2-2 The Efficiency of Algorithms

魏恒峰

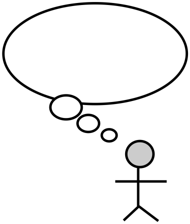
hfwei@nju.edu.cn

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 ($\Theta(n)$)

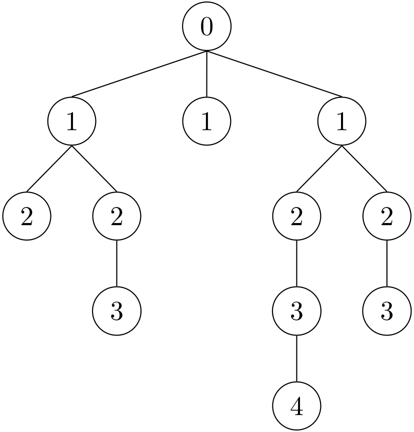


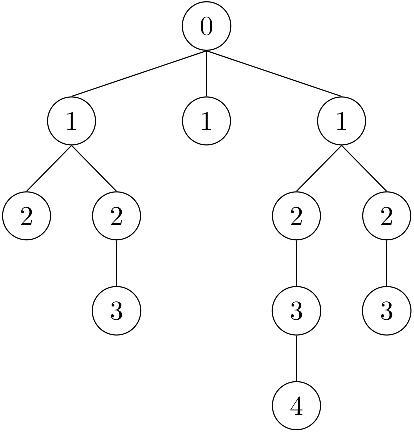
(1) Diameter of Convex Polygon: $\Theta(n)$

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

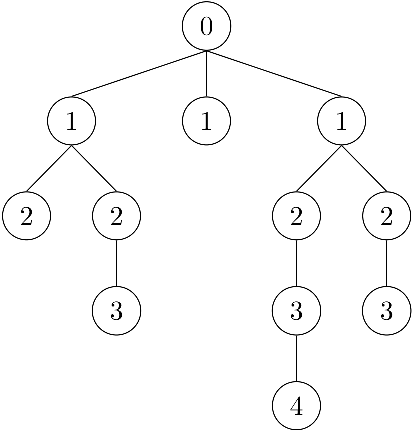
(3) Traversal over Trees: DFS/BFS ($\Theta(n)$)

I have thought that $\cdots$

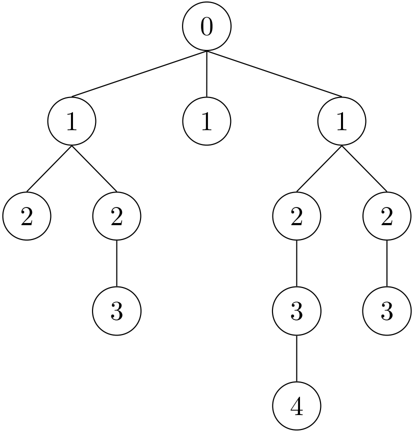

0
1
1
1
2
2
2
2
3
3
3
4

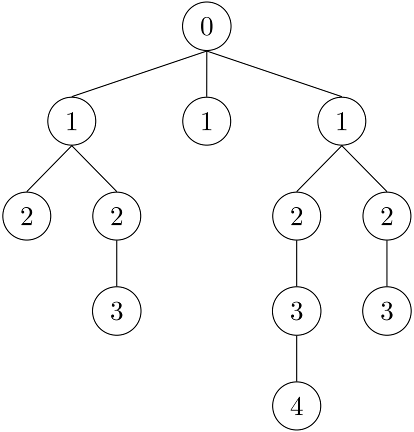


$$\text{sum-of-depths}(r) = \begin{cases} \\ \displaystyle\sum_{v:\text{child of } r} \text{sum-of-depths}(v) + \text{depth of } r, \end{cases}$$



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


0
1
1
1
2
2
2
2
3
3
3
4



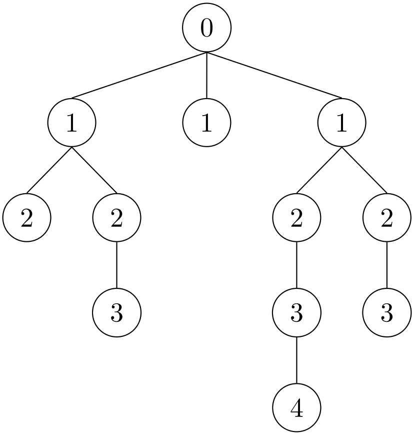
$$\text{sum-of-depths}(r, \color{red}{d}) = \begin{cases} d, & r \text{ is a leaf} \\ \displaystyle\sum_{v:\text{child of } r} \text{sum-of-depths}(v, \color{red}{d+1}) + d, & \text{o.w.} \end{cases}$$

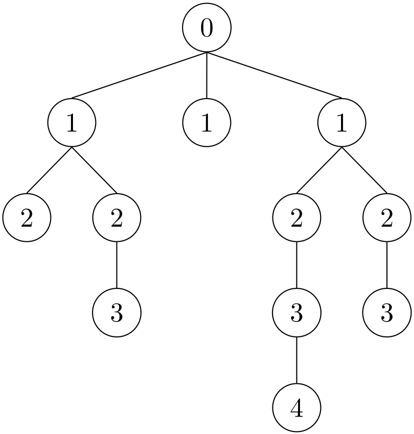
Algorithm 1 Calculate the sum of depths of all nodes of a tree T.

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

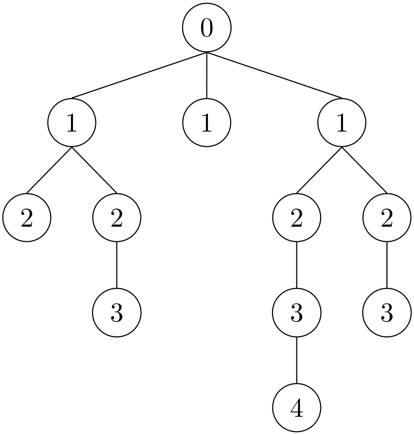
procedure $\textsc{Sum-of-Depths}(r, depth)$ ▷ r: root of a tree
    if T is a leaf **then**
        return $depth$

    for all child vertex v of r **do**
        $depth \leftarrow depth + \textsc{Sum-of-Depths}(v, depth + 1)$
    return $depth$


0
1
1
1
2
2
2
2
3
3
3
4



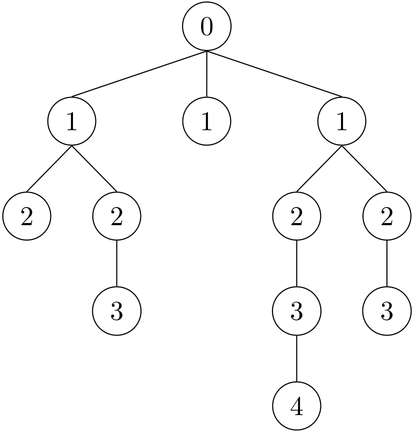
$$\text{nodes-at-depth}(r, k) = \begin{cases} \\ \displaystyle\sum_{v:\text{child of } r} \text{nodes-at-depth}(v, k-1), \end{cases}$$

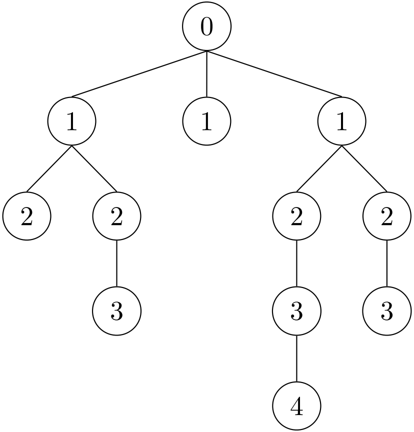


$$\text{nodes-at-depth}(r, k) = \begin{cases} 1, & k = 0 \\ 0, & k > 0 \land r \text{ is a leaf} \\ \displaystyle\sum_{v:\text{child of } r} \text{nodes-at-depth}(v, k-1), & \text{o.w.} \end{cases}$$

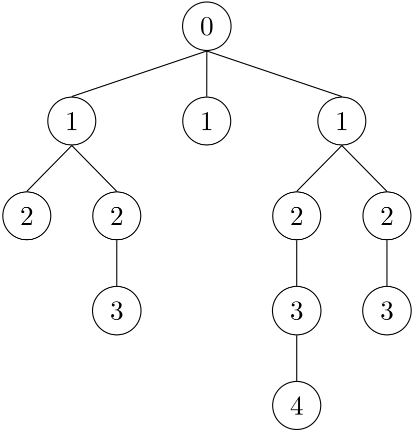
Algorithm 2 Count the number of nodes in T at depth K.

```
procedure NODES-AT-DEPTH()
    return NODES-AT-DEPTH(T, K)

procedure NODES-AT-DEPTH(r, k)                    ▷ r: root of a tree
    if k = 0 then
        return 1

    if r is a leaf then
        return 0

    num ← 0
    for all child vertex v of r do
        num ← num + NODES-AT-DEPTH(v, k − 1)
    return num
```


0
1
1
1
2
2
2
2
3
3
3
4



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



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

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 $\textsc{Leaf-at-Depth}(r, parity)$ $\triangleright$ 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 $result$


Thank
You!