

java实现KMP 算法

转载

闵浮龙

2019-10-17 16:38:37

1882

收藏 11

版权

分类专栏: 算法

一、应用场景-字符串匹配问题

字符串匹配问题:

1. 有一个字符串 str1= "“硅硅谷 尚硅谷你尚硅 尚硅谷你尚硅谷你尚硅你好”", 和一个子串 str2="尚硅谷你尚硅你"
2. 现在要判断 str1 是否含有 str2, 如果存在, 就返回第一次出现的位置, 如果没有, 则返回-1

二、暴力匹配算法

如果用暴力匹配的思路, 并假设现在 str1 匹配到 i 位置, 子串 str2 匹配到 j 位置, 则有:

1. 如果当前字符匹配成功(即str1[i]==str2[j]), 则i++, j++, 继续匹配下一个字符
2. 如果失配(即str1[i]!=str2[j]), 令i=i-(j-1), j=0。相当于每次匹配失败时, i回溯, j被置为0。
3. 用暴力方法解决的话就会有大量的回溯, 每次只移动一位, 若是不匹配, 移动到下一位接着判断, 浪费了大量时间。(不可行!)
4. 暴力匹配算法实现.
5. 代码

```
1 package kmp;
2
3 /**
4  * @program: text
5  * @description: 暴力匹配算法
6  * @author: min
7  * @create: 2019-10-16 17:26
8  */
9 public class ViolenceMatch {
10     public static void main(String[] args) {
11         String str1 = "硅硅谷 尚硅谷你尚硅 尚硅谷你尚硅谷你尚硅你好";
12         String str2 = "尚硅谷你尚硅你~";
13         int index = violenceMatch(str1, str2);
14         System.out.println("index=" + index);
15     }
16
17     /**
18      * 暴力匹配算法实现
19      *
20      * @param str1
21      * @param str2
22      * @return
23      */
24     private static int violenceMatch(String str1, String str2) {
25         char[] s1 = str1.toCharArray();
26         char[] s2 = str2.toCharArray();
27
28         int s1Len = s1.length;
29         int s2Len = s2.length;
30
31         int i = 0; // i索引指向s1
32         int j = 0; // j索引指向s2
```

点赞2

评论2

分享

收藏11

举报

关注

一键三连

```
33
34     while (i < s1Len && j < s2Len) { // 保证匹配时, 不越界
35         if (s1[i] == s2[j]) { // 匹配 ok
36             i++;
37             j++;
38         } else { // 没有匹配成功
39             // 如果失配(即 str1[i] != str2[j]), 令 i = i - (j - 1), j = 0。
40             i = i - (j - 1);
41             j = 0;
42         }
43     }
44
45     // 判断是否匹配成功
46     if (j == s2Len) {
47         return i - j;
48     } else {
49         return -1;
50     }
51 }
52 }
53
```

三、KMP 算法介绍

1. KMP 是一个解决模式串在文本串是否出现过, 如果出现过, 最早出现的位置的经典算法
2. Knuth-Morris-Pratt 字符串查找算法, 简称为“KMP 算法”, 常用于在一个文本串 S 内查找一个模式串 P 的出现位置, 这个算法由 Donald Knuth、Vaughan Pratt、James H. Morris 三人于 1977 年联合发表, 故取这 3 人的姓氏命名此算法。
3. KMP 方法算法就利用之前判断过信息, 通过一个 next 数组, 保存模式串中前后最长公共子序列的长度, 每次回溯时, 通过 next 数组找到, 前面匹配过的位置, 省去了大量的计算时间

四、KMP 算法最佳应用-字符串匹配问题

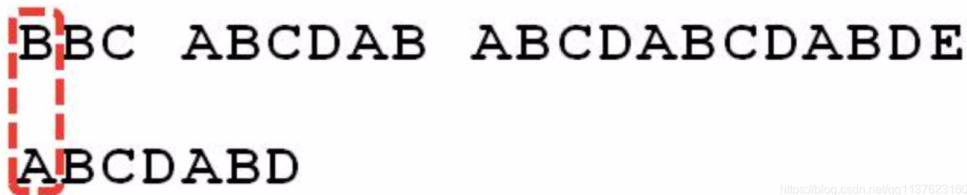
字符串匹配问题:

1. 有一个字符串str1="BBCABCDABABCDABCDABDE", 和一个子串str2="ABCDABD"
2. 现在要判断 str1 是否含有 str2, 如果存在, 就返回第一次出现的位置, 如果没有, 则返回-1
3. 要求:使用KMP算法完成判断, 不能使用简单的暴力匹配算法。

思路分析图解

举例来说, 有一个字符串 Str1 = "BBC ABCDAB ABCDABCDABDE", 判断, 里面是否包含另一个字符串 Str2 = "ABCDABD"?

1. 首先, 用 Str1 的第一个字符和 Str2 的第一个字符去比较, 不符合, 关键词向后移动一位

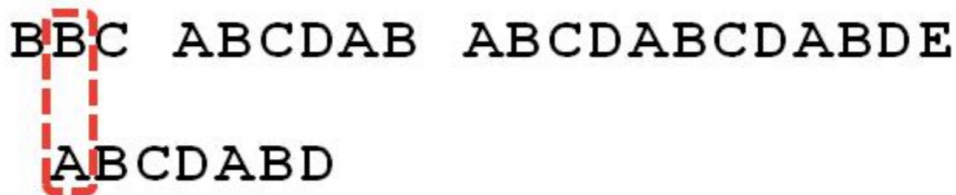


BBC ABCDAB ABCDABCDABDE

ABCDABD

<https://blog.csdn.net/qq1137623160>

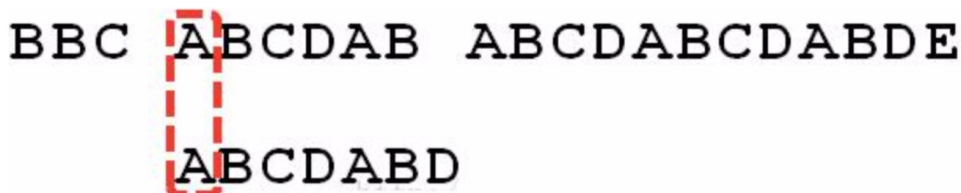
2. 重复第一步, 还是不符合, 再后移



BBC ABCDAB ABCDABCDABDE
ABCDABD

<https://blog.csdn.net/q1137623160>

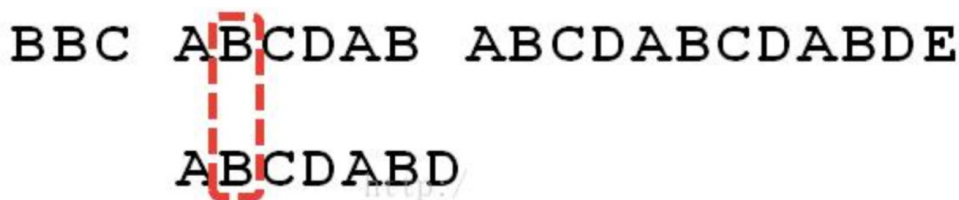
3. 一直重复，直到Str1有一个字符与Str2的第一个字符符合为止



BBC ABCDAB ABCDABCDABDE
ABCDABD

<https://blog.csdn.net/q1137623160>

4. 接着比较字符串和搜索词的下一个字符，还是符合。



BBC ABCDAB ABCDABCDABDE
ABCDABD

<https://blog.csdn.net/q1137623160>

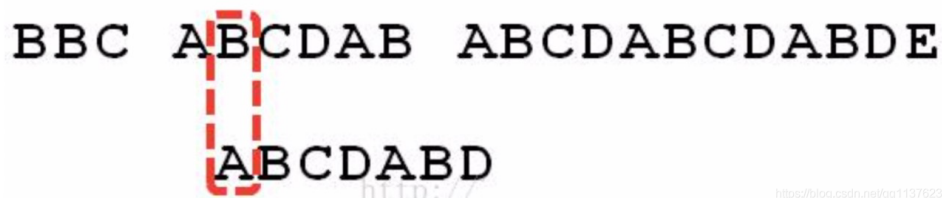
5.遇到 Str1 有一个字符与 Str2 对应的字符不符合。



BBC ABCDAB ABCDABCDABDE
ABCDABD

<https://blog.csdn.net/q1137623160>

6.这时候，想到的是继续遍历 Str1 的下一个字符，重复第 1 步。(其实是很不明智的，因为此时 BCD 已经比较过了，没有必要再做重复的工作，一个基本事实是，当空格与 D 不匹配时，你其实知道前面六个字符是"ABCDAB"。KMP 算法的想法是，设法利用这个已知信息，不要把"搜索位置"移回已经比较过的位置，继续把它向后移，这样就提高了效率。)



BBC ABCDAB ABCDABCDABDE
ABCDABD

<https://blog.csdn.net/q1137623160>

7.怎么做到把刚刚重复的步骤省略掉?可以对 Str2 计算出一张《部分匹配表》，这张表的产生在后面介绍

搜索词	A	B	C	D	A	B	D
部分匹配值	0	0	0	0	1	2	0

8.已知空格与 D 不匹配时，前面六个字符"ABCDAB"是匹配的。查表可知，最后一个匹配字符 B 对应的"部分匹配值"为 2，因此按照下面的公式算出向后移动的位数：

移动位数 = 已匹配的字符数 - 对应的部分匹配值因为 $6 - 2$ 等于 4，所以将搜索词向后移动 4 位。

9.因为空格与C不匹配，搜索词还要继续往后移。这时，已匹配的字符数为 2("AB")，对应的"部分匹配值"为 0。所以，移动位数 = $2 - 0$ ，结果为 2，于是将搜索词向后移 2 位。

BBC ABCDAB ABCDABCDABDE
ABCDABD

10.因为空格与 A 不匹配，继续后移一位。

BBC ABCDAB ABCDABCDABDE
ABCDABD

11.逐位比较，直到发现 C 与 D 不匹配。于是，移动位数 = $6 - 2$ ，继续将搜索词向后移动 4 位。

BBC ABCDAB ABCDABCDABDE
ABCDABD

12.逐位比较，直到搜索词的最后一位，发现完全匹配，于是搜索完成。如果还要继续搜索(即找出全部匹配)，移动位数 = $7 - 0$ ，再将搜索词向后移动 7 位，这里就不再重复了。

BBC ABCDAB ABCDABCDABDE
ABCDABD

13.介绍《部分匹配表》怎么产生的 先介绍前缀，后缀是什么

字符串：**“ bread ”**

前缀：**b , br , bre , brea**

后缀：**read , ead , ad , d**

<https://blog.csdn.net/qq1137623160>

“部分匹配值”就是“前缀”和“后缀”的最长的共有元素的长度。以“ABCDABD”为例，-“A”的前缀和后缀都为空集，共有元素的长度为 0；-“AB”的前缀为[A]，后缀为[B]，共有元素的长度为 0；

-“ABC”的前缀为[A, AB]，后缀为[BC, C]，共有元素的长度 0；

-“ABCD”的前缀为[A, AB, ABC]，后缀为[BCD, CD, D]，共有元素的长度为 0；

-“ABCDAB”的前缀为[A, AB, ABC, ABCD]，后缀为[BCDAB, CDAB, DA, A]，共有元素为“A”，长度为 1；-“

ABCDAB”的前缀为[A, AB, ABC, ABCD, ABCDA]，后缀为[BCDAB, CDAB, DAB, AB, B]，共有元素为“AB”，长度为 2；

-“ABCDABD”的前缀为[A, AB, ABC, ABCD, ABCDA, ABCDAB]，后缀为[BCDABD, CDABD, DABD, ABD, BD, D]，共有元素的长度为 0。

14.“部分匹配”的实质是，有时候，字符串头部和尾部会有重复。比如，“ABCDAB”之中有两个“AB”，那么它的“部分匹配值”就是 2(“AB”的长度)。搜索词移动的时候，第一个“AB”向后移动 4 位(字符串长度- 部分匹配值)，就可以来到第二个“AB”的位置。

搜索词	A	B	C	D	A	B	D
部分匹配值	0	0	0	0	1	2	0

<https://blog.csdn.net/qq1137623160>

到此 KMP 算法思想分析完毕!

五、代码实现

```

1 package kmp;
2
3 import java.util.Arrays;
4
5 /**
6  * @program: text
7  * @description: kmp算法
8  * @author: min
9  * @create: 2019-10-17 16:08
10 */
11 public class KMPAlgorithm {
12     public static void main(String[] args) {
13         String str1 = "BBC ABCDAB ABCDABCDABDE";
14         String str2 = "ABCDABD";
15         //String str2 = "BBC";
16         int[] next = kmpNext("ABCDABD"); //[0, 1, 2, 0]
17         System.out.println("next=" + Arrays.toString(next));
18         int index = kmpSearch(str1, str2, next);
19         System.out.println("index=" + index); // 15 了
20
21     }

```

点赞2

评论2

分享

收藏11

举报

关注

一键三连

```
22
23 /**
24  * @param str1 源字符串
25  * @param str2 子串
26  * @param next 部分匹配表，是子串对应的部分匹配表
27  * @return 如果是-1 就是没有匹配到，否则返回第一个匹配的位置
28  */
29 public static int kmpSearch(String str1, String str2, int[] next) {
30     //遍历
31     for (int i = 0, j = 0; i < str1.length(); i++) {
32         //需要处理 str1.charAt(i) != str2.charAt(j), 去调整 j 的大小
33         //KMP 算法核心点，可以验证...
34         while (j > 0 && str1.charAt(i) != str2.charAt(j)) {
35             j = next[j - 1];
36         }
37
38         if (str1.charAt(i) == str2.charAt(j)) {
39             j++;
40         }
41         if (j == str2.length()) { //找到了 // j = 3 i
42             return i - j + 1;
43         }
44     }
45     return -1;
46 }
47
48 //获取到一个字符串(子串) 的部分匹配值表
49 public static int[] kmpNext(String dest) {
50     //创建一个 next 数组保存部分匹配值
51     int[] next = new int[dest.length()];
52     next[0] = 0; //如果字符串是长度为 1 部分匹配值就是 0
53     for (int i = 1, j = 0; i < dest.length(); i++) {
54         //当 dest.charAt(i) != dest.charAt(j) , 我们需要从 next[j-1]获取新的 j
55         //直到我们发现 有 dest.charAt(i) == dest.charAt(j)成立才退出
56         //这时 kmp 算法的核心点
57         while (j > 0 && dest.charAt(i) != dest.charAt(j)) {
58             j = next[j - 1];
59         }
60
61         //当 dest.charAt(i) == dest.charAt(j) 满足时，部分匹配值就是+1
62         if (dest.charAt(i) == dest.charAt(j)) {
63             j++;
64         }
65         next[i] = j;
66     }
67     return next;
68 }
69 }
70 }
```

转载至：尚硅谷_韩顺平_图解Java数据结构和算法.pdf

《C语言/C++学习指南》加密解密篇（安全相关算法）

07-07

本套视频教程介绍加密解密相关的常见算法，指出每种算法的应用场景，并给出使用示例。具体包含：（1）数据转换...

KMP算法（研究总结，字符串）

dipinzhu4111的博客 2851

KMP算法（研究总结，字符串） 前段时间学习KMP算法，感觉有些复杂，不过好歹是弄懂啦，简单地记录一下，方...



优质评论可以帮助作者获得更高权重



评论



Inmaturity_7: 一看博主就是看的尚硅谷韩老师讲解的视频，在B站，hhh 2月前 回复 ...



COMEFOR: 感谢，讲得非常详细。 3月前 回复 ...

点赞2

评论2

分享

收藏11

举报

关注

一键三连

相关推荐

- java算法篇KMP算法及应用_Just Do It!

3-3

java算法篇KMP算法及应用 算法背景 给定两个字符串,判断是否一个字符串包含另外一个字符串,如果包含,返回起始...
- KMP算法的Java实现(基于阮一峰的博客)_u010698087的博客

3-14

KMP算法的Java实现(基于阮一峰的博客) 这个算法也看了大半天了,仔细看过两个人的博客,一个是传说中的Matrix67,...
- Java实现KMP算法

TKD03072010的专栏 1万+

package arithmetic; /** * Java实现KMP算法 * 思想: 每当一趟匹配过程中出现字符比较不等, 不需要回溯指针, * ...
- KMP (Java实现)

V_0617的博客 3648

看了一下 阮一峰 老师的KMP算法的讲解, 感觉终于对这个算法有了点理解了。于是就用java实现了一下。下面讲解...
- ...KMP算法及java实现_一名普通码农的菜地_kmp算法java...

3-18

下面我将java版的kmp算法(有点简陋)贴出来: package kmp; import java.util.ArrayList; public class KMPTest{ public stati...
- KMP算法JavakMP算法JavakMP算法JavakMP算法Java_kmp算法java,java...

3-22

java实现KMP 算法 1740一、应用场景-字符串匹配问题字符串匹配问题: 有一个字符串 str1= ““硅谷 尚硅谷你尚硅 ...
- 阮一峰: 字符串匹配的KMP算法

Freewind的专栏 1110

字符串匹配是计算机的基本任务之一。 举例来说, 有一个字符串"BBC ABCDAB ABCDABCDABDE", 我想知道...
- 从头到尾彻底理解KMP

omnispace的博客 1666

从头到尾彻底理解KMP 作者: July 时间: 最初写于2011年12月, 2014年7月21日晚10点 全部删除重写成此文, 随后...
- KMP算法--java实现_stevia829的博客

3-11

KMP算法-java实现 与暴力算法的不同:暴力算法匹配不通过的时候就回到最前面 KMP算法引入部分匹配函数的概念 ...
- 字符串匹配-KMP算法 讲解与java代码实现_bury_的博客-C...

3-19

可以使用KMP算法,首先计算字符串s的模式偏移数组next,然后在遍历a查找s的时候可以利用next偏移数组对s进行偏...
- java kmp算法_KMP算法-Java实现

weixin_35582019的博客 6

目的: 为了解决字符串模式匹配历程: 朴素模式匹配: 逐次进行比较KMP算法: 利用匹配失败得到的信息, 来最大限...
- JAVA实现KMP模式匹配算法

菜鸟 1889

获取next()数组 /** * 获取next数组 * data 主字符串 */ public static int[] getNext(String data){ int[] next=new int[data.l...
- KMP算法详解及Java实现_banche163的专栏_java实现kmp算法

2-28

KMP算法详解及Java实现 以前在学习计算机数据结构时,涉及到基础算法KMP算法,学习了好几次,在网上找了很多资...
- JAVA实现KMP算法_javakmp算法,kmp算法java实现-Java其他资源-CSDN...

3-21

JAVA实现KMP算法,使用java语言实现的kmp算法javakmp算法更多下载资源、学习资料请访问CSDN下载频道.
- Java实现KMP算法 (代码)

Novayue的博客 1134

package kmp; /** * @Description 研究kmp算法 * @author daixiaoyong * @date 2019年3月6日 下午5:10:30 */ public...
- KMP算法—终于全部弄懂了

dark_cy的博客 21万+

详细讲解KMP算法, 并对难点 k=next[k] 这条语句重点描述
- 字符串匹配 (KMP) 算法及Java实现

weixin_34001430的博客 535

一、什么是KMP算法? 维基百科的解释是: 在计算机科学中, Knuth-Morris-Pratt字符串查找算法 (简称为KMP算法...
- java kmp算法_KMP算法java版实现

weixin_36132740的博客 11

import java.util.Arrays;public class KMP {private static int[] prefixTable;/** 部分匹配表* @param t* @return*/public i...
- 用Java实现KMP算法

行作笔、心当墨 134

KMP算法是一种非常出名的字符串匹配算法, 其核心思想在于: 当一趟匹配过程中出现字符不匹配时, 不需要回溯主...
- java实现kmp_java 实现KMP算法

weixin_34600909的博客 23

KMP算法是一种神奇的字符串匹配算法, 在对 超长字符串 进行模板匹配的时候比暴力匹配法的效率高不少。接下...
- java KMP 字符串匹配算法

Sun_Ru的博客 899

KMP算法的关键是利用匹配失败后的信息, 尽量减少模式串与主串的匹配次数以达到快速匹配的目的。具体实现就是...
- javaKMP算法

wuhen0616的博客 267

中午吃饭前发了一次, 但吃饭时想到getNext方法有问题。就删了之

1.字符串匹配问题 1.有一个字符串str1 = “硅硅古 尚硅谷你尚硅谷 尚硅谷你尚硅谷你尚硅你好”，和一个字符串str2 = “...



闵浮龙

码龄4年

上海钧正网络科技有...

65

10万+

1万+

45万+



原创

周排名

总排名

访问

等级

4902

128

398

61

288

积分

粉丝

获赞

评论

收藏

私信

关注

搜博文文章



热门文章

- java面试——Hibernate常见面试题 52592
- java面试——springMVC面试题 26017
- 判断StringBuffer是否为空 24644
- 配置redis外网可访问,并只允许指定的ip可访问redis 23728
- FutureTask介绍及使用 19941

分类专栏

-  计算机是如何跑起来的 1篇
-  那些年趟过的坑 38篇
-  感悟 49篇
-  思绪 5篇
-  nginx 8篇
-  面试 13篇

最新评论

- java面试——Hibernate常见面试题
Tisfy: 正如大音希声扫阴霾
- com.mysql.jdbc.exceptions.jdbc4.Comm...
Xiao_yuer_q: 多半是因为两个工程共用了
一个端口的缘故，把db.properties里面的 ...
- java字符串大写转小写，小写转大写
lunvey: 学习了,原来java是这样子的，谢
谢！

Inmaturity_7: 一看博主就是看的尚硅谷韩老师讲解的视频，在B站，hhh

make: *** 没有规则可以创建目标“linux”...

numlock1350: 字不多却铿锵有力

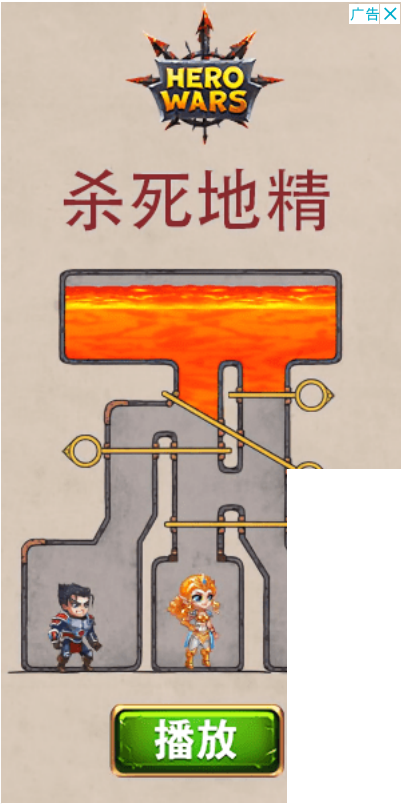
最新文章

Excel表格快速隐藏手机号码中间4位数字，方法就这5种！

一、计算机的三大原则

maven的离线模式

2021年 1篇	2020年 8篇
2019年 33篇	2018年 83篇
2017年 90篇	



目录

- 一、应用场景-字符串匹配问题
- 二、暴力匹配算法
- 三、KMP 算法介绍
- 四、KMP 算法最佳应用-字符串匹配问题
- 字符串匹配问题:
- 思路分析图解
- 五、代码实现