# 2-2 The Efficiency of Algorithms[1]

Hengfeng Wei (hfwei@nju.edu.cn)

16 March 2018 ∼ March 23, 2018

[1]

The key points include:

- Correctness of the rotating caliper algorithm for the convex polygon diameter problem.
- Lower bound proof of comparison-based sorting.
- Depth-first and breadth-first traversal of trees.

## The Convex Polygon Diameter Problem

### Solution

**Theorem 1** *For a convex polygon, a pair of vertices determine the diameter.*

**Proof**

**Definition 1 (Line of Support)**

**Definition 2 (Antipodal)**

**Fact 1** *Not all vertex pairs are antipodal.*

**Proof**

**Theorem 2 (Yaglom)** *The diameter of a convex polygon is the greatest distance between parallel lines of support.*

**Proof**

## Lower Bound for Comparison-based Sorting (UD Problem 6.13)

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

**Algorithm 1** 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)
```

### *Understanding the Problem*

### *Solution*

### *Comments*

## *BFS on Tree (UD Problem 4.3)*

Write algorithms that solve the following problems by performing *breadth-first traversals* of the given trees. You may assume the availability of a queue $Q$. The operations on $Q$ include adding an item to the rear, retrieving and removing an item from the front, and testing $Q$ for emptiness.

(a) Given a tree $T$ whose nodes contain integers, print a list consisting of the sum of contents of nodes at depth 0, the sum of contents of nodes at depth 1, etc.

(b) Given a tree $T$, find the depth $K$ with the maximal number of nodes in $T$. If there are several such $K$s, return their maximum.

### *Solution*

## *Tree Traversal (UD Problem 4.2)*

(a) Write an algorithm which, given a tree $T$, calculates the sum of the depths of all the nodes of $T$.

(b) Write an algorithm which, given a tree $T$ and a positive integer $K$, calculates the number of nodes in $T$ at depth $K$.

(c) Write an algorithm which, given a tree $T$, checks whether it has any leaf at an even depth.

**Algorithm 2** 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)
```

**Algorithm 3** Calculate the sum of depths of all nodes of a tree $T$.

```
procedure Sum-of-Depth()
    return Sum-of-Depth(T, 0)

procedure Sum-of-Depth(r, depth)               ▷ r: root of a tree
    if T is a leaf then
        return depth
    else
        for all child vertex v of r do
            depth ← depth + Sum-of-Depth(v, depth + 1)
        return depth
```

*Solution*

*References*