码农学习小组

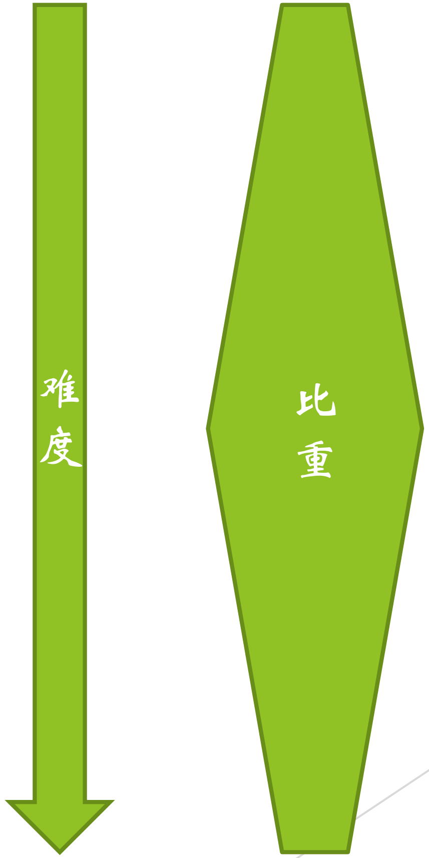
Week 01

码农面试一般过程

- 投简历 / 内推
- Recruiter寒暄
- 电话面试 1-3 轮
- Onsite面试 一整天

码农面试问题分类

- 基础知识
  - CS基础知识 - Max Integer, Max Unsigned Integer, K, M, G, T.
  - 操作系统知识 – 虚拟内存，进程与线程
  - 网络基础 – TCP vs UDP, HTTP
  - Design Pattern
- 算法 Algorithm
- 面向对象的设计 OOD
- 系统设计 System Design



难度

比重

目标

- 立足九章
- 刷会Leetcode Easy to Medium
- 分享经验、树立信心

时间复杂度和空间复杂度
BUD优化 – Bottleneck Unnecessary Duplicated

- Two Sum
  - [2, 0, 8, 3, 4, 9, 5, 1] target=5
  - [2, 3], [0, 5], [4, 1]
- Brute force
- Sort
- HashMap / HashSet

	Time	Extra Space
Brute force	O(n^2)	O(1)
Sort	O(nlgn)	O(1)
HashSet	O(n)	O(n)

九章算法第二节
二分查找 - Binary Search

- 什么是Binary Search?
- 前提条件?
- 时间复杂度?
  - The master theorem
  - $T(n) = aT(n/b) + f(n)$
  - $T(n) = T(n/2) + O(1) \rightarrow O(\lg n)$
  - $T(n) = T(n/2) + O(n) \rightarrow O(n\lg n)$

二分查找模板 - 九章vs传统

```
int start = 0, end = nums.length - 1;
while (start + 1 < end) {
    int mid = start + (end - start) / 2;
    if (nums[mid] < target) {
        start = mid;
    } else {
        end = mid;
    }
}
if (nums[start] == target) {
    return start;
}
if (nums[end] == target) {
    return end;
}
return -1;
```

```
int start = 0, end = nums.length - 1;
while (start < end) {
    int mid = start + (end - start) / 2;
    if (nums[mid] == target) {
        return mid;
    }
    else if (nums[mid] < target) {
        start = mid + 1;
    } else {
        end = mid - 1;
    }
}
return -1;
```

1，1，1，2，2，2，3，3

Example – Search for first or last target

```
int start = 0, end = nums.length - 1;
while (start + 1 < end) {
    int mid = start + (end - start) / 2;
    if (nums[mid] < target) {
        start = mid;
    } else {
        end = mid;
    }
}
if (nums[start] == target) {
    return start;
}
if (nums[end] == target) {
    return end;
}
return -1;
```

```
int start = 0, end = nums.length - 1;
while (start + 1 < end) {
    int mid = start + (end - start) / 2;
    if (nums[mid] <= target) {
        start = mid;
    } else {
        end = mid;
    }
}
if (nums[start] == target) {
    return start;
}
if (nums[end] == target) {
    return end;
}
return -1;
```

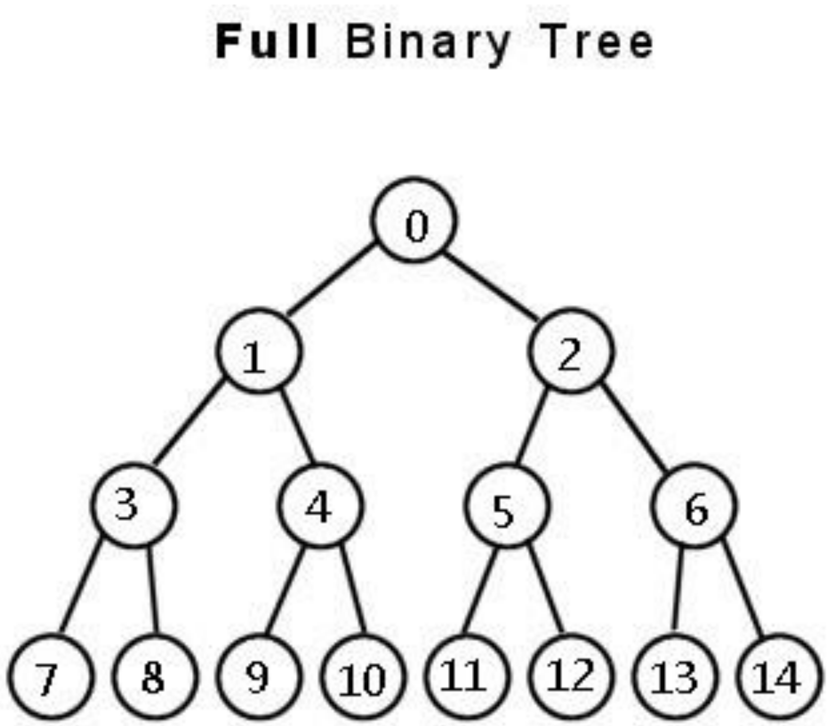
九章算法第三节 - Binary Tree & Divide and Conquer

- 什么是Binary Tree?
  - Balanced Binary Tree
  - Complete Binary Tree
  - Full Binary Tree

  - Binary Search Tree

- 什么是Divide and Conquer
  - 举例?
  - 时间复杂度?
  - 第二大类解题思路

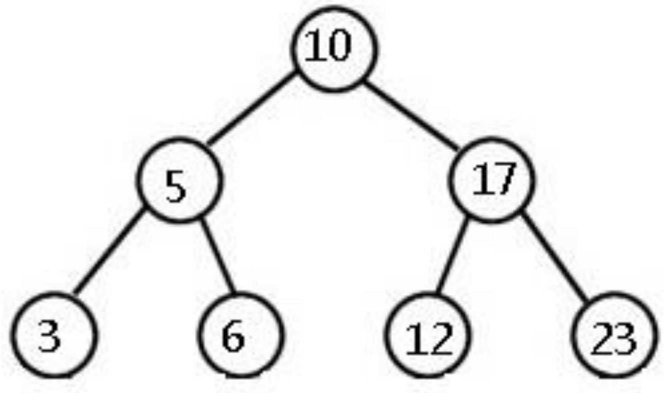
二叉树的重要性质



- 高度为h的Full Binary Tree，节点数n=?
  - $2^0 + 2^1 + \ldots + 2^{(h-1)} = 2^h - 1 = n$
- 节点数为n的Full Binary Tree， 高度是?
  - Lg(n)
- 任意一层的节点数与其上所有节点数的关系?
  - 多1个
- 父节点与子节点的序号关系?
  - left_son = p * 2 + 1; right_son = p * 2 + 2;
  - p = (son – 1) / 2;

二叉树的遍历

- 深度优先DFS，广度优先BFS
- 顺序In-order，先序Pre-order，后序Post-order
- 层序Level-order



Binary Tree DFS recursive traverse 通用模板

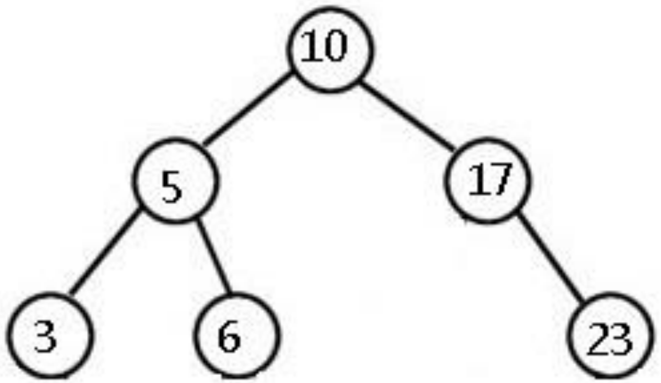
In-order

```
[?] traverse (TreeNode root) {
    if (root == null) // base case
        return something;
    [?] l = traverse(root.left);
    do something with root;
    [?] r = traverse(root.right);
    return something base on l and r;
}
```

Pre-order

```
[?] traverse (TreeNode root) {
    if (root == null) // base case
        return something;
    do something with root;
    [?] l = traverse(root.left);
    [?] r = traverse(root.right);
    return something base on l and r;
}
```

Binary Tree BFS iterative traverse 通用模板



Level-order

```
List<Integer> traverse (TreeNode root) {
    Queue<Integer> queue = new LinkedList<>();
    List<Integer> result = new ArrayList<>();
    queue.add(root);
    while (!queue.isEmpty()) {
        TreeNode node = queue.poll();
        result.add(node.val);
        if (node.left != null)
            queue.add(node.left);
        if (node.right != null)
            queue.add(node.right);
    }
    return result;
}
```

Binary Search Tree

- 查找一个元素
  - 用哪个模板?
  - 如何修改这个模板?
  - 时间复杂度?
- 插入一个元素
- 删除一个元素

Pre-order

```
boolean traverse (TreeNode root, int target) {
    if (root == null) // base case
        return false;
    int value = root.val;
    if (value == target)
        return true;
    else if (value > target)
        return traverse(root.left);
    else
        return traverse(root.right);
}
```

Questions?

Homework!

- https://leetcode.com/problems/first-bad-version/
- https://leetcode.com/problems/search-for-a-range/
- https://leetcode.com/problems/find-peak-element/
- https://leetcode.com/problems/search-insert-position/
- https://leetcode.com/problems/search-in-rotated-sorted-array/
- https://leetcode.com/problems/search-in-rotated-sorted-array-ii/

- https://leetcode.com/problems/balanced-binary-tree/
- https://leetcode.com/problems/same-tree/
- https://leetcode.com/problems/binary-tree-paths/
- https://leetcode.com/problems/lowest-common-ancestor-of-a-binary-search-tree/
- https://leetcode.com/problems/count-complete-tree-nodes/
- https://leetcode.com/problems/validate-binary-search-tree/