KMP 哈希

zty

zucc ACM Group

2022.01.28

目录

目录

KMP

- KMP算法是一种字符串匹配算法

KMP

- KMP算法是一种字符串匹配算法
- 主串长为n，子串长为m，用朴素的匹配算法就要O（n*m）的时间
  主串为s，要匹配的子串为t；
  for(int i = 0; i < s.size(); ++i)
  for (int j = 0; j < t.size(); ++j)

KMP

- KMP算法是一种字符串匹配算法
- 主串长为n，子串长为m，用朴素的匹配算法就要O（n*m）的时间
  主串为s，要匹配的子串为t；
  for(int i = 0; i < s.size(); ++i)
  for (int j = 0; j < t.size(); ++j)
- KMP算法就可以降到O(n+m)，KMP算法主要就是找一个最长公共前后缀
  一个字符串abcdef
  前缀是a，ab，abc，abcd，abcde，abcdef
  后缀就是f，ef，def，cdef，bcdef，abcdef
  对于字符串abcabc
  遍历他，在a的时候最长公共前后缀是a，ab的时候没有，abc也没有，abca就是a因为前面和后面都有a，abcab就是ab，abcabc就是abc

- 你abcabcabd和abcabd匹配

- 你abcabcabd和abcabd匹配
- abcabcabd
  abcabd s[5]!=t[5]你就不需要回溯到第1个位置

- 你abcabcabd和abcabd匹配
- abcabcabd
  abcabd s[5]!=t[5]你就不需要回溯到第1个位置
- abcabcabd
  ……abcabd

- 你abcabcabd和abcabd匹配
- abcabcabd
  abcabd s[5]!=t[5]你就不需要回溯到第1个位置
- abcabcabd
  ……abcabd
- 只需要比较10次而朴素的要14次

- 你abcabcabd和abcabd匹配
- abcabcabd
  abcabd s[5]!=t[5]你就不需要回溯到第1个位置
- abcabcabd
  ……abcabd
- 只需要比较10次而朴素的要14次
- 那么现在要解决的就是怎么去求这个next数组，next数组表示的不仅是最长公共子序列的长度还是你匹配失败时要回溯到的位置，比如说next[5]就是2

KMP

- abcabcabd

KMP

- abcabcabd
- 我们现在手动求一下这个next数组，比较第一个b失败的时候你就要去比较这个字符失败是否第一个a相同，所以next[1] = 0;你第一个c和第二个a比较失败时也是要比较第一个a，next[2] = next[3] = 0;当你第二个b失配的时候你可以知道你已经有一个a与模式串匹配了,所以next[4] = 1;next[5] = 2;next[6] = 3; next[7] = 4;next[8] = 5;

KMP

- abcabcabd
- 我们现在手动求一下这个next数组，比较第一个b失败的时候你就要去比较这个字符失败是否第一个a相同，所以next[1] = 0;你第一个c和第二个a比较失败时也是要比较第一个a，next[2] = next[3] = 0;当你第二个b失配的时候你可以知道你已经有一个a与模式串匹配了,所以next[4] = 1;next[5] = 2;next[6] = 3; next[7] = 4;next[8] = 5;
- 那么程序该怎么写呢，我们可以用两个变量i，j分别指向前缀和后缀
  一开始i=-1，j=0，初始next[0]=-1因为第一个字符前面没东西，i=-1就意味着现在没有公共前后缀，所以next[j+1] = 0;而
  当s[i]==s[j] 的时候表面前后缀匹配成功了，那么当这个位置后一个位置失配的时候前面也有i个已经成功匹配,所以next[j+1]=i+1 匹配失败的时候next[i] = i;

目录

- 哈希其实就类似于进制转换

- 哈希其实就类似于进制转换
- 1111B转10进制就是 $1 + 1 * 2 + 1 * 4 + 1 * 8 = 15$

- 哈希其实就类似于进制转换
- 1111B转10进制就是 $1 + 1 * 2 + 1 * 4 + 1 * 8 = 15$
- 字符串哈希就改成p进制，这个p一般取素数（131，13331），但是你这么乘肯定会爆，所以就在对1e9+7取个模,p的指数次也可以预处理存在数组里

- 哈希其实就类似于进制转换
- 1111B转10进制就是 $1 + 1 * 2 + 1 * 4 + 1 * 8 = 15$
- 字符串哈希就改成p进制，这个p一般取素数（131，13331），但是你这么乘肯定会爆，所以就在对1e9+7取个模,p的指数次也可以预处理存在数组里
- abcd哈希，hash[0] = 0*p[0],hash[1] = hash[0] + 1*p[1] hash[2]=hash[1]+2*p[2],hash[3]=hash[2]+3*p[3]; 想要知道(l,r)是多少可以用(hash[r]-hash[l-1])/p[l];

- 哈希其实就类似于进制转换
- 1111B转10进制就是 $1 + 1 * 2 + 1 * 4 + 1 * 8 = 15$
- 字符串哈希就改成p进制，这个p一般取素数（131，13331），但是你这么乘肯定会爆，所以就在对1e9+7取个模,p的指数次也可以预处理存在数组里
- abcd哈希，hash[0] = 0*p[0],hash[1] = hash[0] + 1*p[1] hash[2]=hash[1]+2*p[2],hash[3]=hash[2]+3*p[3]; 想要知道(l,r)是多少可以用(hash[r]-hash[l-1])/p[l];
- 当然两个不同的数取模以后可能会相同，所以会有冲突，hash值相同可能是不同的串，(wa了可以尝试换个模数)

- 哈希其实就类似于进制转换
- 1111B转10进制就是 $1 + 1 * 2 + 1 * 4 + 1 * 8 = 15$
- 字符串哈希就改成p进制，这个p一般取素数（131，13331），但是你这么乘肯定会爆，所以就在对1e9+7取个模,p的指数次也可以预处理存在数组里
- abcd哈希，hash[0] = 0*p[0],hash[1] = hash[0] + 1*p[1] hash[2]=hash[1]+2*p[2],hash[3]=hash[2]+3*p[3]; 想要知道(l,r)是多少可以用(hash[r]-hash[l-1])/p[l];
- 当然两个不同的数取模以后可能会相同，所以会有冲突，hash值相同可能是不同的串，(wa了可以尝试换个模数)
- 最万无一失的办法就是双哈希，你用不同的p和模数取两个哈希值,然后用map¡ pair¡hash1,hash2¿,int¿ 去映射，基本上不可能冲突了，很安全