字符串总结

MLEAutoMaton

长郡中学

2019 年 3 月 28 日

欢迎交流 QQ:3099895260

字符串 Hash

字符串 Hash

字符串 Hash 好像确实不是很难.

字符串 Hash

字符串 Hash 好像确实不是很难.

下面是代码.

字符串 Hash

字符串 Hash 好像确实不是很难.

下面是代码.

```
const int N=5000010;
#define ull unsigned long long
const ull Base=19260817;//(19260817,19491001,1e9+7) 随便选
ull Hash[N];
char s[N];
void hAsh()
{
        for(int i=1;i<=l;i++)
        Hash[i]=Hash[i-1]*Base+s[i];
}
```

是不是特别的简单?

[Ctsc2014] 企鹅 QQ

[Ctsc2014] 企鹅 QQ

小 Q 定义，若两个账户名称是相似的，当且仅当这两个字符串等长且恰好只有一位不同。例如“Penguin1”和“Penguin2”是相似的，但“Penguin1”和“2Penguin”不是相似的。而小 Q 想知道，在给定的 N 个账户名称中 (长度都为 L)，有多少对是相似的。

$N <= 30000, L <= 200$

[Ctsc2014] 企鹅 QQ

[Ctsc2014] 企鹅 QQ

发现我们可以通过枚举是哪一位不同来确定，但是这样子复杂度很高，$O(N^2 * L)$ 的，怎么优化匹配的时间呢？

[Ctsc2014] 企鹅 QQ

发现我们可以通过枚举是哪一位不同来确定，但是这样子复杂度很高，$O(N^2 * L)$ 的，怎么优化匹配的时间呢？
前缀 Hash 后在来一个后缀 Hash，把这两个 Hash 值合并就可以当做一个字符串在当前这一位没有影响的特征值.

[Ctsc2014] 企鹅 QQ

发现我们可以通过枚举是哪一位不同来确定，但是这样子复杂度很高，$O(N^2 * L)$ 的，怎么优化匹配的时间呢？
前缀 Hash 后在来一个后缀 Hash，把这两个 Hash 值合并就可以当做一个字符串在当前这一位没有影响的特征值.
然后在这一段里面把相同的算一下贡献就好了.

[Ctsc2014] 企鹅 QQ

[Ctsc2014] 企鹅 QQ

当然还有其他的方法啊.

[Ctsc2014] 企鹅 QQ

当然还有其他的方法啊.
只需要前缀 Hash, 然后考虑每一段的贡献就好了!!!

POJ3974 Palindrome

POJ3974 Palindrome

有 T 次询问，每一次给出一个串

POJ3974 Palindrome

有 T 次询问，每一次给出一个串

求出一个串的最长回文子串.

POJ3974 Palindrome

有 T 次询问, 每一次给出一个串

求出一个串的最长回文子串.

$N <= 1000000$

POJ3974 Palindrome

POJ3974 Palindrome

考虑回文子串不仅可以 $manacher$, 也可以用二分 +Hash 求解.

POJ3974 Palindrome

考虑回文子串不仅可以 $manacher$, 也可以用二分 +Hash 求解.

注意分两种情况讨论 (奇数长度和偶数长度)

POJ2774 Long Long Message

POJ2774 Long Long Message

给你两个串，求这两个串的最长公共子串.

$N <= 10^5$

POJ2774 Long Long Message

POJ2774 Long Long Message

这个可以 SA 做 (当然 SA 后面会讲的啦.)

POJ2774 Long Long Message

这个可以 SA 做 (当然 SA 后面会讲的啦.)
考虑怎么 Hash, 肯定有的是二分答案 (二分串的长度)

POJ2774 Long Long Message

这个可以 SA 做 (当然 SA 后面会讲的啦.)
考虑怎么 Hash, 肯定有的是二分答案 (二分串的长度)
接着就是把一个串里面的所有长度为这个的 Hash 值抠出来, 然后再到另一个串里面找 (这个时候可以二分).

POJ2774 Long Long Message

这个可以 SA 做 (当然 SA 后面会讲的啦.)
考虑怎么 Hash, 肯定有的是二分答案 (二分串的长度)
接着就是把一个串里面的所有长度为这个的 Hash 值抠出来, 然后再到另一个串里面找 (这个时候可以二分).
然后就可以判断是不是为 LCP 了.

前言

下面是一些丧心病狂的东西，如果没有做好准备，就不要往下看了.

Hash Killer I

已知下面这一段 Hash 代码，求一组卡 Hash 的数据.

```
u64 hash_pow_l = 1;
for (int i = 1; i <= l; i++)
        hash_pow_l *= base;
int li_n = 0;
static u64 li[MaxN];
u64 val = 0;
for (int i = 0; i < l; i++)
        val = val * base + s[i] - 'a';
li[li_n++] = val;
for (int i = l; i < n; i++)
{
        val = val * base + s[i] - 'a';
        val -= (s[i - l] - 'a') * hash_pow_l;
        li[li_n++] = val;
}
```

$N <= 10^5$

Hash Killer I

考虑它是自然溢出，相当于就是对 2^{63} 取膜

Hash Killer I

考虑它是自然溢出，相当于就是对 2^{63} 取膜
那么就有 $aaaaa...aaa$(多于 64 个) 和 $baaaa...aaa$(多于 64 个) 的 $Hash$ 相同

Hash Killer I

考虑它是自然溢出，相当于就是对 2^{63} 取膜
那么就有 $aaaaa...aaa$(多于 64 个) 和 $baaaa...aaa$(多于 64 个) 的 $Hash$ 相同
当然，这是对于偶数的 Base.
奇数直接再反转一遍拼上去就可以了.

Hash Killer II

Hash 代码, 求 Hack!!!

```
const int Mod = 1000000007;
u64 hash_pow_l = 1;
for (int i = 1; i <= l; i++)
        hash_pow_l = (hash_pow_l * base) % Mod;

int li_n = 0;
static int li[MaxN];

u64 val = 0;
for (int i = 0; i < l; i++)
        val = (val * base + s[i] - 'a') % Mod;
li[li_n++] = val;
for (int i = l; i < n; i++)
{
        val = (val * base + s[i] - 'a') % Mod;
        val = (val + Mod - ((s[i - l] - 'a') * hash_pow_l) % Mod) % Mod;
        li[li_n++] = val;
}
```

$N <= 10^5$

Hash Killer II

这道题目好像题面里面给了提示 (当然没给就有点难想了.)

Hash Killer II

这道题目好像题面里面给了提示 (当然没给就有点难想了.)

菊开讲过一个叫做生日悖论的, 不知道还有多少人记得 (菊队~~长~~!!!)

Hash Killer II

这道题目好像题面里面给了提示 (当然没给就有点难想了.)

菊开讲过一个叫做生日悖论的, 不知道还有多少人记得 (~~菊队长!!!~~)

考虑相同的可能性大概是 $\sqrt{n}$ 的, 所以直接随机一下就好了 (概率 ~~AC~~).

Hash Killer III

限于篇幅，这里就不放代码了，告诉你是一个双模数 Hash，求 Hack!!!

Hash Killer III

要是有谁做出来了，可以写一篇论文然后直接被清北录取啊.

Hash Killer III

要是有谁做出来了，可以写一篇论文然后直接被清北录取啊.

这题要是写出来了就是 Hash 的末日了.

双模数 Hash

你可以选择用 $1e9+7$ 与 19260817 的组合，当然如果你写自然溢出写的很多，不想换了怎么办呢？

双模数 Hash

你可以选择用 $1e9+7$ 与 19260817 的组合，当然如果你写自然溢出写的很多，不想换了怎么办呢？
来和我一起感受自然溢出的快感吧!!!

双模数 Hash

你可以选择用 $1e9+7$ 与 19260817 的组合，当然如果你写自然溢出写的很多，不想换了怎么办呢？
来和我一起感受自然溢出的快感吧!!!
仔细思考一下自然溢出是个什么东西？不就是相当于取模吗？
然后就可以使用 $unsigned\ int$ 和 $unsigned\ long\ long$

KMP 概念

这个东西很好理解，所以还是~~背背板子~~理解一下的好.
考虑我们令 nxt_i 表示 1 i 之间的最长公共前后缀.
如果我们匹配到某一个位置无法匹配，就跳一下 nxt 就好了.
具体来说这个东西很玄学:(下面是 30s 口胡过程)
因为现在我们是第 i 为无法与 j 匹配，所以我们要找一个可以匹配的 j 或者是不匹配了.
那么如果长度减了之后，显然在 i 串上面形如一段后缀，在 j 上面形如一段前缀，然后这个定义的来由就差不多讲清楚了.
那么怎么求呢？直接把模式串与模式串匹配就好了. 相当于是在模式串里面找模式串.

AC 自动机概念

今天才学 (3.27), 以前好像学过, 但是没学懂...(我果然是奶王, 刚学第二天考试就考了)
考虑如果有很多个模式串, 单文本串匹配怎么办呢?
我们把模式串丢到一个 $Tire$ Trie 树上面去.
那么剩下的过程就是解我们的 nxt 数组了对吧.
这个东西直接跑一个 bfs 求解就好了.
自己画画图分析, 应该就没错了.

SAM 概念

怕不是我写过的最多的字符串题吧. 谢罪~~了~~
这个构建自己随便 yy 一下就好了, 大致流程参考 yyb 的 Blog...
放一个 extend 的模板 (主要是怕自己忘了)

板子

```
void extend(int c)
{
        int np=++tot,p=last;last=tot;
        t[np].len=t[p].len+1;
        while(p && !t[p].son[c])t[p].son[c]=np,p=t[p].ff;
        if(!p)t[np].ff=1;
        else
        {
                int q=t[p].son[c];
                if(t[p].len+1==t[q].len)t[np].ff=q;
                else
                {
                        int nq=++tot;
                        t[nq]=t[q];t[q].ff=t[np].ff=nq;
                        t[nq].len=t[p].len+1;
                        while(p && t[p].son[c]==q)t[p].son[c]=nq,p=t[p].ff;
                }
        }
        siz[np]=1;
}
```

~~亲测没有什么问题~~

SA 概念

难道我要说不会？好像确实是的...
大概背个模板就差不多了.
给出模板!!
代码戳这里

最小表示法

把串复制一遍放到末尾，然后建后缀自动机，在上面跑 n 次就好了.

你问线性的?(当然会啦)

考虑我们定义两个指针 i, j 分别指向 a_1, a_2，定义 k 表示匹配的长度，那么对于:$i+k-1\%n+1$ 和 $j+k-1\%n+1$ 这样子的两个位置，显然有如下三种情况:

- a[(i+k-1)%n+1] = a[(j+k-1)%n+1],k++.
- a[(i+k-1)%n+1] < a[(j+k-1)%n+1]，显然 j 不优，那么就把 j 跳到 $j+k$,k=0.
- a[(i+k-1)%n+1] > a[(j+k-1)%n+1]，显然 i 不优，那么就把 i 跳到 $j+i$,k=0.

然后这个东西就做到比较快的了.

回文自动机简介

自动机里面除了 AC 自动机之外最简单的吧.
考虑回文串有两种, 一个是奇数长度的, 另一个是偶数长度的.
回文自动机上面的节点应该代表着是两个字符, 如:
$aba->c->cabca$
那么我们如果要在回文自动机跑东西怎么办呢?
考虑令 $fail_i$ 表示以 i 结尾的最长的回文后缀 (不包括自己).
那么每加入一个节点, 只要和上一次的比一下是否匹配就好了.
接着考虑插入这一个字符, 如果存在就不要管对吧.
不存在怎么做?
从 $last$ 的祖先中找一个可以匹配的上的回文串就好了.
长度每一次是 $+2$.
具体代码可以到我的 cnblogs内查看.

完

谢谢大家

祝大家 $HNOI2019$ 顺利。