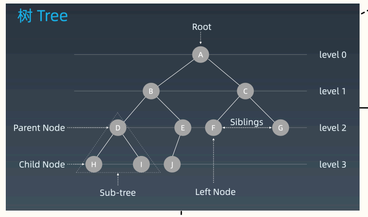
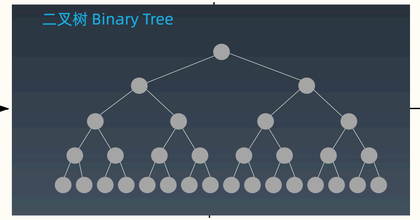
树、二叉树



树和图最关键的区别就是看有没有环

Linked List 是特殊化的 Tree

Tree 是特殊化的 Graph



儿子节点只有两个的树

🤪让树节点变得有序，这样查找和维护起来就更加有效

二叉树的遍历

前序：根-左-右
中序：左-根-右
后序：左-右-根

二叉搜索树

🤪中序遍历为升序

左子树的**所有节点的值**均小于根节点的值，右子树**所有节点的值**均大于根节点的值。依次类推，左右子树也都为二叉查找树。

🤪查询 、插入、删除 log(n)

相当于链表升了一个维度

实现

查找

类似二分查找，比如查找 19，先和根节点比较，比根节点小，往左边走，然后与 14 节点比较，比 14 大往右边走

插入

就是先查找该数据，无法继续向下查找的地方就是该节点需要插入的地方

删除

删除的是根节点 35，就从 35 的子节点中找到刚好大于 35 的节点 42，把 42 替换到 35 的位置

代码节点

```
struct TreeNode {
    int val;
    TreeNode *left;
    TreeNode *right;
    TreeNode(x):val(x),left(NULL),right(NULL){};
}
```

查找 O(1)
插入 O(logn) 或 O(1)
删除 O(logn)

常见的堆有二叉堆和斐波那契堆，二叉堆是时空复杂度刚合格的堆

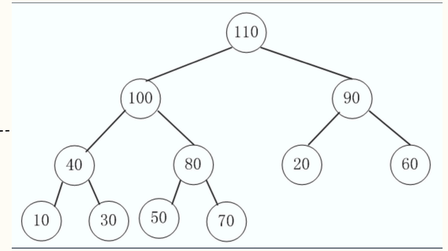
堆 heap

堆是可以迅速找到一堆数据中最大或最小值的数据结构

将根节点为最大值的堆叫做大顶堆或大根堆，相反则是小顶堆或小根堆

二叉堆

1.是一个完全二叉树
2.任意节点的值都大于其子节点



实现

一般由数组实现

假设根节点索引为 0；则父节点和子节点的位置关系如下：
索引为 i 的左还是是 2i+1,右孩子为 2i+2;
索引为 i 的父节点的索引为 floor((i-1)/2)

[110, 100, 90, 40, 80, 20, 60, 10, 30, 50, 70]

插入

1.先将节点放到数组最后
2.然后在和其父节点依次比较，如果大于其父节点则交换位置；

删除堆顶

1.将堆尾元素和堆顶交换，size -1，即删除了堆顶
2.然后在向下调整堆，依次和儿子中较大的一个交换位置

哈希表

1.是根据关键码值而直接进行访问的数据结构
2.它通过关键码值映射到表中某一个位置来访问记录，加快查找的速度

这个映射函数叫散列函数，存放记录的数组叫哈希表或散列表

不同的存储数据经过哈希函数算出来映射到同一个位置，叫哈希碰撞

可以在该位置上拉出一个链表存储多个数据