获取客户端真实IP winrar优化

Manacher's ALGORITHM: O(n)时间求字符串的最长回文子串

OCT 13

felix021 @ 2011-10-13 12:00 [IT » 程序设计] 评论(43) , 引用(0) , 阅读(115487) | Via 本站原创

大 | 中 | 小

Translated to ENGLISH VERSION

源于这两篇文章:
http://blog.csdn.net/ggggiqnypgjg/article/details/6645824
http://zhuhongcheng.wordpress.com/2009/08/02/a-simple-linear-time-algorithm-for-finding-longest-palindrome-sub-string/

这个算法看了三天，终于理解了，在这里记录一下自己的思路，免得以后忘了又要想很久- -.

首先用一个非常巧妙的方式，将所有可能的奇数/偶数长度的回文子串都转换成了奇数长度：在每个字符的两边都插入一个特殊的符号。比如 abba 变成 #a#b#b#a#， aba变成 #a#b#a#。 为了进一步减少编码的复杂度，可以在字符串的开始加入另一个特殊字符，这样就不用特殊处理越界问题，比如 $#a#b#a#（注意，下面的代码是用C语言写就，由于C语言规范还要求字符串末尾有一个'\0'所以正好OK，但其他语言可能会导致越界）。

下面以字符串12212321为例，经过上一步，变成了 S[] = "$#1#2#2#1#2#3#2#1#";

然后用一个数组 P[i] 来记录以字符S[i]为中心的最长回文子串向左/右扩张的长度（包括S[i]，也就是把该回文串"对折"以后的长度），比如S和P的对应关系：

```
S  #  1  #  2  #  2  #  1  #  2  #  3  #  2  #  1  #
P  1  2  1  2  5  2  1  4  1  2  1  6  1  2  1  2  1
(p.s. 可以看出，P[i]-1正好是原字符串中回文串的总长度）
```

那么怎么计算P[i]呢？该算法增加两个辅助变量（其实一个就够了，两个更清晰）id和mx，其中 id 为已知的 {右边界最大} 的回文子串的中心，mx则为id+P[id]，也就是这个子串的右边界。

然后可以得到一个非常神奇的结论，这个算法的关键点就在这里了：如果mx > i，那么P[i] >= MIN(P[2 * id - i], mx - i)。就是这个串卡了我非常久。实际上如果把它写得复杂一点，理解起来会简单很多：

```
//记j = 2 * id - i，也就是说 j 是 i 关于 id 的对称点(j = id - (i - id))
if (mx - i > P[j])
    P[i] = P[j];
```

联系我

i@felix021.com

最新评论

grin
为什么不用js写
faint
最近也在用golang写番羽算法..
惊了，09年！我给一个cpp版本..
Python的gc真的很坑
grin
按照大佬的方法，完美解决问..
谢谢大佬。
妙呀

最新日志

程序员面试指北：面试官视角..
使用pprof和go-torch排查gola..
记 python 超时的一个坑
goroutine调度"导致&quo..
从堆和栈开始的一些问题..
英语流利说 TED 课程
Golang rand.Rand 并发panic:..
vscode 的 tab
如何识别聪明和优秀的人？ ..
用 go channel 实现素数迭代..

分类

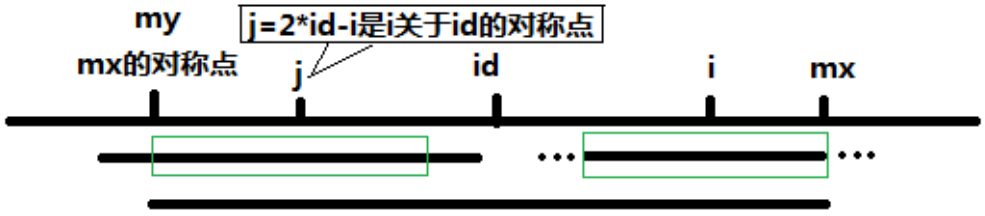
随想 [23] RSS
IT [767] RSS
» Blockchain [3] RSS
» 云 [3] RSS
» 操作系统 [134] RSS
» shell [2] RSS
» Python [14] RSS
» 探索设计模式 [5] RSS
» 软件 [93] RSS

```
else /* P[j] >= mx - i */
    P[i] = mx - i; // P[i] >= mx - i，取最小值，之后再匹配更新。
```

当然光看代码还是不够清晰，还是借助图来理解比较容易。

当 mx - i > P[j] 的时候，以S[j]为中心的回文子串包含在以S[id]为中心的回文子串中，由于 i 和 j 对称，以S[i]为中心的回文子串必然包含在以S[id]为中心的回文子串中，所以必有 P[i] = P[j]，见下图。

当 P[j] >= mx - i 的时候，以S[j]为中心的回文子串不一定完全包含于以S[id]为中心的回文子串中，但是基于对称性可知，下图中两个绿框所包围的部分是相同的，也就是说以S[i]为中心的回文子串，其向右至少会扩张到mx的位置，也就是说 P[i] >= mx - i。至于mx之后的部分是否对称，就只能老老实实去匹配了。



对于 mx <= i 的情况，无法对 P[i]做更多的假设，只能P[i] = 1，然后再去匹配了。

于是代码如下：

```
//输入，并处理得到字符串s
int p[1000], mx = 0, id = 0;
memset(p, 0, sizeof(p));
for (i = 1; s[i] != '\0'; i++) {
    p[i] = mx > i ? min(p[2*id-i], mx-i) : 1;
    while (s[i + p[i]] == s[i - p[i]]) p[i]++;
    if (i + p[i] > mx) {
        mx = i + p[i];
        id = i;
    }
}
//找出p[i]中最大的
```

OVER.

#UPDATE@2013-08-21 14:27
@zhengyuee 同学指出，由于 P[id] = mx，所以 S[id-mx] != S[id+mx]，那么当 P[j] > mx - i 的时候，可以肯定 P[i] = mx - i，不需要再继续匹配了。不过在具体

» 硬件 [43] RSS
» 手机 [12] RSS
» 程序设计 [170] RSS
» 网络 [203] RSS
» 数据库 [21] RSS
» 病毒 [8] RSS
» 其他 [56] RSS

其他

登入
注册
RSS： 日志 | 评论
编码：UTF-8
XHTML 1.0

统计

访问次数 4427118
今日访问 242
日志数量 791
评论数量 2454
引用数量 1
留言数量 151
注册用户 416
在线人数 34

链接

默认链接组

» WHU微软俱乐部
» 谁见过风？（国际版）

朋友们的据点

» FOUR
» 笨狗又一窝
» czyhd's Blog
» Kid的原创Blog
» [GCC]Feli
» dutor
» 不敢流泪
» Sheen
» 姜南的BLOG
» Liuw's Thinkpad
» 张磊的blog
» intijk
» PortWatcher's Blog

实现的时候即使不考虑这一点，也只是多一次匹配（必然会fail），但是却要多加一个分支，所以上面的代码就不改了。

转载请注明出自 **https://www.felix021.com/blog/read.php?2040** ，如是转载文则注明原出处，谢谢:)
RSS订阅地址: **http://www.felix021.com/blog/feed.php** 。

» 美德瑞钢琴

啊啊 2020-2-1 20:08

为什么不用js写

kjkjk 2019-11-11 04:37

1233211234567 ✉ 2019-6-15 11:11

ruibinhong ✉ 2019-4-21 14:53

给定输入 character,你的算法输出是3,4,得到的回文串是ara;但有的要求输出字符串里所有能构成的回文串的字符组合（保持相对顺序一致），并计算里面最长的，这种条件下最长的回文串是carac

> **felix021 回复于 2019-4-22 18:16**
>
> 嗯，那是另一个问题了，相对简单一些

sarag 2019-3-3 13:35


厉害了，看一遍就懂了

..... 2019-2-12 11:52

小火汁 2018-11-7 00:08

newbee

ahlixinjie 2018-10-28 10:14

如果没看懂的话建议看一下英文版，每一步怎么来怎么去的话写的非常清楚
https://articles.leetcode.com/longest-palindromic-substring-part-ii/

Entropy 2018-10-21 05:26

感谢楼主 果然一遍就懂~

dengjiangzhou 2018-10-14 14:35

倒着讲解的，讲的很好

3344aaxgl 2018-10-12 11:29

确实一遍就懂了

657657 2018-9-28 03:18

234324923432 2018-9-25 12:53

无 2018-8-22 21:14

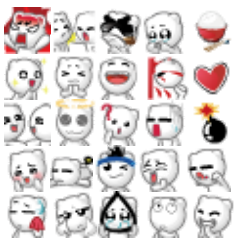
给跪 2018-8-20 07:50

大家都看懂了，厉害厉害，已放弃，打扰了 

分页： 1/3 1

发表评论

表情



昵称 密码 *非必须

网址 电邮 [注册]

B I U

- [ ] 打开HTML
- [ ] 打开UBB
- [x] 打开表情
- [ ] 隐藏
- [ ] 记住我

提交 重置