sorting lower bound

许多sorting algorithms 都是基于比较的

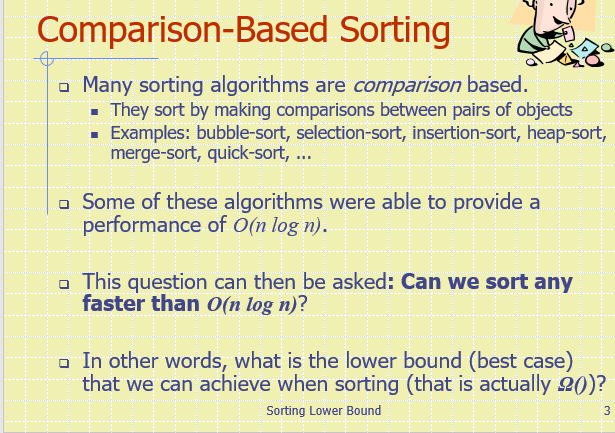
他们通过比较object来进行sort

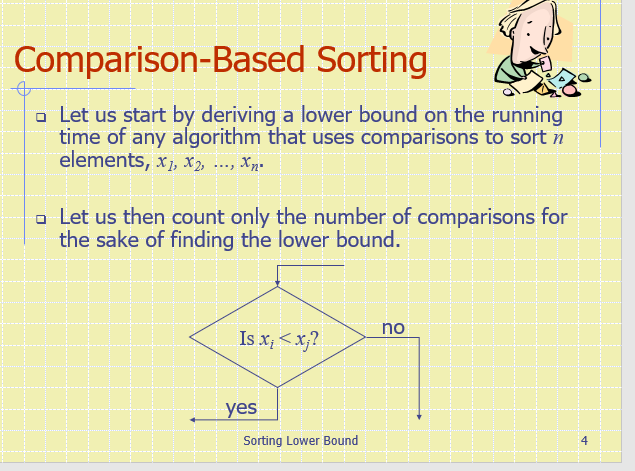
比如：bubble,selection,insertion,heap,quick

这些算法最快就是onlogn

那么我们能超越Onlogn吗

换句话说 我们在sorting的时候能达到怎样的LOWER BOUND（BEST case）呢 （实际是是Ω()）





基于比较的sorting

为了找到Lower bound，我们只需要数有几次comjparison比较

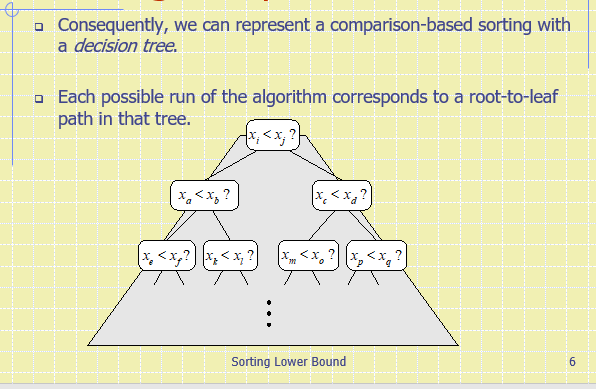
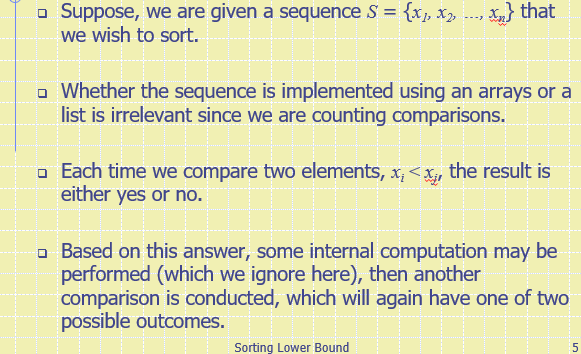
假设我们有一个sequence S，并且我们希望sort他

这个sequence是array或List与我们计算comparison数量无关

每当我们compare两个 element的时候，xi<xj这个比较的答案永远是yes or no

基于xi<xj这个问题的答案，可以进行一些内部计算，(我们在这里忽略了) //内部计算就是XI<XJ后执行的语句

然后进行另一个比较，这将再次得到两种可能结果之一。



最终，我们可以吧基于比较的sorting表示成一个decision tree

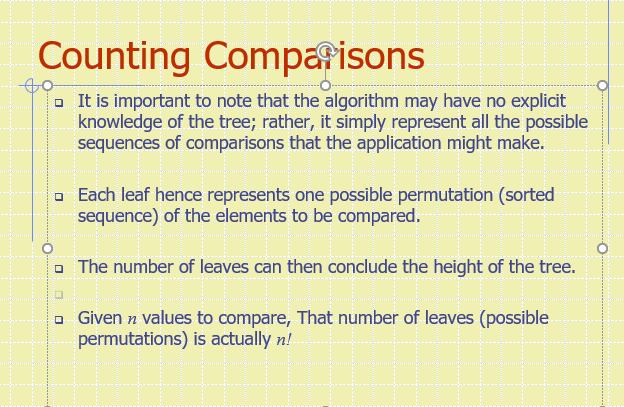
每一个算法的可能路径 代表这一条root to leaf的path

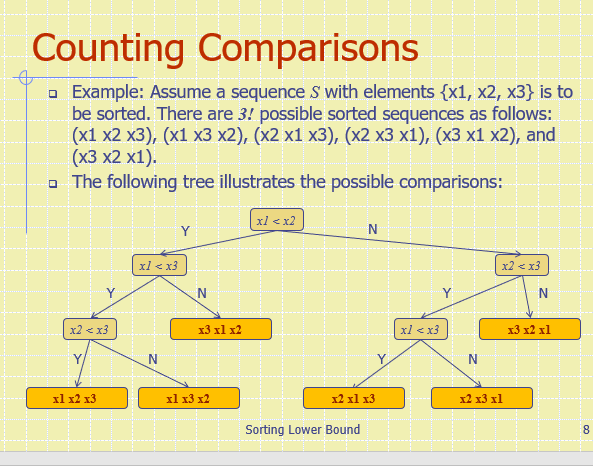
需要注意的是，algorithm可能没有树的显式知识;相反，它只是表示应用程序可能进行的所有可能的comparison sequence。

每一个leaf代表着一个可能的排列（sorted sequence） ，这个排列是由需要被比较的element们组成的

通过leaf的数量我们能得到tree的高度

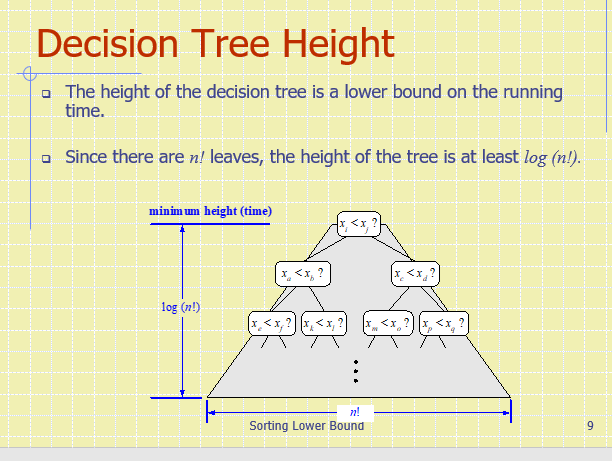
如果有n个数要比较，那么Leaf的数量就是n!





假设有一组element为x1 x2 x3的sequence S, 我们想要sort他

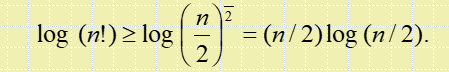
那么就有3!组可能的sorted sequence 也就是6



decision tree的高度就是running time的lower bound

因为有n! leaves，高度至少是Log n!,.

因此这类 comnparison-based sortring 算法至少使用log n! time



二log n!>=nlogn

所以任何 comparison-based sorting algorithm至少要用 nlogn ，不可能比这还好

