

xuanflyer

xuanflyer is fighting!!!

目录视图 摘要视图 RSS 订阅

个人资料



xuanflyer

访问： 38600次

积分： 1025分

排名： 第13455名

原创： 67篇 转载： 1篇

译文： 0篇 评论： 30条

文章搜索

文章分类

ACMICPC (65)

数据结构 (50)

计算几何 (4)

比赛 (1)

杂 (12)

心得 (2)

技术博客

July

Cool Shell

文章存档

2014年01月 (1)

2012年09月 (1)

2011年11月 (2)

2011年10月 (1)

2011年09月 (4)

展开

英雄会第四届在线编程大赛·线上初赛：带通配符的数 CSDN社区“三八节”特别活动 开发者职业生涯调查之未来

Manacher算法--O (n) 回文子串算法

分类： 数据结构 ACMICPC 2011-07-30 00:41 13819人阅读 评论(3) 收藏 举报

算法

O (n) 回文子串算法

注：转载的这篇文章，我发现下面那个源代码有点bug。。。在下一篇博客中改正了。。

这里，我介绍一下O (n) 回文串处理的一种方法。Manacher算法。

原文地址：

<http://zhuhongcheng.wordpress.com/2009/08/02/a-simple-linear-time-algorithm-for-finding-longest-palindrome-sub-string/>

其实原文说得是比较清楚的，只是英文的，我这里写一份中文的吧。

首先：大家都知道什么叫回文串吧，这个算法要解决的就是一个字符串中最长的回文子串有多长。这个算法可以在O (n) 的时间复杂度内既线性时间复杂度的情况下，求出以每个字符为中心的最长回文有多长，

这个算法有一个很巧妙的地方，它把奇数的回文串和偶数的回文串统一起来考虑了。这一点一直是在做回文串问题中时比较烦的地方。这个算法还有一个很好的地方就是充分利用了字符匹配的特殊性，避免了大量不必要的重复匹配。

算法大致过程是这样。先在每两个相邻字符中间插入一个分隔符，当然这个分隔符要在原串中没有出现过。一般可以用‘#’分隔。这样就非常巧妙的将奇数长度回文串与偶数长度回文串统一起来考虑了（见下面的一个例子，回文串长度全为奇数了），然后用一个辅助数组P记录以每个字符为中心的最长回文串的信息。P [id] 记录的是以字符str [id] 为中心的最长回文串，当以str [id] 为第一个字符，这个最长回文串向右延伸了P [id] 个字符。

原串： w aa b w s w f d

新串： # w # a # a # b # w # s # w # f # d #

辅助数组P： 1 2 1 2 3 2 1 2 1 2 1 4 1 2 1 2 1 2 1

这里有一个很好的性质，P [id] -1就是该回文子串在原串中的长度（包括‘#’）。如果这里不是特别清楚，可以自己拿出纸来画一画，自己体会体会。当然这里可能每个人写法不尽相同，不过我想大致思路应该是一样的吧。

好，我们继续。现在的关键问题就在于怎么在O (n) 时间复杂度内求出P数组了。只要把这个P数组求出来，最长回文子串就可以直接扫一遍得出来了。

由于这个算法是线性从前往后扫的。那么当我们准备求P [i] 的时候，i以前的P [j] 我们是

阅读排行

Manacher算法--O(n)	(13818)
hdu3068回文串Manacher	(1838)
hdu 4027 - 线段树 -4	(892)
pku1095卡特兰数+递归	(876)
pku2761区间第k大数-二	(741)
hdu 4039 - 杂-1	(692)
为什么我们同时搞acm,	(643)
hdu 3948后缀树组-4	(642)
pku2140任意区间第k小	(600)
hdu3901 带通配符*和?	(599)

推荐文章

- * acegi security实践教程—把用户信息存放到数据库
- * 在Android中显示GIF动画
- * Ubuntu 下 Android Studio 开发工具使用详解
- * Android 4.4 Kitkat Phone工作流程浅析(三)_MO(去电)流程分析
- * 如何为豆瓣FM写一个chrome的歌词插件
- * 《使用Python进行自然语言处理》学习笔记五

最新评论

csdn再启动寄语
xuanflyer: @magicnumber: 赋害 your sister

csdn再启动寄语
magicnumber: 好赋害!

Manacher算法--O(n) 回文子串
tli600605: 其实当 $mx - i \leq P$ 时, p 只能为 $mx - i$ 。不会在往后扩展了。因为如果能扩展, 就意味着原来以 i ...

Manacher算法--O(n) 回文子串
grantbx: lz说的bug是比较字符时可能出现的越界问题吧

Manacher算法--O(n) 回文子串
grantbx: 通过记录已匹配的最右位置和对应对称中心来跳过一些没用的比较, 算法太赞了!

hdu3068回文串Manacher算法
xufeng01: 一定要 $str = \$$ 吗, 可不可以直接 $str = \#$, $str = s \ str = \#$

pku2104 第k大数-划分树做法
qqspeed: 90行的r2应该是之间被分到右子树的吧, 不是左子树, 左子树的就是l2;

hdu3887 树状数组--思维还是有
hnust_xiehonghao: mark 标记下

为什么我们同时搞acm, 结果是Ice_Crazy: 一语道破天机orz。。。

hdu3256计算几何2
liyongqiang000: 为什么不能用double直接输入呢, double的精度不是可以有15位有效数字么

已经得到了的。我们用 mx 记在 i 之前的回文串中, 延伸至最右端的位置。同时用 id 这个变量记下取得这个最优 mx 时的 id 值。(注: 为了防止字符比较的时候越界, 我在这个加了‘#’的字符串之前还加了另一个特殊字符‘\$’, 故我的新串下标是从1开始的)
好, 到这里, 我们可以先贴一份代码了。

复制代码

```
1.
void pk()
{
    int i;
    int mx = 0;
    int id;
    for(i=1; i<n; i++)
    {
        if( mx > i )
            p[i] = MIN( p[2*id-i], mx-i );
        else
            p[i] = 1;
        for(; str[i+p[i]] == str[i-p[i]]; p[i]++)
            ;
        if( p[i] + i > mx )
        {
            mx = p[i] + i;
            id = i;
        }
    }
}
```

代码是不是很短啊, 而且相当好写。很方便吧, 还记得我上面说的这个算法避免了很多不必要的重复匹配吧。这是什么意思呢, 其实这就是一句代码。

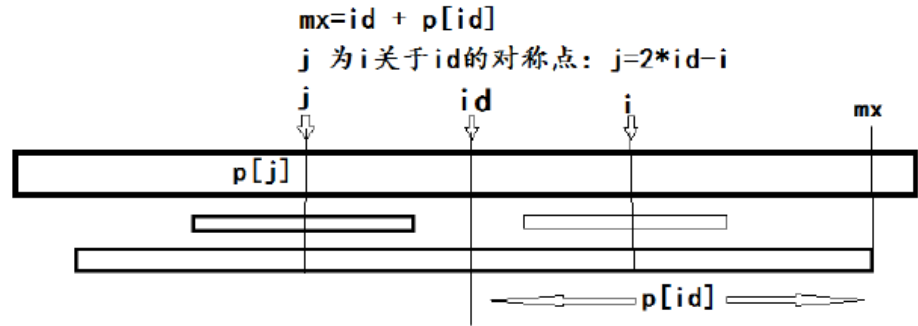
```
if( mx > i )
    p[i] = MIN( p[2*id-i], mx-i );
```

就是当前面比较的最远长度 $mx > i$ 的时候, $p[i]$ 有一个最小值。这个算法的核心思想就在这里, 为什么 p 数组满足这样一个性质呢?

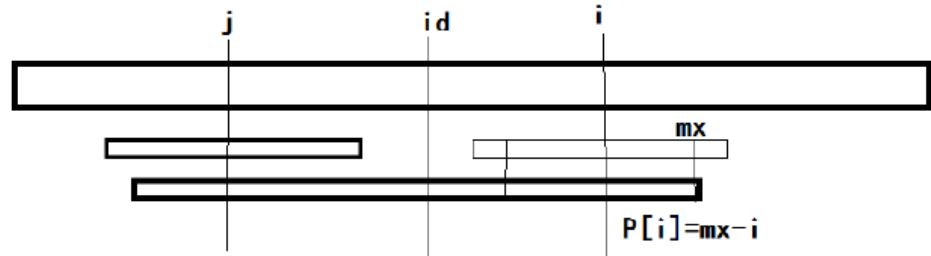


评论排行

- zsj3513 杂 (10)
- Manacher算法--O(n) (3)
- hdu3887 树状数组--思维 (3)
- hdu2222, 第一个AC自 (3)
- csdn再启动寄语 (2)
- pku3294-后缀数组-3 (2)
- 为什么我们同时搞acm, (2)
- hdu3068回文串Manacher (1)
- pku2104 第k大数-划分树 (1)
- hdu3256计算几何2 (1)



在这种情况下 $p[i]$ 默认 $\geq p[j]$, 因为回文串你把它左右翻转之后它还是回文的吧, 不然就不叫回文串了。这就是上面那个等式的前半部分。那么为什么还会有后半部分呢。



这就是为什么那个等式是取个最小值的原因了, 因为后面的那部分你可能还没有匹配过, 在这种情况下你怎么能保证 $p[i] \geq p[j]$ 呢。你还是得老老实实去匹配了才知道吧。

下面我们来大概看一看这个算法为什么是线性的。你会发现在上图中第一种情况下, 下一次比较一写是失败的, 而真正有意义的比较是从 mx 这一位开始的, 在这种情况下你可能会得到一些有相等的比较。而一旦匹配成功之后这个 mx 值是在不断增加的。所以这个算法总体是严格线性的。

看完这个算法, 你有可能会觉得这种算法在哪会用到呢? 其实回文串后缀数组也可以做。只是复杂度是 $O(n \log n)$ 的, 而且一般情况下也不会刻意去卡一个 $\log n$ 的算法。可正好hdu就有这么一题, 你用后缀数组写怎么都得T (当然应该是我写得太烂了)。不信的话大家也可以去试试这题。

<http://acm.hdu.edu.cn/showproblem.php?pid=3068>

另外, 顺便附一份AC代码。

<http://acm.hust.edu.cn:8080/judge/problem/viewSource.action?id=140283>

更多 1

上一篇: [pku3321树状数组](#)

下一篇: [hdu3068回文串Manacher算法](#)

相关主题推荐 算法 algorithm 源代码 string 博客

相关博文推荐

Combinations -- Leet...

归并排序算法的原理及JAVA实现

《提高C++性能的编程技术》总结

考研复习 (5) - 队列操作

cocos2d-x定时器机制


面试题总结15 自己构建一个哈希表

面试题总结14 动态规划

考研复习 (4) - 栈操作


查看评论

3楼 [iii600605](#) 2013-10-09 15:38发表




其实当 $mx - i \leq P[j]$ 时， $p[i]$ 只能为 $mx - i$ 。不会在往后扩展了。因为如果能扩展，就意味着原来以 id 为中心的回文串的边界 $str[id + p[id]]$ 和 $str[id - p[id]]$ 是相等的。那么原来求的 $p[id]$ 就会是错误的。

2楼 grantbx 2013-09-30 22:30发表



lz说的bug是比较字符时可能出现的越界问题吧

1楼 grantbx 2013-09-30 21:52发表



通过记录已匹配的最右位置和对应的对称中心来跳过一些没用的比较，算法太赞了！

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

专区推荐内容

#HTML(HTML5): 多媒体...
[C/C++]类别互相引用
超极本 和平板电脑 Window...
以用户为中心实现绝佳的用户体验
高性能计算相关资源
win8玩游戏不能全屏怎么办

<< >>

更多招聘职位

我公司职位也要出现在这里

核心技术类目

全部主题

Java VPN Android iOS ERP IE10 Eclipse CRM JavaScript Ubuntu NFC

WAP jQuery 数据库 BI HTML5 Spring Apache Hadoop .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 Spark HBase Pure Solr Angular Cloud Foundry Redis Scala Django

Bootstrap

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

客服1 客服2 微博客服 webmaster@csdn.net 400-600-2320

京 ICP 证 070598 号

北京创新乐知信息技术有限公司 版权所有

江苏乐知网络技术有限公司 提供商务支持

Copyright © 1999-2014, CSDN.NET, All Rights Reserved 