马拉车算法（Manacher's Algorithm）

acher's Algorithm）

读 8,542

7篇原创

马拉车算法，是一位名叫Manacher的人在1975年提出的一种算，神奇之处在于将算法的时间复杂度精进到了`O(N)`，下面我们

01 算法由来

在求解最长回文子串的问题时，一般的思路是以当前字符为中心，向其左右两边扩展寻找回文，但是这种解法的时间复杂度是O(N^2)，那么能不能将时间复杂度再降低一点？做到线性？马拉车算法就完美地解决了这个问题。

02 预处理

回文字符串以其长度来分，可以分为奇回文（其长度为奇数）、偶回文（其长度为偶数），一般情况下需要分两种情况来寻找回文，马拉车算法为了简化这一步，对原始字符串进行了处理，在每一个字符的左右两边都加上特殊字符（肯定不存在于原字符串中的字符），让字符串变成一个奇回文。例如：

原字符串：abba，长度为4
预处理后：#a#b#b#a#，长度为9

原字符串：aba，长度为3

以字符串`"cabbaf"`为例，将预处理后的新字符串`"#c#a#b#b#a#f#"`变成一个字符数组arr，定义一个辅助数组`int[] p`，`p`的长度与`arr`等长，`p[i]`表示以`arr[i]`字符为中心的最长回文半径，`p[i]=1`表示只有`arr[i]`字符本身是回文子串。

```
i      0 1 2 3 4 5 6 7 8 9 10 11 12
arr[i] # c # a # b # b # a #  f  #
p[i]   1 2 1 2 1 2 5 2 1 2 1  2  1
```

我们来比对分下一下最长回文半径和原字符串之间的关系。在上面例子中，最长回文子串是`"#a#b#b#a#"`，它以arr[6]为中心，半径是5，其代表的原始字符串是`"abba"`，而`"abba"`的长度为4，可以通过5减去1得到，是字符串`"cabbaf"`中的最长回文子串，那么我们是不是可以得出最长回文半径和最长回文子串长度之间的关系？

让我们再多看几个例子，如`"aba"`，转换后是`"#a#b#a#"`，以字符`'b'`为中心的回文，半径是4，减1得到3，3是原字符串的最长回文子串长度。

再例如`"effe"`，转换后是`"#e#f#f#e#"`，以最中间的'#'为中心的回文，半径是5，减1得到4，4是原字符串的最长回文子串长度。

因此，最后我们得到最长回文半径和最长回文子串长度之间的关系：`int maxLength = p[i]-1`。`maxLength`表示最长回文子串长度。

04 计算最长回文子串起始索引

知道了最长回文子串的长度，我们还需要知道它的起始索引值，这样才能截取出完整的最长回文子串。

继续以第三步中的字符串`"cabbaf"`为例，`p[6]=5`，是最长半径，用6(i)减去最长半径5(p[i])得到1，而1恰好是最长回文子串"abba"的起始索引。

我们再来看一个奇回文的例子。例如`"aba"`，转换后是`"#a#b#a#"`，p[3]=4，最长半径是4，i为3，用i减去4得到-1，数组下标越界了。

在偶回文的情况下，可以满足i减最长半径，而奇回文却会下标越界，我们需要在转换后的字符串前面再加一个字符，解决下标越界的问题，不能是`'#'`，那就加个`'$'`字符吧，但是加过一个字符后，字符串的长度不是奇数了，只能在尾部再加一个不会重复出现的字符，比如`'@'`，这样字符串的长度依旧是奇数了，满足前面第三部分的条件。

加多一个字符后，奇回文可以正常做减法了，偶回文呢？

```
i      0 1 2 3 4 5 6 7 8 9 10 11 12 13
arr[i] $ # c # a # b # b # a  #  f  #
p[i]     1 2 1 2 1 2 5 2 1 2  1  2  1
```

在补上字符`'$'`后，p[7]=5，用i减去最长半径，7-5=2，而理想的结果应该是1，那就再除以2吧，这样就能得到1了。而奇回文"aba"在用i减去最长半径后得到的是0，除以2后还是0，可以完美解决下标越界的问题。

结论：最长回文子串的起始索引`int index = (i - p[i])/2`。

05 计算p数组

在第三步和第四步中我们都用到了一个关键对象p数组，存放的是最长回文子串半径，那么它是怎么来的呢？
还是以上面的例子配合着看，

```
i      0 1 2 3 4 5 6 7 8 9 10 11 12 13 14
arr[i] $ # c # a # b # b # a  #  f  #  @
p[i]     1 2 1 2 1 2 5 2 1 2  1  2  1
```

设置两个变量id和mx，id是所有回文子串中，能延伸到最右端位置的那个回文子串的中心点位置，mx是该回文串能延伸到的最右端的位置。

当i等于7时，id等于7，p[id] = 5，在以位置7为中心的回文子串中，该回文子串的右边界是位置12。

当i等于12时，id等于12，p[id] = 2，在以位置12为中心的回文子串中，该回文子串的右边界是位置14。

由此我们可以得出回文子串右边界和其半径之间的关系：`mx = p[id]+id`。

image

因为回文字符串是中心对称的，知道中心点位置id，如果一个位置的回文子串以i为中心，并且包含在以id为中心的回文子串中，即 `mx > i`，那么肯定会存在另外一个以j为中心回文子串，和以i为中心的回文子串相等且对称，即 `p[j] = p[i]`，而i和j是以id为中心对称，即 `i+j=2*id`，如果知道了i的值，那么 `j = 2*id - i`。

但是我们需要考虑另外一种情况，如果存在一个以i为中心的回文子串，依旧有mx > i，但是以i为中心的回文子串右边界超过了mx，在i到mx的这段回文子串中，与另一端对称的以j为中心的回文子串还是相等的，此时`p[i] = mx - i`，`p[j] = [pi]`，至于右边界mx之外的子串，即以i为中心的回文子串超出的部分是否还是满足上述条件就需要遍历比较字符了。

因此，在`mx > i`的情况下，`p[i] = Math.min(p[2*id - i], mx - i)`。
另外如果i大于mx了，也即是边界mx后面的子串，依旧需要去比较字符计算。

```
public static String Manacher(String s) {
    if (s.length() < 2) {
        return s;
    }
    // 第一步：预处理，将原字符串转换为新字符串
    String t = "$";
    for (int i=0; i<s.length(); i++) {
        t += "#" + s.charAt(i);
    }
    // 尾部再加上字符@，变为奇数长度字符串
    t += "#@";
    // 第二步：计算数组p、起始索引、最长回文半径
    int n = t.length();
    // p数组
    int[] p = new int[n];
    int id = 0, mx = 0;
    // 最长回文子串的长度
    int maxLength = -1;
    // 最长回文子串的中心位置索引
```

```
    int index = 0;
    for (int j=1; j<n-1; j++) {
        // 参看前文第五部分
        p[j] = mx > j ? Math.min(p[2*id-j], mx-j) : 1;
        // 向左右两边延伸，扩展右边界
        while (t.charAt(j+p[j]) == t.charAt(j-p[j])) {
            p[j]++;
        }
        // 如果回文子串的右边界超过了mx，则需要更新mx和id的值
        if (mx < p[j] + j) {
            mx = p[j] + j;
            id = j;
        }
        // 如果回文子串的长度大于maxLength，则更新maxLength和index的值
        if (maxLength < p[j] - 1) {
            // 参看前文第三部分
            maxLength = p[j] - 1;
            index = j;
        }
    }
    // 第三步：截取字符串，输出结果
    // 起始索引的计算参看前文第四部分
    int start = (index-maxLength)/2;
    return s.substring(start, start + maxLength);
}
```

06 小结

马拉车算法将求解最长回文子串的时间复杂度降低到了`O(N)`，虽然也牺牲了部分空间，其空间复杂度为`O(N)`，但是其算法的巧妙之处还是值得学习和借鉴的。

算法专题目前已连续日更超过六个月，算法题文章211+篇，公众号对话框回复【数据结构与算法】、【算法】、【数据结构】中的任一关键词，获取系列文章合集。

以上就是全部内容，如果大家有什么好的解法思路、建议或者其他问题，可以下方留言交流，好看、留言、转发就是对我最大的回报和支持！

程序员小川 公众号：程序员小川，现Java程序员，终身学习践行者，专注技术、认知、学习、理财，技术人不1
总资产146 (约11.77元) 共写了22.9W字 获得869个赞 共2,843个粉丝

全部评论 2

只看作者

按时间倒序

按时间正序

Dashy

4楼 02.15 10:00

谢谢分享，只是有一点瑕疵，第四部分末尾用到的数据和第五部分开始用到的数据不一致，少了末尾的@吧?

赞 回复

徐绍深

3楼 01.04 14:26

清晰透彻，赞

赞 回复

被以下专题收入，发现更多相似内容

收入我的专题

算法 算法提高之Le... 程序园 数据结构和算法分析 Android进阶

推荐阅读更多精彩内容

经典算法问题：最长回文子串之 Manacher 算法

title: 经典算法问题：最长回文子串之 Manacher 算法date: 2019-02-17 08:00:0...

李威威 阅读 403 评论 0 赞 2

经典算法问题：最长回文子串之 Manacher 算法

维基百科中对于“最长回文子串”介绍如下。 在计算机科学中，最长回文子串或最长对称因子问题是在一个字符

串中查找一个最...

李威威 阅读 101 评论 0 赞 1

HDU - 3068 最长回文（马拉车算法模板题）

HDU原题链接：传送门 最长回文 Time Limit: 4000/2000 MS (Java/Others) ...

萌级樱花草 阅读 385 评论 0 赞 0

Manacher算法--最长回文 HDU - 3068

工具：char Ma[2maxn];//作为一个转化数组int Mp[maxn2];//该点为中点时的半径int ...

laochonger 阅读 174 评论 0 赞 0

经典问题与算法：最长回文子串问题与Manacher算法

问题描述：给定一个字符串，求出其最长回文子串的长度例如：对于字符串s="acaacdbab"而言，其回文子串分别为...

进击的Lancelot 阅读 833 评论 0 赞 1