A Classic and Beautiful Divide-Conquer-Combine Algorithm:



Section 33.4, CLRS



- (a) Given an array $A[0 \cdots n - 1]$, to determine whether there is a value that *occurs more than $\lfloor n/k \rfloor$ times* in $\Theta(n \lg k)$ time and $\Theta(k)$ extra space.
- (b) Prove that the *lower bound* of this problem is $\Theta(n \lg k)$.

TIP#1

Take $k = 2$.

$\Theta(n)$ time & $\Theta(1)$ space

Thank
You!



Office 302

Mailbox: H016

hfwei@nju.edu.cn