数学之美番外篇：快排为什么那样快

By
刘未鹏[1]

– June 13, 2008**Posted in:** 数学[2], 算法[3], 计算机科学[4]

目录

0. 前言

知道这个理论是在TopLanguage[5]上的一次讨论，先是g9转了David MacKay的一篇文章[6]，然后引发了牛人们的一场关于信息论的讨论[7]。Anyway，正如g9很久以前在Blog[8]里面所说[9]的：

> *有时无知是福。俺看到一点新鲜的科普也能觉得造化神奇。刚才读Gerald Jay Sussman（SICP[10]作者）的文章，Building Robust Systems – an essay[11]，竟然心如小鹿乱撞，手心湿润，仿佛第一次握住初恋情人温柔的手。*

而看到MacKay[12]的这篇文章我也有这种感觉——以前模糊的东西忽然有了深

刻的解释，一切顿时变得明白无比。原来看问题的角度或层面能够带来这么大的变化。再一次印证了越是深刻的原理往往越是简单和强大。所以说，土鳖也有土鳖的幸福:P

这篇文章相当于MacKay原文[13]的白话文版。MacKay在原文中用到了信息论的知识，后者在我看来并不是必须的，尽管计算的时候方便，但与本质无关。所以我用大白话解释了一通。

1. 猜数字

我们先来玩一个猜数字游戏：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log_2{n}次以内猜中。用算法的术语来说就是它的下界是最好的。

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。

徐宥[14]在讨论中提到，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）。

2. 称球

12个小球，其中有一个是坏球。有一架天平。需要你用最少的称次数来确定哪个小球是坏的并且它到底是轻还是重。

这个问题是一道流传已久的智力题。网络上也有很多讲解，还有泛化到N个球的情况下的严格证明。也有零星的一些地方提到从信息论的角度来看待最优解法。本来我一直认为这道题目除了试错之外没有其它高妙的思路了，只能

一个个方法试，并尽量从结果中寻找信息，然后看看哪种方案最少。

然而，实际上它的确有其它的思路，一个更本质的思路，而且根本用不着信息论这么拗口的知识。

我们先回顾一下猜数字游戏。为了保证任何情况下以最少次数猜中，我们的策略是每次都排除恰好一半的可能性。类比到称球问题上：坏球可能是12个球中的任意一个，这就是12种可能性；而其中每种可能性下坏球可能轻也可能重。于是“坏球是哪个球，是轻是重”这个问题的答案就有12×2=24种可能性。现在我们用天平来称球，就等同于对这24种可能性发问，由于天平的输出结果有三种“平衡、左倾、右倾”，这就相当于我们的问题有三个答案，即可以将所有的可能性切成三份，根据猜数字游戏的启发，我们应当尽量让这三个分支概率均等，即平均切分所有的可能性为三等份。如此一来的话一次称量就可以将答案的可能性缩减为原来的1/3，三次就能缩减为1/27。而总共才有24种可能性，所以理论上是完全可以3次称出来的。

如何称的指导原则有了，构造一个称的策略就不是什么太困难的事情了。首先不妨解释一下为什么最直观的称法不是最优的——6、6称：在6、6称的时候，天平平衡的可能性是0。刚才说了，最优策略应该使得天平三种状态的概率均等，这样才能三等分答案的所有可能性。

为了更清楚的看待这个问题，我们不妨假设有6个球，来考虑一下3、3称和2、2称的区别：

在未称之前，一共有12种可能性：1轻、1重、2轻、2重、...、6轻、6重。现在将1、2、3号放在左边，4、5、6放在右边3、3称了之后，不失一般性假设天平左倾，那么小球的可能性就变成了原来的一半（6种）：1重、2重、3重、4轻、5轻、6轻。即这种称法能排除一半可能性。

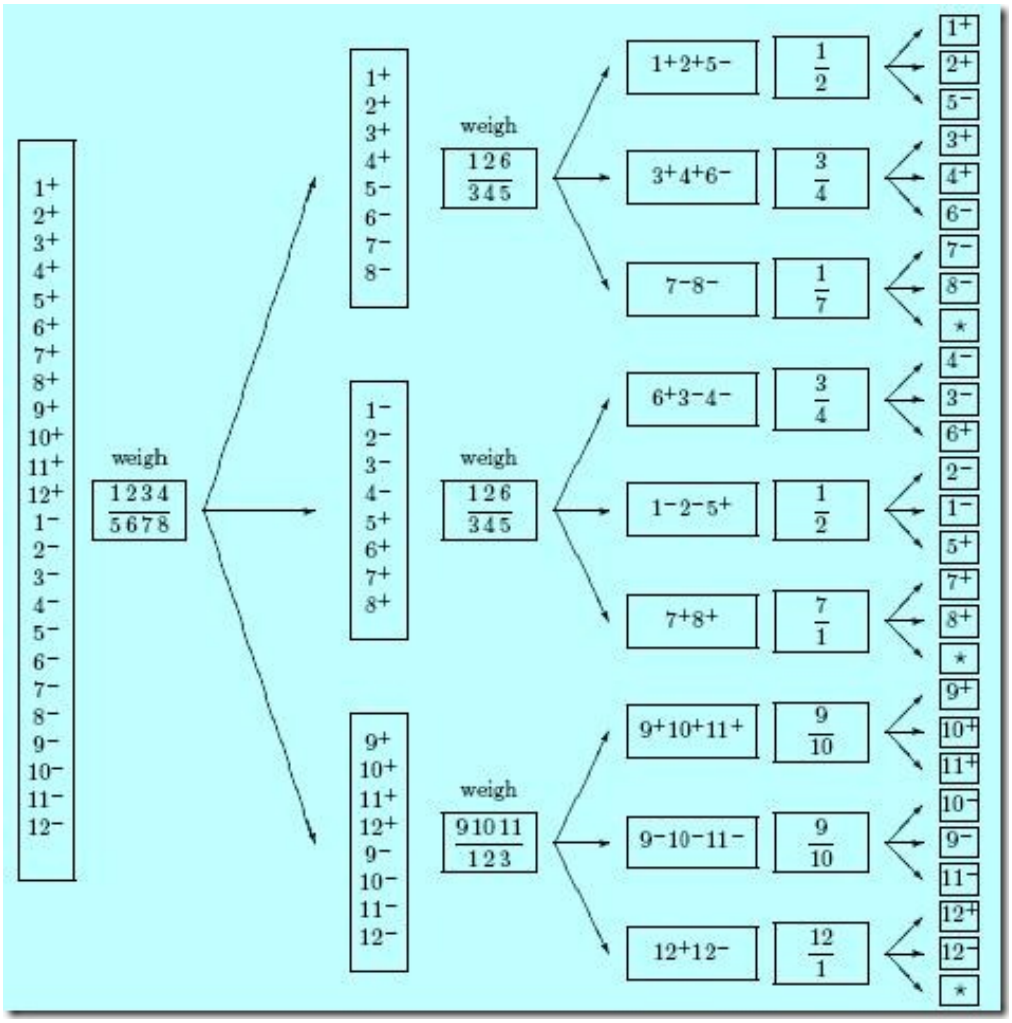
现在再来看2、2称法，即1、2放左边，3、4放右边，剩下的5、6不称，放一边。假设结果是天平平衡，那么可能性剩下——4种：5重、5轻、6重、6轻。假设天平左倾，可能性也剩下4种：1重、2重、3轻、4轻。右倾和左倾的情况类似。总之，这种称法，不管天平结果如何，情况都被我们缩小到了原来的三分之一！我们充分利用了“天平的结果状态可能有三种”这个条件来三等分所有可能性，而不是二等分。

说到这里，剩下的事情就实在很简单了：第二步称法，只要记着这样一个指导思想——你选择的称法必须使得当天平平衡的时候答案剩下的可能性和天平

左倾（右倾）的时候答案剩下的可能性一样多。实际上，这等同于你得选择一种称法，使得天平输出三种结果的概率是均等的，因为天平输出某个结果的概率就等同于所有支持这个结果（左倾、右倾、平衡）的答案可能性的和，并且答案的每个可能性都是等概率的。

MacKay在他的书《Information Theory: Inference and Learning Algorithms》（作者开放免费电子书[15]）里面4.1节专门讲了这个称球问题，还画了一张不错的图，我就照抄了：

[16]



[17]

图中“1+”是指“1号小球为重”这一可能性。一开始一共有24种可能性。4、4称了之后不管哪种情况（分支），剩下来的可能性总是4种。这是一个完美的三分。然后对每个分支构造第二次称法，这里你只要稍加演算就可以发现，分支1上的第二次称法，即“1、2、6对3、4、5”这种称法，天平输出三种结果的可能性是均等的（严格来说是几乎均等）。这就是为什么这个称法能够在最

坏的情况下也能表现最好的原因，没有哪个分支是它的弱点，它必然能将情况缩小到原来的1/3。

3. 排序

用前面的看问题视角，排序的本质可以这样来表述：一组未排序的N个数字，它们一共有N!种重排，其中只有一种排列是满足题意的（譬如从大到小排列）。换句话说，排序问题的可能性一共有N!种。任何基于比较的排序的基本操作单元都是“比较a和b”，这就相当于猜数字游戏里面的一个问句，显然这个问句的答案只能是“是”或“否”，一个只有两种输出的问题最多只能将可能性空间切成两半，根据上面的思路，最佳切法就是切成1/2和1/2。也就是说，我们希望在比较了a和b的大小关系之后，如果发现a<b的话剩下的排列可能性就变成N!/2，如果发现a>b也是剩下N!/2种可能性。由于假设每种排列的概率是均等的，所以这也就意味着支持a<b的排列一共有N!/2个，支持a>b的也是N!/2个，换言之，a<b的概率等于a>b的概率。

我们希望每次在比较a和b的时候，a<b和a>b的概率是均等的，这样我们就能保证无论如何都能将可能性缩小为原来的一半了！最优下界。

一个直接的推论是，如果每次都像上面这样的完美比较，那么N个元素的N!种可能排列只需要log_2{N!}就排查玩了，而log_2{N!}近似于NlogN。这正是快排的复杂度。

3.1 为什么堆排比快排慢

回顾一下堆排的过程：

1. 建立最大堆（堆顶的元素大于其两个儿子，两个儿子又分别大于它们各自下属的两个儿子... 以此类推）

2. 将堆顶的元素和最后一个元素对调（相当于将堆顶元素（最大值）拿走，然后将堆底的那个元素补上它的空缺），然后让那最后一个元素从顶上往下滑到恰当的位置（重新使堆最大化）。

3. 重复第2步。

这里的关键问题就在于第2步，堆底的元素肯定很小，将它拿到堆顶和原本属于最大元素的两个子节点比较，它比它们大的可能性是微乎其微的。实际上

它肯定小于其中的一个儿子。而大于另一个儿子的可能性非常小。于是，这一次比较的结果就是概率不均等的，根据前面的分析，概率不均等的比较是不明智的，因为它并不能保证在糟糕情况下也能将问题的可能性削减到原本的1/2。可以想像一种极端情况，如果a肯定小于b，那么比较a和b就会什么信息也得不到——原本剩下多少可能性还是剩下多少可能性。

在堆排里面有大量这种近乎无效的比较，因为被拿到堆顶的那个元素几乎肯定是很小的，而靠近堆顶的元素又几乎肯定是很大的，将一个很小的数和一个很大的数比较，结果几乎肯定是“小于”的，这就意味着问题的可能性只被排除掉了很小一部分。

这就是为什么堆排比较慢（堆排虽然和快排一样复杂度都是O(NlogN)但堆排复杂度的常系数更大）。

MacKay也提供了一个修改版的堆排：每次不是将堆底的元素拿到上面去，而是直接比较堆顶（最大）元素的两个儿子，即选出次大的元素。由于这两个儿子之间的大小关系是很不确定的，两者都很大，说不好哪个更大哪个更小，所以这次比较的两个结果就是概率均等的了。具体参考这里[18]。

3.2 为什么快排其实也不是那么快

我们考虑快排的过程：随机选择一个元素做“轴元素”，将所有大于轴元素的移到左边，其余移到右边。根据这个过程，快排的第一次比较就是将一个元素和轴元素比较，这个时候显而易见的是，“大于”和“小于”的可能性各占一半。这是一次漂亮的比较。

然而，快排的第二次比较就不那么高明了：我们不妨令轴元素为pivot，第一次比较结果是a1<pivot，那么可以证明第二次比较a2也小于pivot的可能性是2/3！这容易证明：如果a2>pivot的话，那么a1，a2，pivot这三个元素之间的关系就完全确定了——a1<pivot<a2，剩下来的元素排列的可能性我们不妨记为P（不需要具体算出来）。而如果a2<pivot呢？那么a1和a2的关系就仍然是不确定的，也就是说，这个分支里面含有两种情况：a1<a2<pivot，以及a2<a1<pivot。对于其中任一种情况，剩下的元素排列的可能性都是P，于是这个分支里面剩下的排列可能性就是2P。所以当a2<pivot的时候，还剩下2/3的可能性需要排查。

再进一步，如果第二步比较果真发现a2<pivot的话，第三步比较就更不妙了，模仿上面的推理，a3<pivot的概率将会是3/4！

这就是快排也不那么快的原因，因为它也没有做到每次比较都能将剩下的可能性砍掉一半。

3.3 鸡排为什么又那么快呢?

传统的解释是：基排[19]不是基于比较的，所以不具有后者的局限性。话是没错，但其实还可以将它和基于比较的排序做一个类比。

基排的过程也许是源于我们理顺一副牌的过程：如果你有N（N<=13）张牌，乱序，如何理顺呢？我们假象桌上有十三个位置，然后我们将手里的牌一张一张放出去，如果是3，就放在位置3上，如果是J，就放在位置11上，放完了之后从位置1到位置13收集所有的牌（没有牌的位置上不收集任何牌）。

我们可以这样来理解基排高效的本质原因：假设前i张牌都已经放到了它们对应的位置上，第i+1张牌放出去的时候，实际上就相当于“一下子”就确立了它和前i张牌的大小关系，用O(1)的操作就将这张牌正确地插入到了前i张牌中的正确位置上，这个效果就相当于插入排序的第i轮原本需要比较O(i)次的，现在只需要O(1)了。

但是，为什么基排能够达到这个效果呢？上面只是解释了过程，解释了过程不代表解释了本质。

当i张牌放到位之后，放置第i+1张牌的时候有多少种可能性？大约i+1种，因为前i张牌将13个位置分割成了i+1个区间——第i+1张牌可以落在任意一个区间。所以放置第i+1张牌就好比是询问这样一个问题：“这张牌落在哪个区间呢？”而这个问题的答案有i+1种可能性？所以它就将剩下来的可能性均分成了i+1份（换句话说，砍掉了i/i+1的可能性！）。再看看基于比较的排序吧：由于每次比较只有两种结果，所以最多只能将剩下的可能性砍掉一半。

这就是为什么基排要快得多。而所有基于比较的排序都逃脱不了NlogN的宿命。

4. 信息论！信息论?

本来呢，MacKay写那篇文章是想用信息论来解释为什么堆排慢，以及为什么快排也慢的。MacKay在他的文章中的解释是，只有提出每种答案的概率都均等的问题，才能获得最大信息量。然而，仔细一想，其实这里信息论并不是因，而是果。这里不需要用信息论就完全能够解释，而且更明白。信息论只

是对这个解释的一个形式化。当然，信息论在其它地方还是有应用的。但这里其实用不着信息论这么重量级的东西（也许具体计算一些数据的时候是需要的），而是只需要一种看问题的本质视角：将排序问题看成和猜数字一样，是通过问问题来缩小/排除（narrow down）结果的可能性区间，这样一来，就会发现，“最好的问题”就是那些能够均分所有可能性的问题，因为那样的话不管问题的答案如何，都能排除掉k-1/k（k为问题的答案有多少种输出——猜数字里面是2，称球里面是3）种可能性，而不均衡的问题总会有一个或一些答案分支排除掉的可能性要小于k-1/k。于是策略的下界就被拖累了。

5. 小结

这的确是“小结”，因为两点：

1. 这个问题可以有信息论的理论解释，而信息论则是一个相当大的领域了。

2. 文中提到的这种看问题的视角除了用于排序、称球，还能够运用到哪些问题上（比如搜索）。

Update(06/13/2008)：徐宥[20]在讨论中继续提到[21]：

> *另外，这几天我重新把TAOCP 第三卷(第二版)翻出来看了看 Knuth 怎么说这个问题的, 发现真是牛大了：*
>
> *先说性能：*
>
> *pp148, section 5.2.3 说：*
>
> *When N = 1000, the approximate average runiing time on MIX are*
> *160000u for heapsort*
> *130000u for shellsort*
> *80000u for quicksort*
>
> *这里，Knuth 同学发现一般情况下 heapsort 表现很不好. 于是，在下文他就说，习题18 (pp156, 难度21)*
>
> *(R.W.Floyd) During the selection phase of heapsort, the key K tends to be quite small, so that nearly all the comparisons in step H6 find*

> *K<K_j. Show how to modify the algorithm so that K is not compared with*
> *K_j in the main loop of the computation, thereby nearly cutting the average number of comparisons in half.*
>
> *答案里面的方法和DMK的方法是一样的。(我觉得DMK是看了这个论文或者TAoCP的) 这里说 by half，就正好和快排差不多了。*
>
> *再说信息论分析：*
>
> *在5.3.1 (pp181) 高爷爷就说,“排序问题可以看成是一个树上的鸟儿排排站的问题. (还特地画了一棵树), 下一段就说, 其实这个也有等价说法, 就是信息论, 我们从称球问题说起...”*
>
> *然后后面一直讲信息论和最小比较排序...*

高爷爷真不愧是姓高的，囧rz..

Links

1. http://mindhacks.cn/author/pongba/
2. http://mindhacks.cn/topics/math/
3. http://mindhacks.cn/topics/algorithms/
4. http://mindhacks.cn/topics/computer-science/
5. http://groups.google.com/group/pongba
6. http://groups.google.com/group/pongba/msg/f95aa12feb4dfd67
7. http://groups.google.com/group/pongba/browse_frm/thread/28ac39e0222becf2
8. http://blog.csdn.net/g9yuayon
9. http://blog.csdn.net/g9yuayon/archive/2007/04/22/1574518.aspx
10. http://mitpress.mit.edu/sicp/
11. http://swiss.csail.mit.edu/classes/symbolic/spring07/readings/robust-systems.pdf
12. http://users.aims.ac.za/~mackay/

13. http://users.aims.ac.za/~mackay/sorting/sorting.html
14. http://blog.youxu.info/
15. http://users.aims.ac.za/~mackay/itila/book.html
16. http://mindhacks.cn/wp-content/uploads/2009/02/23131201.jpg
17. http://mindhacks.cn/wp-content/uploads/2009/02/23131201.jpg
18. http://users.aims.ac.za/~mackay/sorting/sorting.html
19. http://en.wikipedia.org/wiki/Radix_sort
20. http://blog.youxu.info/
21. http://groups.google.com/group/pongba/msg/07493e329ed920ff