

个人资料



E_S_Promise

访问： 113768次
积分： 1941
等级： 4
排名： 第18690名

原创： 90篇 转载： 13篇
译文： 0篇 评论： 7条

文章搜索

文章分类

[ACM——数论 \(10\)](#)
[ACM——排序 \(2\)](#)
[ACM——博弈论 \(3\)](#)
[ACM——数据结构 \(18\)](#)
[ACM——大数运算 \(2\)](#)
[ACM——动态规划 \(19\)](#)
[ACM——贪心算法 \(10\)](#)
[ACM——模拟 \(2\)](#)
[JAVA \(6\)](#)
[ACM \(2\)](#)
[ACM——分治 \(1\)](#)
[ACM——搜索 \(14\)](#)
[ACM——并查集（数据结构） \(3\)](#)
[ACM——树状数组（数据结构） \(8\)](#)
[ACM——线段树 \(3\)](#)
[HTML5 \(1\)](#)
[ACM——图论 \(17\)](#)
[概率题 \(1\)](#)
[ACM——差分约束 \(1\)](#)
[w \(0\)](#)
[Windows \(1\)](#)
[sh \(0\)](#)
[DBMS \(1\)](#)

文章存档

[2015年08月 \(4\)](#)

[【活动】2017 CSDN博客专栏评选](#) [【5月书讯】流畅的Python，终于等到你！](#) [CSDN日报20170519 ——《思维的局限》](#)
[Kotlin 专场](#)

快速幂（C语言实现）超详细（转载）

标签： [ACM](#) [acm](#) [快速幂](#) [数论](#) [算法](#)

2013-02-12 00:50

22740人阅读

[评论\(6\)](#)[收藏](#)[举报](#)分类： [ACM——数论 \(9\)](#)

快速幂取模算法

在网站上一直没有找到有关于快速幂算法的一个详细的描述和解释，这里，我给出快速幂算法的完整解释，用的是C语言，不同语言的读者只好换个位啦，毕竟读C的人较多~

所谓的快速幂，实际上是快速幂取模的缩写，简单的说，就是快速的求一个幂式的模(余)。在程序设计过程中，经常要去求一些大数对于某个数的余数，为了得到更快、计算范围更大的算法，产生了快速幂取模算法。[有读者反映在讲快速幂部分时有点含糊，所以在这里对本文进行了修改，作了更详细的补充，争取让更多的读者一目了然]

我们先从简单的例子入手：求 a^b 几。

算法1.首先直接地来设计这个算法：

```
int ans = 1;
```

```
for(int i = 1;i<=b;i++)
```

```
{
```

```
    ans = ans * a;
```

```
}
```

```
ans = ans % c;
```

这个算法的时间复杂度体现在for循环中，为 $O(b)$ 。这个算法存在着明显的问题，如果a和b过大，很容易就会溢出。

那么，我们先来看看第一个改进方案：在讲这个方案之前，要先有这样一个公式：

这个公式大家在离散数学或者数论当中应该学过，不过这里为了方便大家的阅读，还是给出证明：

引理1：

上面公式为下面公式的引理，即积的取余等于取余的积的取余。

证明了以上的公式以后，我们可以先让a关于c取余，这样可以大大减少a的大小，

于是不用思考的进行了改进：

算法2：

```
int ans = 1;
```

```
a = a % c; //加上这一句
```

```
for(int i = 1;i<=b;i++)
```

```
{
```

2014年03月 (3)
2014年01月 (4)
2013年10月 (5)
2013年09月 (9)

展开

阅读排行

快速幂（C语言实现）超 (22734)
HTML5 地理定位详解（！ (5555)
JAVA中的字节流和字符 (5490)
LPCTSTR, LPWSTR, P (4431)
HDOJ 2037 今年暑假不 (4043)
配置Vim——自动缩进 (3551)
UVa 714 Copying Books (3057)
UVa 10193 All You Neec (2029)
康拓展开 (1835)
递归回溯 暴力枚举（总！ (1659)

评论排行

快速幂（C语言实现）超 (6)
UVa 558 Wormholes 判 (1)
UVa 550 - Multiplying by (0)
POJ 2506 递归 + 高精度 (0)
POJ 1047 Round and Re (0)
POJ 1363 Rails 栈的应用 (0)
堆排序 (0)
ACM排序 (0)
POJ 1067 威佐夫博弈(W (0)
配置Vim——自动缩进 (0)

推荐文章

* 程序员4月书讯：Angular来了！
* CSDN日报20170516 —— 《驱动小白和硬件老司机关于硬件那点事儿的一次密谈》
* 简单粗暴地入门机器学习
* AntShares区块链的节点部署与搭建私有链
* 分布式机器学习的集群方案介绍之HPC实现
* Android 音频系统：从 AudioTrack 到 AudioFlinger

最新评论

快速幂（C语言实现）超详细（qq_1319540839: http://wenku.baidu.com/link?url=bRdA7fj0EybEc28TEZ...
快速幂（C语言实现）超详细（Syn99: 膜
快速幂（C语言实现）超详细（qq_36473502: 谢谢博主分享~ 博主真的强
快速幂（C语言实现）超详细（Saitama_: @qq_34552944:c代表对c取余，a%c就是求a除以c的余数。
快速幂（C语言实现）超详细（qq_34552944: 想知道那个c代表什么
快速幂（C语言实现）超详细（lcoding_F2014: Good!

```
ans = ans * a;  
  
}  
  
ans = ans % c;
```

聪明的读者应该可以想到，既然某个因子取余之后相乘再取余保持余数不变，那么新算得的ans也可以进行取余，所以得到比较良好的改进版本。

算法3:

```
int ans = 1;  
  
a = a % c; //加上这一句  
  
for(int i = 1;i<=b;i++)  
{  
  
    ans = (ans * a) % c;//这里再取了一次余  
  
}  
  
ans = ans % c;
```

这个算法在时间复杂度上没有改进，仍为 $O(b)$ ，不过已经好很多的，但是在c过大的条件下，还是很有可能超时，所以，我们推出以下的快速幂算法。

快速幂算法依赖于以下明显的公式，我就不证明了。

有了上述两个公式后，我们可以得出以下的结论：

1.如果b是偶数，我们可以记 $k = a2 \bmod c$ ，那么求 $(k)b/2 \bmod c$ 就可以了。

2.如果b是奇数，我们也可以记 $k = a2 \bmod c$ ，那么求

$((k)b/2 \bmod c \times a) \bmod c = ((k)b/2 \bmod c * a) \bmod c$ 就可以了。

那么我们可以得到以下算法：

算法4:

```
int ans = 1;  
  
a = a % c;  
  
if(b%2==1)  
  
    ans = (ans * a) mod c; //如果是奇数，要多求一步，可以提前算到ans中  
  
k = (a*a) % c; //我们取a2而不是a  
  
for(int i = 1;i<=b/2;i++)  
  
{  
  
    ans = (ans * k) % c;  
  
}  
  
ans = ans % c;
```

我们可以看到，我们把时间复杂度变成了 $O(b/2)$ 。当然，这样子治标不治本。但我们可以看到，当我们令 $k = (a * a) \bmod c$ 时，状态已经发生了变化，我们所要求的最终结果即为 $(k)b/2 \bmod c$ 而不是原来的 $ab \bmod c$ ，所以我们发现这个过程是可以迭代下去的。当然，对于奇数的情形会多出一项 $a \bmod c$ ，所以为了完成迭代，当b是奇数时，我们通过

$ans = (ans * a) \% c$;来弥补多出来的这一项，此时剩余的部分就可以进行迭代了。

形如上式的迭代下去后，当b=0时，所有的因子都已经相乘，算法结束。于是便可以在 $O(\log b)$ 的时间内完成了。于是，有了最终的算法：快速幂算法。

算法5：快速幂算法

```
int ans = 1;
```

```
a = a % c;
```

```
while(b>0)
```

```
{
```

```
if(b % 2 == 1)
```

```
ans = (ans * a) % c;
```

```
b = b/2;
```

```
a = (a * a) % c;
```

```
}
```

将上述的代码结构化，也就是写成函数：

```
int PowerMod(int a, int b, int c)
```

```
{
```

```
int ans = 1;
```

```
a = a % c;
```

```
while(b>0)
```

```
{
```

```
if(b % 2 == 1)
```

```
ans = (ans * a) % c;
```

```
b = b/2;
```

```
a = (a * a) % c;
```

```
}
```

```
return ans;
```

```
}
```

本算法的时间复杂度为 $O(\log b)$ ，能在几乎所有的程序设计（竞赛）过程中通过，是目前最常用的算法之一。

以下内容仅供参考：

扩展：有关于快速幂的算法的推导，还可以从另一个角度来想。

=? 求解这个问题，我们也可以从进制转换来考虑：

将10进制的 b 转化成2进制的表达式：

那么，实际上，

所以

注意此处的要么为0，要么为1，如果某一项，那么这一项就是1，这个对应了上面算法过程中 b 是偶数的情况，为1对应了 b 是奇数的情况[不要搞反了，读者自己好好分析，可以联系10进制转2进制的方法]，我们从依次乘到。对于每一项的计算，计算后一项的结果时用前一项的结果的平方取余。对于要求的结果而言，为时 ans 不用把它乘起来，[因为这一项值为1]，为1项时要乘以此项再取余。这个算法和上面的算法在本质上是同样的，读者可以自行分析，这里我说不多说了，希望本文有助于读者掌握快速幂算法的知识点，当然，要真正的掌握，不多练习是不行的。

By 夜せ | 深

上一篇 POJ 1363 Rails 栈的应用

下一篇 POJ 1047 Round and Round We Go (大数乘法) 水

相关文章推荐

- C语言字符串操作总结大全超详细
- C语言运算符优先级超详细
- C语言字符串操作总结大全超详细
- C语言字符串操作总结大全超详细
- C语言字符串操作总结大全超详细
- C语言字符串操作总结大全超详细
- C语言字符串操作总结大全超详细
- C语言字符串操作总结大全超详细
- 转载点评详细解析C语言中的sizeof
- linux下解决c语言undefined reference to sin



展馆设计



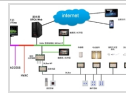
app开发报价



工地洗车机



立体车库



智能照明控制



视频剪辑学习



远程视频会议

参考知识库



C语言知识库

9013 关注 | 3465 收录



算法与数据结构知识库

16294 关注 | 2320 收录

猜你在找

- C语言及程序设计提高
- 《C语言/C++学习指南》加密解密篇（安全相关算法）
- C语言及程序设计初步
- C语言系列之 快速排序与全排列算法
- C语言系列之 数组与算法实战
- C语言系列之 字符串相关算法
- C语言系列之 递归算法示例与 Windows 趣味小项目
- C语言系列之 字符串压缩算法与结构体初探
- C语言及程序设计进阶
- Linux下的C语言程序设计

- 1 视频通话API，4行代码接入
- 2 烟台大学校企合作-韩国留学
- 3 jQuery MiniUI，加速Web开发
- 4 专业资金盘和资金类游戏开发
- 5 火爆 | 2017年热门网页游戏大全
- 6 珍珠美人-优质珍珠首饰品牌.
- 7 爱聊视频聊天室
- 8 网吧营销活动-网吧营销大师
- 9 每个企业都在用的在线客服系

查看评论

5楼 qq_1319540839 2017-03-12 08:53发表



<http://wenku.baidu.com/link?url=bRdA7fj0EybEc28TEZ01E8AOQkuOVA2Xb9fJ3wYKVSt3o4WlbYzhx4vP5EJEoGnejptJCFzo5EtMj14ADN4F3QR6JQBEm>

4楼 Syn99 2017-01-04 19:10发表



膜

3楼 qq_36473502 2016-10-24 15:53发表



谢谢博主分享~
博主真的强

2楼 qq_34552944 2016-04-05 23:12发表



想知道那个c代表什么

Re: Sailama_ 2016-06-12 20:37发表



回复qq_34552944：c代表对c取余，a%c就是求a除以c的余数。

1楼 [lcoding_F2014](#) 2015-09-28 19:41发表



Good!

您还没有登录,请[\[登录\]](#)或[\[注册\]](#)

* 以上用户言论只代表其个人观点，不代表CSDN网站的观点或立场

核心技术类目

[全部主题](#) [Hadoop](#) [AWS](#) [移动游戏](#) [Java](#) [Android](#) [iOS](#) [Swift](#) [智能硬件](#) [Docker](#) [OpenStack](#)
[VPN](#) [Spark](#) [ERP](#) [IE10](#) [Eclipse](#) [CRM](#) [JavaScript](#) [数据库](#) [Ubuntu](#) [NFC](#) [WAP](#) [jQuery](#)
[BI](#) [HTML5](#) [Spring](#) [Apache](#) [.NET](#) [API](#) [HTML](#) [SDK](#) [IIS](#) [Fedora](#) [XML](#) [LBS](#) [Unity](#)
[Splashtop](#) [UML](#) [components](#) [Windows Mobile](#) [Rails](#) [QEMU](#) [KDE](#) [Cassandra](#) [CloudStack](#) [FTC](#)
[coremail](#) [OPhone](#) [CouchBase](#) [云计算](#) [iOS6](#) [Rackspace](#) [Web App](#) [SpringSide](#) [Maemo](#)
[Compuware](#) [大数据](#) [aptech](#) [Perl](#) [Tornado](#) [Ruby](#) [Hibernate](#) [ThinkPHP](#) [HBase](#) [Pure](#) [Solr](#)
[Angular](#) [Cloud Foundry](#) [Redis](#) [Scala](#) [Django](#) [Bootstrap](#)

[公司简介](#) | [招贤纳士](#) | [广告服务](#) | [联系方式](#) | [版权声明](#) | [法律顾问](#) | [问题报告](#) | [合作伙伴](#) | [论坛反馈](#)

[网站客服](#) [杂志客服](#) [微博客服](#) [webmaster@csdn.net](#) 400-660-0108 | 北京创新乐知信息技术有限公司 版权所有 | 江苏知之为计算机有限公司 |

江苏乐知网络技术有限公司

京 ICP 证 09002463 号 | Copyright © 1999-2017, CSDN.NET, All Rights Reserved 