算法 计算机

有哪些令人拍案叫绝的算法?

添加评论 分享

25 个回答 默认排序

Liang Miya， 听雪江湖

781 人赞同

在Matrix67的blog 上看的一道题：
有一个黑匣子，黑匣子里有一个关于 x 的多项式 p(x)。我们不知道它有多少项，但已知所有的系数都是正整数。每一次，你可以给黑匣子输入一个整数，黑匣子将返回把这个整数代入多项式后的值。那么，最少需要多少次， 我们可以得到这个多项式每项的系数呢?
答案是两次。
/*----------答案分界线----------*/

第一次，输入 1 ，于是便得到整个多项式的所有系数之和。记作 S 。
第二次，输入 S + 1 ，于是黑匣子返回的是
$a_n * (S1)^n a_{n-1} * (S1)^{n-1} ... a_1 * (S1) a_0$ 的值。
我们要得到 $a_n, ..., a_0$，只需要把这个值换成 S+1 进制， 然后依次读出每一位上的数。

P.S. 第一次得到S是为了保证对于任意系数 $a_i, a_i <= S$

编辑于 09:06 84 条评论 感谢 分享 收藏 • 没有帮助 • 举报 • 作者保留权利

知乎用户， nothing

225 人赞同

刷leetcode时碰到的，题目为Single Number | LeetCode OJ 和Single Number III | LeetCode OJ 。

这个题目是说，有一个n个元素的数组，除了一个元素只出现一次外，其他元素都出现两次，让你找出这个只出现一次的元素是几，要求时间复杂度为O(n)且不再开辟新的内存空间。
————————————答案分割线————————————

解法是将所有元素做异或运算，即a[1] XOR a[2] XOR a[3] XOR…XOR a[n]，所得的结果就是那个只出现一次的数字，时间复杂度为O(n)。

当时看了答案后就一拍脑瓜，感叹这个技巧真不错。然而直到我遇见了这道题的进阶版，当时真是恍然大悟拍案叫绝。

进阶版:有一个n个元素的数组，除了两个数只出现一次外，其余元素都出现两次，让你找出这两个只出现一次的数分别是几，要求时间复杂度为O(n)且再开辟的内存空间固定(与n无关)。
————————————答案分割线————————————

首先，仿照前面的算法，把所有元素异或，得到的结果就是那两个只出现一次的元素异或的结果。

然后，重点来了，因为这两个只出现一次的元素一定是不相同的，所以这两个元素的二进制形式肯定至少有某一位是不同的，即一个为0，另一个为1，找到这一位。

可以根据前面异或得到的数字找到这一位，怎么找呢？稍加分析就可以知道，异或得到这个数字二进制形式中任意一个为1的位都是我们要找的那一位，找到这一位就可以了(这很容易)。

再然后，以这一位是1还是0为标准，将数组的n个元素分成了两部分，将这一位为0的所有元素做异或，得出的数就是只出现一次的数中的一个;将这一位为1的所有元素做异或，得出的数就是只出现一次的数中的另一个。从而解出题目。忽略寻找不同位的过程，总共遍历数组两次，时间复杂度为O(n)。

时宇电， 微信公众号:给个策略

212 人赞同

一道很有意思的题目，曾经在面试腾讯关键时刻出现过，感恩。

问:
有一种玻璃杯质量确定但未知，需要检测。
有一栋100层的大楼，该种玻璃杯从某一层楼扔下，刚好会碎。
现给你两个杯子，问怎样检测出这个杯子的质量，即找到在哪一层楼刚好会碎?

------- 一条分割线的时间思考 ---------

分析：
首先理解题意，将这个杯子从某一层楼扔下，如果没碎，我还可以再利用它测试。
如果碎了的话，就不能再继续用了。
如果我从x楼扔下，没碎，在x+1楼扔下，碎掉了，即证明找到了x+1是刚好碎掉的楼层。

问题的关键是，怎么快速找到这个楼层呢？这是一个**查找**问题。
我们需要一个**策略方法**来快速地找到它，就看谁的方法比较优秀拉。
而优秀的方法其评价标准显而易见：**各种情况下都能快速地找到目标楼层。**

------- 再一条分割线的时间思考 ---------

思考路径：
如果只有一个杯子的话，应该怎么做呢?

稍微想一下也可以知道，必定只能一层一层地扔，1楼没碎扔2楼，2楼没碎扔3楼，直到碎掉。

现在我有两个杯子。

学习过算法和程序的人应该都知道二分法，很容易想到这样去做，因为面对的是一个搜索问题。
所以可能会给出这样的策略：
从50楼扔下，没碎的话，再扔75楼，再没碎我扔88楼，依次下去很快就可以锁定楼层？
很快你会意识到问题所在，万一第一次从50层楼扔下去，碎了咋整，难道又一层一层地扔？
杯子的质量是刚好在49层碎掉的话。最差的情况我需要扔50次，这方法不行。

再一个比较常见的方法是，先分区间的扔，再慢慢地一层一层地扔，隐含着分段查找的策略。
具体操作方式是：
先从第10楼扔，再从第20楼扔，依次下去，如果到某一层碎掉，比如50层碎掉了，我再从41楼开始扔，这样的话应该算是比较快了把？
这个方法是要快一点不过如果我杯子的质量比较好，在99楼才会刚好碎掉。
这样，最差的情况下，需要扔19次才能找到目标楼层，还是不能让面试官满意。

我们需要的方法是无论杯子的质量如何，不论是在1楼碎，49楼碎，99楼碎都要能快速锁定的方法。

继续思考刚才方法的缺陷，当杯子质量比较差的时候，此方法还是比较快速的找到的。比如杯子是在19楼刚好碎，我只需要扔11次，比99楼刚好碎的情况要少很多次。

所以我们的愿望是：
杯子的质量无论分布在哪个查找区间，都可以快速地找到。所以我想到的是可以“匀”一下刚才的方法。即最开始我需要大胆地扔，然后再慢慢小心地扔。

具体方法设计：
每次扔的区间减少一层，这样做可以保证每个区间查找的数期望次数一致。
假定第一步在15楼扔，没碎的话则下一步在29楼扔，没碎下一步在42楼扔....碎掉之后则在上一次没碎的楼层开始向上扔。那么最开始在哪一层开始扔呢？？
这里我们需要拿支笔算一下：
x+(x-1)+(x-2)+...+2 >=100
求解出答案为14。

即最终给出的解决方案是：
最开始从14楼开始扔，没碎的话在27楼扔，再没碎的话在39楼扔.....一旦碎掉，则从上一次没碎的楼层逐层往上扔，即可快速确认杯子在哪一层刚好会碎掉。

这样的方法可以保证在最差的情况下也能在14次内找到楼层，平均需要的次数不到10次。
（列式子算了下期望是9次多）

这是我知道的最好的方法，有意思。

清风

update:
出乎我的意料，评论区在一本正经地讨论这个算法的原理和实用性。这个算法确实没什么卵用，也没有哪个库会用它(有的话，请打脸，我服...)。我说这个的本意是想图个乐，而且确实惊叹于作者的思路，这种不按套路出牌的算法着实无比惊艳。所以，各位就当我抖了个机灵，不必太较真。

<!-- 原答案在下面 -->

睡眠排序，简直魔性...

大体思路是对每个元素开一个线程，线程的工作是睡眠该元素的值那么多的时间，然后将元素的值打印出来。这样一来，值越大，睡眠时间越长，排位越靠后，从而完成排序。

第二大魔王， 纵情豁达

鸡兔同笼吧
已知共有鸡和兔15只,共有40只脚,问鸡和兔各有几只.
算法:假设鸡和兔能听懂口令,吹一声哨,它们抬起一只脚。再吹一声哨,它们又抬起一只脚 ,这时鸡都一屁股坐地上了,兔子还两只脚立着。所以,兔子有10/2=5只,鸡有

15-5=10只

发布于 2015-01-12　23 条评论　感谢　分享　收藏 • 没有帮助 • 举报 • 作者保留权利

魂魄妖梦， 半分幻の庭师

12 人赞同

一些蚂蚁在树枝上向其中一端爬，爬到末端就掉下去，如果相遇则向反方向爬，蚂蚁速度相同，给定位置和方向，求所有蚂蚁从树枝上落下来的时间

发布于 昨天 07:03　6 条评论　感谢　分享　收藏 • 没有帮助 • 举报 • 作者保留权利

郁白， 分布式与数据库，选择性回复，不跟小白辩论

1 人赞同

无限循环小数化分数的方法

发布于 06:30　添加评论　感谢　分享　收藏 • 没有帮助 • 举报 • 作者保留权利

阮行止， 高中生,oier,螳臂当车的歹徒

19 人赞同

快速傅里叶变换（FFT）。

考虑多项式 $A=\sum_{i=0}^{n}a_ix^i, B=\sum_{i=0}^{n}b_ix^i$。现在要计算这两个多项式的卷积（就是乘积）。

普通的多项式乘法，总是系数相乘，走不出 $O(n^2)$ 的复杂度。

```
for(i=0;i<lenA;i++)
    for(j=0;j<lenB;j++)
        ans[i+j]+=A[i]*B[j];
```

于是就有了FFT的令人拍案叫绝之处：用点值来表达一个多项式。
我们知道，给定两个点就可以求出一个一次函数的解析式，给定三个点就能求出一个二次函数的解析式……这启示我们，可以将多项式以点值的形式表示。

选定 $n1$ 个取值点，求出这些点上的值。我们拿到这些值，就可以还原出整个多项式。
例如： $f(x)=2x^2x-1$，选定取值点 $x=\{0,1,2\}$，于是我们得到这个多项式的点值表达： $\{(0,-1)(1,2)(2,9)\}$

有什么用呢？详见我的博客：FFT入门 · blue's Blog　。

FFT令人拍案叫绝的地方，就是想到了用点值来表达多项式，并**在点值上计算乘积**。
凭借对单位复数根的灵活应用，多项式乘法的复杂度被降到了 $O(n\log n)$。

与之类似，快速数论变换（NTT）也是令人拍案叫绝的算法。但是由于与FFT思路相似，不再赘述。
noip2016 RP++.

编辑于 昨天 21:23　6 条评论　感谢　分享　收藏 • 没有帮助 • 举报 • 作者保留权利

知乎用户， 学习编程中...

26 人赞同

```
struct BinTree {
        BinTree* left;
        BinTree* right;
        int value;
};

struct InvertBinTree {
        InvertBinTree* right;
        InvertBinTree* left;
        int value;
};

BinTree* binTree;
/* *** */
InvertBinTree* invertBinTree = (InvertBinTree*)binTree;
```

好像可以O(1)翻转二叉树

编辑于 2016-11-13　51 条评论　感谢　分享　收藏 • 没有帮助 • 举报 • 作者保留权利

李晨曦， 计算机硕士在读

4 人赞同

看了一个小算法，被深深的震撼了，原来编程中被我们认为理所当然应该这样做的东西，实际上还有很多优化的空间。
在一个文件中有很多行这样的数据
134|32|35|2|43234|.....|12'\n'
25|5|286|16|58|.....|614'\n'
8731|8|98|22|4823|.....|62'\n'
.....
.....代表还有很多列，很多行，我的实验里一共15列，600万行
需求是求所有第二列数字的和，在这个例子里是32+5+8=45

1.最简单的方法就是用C++中的getline读一行数据，然后逐个字符判断，如果是第二个'|'，就把第一个和第二个'|'中间的数字取出来就可以了，然后处理下一行。
这种方法时间是：0.8秒

2.第1种方法用getline，它帮我们读取文件的时候，实际上是要逐个字符比较'\n'的，才能取出来这一行，然后我们查找'|'的时候，又遍历了一下这一行的其中一部分。实际上我们只要遍历一次就可以了，于是可以用mmap函数把整个文件映射进内存，然后逐个字符判断是'|'还是'\n'来取得我们想要的结果。
这种方法时间是：0.65秒

3.前面都是铺垫，这个才是我想说的
如果我们不是每次只处理一个字符，而是一次处理多个字符呢?
查找第二列的时候还是和2中方法相同，关键是找'\n'的时候
这种方法的时间是：0.45秒

```
//char *current = 文件初始位置, safeLimit是文件结尾（有些细节就先不处理了）
while (current<safeLimit) {
//把8个char当作一个64位的无符号整型处理，也就是同时处理8个字符!
    uint64_t block=*reinterpret_cast<const uint64_t*>(current);
//单独拿出来一个字符看，1000,0000，实际上是8个这样的
    constexpr uint64_t highBits=0x8080808080808080ull;
//0111,1111
    constexpr uint64_t lowBits=~highBits;
//0000,1010，这个是'\n'（Ascii码）
    constexpr uint64_t pattern=0x0A0A0A0A0A0A0A0Aull;
//这块有点神奇，你自己用一个例子试一下
    uint64_t lowChars=(~block)&highBits;
    uint64_t found0A=~((((block&lowBits)^pattern)+lowBits)&highBits);
    uint64_t match=found0A&lowChars;
//如果match是1000,0000，就说明遇到'\n'了
//如果这8个字节里没有'\n'，直接进入下8个字节
    if (!match) {
        current+=8;
        continue;
    }
//把match的字节前后交换一下位置
    match=__builtin_bswap64(match);
//找到了'\n'，current直接移动到下一行
    current+=(__builtin_clzll(match)/8);
}
```

编辑于 昨天 06:27　32 条评论　感谢　分享　收藏 • 没有帮助 • 举报 • 作者保留权利

李博， 学计算机的

1 人赞同

SA
SAM
DLX
平衡树的各种旋转维护操作

发布于 08:48　添加评论　感谢　分享　收藏 • 没有帮助 • 举报 • 作者保留权利

湛兰， 此人编程不发达 大家都要支援他！

9 人赞同

自从学了算法之后，Leetcode见的多了，各种算法接触的多了，我的内心对于神奇算法基本已经毫无波动。

讲一个虽然幼稚，但是在小学时候，也曾经惊艳过我的算法。题目：甲乙两个人相向而行，相距L，出发同时甲的狗也出发在甲乙之间往返。已知甲，乙，狗速度分别为A，B，C且有 C>A>B。
求两人相遇时狗跑过的距离。

在小时候我怎么都做不出来，因为算出来 我列出了一个无数项求和的式子。在当时的我根本求不出来。

结果当我问了我爹后= =我爹直接就写下了答案

===我是奥术（划掉）奥数的分割线===
答案：
L/(A+B)*C。
算法：
先求出甲乙相遇时间，再乘以狗的速度=。=

当时这个算法震撼了我幼小的心灵，原来同样问题换个算法复杂度会下降这么多！

===我是高数的分割线===

后来当接触到收敛概念的时候。

第一反应就是：

妈的，小时候写的那个无限级数是收敛的，其实能算出来!

有奥数童年的小伙们求赞~\(≧▽≦)/~

编辑于 00:57　8 条评论　感谢　分享　收藏 • 没有帮助 • 举报 • 作者保留权利

torival， 学生

6 人赞同

就说一个小巧、精致、但又容易被人忽略的一个小算法吧。

1、假设有一篇文章，然后呢，让你统计每个字符出现的次数。

我想绝大多数人，还是能找到这样的办法的。定义一个包含26个元素的int数组，并初始化；循环读取文件里面内容，然后把每个字符映射为下标，直接统计。
空间：O(1)，时间：O(N)

2、对于年龄进行排序。

可能有一部分人，在想快排？归并？不不不，都不用。还是上面的想法，通过定义一个包含100个元素的数组，然后把每个人的年龄做下标，统计个数。之后从0到100，把个数不为零的输出，每输出一个，对应的个数减一。
空间：O(1)，时间：O(N)

3、按学号排序。

很多人想到了，既然我说只讲一个小算法，那么第三个肯定也是定义一个很大很大的数组，然后学号做下标。不不不，又错了。如果真是这么搞，得浪费多少资源。咱们可以这么搞，假定学号的上限为N，然后定义一个N/32 + 1的数组，之后把这个区间的每一位看做一个数，比如a[0]就代表0-31这些数，注意每一位看做一个数。我假定int是4个字节。然后，这个排序有没有大量节省空间？
空间：O(?)，时间：O(N)

注：第二个排序方法叫，计数排序。
第三个排序方法叫，位图排序。

如果你懂数据结构的话，其实这三个问题都是通过构造了一个简单的hash table，然后哈希函数选择了直接映射的函数。

半夜手机撸出来的，感觉可以，赞一个吧！
(๑•̀ㅁ•́๑)

发布于 01:04　3 条评论　感谢　分享　收藏 • 没有帮助 • 举报 • 作者保留权利

尼不要逗了， I am here　　legendtkl.com

5 人赞同

说一个刷题经常用的小技巧: 链表的假头 dummy head。有链表的题操作链表元素一般都需要考虑被操作的这个元素是不是表头，如果是表头往往要做特殊处理。

举个例子，删除操作。
1.如果不是表头，直接把这个节点的pre指向其next，然后删除该节点。
2.如果是表头，把head暂存起来tmp，head指向next，删除tmp。

使用dummy head的话，删除原链表的表头也转变成第一种情况了。因为原表头现在也有pre了，也就是dummy head。
类似这种把edge case转化为normal case的小技巧总是让人耳目一新。

评论区都说链表这个太基础了，但是这种思想确实很巧妙。那么再说几个：

1. LIS问题的二分实现

LIS问题是典型的DP问题，简单描述：有一个长为n的数组，求出这个序列中最长的上升子序列的长度。对于这个问题的解法这里就不赘述了。使用DP解法可以达到O(n^2)的时间复杂度，如果再使用二分优化就可以达到O(nlgn)。二分的实现千差万别，下面这种实现尤为巧妙。

```
int LIS(vector<int> nums) {
    vector<int> seq(nums.size(), INF);//INF是一个比nums数组中所有数都要大的数
    for(int i=0; i<nums.size(); i++)
        *lower_bound(nums.begin(), nums.end(), nums[i]) = nums[i]
    return lower_bound(seq.begin(), seq.end(), INF) - seq.begin();
}
```

编辑于 08:49　6 条评论　感谢　分享　收藏 • 没有帮助 • 举报 • 作者保留权利

xian hong

3 人赞同

dancing links

发布于 昨天 10:25　添加评论　感谢　分享　收藏 • 没有帮助 • 举报 • 作者保留权利

知乎用户， IT码农

2 人赞同

平方根倒数速算法

发布于 昨天 11:42　1 条评论　感谢　分享　收藏 • 没有帮助 • 举报 • 作者保留权利

是个胖猴子， 胖

1 人赞同

我就说一些zero knowledge proof的算法吧，分为interactive和noninteractive，大概意思就是Alice要向Bob证明一件事情，但是两人都不能获得额外的信息。场景就是你向对方证明你知道密码，但是对方又不知道你的密码是什么。举几个简单例子。

1. 有n>2个交易员，他们想知道他们平均的薪水，但是又不想任何人知道自己的薪水。

算法就是，第一个交易员random选一个数，加在自己的薪水上，然后告诉第二个交易员，然后第二个再把自己薪水加上去告诉第三个交易员，。。。。。，第n个把自己的薪水加上去告诉第一个交易员，第一个交易员把自己最初选的随机数减掉，然后把结果/n告诉大家。

2. Alice是色盲，她有两个球，外观完全一样，Bob告诉她这两个球颜色不同，问Bob怎么向Alice证明这俩球颜色不同。

把两个球分别称为A球B球，Alice放在背后，每次拿出来一个让Bob判断是哪个，重复足够多次，say100000次，如果Bob每次都说对，那么两个球颜色不同的概率>1-(1/2)^100000。

3. 判断两个图G和H是同构的是NPC的，Alice claim这两个图是同构的，事实上，她只要给出G和H顶点的对应关系，Bob就可以检验，但是她又不想让Bob知道同构的证据，那么如何证明。

每一次，Alice都生成一个G的同构图G'，然后由Bob提出由Alice给出G和G'的顶点对应关系，还是H和G'的顶点对应关系。重复足够多次，如果Bob每次都说对，那么G和H同构的概率就非常大。

4. 还有一个印象很深的公钥秘钥的问题，记不清了，我想想看。

编辑于 10:31　添加评论　感谢　分享　收藏 • 没有帮助 • 举报 • 作者保留权利

知乎用户， 烦躁的程序狗！

1 人赞同

鸵鸟算法

wapbaike.baidu.com/item ...

编辑于 00:07 添加评论 感谢 分享 收藏 • 没有帮助 • 举报 • 作者保留权利

王睿， 北理工统计phd

1 人赞同

我觉得统计假设检验中的随机化检验这类算法很绝。比如广泛使用的两样本检验中用的置换方法。

这些算法用来确定检验的临界值。不管多复杂的检验统计量，都能被很好的控制住水平。

还有最小角回归算法，竟然可以用来解lasso。算法的复杂度竟然和最小二乘法是一样的，好神奇的。

发布于 昨天 23:03 添加评论 感谢 分享 收藏 • 没有帮助 • 举报 • 作者保留权利

陈源， 非典型物理系学生，音乐爱好者

求矩阵特征值的QR iteration.

给定可对角化的矩阵A，做以下迭代：

A=QR(QR分解)

A <- RQ

最后A将变成一个上三角矩阵，特征值出现在对角元上。

第一次看到时惊呆了，这都可以...

发布于 01:23 添加评论 感谢 分享 收藏 • 没有帮助 • 举报 • 申请转载

更多

我来回答这个问题

写回答…

我要回答

关注问题 1893 人关注该问题

相关问题 换一换

编程到底难在哪里？ 118 个回答

香农的信息论究竟牛在哪里？ 70 个回答

你碰到过的最难调试的 Bug 是什么样的？ 877 个回答

电脑取随机数是什么原理，是真正的随机数吗？ 16 个回答

作为计算机专业学生，最应该学习的课程前五位是什么？ 130 个回答