[**快速排序里的学问：从猜数字开始**](http://www.nowamagic.net/librarys/veda/detail/2387)

我们先来玩一个[猜数字](http://www.nowamagic.net/librarys/veda/tag/%E7%8C%9C%E6%95%B0%E5%AD%97)游戏：

我心里默念一个1~64之间的数，你来猜（你只能问答案是“是”或“否”的问题）。为了保证不论在什么情况下都能以尽量少的次数猜中，你应该采取什么策略呢？

很显然，二分。先是猜是不是位于1~32之间，排除掉一半可能性，然后对区间继续二分。这种策略能够保证无论数字怎么跟你捉迷藏，都能在log2n次以内猜中。用算法的术语来说就是它的下界是最好的。（算法的下界 ：一个问题的下界是用来解决该问题的任意算法所需要的最小时间复杂度。 ）

我们再来回顾一下这个游戏所蕴含的本质：为什么这种策略具有最优下界？

答案也很简单，这个策略是平衡的。反之如果策略不是平衡的，比如问是不是在1~10之间，那么一旦发现不是在1~10之间的话就会剩下比N/2更多的可能性需要去考察了。

这种策略的本质可以概括成“让未知世界无机可乘”。它是没有“弱点的”，答案的任何一个分支都是等概率的。反之，一旦某个分支蕴含的可能性更多，当情况落到那个分支上的时候你就郁闷了。比如猜数字游戏最糟糕的策略就是一个一个的猜：是1吗？是2吗？… 因为这种猜法最差的情况下需要64次才能猜对，下界非常糟糕。二分搜索为什么好，就是因为它每次都将可能性排除一半并且无论如何都能排除一半（它是最糟情况下表现最好的）。

**猜数字的时间复杂度**

猜数字的时间复杂度其实就是[二分](http://www.nowamagic.net/librarys/veda/tag/%E4%BA%8C%E5%88%86)查找的复杂度。

二分查找的基本思想是将n个元素分成大致相等的两部分，然后用a[n/2]与x做比较，如果x=a[n/2]，则找到x，算法中止；如果x < a[n/2]，则只要在数组a的左半部分继续搜索x，如果x>a[n/2]，则只要在数组a的右半部搜索x。

所以二分查找的时间复杂度无非就是while循环的次数。

我们可以跟着循环去想。总共有n个元素，跟下去就是n, n/2, n/4, ...., n/2k，其中k就是循环的次数。

每比较一次，查找范围被缩短为原来1/2，当范围长度被缩短为1的时候，就完成查找了，然后要比较多少次，高中时的数学问题了。

另 n/2k = 1，即 2k = n，所以 k = log2n。

k 为次数，时间复杂度可以表示O()=O(logn)。