1.绪论

下界

代数判定树

两个吃罢饭，又走了四五十里，却来到一市镇上，地名唤做瑞龙镇，却是个三岔路口。宋江借问那里人道："小人们欲投二龙山、清风镇上，不知从那条路去？"

邓俊辉

deng@tsinghua.edu.cn

难度与下界

❖ 由前述实例可见，同一问题的不同算法，复杂度可能相差悬殊

❖ 在可解的前提下，可否谈论问题的 $\boxed{\text{难度}}$ ？如何比较不同问题的难度？

❖ 问题P若存在算法，则所有算法中 $\boxed{\text{最低}}$ 的复杂度称为P的难度

❖ 为什么要确定问题的难度？给定问题P，如何确定其难度？

❖ 两个方面着手：设计复杂度更低的算法 + 证明更高的问题难度下界

❖ 一旦算法的复杂度达到难度下界，则说明

就大$\mathcal{O}$记号的意义而言，算法已经最优

❖ 例如，排序问题下界为$\Omega(n\log n)$，而且是紧的...

排序

- 任给n个元素$\{ R_1, \ldots, R_n \}$，对应关键码$\{ K_1, \ldots, K_n \}$

  需按某种$\boxed{\text{次序}}$排列 //≤：偏序/全序

- 亦即，找出 $\langle 1, \ldots, n \rangle$ 的一个$\boxed{\text{排列}}$

  $\langle i_1, \ldots, i_n \rangle$，使得

  $K_{i1} \le \ldots \le K_{in}$

- 例如： $\{ 3, 1, 4, 1, 5, 9, 2, 6 \}$

  $\rightarrow$ $\{ 1, 1, 2, 3, 4, 5, 6, 9 \}$

- 注意：此处的关键码集是复集$\boxed{\text{multiset}}$ //可能存在重复关键码

- 应用：25~50%的计算都可归于排序

算法分类

❖ 直接算法	直接移动元素本身 //元素结构简单时适宜采用
❖ 间接算法	下标 + 关键码 + 指针 //元素结构复杂时适宜采用
❖ 内部 / 外部	internal / external
❖ 脱机 / 在线	offline / online
❖ 串行 / 并行	sequential / parallel
❖ 确定性 / 随机	deterministic / randomized
❖ 基于比较式 / 散列式	comparison-based / hash-based

时空性能、稳定性

❖ 多种角度估算的时间、空间复杂度

最好 / best-case　　　最坏 / worst-case

平均 / average-case　　分摊 / amortized

❖ 其中，对最坏情况的估计最保守、最稳妥

因此，首先应考虑 最坏情况最优 的算法　　　//worst-case optimal

❖ 排序所需的时间，主要取决于

关键码比较的次数 / # {key comparison}

元素交换的次数 / # {data swap}

❖ 就地 in-place ：除输入数据本身外，只需$\mathcal{O}(1)$附加空间

❖ 稳定 stability ：关键码雷同的元素，在排序后相对位置保持

最坏情况最优 + 基于比较

❖ 排序算法，最快能够有多快？

语境1：就最坏情况最优而言

语境2：就$\boxed{\text{某一大类}}$主流算法而言...

❖ 基于比较的算法（comparison-based algorithm）

算法执行的进程，取决于一系列的数值（这里即关键码）比对结果

比如，max() 和 bubbleSort()

❖ 任何CBA在最坏情况下，都需$\Omega(n\log n)$时间才能完成排序

判定树

❖ 每个CBA算法，都对应于一棵判定树

- 从根节点通往任一叶子的路径，都对应于算法的某次运行过程

- 每一可能的输出，都对应于

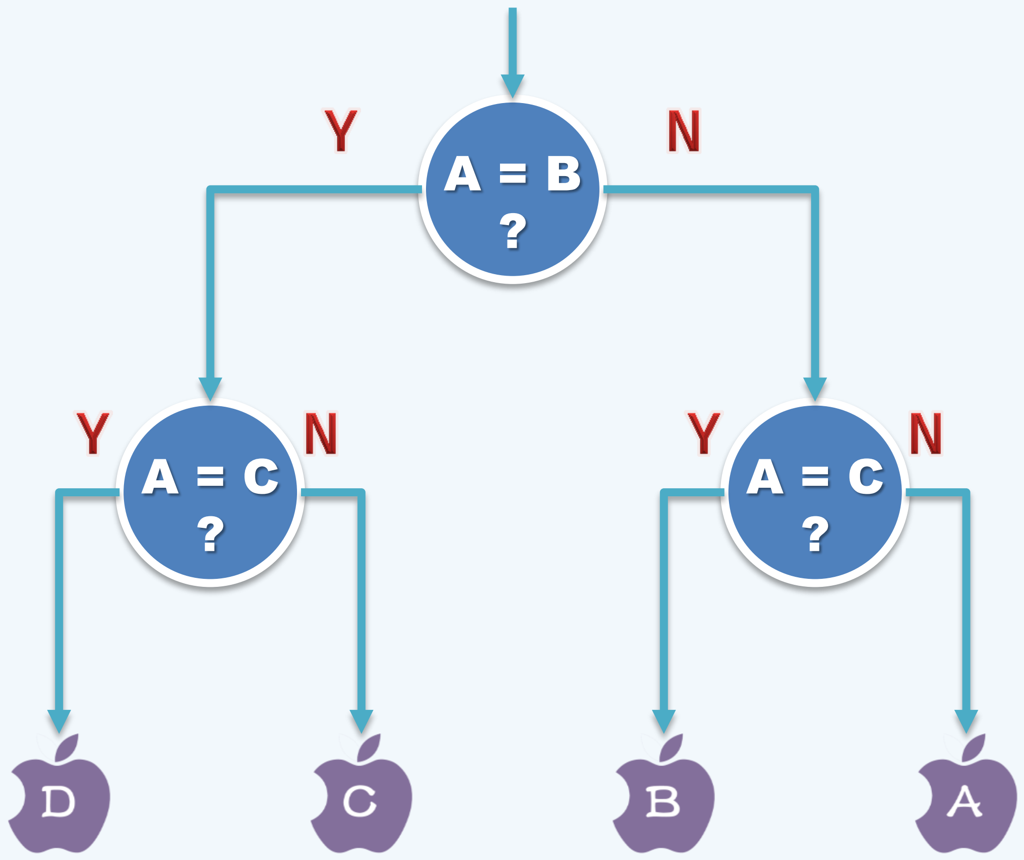
  $\boxed{\text{至少}}$一匹叶子（一条通往叶子的路径）

❖ 实例： 经过2/4次称量

必可从4/16只苹果中

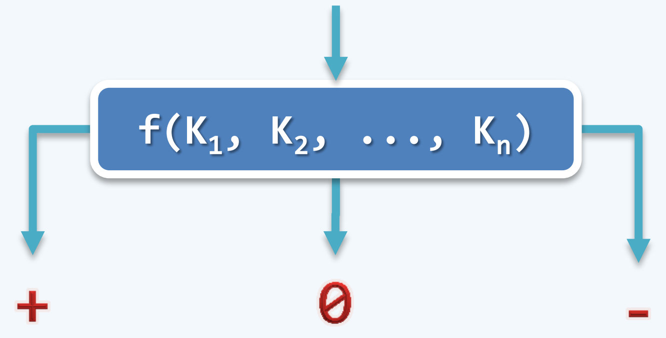
找出唯一的重量$\boxed{\text{不同}}$者

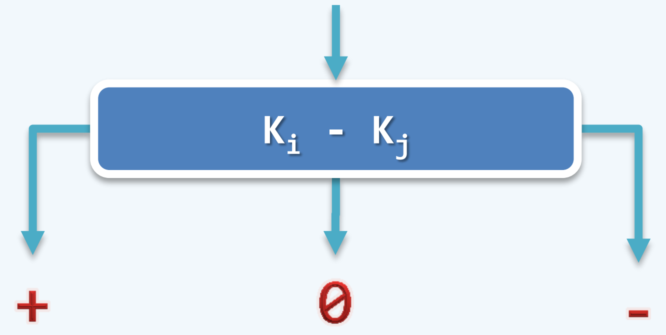
❖ 问题： 称量次数可否更少？



代数判定树

- Algebraic Decision Tree
  - 针对“比较-判定”式算法的计算模型
  - 给定输入的规模，将所有可能的输入所对应的一系列判断表示出来
- 代数判定：
  - 使用某一常次数代数多项式将任意一组关键码做为变量，对多项式求值
  - 根据结果的符号，确定算法推进方向
- Comparison Tree：最简单的ADT，二元一次多项式，形如：$K_i - K_j$





下界：Ω(nlogn)

❖ 比较树是三叉树（ternary tree）

每个内节点至多三个分支 //+、0、-

从根节点到叶子的每一路径，对应于算法的某次运行过程

每匹叶子对应于一个输出 //在此，即排序后的序列

树高 = 最坏情况下所需的比较次数 //最坏情况最优

树高的下界 = 所有CBA的时间复杂度下界

❖ 对n个元素进行排序的任何一棵ADT，$\boxed{\text{高度}}$至少为Ω(nlogn)

#叶子 ≥ #可能的输出 = n个元素可能的排列 = n!

树高 $\geq log_3 n! = log_3 e \cdot ln(n!)$

$= log_3 e \cdot [nlnn - n + O(lnn)] = \Omega(nlogn)$ //Stirling approximation

❖ 上述结论，可进一步推广至理想平均情况、随机情况（概率≥25%）...