

# 字符串：字符串hash、trie树、kmp、manacher

邓丝雨

# 字符串hash



## 哈希 (hash)

- Hash, 一般翻译做“散列”, 也有直接音译为“哈希”的, 就是把任意长度的输入 (又叫做预映射, pre-image), 通过哈希算法, 变换成固定长度的输出, 该输出就是哈希值。
- 哈希值的空间通常远小于输入的空间, 不同的输入可能会哈希成相同的输出, 所以不可能从哈希值来确定唯一的输入值。



- 字符串hash —— 把一个字符串变成一个数字（一般是小于int的）
- 怎么变？



- 先想一个特例：
- 01串整么对应成一个数？
- ——当成一个2进制数
- 把这个特例变强一点：
- 一个数字串怎么对应成一个数？
- ——当成一个十进制数



- $\text{hash}[i] = (\text{hash}[i-1] * p + \text{idx}(s[i])) \% \text{mod}$
- 字符串: abc, bbc, aba, aadaabac
- 字符串下标从0开始,  $\text{idx}(a)=1, \text{idx}(b)=2, \text{idx}(c)=3, \text{idx}(d)=4$ ;
- 取  $p=13$ ,  $\text{mod}=101$
- $\text{hash}[0]=1$ , 表示 a 映射为1
- $\text{hash}[1] = (\text{hash}[0] * p + \text{idx}(b)) \% \text{mod} = 15$ , ab 映射为 15
- $\text{hash}[2] = (\text{hash}[1] * p + \text{idx}(c)) \% \text{mod} = 97$
- 这样, 我们就把 abc 映射为 97 这个数字了



## 应用

- 判断两个字符串是否一致
- Hash值不同肯定不同
- 认为hash值一致就是字符串一致



- 判断B是不是A的一个子串





- 若A的某个子串的hash值等于B的hash值则认为是，否则肯定不是
- A某个子串的hash值怎么求




- 我们看下hash的公式:
- $\text{hash}[i] = (\text{hash}[i-1] * p + \text{idx}(s[i])) \% \text{mod}$
- 这表示第  $i$  个前缀的hash值
- 那么, 我要求  $S[l \dots r]$  这个子串的hash值
- $\text{hash}[l \dots r] = (\text{hash}[r] - \text{hash}[l-1] * (p^{(r-l+1)})) \% \text{mod}$
- (假设字符串下标从1开始)



## 双hash

- $\text{hash1}[i] = (\text{hash1}[i-1] * p + \text{idx}(s[i])) \% \text{mod1}$
- $\text{hash2}[i] = (\text{hash2}[i-1] * p + \text{idx}(s[i])) \% \text{mod2}$
- mod1一般取 $1e9+7$ , mod2一般取 $1e9+9$ 为什么这么取?
- 1000000007和1000000009是一对孪生素数, 取它们, 冲突的概率极低

# Trie树

- 
- trie, 又称**前缀树**或**字典树**, 是一种有序树, 用于保存关联数组, 其中的键通常是字符串。与二叉查找树不同, 键不是直接保存在节点中, 而是由节点在树中的位置决定。一个节点的所有子孙都有相同的前缀, 也就是这个节点对应的字符串, 而根节点对应空字符串。一般情况下, 不是所有的节点都有对应的值, 只有叶子节点和部分内部节点所对应的键才有相关的值。
  - 基本性质
    - 1, 根节点不包含字符, 除根节点意外每个节点只包含一个字符。
    - 2, 从根节点到某一个节点, 路径上经过的字符连接起来, 为该节点对应的字符串。
    - 3, 每个节点的所有子节点包含的字符串不相同。



## 例1: poj3630

- 给定一系列电话号码，查看他们之间是否有 $i, j$ 满足，号码 $i$ 是号码 $j$ 的前缀子串。



- 1.暴力：即从字符串集中从头往后搜，看每个字符串是否为字符串集中某个字符串的前缀，复杂度为 $O(n^2)$ 。





- 2、使用hash：我们用hash存下所有字符串的所有的前缀子串。建立存有子串hash的复杂度为 $O(n \times \text{len})$ 。查询的复杂度为 $O(n) \times O(1) = O(n)$ 。



- 3、使用Trie：因为当查询如字符串123是否为某个字符串的前缀时，显然以b、c、d....等不是以a开头的字符串就不用查找了，这样迅速缩小查找的范围和提高查找的针对性。所以建立Trie的复杂度为 $O(n \cdot \text{len})$ ，查找的时候只要顺着找下去就可以了。



## 例2:poj2503

- 一本字典，英语对应外语，现在要求输入一个外语单词，如果字典中相应的英语解释就输入对应的英语单词，否则输出 `e n`；



## 01-trie

- 我们的字符集只有01两种元素的时候，我们可以构造一棵二叉树，0的分支在左边，1的分支在右边，这就是01trie树。它常被用来处理这样一个经典问题：给你若干个数，选出其中两个数使得他们异或之后最小/最大。（这里以最小为例）
- 我们遍历数组中的元素，对于每个元素在他前面找和他异或起来最小的值，显然，要最小肯定是这两个数要从最高位开始尽量一致，我们把它之前的数都按照二进制串的样子加入到01trie里面（所有数的位数统一成一致，不够的补零）。然后根据当前这个数对应的01串在trie上从根向下走，如果当前数这一位是0，在trie树上我们更希望走0的那一边，有0就往0那边走，没有0就只能走1了。



## 例：NC22998 奶牛异或

- 农民约翰在喂奶牛的时候被另一个问题卡住了。他的所有 $N$  ( $1 \leq N \leq 100,000$ ) 个奶牛在他面前排成一行(按序号 $1..N$ 的顺序)，按照它们的社会等级排序。奶牛#1有最高的社会等级，奶牛#N最低。每个奶牛同时被指定了一个不唯一的附加值，这个数在  $[0, 2^{21} - 1]$  的范围内。
- 帮助农民约翰找出应该从哪一头奶牛开始喂，使得从这头奶牛开始的一个连续的子序列上，奶牛的附加值的异或最大。
- 如果有多个这样的子序列，选择结尾的奶牛社会等级最高的。如果还不唯一，选择最短的。



- 区间 $[l, r]$ 的异或和等于前 $r$ 个数的异或和和前 $l-1$ 个数的异或和异或，所以，我们只需要在前缀异或和数组里面任选两个数让他们异或和最大即可。

# KMP



## 问题引入

- 给你两个字符串，寻找其中一个字符串是否包含另一个字符串，如果包含，返回包含的起始位置





- Eg :
- 目标串 `abccabccabd`
- 模式串 `abccabd`




- 失配指针 next
- 当在i位置发生失配以后，不需要从头开始比较，只需要将模式串移动到某特定位置然后还从刚才的位置开始比较。
- 只和模式串有关 和目标串没有关系



- **Next[i]在数值上等于最长公共前后缀的长度-1**

a	b	c	a	b	d
-1	0	0	0	1	2



```
20     nxt[0] = -1;
21     for(int i = 1, j = -1; i <= lent; i++)
22     {
23         while(j > -1 && t[i] != t[j+1]) j = nxt[j];
24         if(t[i] == t[j+1]) j++;
25         nxt[i] = j;
26     }
```



- 一个小优化:
- `abcabd`
- `-100012`
- 如果第二个b失配, 移动到第一个b还是依然会失配
- 所以如果`next[i]`与i的字母相同, 那么`next[i]`可以直接赋值为`next[next[i]]`



## 例1: POJ 2752

- 给你一个字符串str 求str中有多少个子串, 既是str的前缀又是str的后缀



## 例2: POJ 1961

- 给定一个字符串，问这个字符串的所有前缀中，有哪些前缀可以由某个串重复 $k$ 次组成，

# Manacher





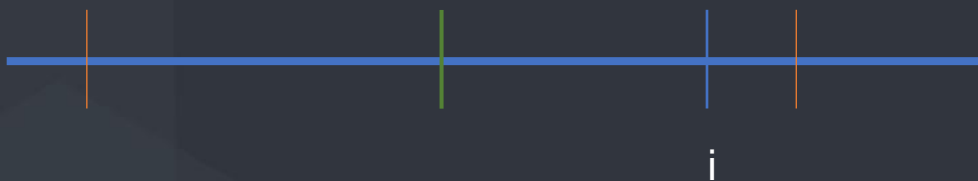
- 1975年, Manacher发明了Manacher算法 (中文名: 马拉车算法), 是一个可以在 $O(n)$ 的复杂度中返回字符串s中最长回文子串长度的算法, 十分巧妙。



- 回文串有奇数和偶数长度两种，分类讨论增加工作量
- 怎么办？
- 给每个字母后面和整个串的最前面加一个没有在字符串里面出现的字符（比如\$）
- Eg: abcbcbbaabbc
- \$a\$b\$c\$b\$c\$b\$a\$a\$b\$b\$c\$
- 奇数和偶数长度的回文串都变成了奇数长度的



- 枚举回文对称中点
- —— $p[n]$ 代表的是以 $n$ 这个位置为中心的子串，它的右边一半的长度





```
9      for(int i = 1 ;i < len; i++)
10     {
11         if(i < mx) p[i] = min(mx-i, p[mid*2-i]);
12         else p[i] = 1;
13         while(s[i-p[i]] == s[i+p[i]])
14             p[i]++;
15         if(i + p[i] > mx)
16         {
17             mid = i;
18             mx = i + p[i];
19         }
20         ans=max(ans, p[i]-1);
21     }
22
```



- 每个字符最多被遍历两次—— $O(n)$



## 例1：NC14894最长回文

- 有两个长度均为 $n$ 的字符串 $A$ 和 $B$ 。可以从 $A$ 中选一个可以为空的子串 $A[l1..r1]$ ， $B$ 中选一个可以为空的子串 $B[l2..r2]$ ，满足 $r1=l2$ ，然后把它们拼起来 ( $A[l1..r1]+B[l2..r2]$ )。求用这样的方法能得到的最长回文串的长度。注意：求的不是本质不同的回文串个数哦！！



- 题解:对字符串A和字符串B各自进行一次manacher, 求出p数组
- 然后枚举回文中心, 我们在 $pA[i]$ 和 $pB[i]$ 取一个max, 表示我们暂时先只取两个串中较长的回文串, 然后对于这个回文串, 我们可以像两边拓展, 因为假设A的回文串的左边的字符不在A的回文串中, 但是可以和B右边的字符相同的话, 就可以形成回文, 边枚举边拓展取最大值即可得到A串和B串取出一段来形成回文串的最大值



## 例2: CF1080E Sonya and Matrix Beauty

- 给定一个  $n \times m$  的字符矩阵，请求出有多少个子矩阵在重排子矩阵每一行的字符后，使得子矩阵的每行每列都是回文串。



- 如果一行能构成回文串，那么最多只能有一种字符出现奇数次。
- 如果一个矩阵的每一行和每一列都是回文串，那么除了满足上面的要求外，第 $i$ 行和第 $n-i+1$ 的每种字母出现的次数必须都相同。
- 所以我们可以枚举两列，然后对每一行的字母把出现次数hash起来，然后就是求由每一行的哈希值构成的序列的回文子串个数，manacher解决即可。
- 另外要注意的是如果某一行不能构成回文串，那么不应该计入到manacher的统计中去。



## 作业

- <https://ac.nowcoder.com/acm/problem/collection/1266>



- POJ3461 Oulipo
- POJ2774 Long Long Message
- POJ2406 Power Strings
- NC53325 Forsaken喜欢字符串
- NC53679 「水」悠悠碧波
- NC19822 我不爱她
- NC20862 救救企鹅
- NC20947 喵喵体
- NC15049 假的字符串
- POJ3630 Phone List
- POJ2503 Babelfish
- NC19249 随风飘
- NC22998 奶牛异或
- POJ2752 Seek the Name, Seek the Fame
- POJ1961 Period
- NC15071 数一数
- NC14894 最长回文
- CF1080E Sonya and Matrix Beauty
- POJ3974 Palindrome
- CF17E Palisection
- NC17062 回文
- UVALive7239 Bazinga