第138封信 | 32位的二进制里有多少个1？

吴军

第138封信 | 32位的二进制...
11:39 5.46MB

信件朗读者：宝木

小师弟，你好！

这周我们继续讲几道Google和微软的面试题，透过这些问题，你可以理解计算机内存和速度的关系，空间和时间的关系，并且掌握计算机科学的一个重要概念和工具：抽象的图（它不是地图，但是和地图有相关性）。

另外，之前有朋友问Google下载所有的网页难道都是自动进行的么？我给了一个简单、肯定的答案，这一周，我们也要用图这个工具说明Google是怎样做的。通过Google的算法，你可以理解工作中和管理上的几个概念，比如什么是瓶颈。

今天我们从一个看似简单的问题讲起。问题只有一句话：

一个32位或者64位的二进制中有多少个1，多少个0？

我们知道计算机内部都是用二进制存储的。为了简单高效，计算机里面是分组存储的，每八个二进制称为一个字节，能够表示2的8次方个数字，从0到255，这八个一组被称为一个字节。当然，两个字节就能表示大约255的平方个数字了。

十几年前计算机的容量有限，处理器的精度也有限，因此常常使用32位二进制，也就是4个字节表示一个整数，它从负的20亿到正的20亿，通常也够用了，今天大部分计算机都是64位的，可以表示的数据范围也大了很多。这一道题是过去微软和Google经常考的面试题，为了便于你理解，我给你看两个具体的例子：

在二进制10110000中有三个1，在01000100中有两个1。

这个问题看起来很简单，实际上开始的时候我也不知道这样的问题有什么实际应用价值，以为仅仅是考察计算机科学的功底，到Google一个月后，在工作中还真遇到了要数一数32位或者64位二进制整数中有多少个1这种情况了，因此它还是有点实际价值的。

对于上述问题最简单的做法是挨个数一遍，32位数就数32次。这种做法当然没有问题，但是显然不是效率最高的做法。

在面试时，稍微聪明一点的面试者应该知道无论是微软还是Google并不期望这样的答案，因此他们在想不出更好的答案时，不妨这样回答，**"我知道挨个直接数32次不是好的做法，但是可以得到结果"**，表示至少对这个问题有更深入一点的认识。如果一个计算机专业的学生专业知识学得比较扎实，他应该知道关于二进制下面这样一个事实：

10000000（1后面七个0）和01111111（0后面七个1）大小只相差1，具体说后者是前者减1。

有趣的是，这两个数字所有的位数都不相同，1和0正好相反，在计算机里判断两个数是否所有的位数都相反只需要一次操作。这种方法的本质实际上是判断出一个数字是否是2的整数次方。

比如例子中的第一个数字是2的7次方。对于其他数字，比如10001000，它就是2的7次方，加上2的3次方。也就是说，可以通过两次判断判别出来。推而广之，如果一个二进制数字中有N个1，只要判断N次就知道有多少个1，而不需要把二进制的每一位都看一遍。这种算法的细节如果你看不明白也没关系，只需要记住两条即可：

1. 从理论上讲，二进制数字中有多少个1，就需要多少次操作。如果你的程序做了更多的操作，说明做了无用功。

2. 这种算法因为用到逻辑操作"与"，我们不妨称之为"与"算法。

"与"算法需要对计算机科学有比较透彻的理解才能想到。如果你的计算机课程平均成绩得了B，恐怕自己是想不出这个算法的。当然，面试前可以刷题，网上有这段程序，不过刷题的人通常理解不了其中的原理，因此，在Google好一点的面试官很容易判断面试者的答案是刷题刷出来的，还是自己理解做出来的。

对于这个问题，我遇到的回答得最好的人是我在清华的学生戴祥天，他后来在约翰·霍普金斯读了博士，之后面试Google，我问了他这个问题，他不仅把上述算法讲清楚，而且还从二进制推广到任意的N进制，而他在和其他面试官面试时，也总是能给出超出预期的答案，最后大家一致认为这是个天才，马上给了他Offer。从这个例子你可以看到学完计算机科学，大家对它的理解程度可以相差非常多。

上述问题还有一个更简单直接的答案，就是微软通常给出的答案。简单地讲，就是对所有的二进制编一张大表，表中直接记录这个二进制中有多少个1即可，比如10001000有两个1，11011011有六个1。这种算法，只要进行一次操作，就可以知道一个二进制数中有多少个1了，因此比上述的"与"算法还快。

然而，凡事都有成本，这种查表的做法在节省时间的时候显然要用其他资源作代价，那就是存储这张大表所需要的空间。如果是32位数，则需要4GB的存储空间，如果你放在计算机里，一半的内存就没了。如果是64位，全世界所有计算机的内存都用上也放不下。你可能会奇怪为什么增长这么多，这就是指数增长的可怕之处。

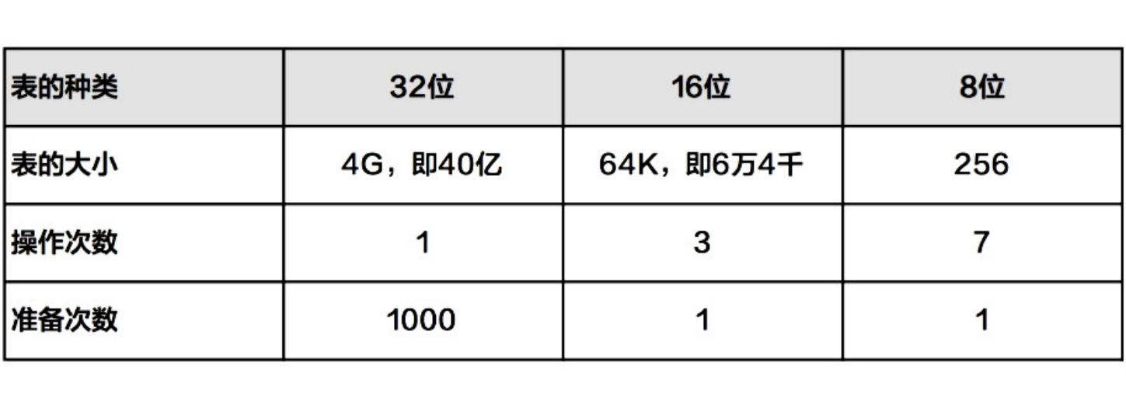
那么有没有好一点的，折衷的做法呢？有的。我的一个同事Z在面试Google时就被问到了这个问题。当时公司还小，因此主管研究的总监，也就是后来Google主管工程的第一把手尤斯塔斯也要亲自面试工程师，我的同事Z当时就遇到了他，而且被问了这个问题。他先给出了微软的这种查表方法，尤斯塔斯马上追问，这种方法是否实用？Z很聪明，马上讲解了为什么不实用，因为占用内存太多。尤斯塔斯又追问有什么补救方法。Z说，如果把32位数字变成两组16位的数字，建立一个16位数字的表格，32位数字前后查表两次，然后把两次的结果加起来即可。而16位数字的表格只占64K空间，只是4G的零头。当然，我的朋友讲，还可以将32位二进制数变成四个八位数（也就是四个字节），对八位数建表，只需要256个字节的存储单元就够了。当然，这样需要查表四次，做三次加法。

如果回答到这个地步，在微软的面试可以得满分了，在Google的面试中也能通过了。不过，尤斯塔斯随后又问出一个问题，让我这个同事对Google佩服不已。

尤斯塔斯问，假如不考虑成本，建立一个4G字节的大表，是否真的就比64K或者256字节的小表来得快？如果没有系统学过计算机课程的人肯定会回答大表来得快，因为从理论上讲确实如此，一次查表操作，要比3次操作或者7次来得快。

但是，在真实的计算机中，内存和处理器之间还有一个高速缓存，程序和数据要先从内存进入高速缓存，才能运行。高速缓存的空间非常有限，通常只有几兆，4G内存上的内容是缓存容量的上千倍，肯定是放不下的，遇到这种情况，计算机本身要进行上千次额外操作，把内存的内容往缓存倒腾。也就是说，**如果建立一个大表，虽然查表只需要做一次，但是准备工作可能要做上千次。**

如果你对上面的解释还不是很理解，不妨看一眼下表，给出了建立32位二进制大表、16位二进制中表和8位二进制小表所需要的内存资源，和所需要的计算操作次数。



表的种类	32位	16位	8位
表的大小	4G，即40亿	64K，即6万4千	256
操作次数	1	3	7
准备次数	1000	1	1

从这个表中你可以看出建立大表的方法只是在理论上有效（操作次数是1），在实际工程上根本不可行（准备次数太高）。显然，要回答好这道面试题，光是把计算机算法学好还不够，还要对计算机系统结构有精深的了解。

我的同事Z当然答出了这个问题，但是他感叹不在Google里面负责一线开发，而是负责工程管理的高管对计算机的原理能如此了解，这个公司的技术水平肯定没得说。可以讲，是尤斯塔斯的这个问题让他下定决心加入Google。

Google的工程师文化实际上植根于它的基因，负责工程的高管虽然后来不可能自己写代码了，但是不等于这些人不会写。尤斯塔斯的水平通过这个例子可以见识到。同尤斯塔斯一同主管Google工程的