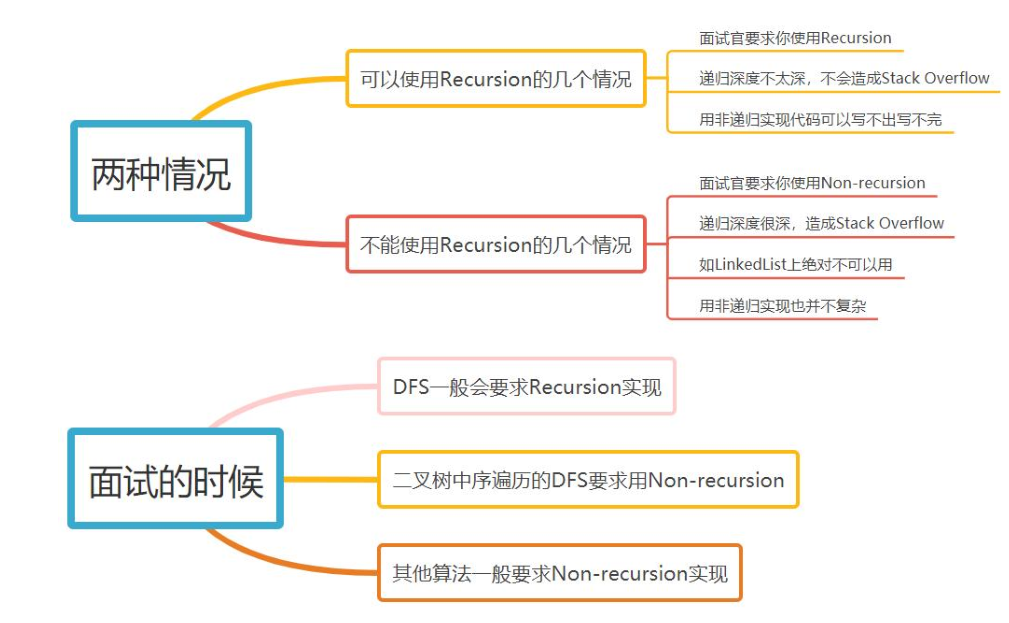
根据时间复杂度来倒推算法是面试中常见的策略，如：比更优的时间复杂度只能是二分法的。

二分法是一种将数组由大化小的思路，其它的例如递归，动态规划，分治也属于由大化小的思路。

二分法也可以用于非排序的数组。

时间复杂度能达到的树，只有平衡二叉树(AVL树，红黑树，红黑树是平衡二叉查找树)，其它的如二叉树或查找二叉树都不是，例如一个树，所有结点只有左孩子，没有右孩子，对它的增加/删除/访问都不可能是。

二分法的递归和非递归，尽可能的使用非递归。



程序中的存储空间主要是堆(heap)和栈(stack)，堆的空间一般和内存大小有关，它是共享的，某个程序释放了堆内存后，其它程序就可以使用，而栈是程序独享的，一般比较小，只有8M，发生函数调用时，被调函数将调用函数的运行环境和返回地址保存在stack中，使得被调函数运行完成后，恢复调用函数的状态。

递归产生stackoverlflow，如果递归函数中没有特别大的内存空间，一般需要大约10000层左右。在python中，递归调用时，编译器就会计数，超过10000后就会停止。

二分法在无穷大的数组中查找target，该问题的难点在于无法知道右边界。<https://www.lintcode.com/problem/search-in-a-big-sorted-array/>

倍增的思想：C++中的vector，Java中的ArrayList，当往里面添加元素时，如果vector内部的存储数组已满，就会新建一个2倍于之前长度的新数组，然后将旧数组中的元素挪移到新数组中，并销毁旧数组。

在浏览器在进行网络连接时，如果连接不上，就会进行重试，重试的时间间隔也是倍增，第1次100ms以后，第2次200ms以后，依次类推。

没有三分法，四分法的原因是时间复杂度不会提高，二分法判断是否找到target，每次只需要比较1次，三分，四分会比较多次，而程序变得更复杂。

lintcode 460, 585中的二分法，左右两根指针，在逐渐逼近要查找的target的位置，while循环退出后，left, right其中之一就是要找的位置。

lintcode 62，159，二分法的原理是：每次都数组进行二分后，得到mid，无论移动left到mid，还是移动right到mid，要保证移动后查找的target所在的位置一定在left和right之间，这样才能慢慢逼近target。

二分法的第三重境界：在未排序的数组上的二分。

lintcode 390，数组未排序，通过对n行m列的矩阵同时进行行和列的二分，其时间复杂度为。分析过程如下：

由于是同时对矩阵的行和列进行二分，并且要搜索中间行和中间列的最大值，二分后要搜索的矩阵就变成了n/2行和m/2列，所以：



第二次二分后，得到：



递推可得：



其中。由等比数列的性质可得：



所以，其时间复杂度为：