nnr nnr nnr

$(AB)^i C$

$A = nn \quad B = r \quad i = 2$

$C = nnr$

$A = S[1 \cdots a]$

$B = S[a+1 \cdots b]$

$i \cdots$

$C = S[ib+1 \cdots n]$

for $a = 1$ to n $\qquad O(n^3)$

for $b = a+1$ to n

for $i = 1$ to $\frac{n}{b}$

已知：$S[1 \cdots (i-1)b] = (AB)^{i-1}$

只需检查 $S[(i-1)b+1 \cdots ib] = AB \cdots O(b)$

检查 $F(A) \le F(C)$.

（预处理 $pre[i] = F(S[1 \cdots i])$ $O(n)$

$suf[i] = F(S[i \cdots n])$）

只需检查 $pre[a] \le suf[ib+1] \cdots O(1)$

预处理 $pre[i] = F(S[1 \cdots i])$ $O(n)$

```
int cnt[26];
cnt[c] 表示 'a' + c 出现了几次

for i = 1 to n {
    cnt[s[i] - 'a'] ++ ;
    if ( cnt[s[i] - 'a'] % 2 == 1 ) pre[i] = pre[i-1] + 1 ;
    else pre[i] = pre[i-1] - 1 ;
}
```

$suf[i] = F(S[i \cdots n])$ 类似的.

```
int g[N][27];
g[i][k]:  pre[1] .. pre[i] 中 ≤ k 的有几个
g[i-1][k]:  pre[1] .. pre[i-1] 中 ≤ k 的有几个

for i = 1 to n
    for k = 0 to 26
        if (pre[i] ≤ k)  g[i][k] = g[i-1][k] + 1
        else  g[i][k] = g[i-1][k].
```

$pre[i] = F(S[1 \cdots i])$

$suf[i] = F(S[i \cdots n])$

$g[i][k]$: $pre[1] .. pre[i]$ 中 $\leq k$ 的有几个

枚举 b，令 $AB = S[1 \cdots b]$

枚举 i，$(AB)^i = S[1 \cdots ib]$，$C = S[ib+1 \cdots n]$

寻找 i 的上限 I，使

$S[1 \cdots b]$ 确实是 $S[1 \cdots Ib]$ 的循环节

（自然，也会是 $S[1 \cdots (I-1)b]$ 的循环节）

预先求 S 的 z 函数！

$$z_i = Lcp(S, S[i \cdots n])$$

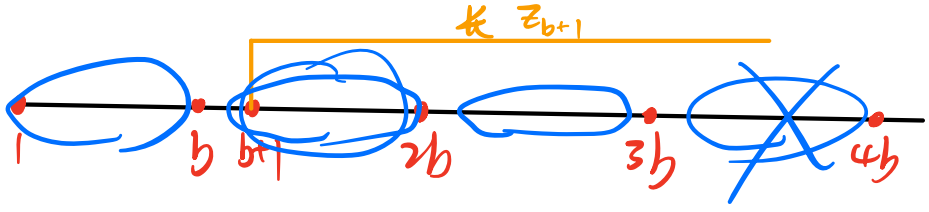
判断 $S[1 \cdots t]$ 是 S 的循环节，充要条件：

$t \mid n$ 且 $z_{t+1} == n - t$

$S[1 \cdots b]$ 是 $S[1 \cdots ib]$ 的循环节的充要条件是什么？

$i=2$ ：可以

$i=3$ ：可以



条件：$ib \le z_{b+1} + b$

最后一个问题：划分 A 和 B

满足 $F(A) \le F(C)$

（假如划分成 $A = S[1 \cdots a]$，$B = S[a+1 \cdots b]$）

$F(A)$ = $pre[a]$

$F(C)$ = $suf[ib+1]$

找 $a \in [1, b-1]$ 有 n 个 a 满足 $pre[a] \le suf[ib+1]$

$\Rightarrow g[b-1][suf[ib+1]]$

保证 C 非空

```
for b = 1 to n
    for (i=1; ib <= b + z_{b+1} && ib < n; i++) {
        ans += g[b-1][suf[ib+1]];
    }
```

$b=1$ 内：n 次

$b=2$ 内：$\frac{n}{2}$ 次

⋮

$$n + \frac{n}{2} + \frac{n}{3} + \cdots + \frac{n}{n} \le n + n \ln n$$

$O(n \ln n)$

```
map<string, int> p;

p["hello"] = 1;
p["world"] = 2;
```

元素个数

单次操作复杂度：$O(\log n \cdot \text{cmp复杂度})$

单次比大小操作

对于 int，cmp 复杂度 O(1)

对于 string，cmp 复杂度 O(长度)

```
map<long long, bool> p;
```

下标、有的值

```
p[998244353] = true;
```

p.count(1926) 查询是否用过 p[1926] 这个东西

p.size() 查询一共用了几个东西

```
unordered_map<long long, bool> p
```

用法和 map 一样。

单次操作复杂度：$O(1 \cdot \text{cmp复杂度})$

不可用于 string

hash 映射；散列

让每个字符串对应一个整数

下面只有小写字母

a	1
⋮	⋮
z	26
aa	27
ab	28
⋮	⋮
az	52
ba	53
⋮	
bz	

h	e	l	l	o
8	5	12	12	15
$\times 26^4$	$\times 26^3$	$\times 26^2$	$\times 26$	$\times 1$

$+ \quad \cdots$

很有效，保证了字符串和数字唯一对应

但没用，因为 26^n 太大了

那就 $\%p$ 吧，比如 $p = 10^9 + 7$

（一般 p 是质数）

$p = 998244353$

```
long long calc(string s) {
    long long res = 0;
    for i = s.size()-1 to 0
        res = (res x 26 + (s[i] - 'a' + 1)) % p;
    return res;
}
```

233

取一个大的质数

```
unordered_map<int, bool> g;

cin >> m
    for i = 1 to m {
        cin >> s
        g[calc(s)] = true;
    }
cout << g.size() << endl;
```

```
map<string, bool> g

g[s] = true
```

双模数 hash

```
long long calc(string s, int p, int b) {
    long long res = 0;
    for i = s.size()-1 to 0
        res = (res x b + (s[i] - 'a' + 1)) % p;
    return res;
}
```

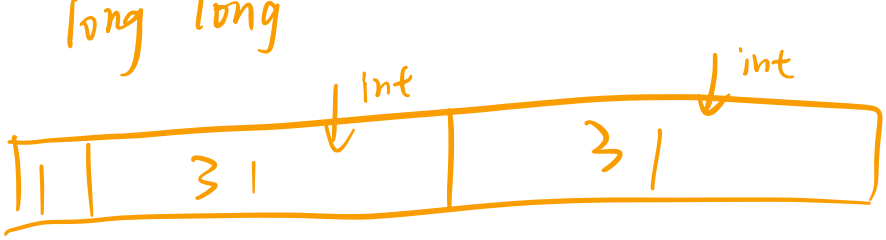
233

取一个大的质数

```
unordered_map<long long, bool> g;

cin >> m
    for i = 1 to m {
        cin >> s
        long long h1 = calc(s, p1, b1)
        long long h2 = calc(s, p2, b2)
        g[ h1<<31 | h2 ] = true;
    }
cout << g.size() << endl;
```

p1 和 p2 是两个不同的质数



下面只有小写，基是 b，模数是 p

可以 O(n) 预处理 s.

以后 O(1) 得到 s 每个子串的哈希值

pre[i]：s[1…i] 的哈希值

$$pre[i] = (pre[i-1] \times b + (s[i] - 'a' + 1)) \% p$$

$$pw[i] = b^i \% p$$

考虑 s[l…r]

s[1 …. l … r]　　pre[r]

s[1 …. l-1] + 00000　　pre[l-1] × b^{r-l+1}

（00000 为 r-l+1 个）

↓

pre[l-1]

$$((pre[r] - pre[l-1] \times b^{r-l+1}) \% p + p) \% p$$