知识点 - KMP

解决问题类型：

前缀函数

查找子串

每个前缀的出现次数

本质不同子串的个数O(n^2)

字符串压缩：找到t使得s可以被多个t重复出现得到

前缀自动机

Z函数

查找子串

本质不同子串的个数O(n^2)

字符串压缩

定义与代码：

前缀函数π(prefix function): $\pi[i]$代表子串$s[0 \dots i]$与其后缀相等的最长真前缀（proper prefix，即不包含本身的前缀）。

$$\pi[i] = \max_{k=0 \dots i} \{k : s[0 \dots k-1] = s[i-(k-1) \dots i]\}$$

"aabaaab" - $[0, 1, 0, 1, 2, 2, 3]$.

$O(n)$，在线

```
vector<int> prefix_function(string s) {
    int n = (int)s.length();
    vector<int> pi(n);
    for (int i = 1; i < n; i++) {
        int j = pi[i-1];
        while (j > 0 && s[i] != s[j])
            j = pi[j-1];
        if (s[i] == s[j])
            j++;
        pi[i] = j;
    }
    return pi;
}
```

Z函数:$z[i]$代表串s 和其后缀$s[i \dots n-1]$ 的LCP， 定义z[0]=0

"abacaba" - $[0, 0, 1, 0, 3, 0, 1]$

$O(n)$,

```
vector<int> z_function(string s) {
    int n = (int) s.length();
    vector<int> z(n);
    for (int i = 1, l = 0, r = 0; i < n; ++i) {
        if (i <= r)
            z[i] = min (r - i + 1, z[i - l]);
        while (i + z[i] < n && s[z[i]] == s[i + z[i]])
            ++z[i];
        if (i + z[i] - 1 > r)
            l = i, r = i + z[i] - 1;
    }
    return z;
}
```

前缀函数应用

1 查找子串

Q:文本串t，模式串s，求t中s所有出现的位置。

A:构造传 s&t, 考虑其前缀数组，若$\pi[i]=s.length()$则说明出现了以i为结尾的s。注意由于$\pi[j]$往前递推的范围不会超过s.length（），所以只要存s的前缀数组即可。

2 所有前缀出现的次数

$\pi[i]$存的是最长的以i结尾的，如何找到更短的以i结尾的匹配前缀的串？这些串长度为：$\pi[\pi[i]-1],\pi[\pi[\pi[i]-1]-1]\ldots$。考虑倒着更新：

```
vector<int> ans(n + 1);
for (int i = 0; i < n; i++)
    ans[pi[i]]++;
for (int i = n-1; i > 0; i--)
    ans[pi[i-1]] += ans[i];
for (int i = 0; i <= n; i++)
    ans[i]++;
```

3 本质不同的子串O(N^2)

考虑每次末尾加一个c时出现了多少新的串。

我们把串reverse一下，问题变为出现了多少新前缀。对新串算一下前缀数组，找到最大的$\pi_{\max}$,由于每个长度小于$\pi_{\max}$的前缀都出现过，所以未出现过的前缀数为：

$$|s|+1-\pi_{\max}$$

4. 压串（循环节）

Q:给定s，求最小的t使得s能由多个t连接而成。

结论：令

$$k=n-\pi[n-1]$$

我们有 如果$k|n$ 则k为t的长度。

证明：画一下，$\pi[n-1]=n-k$，说明长度为k的最后一块和前面的长度为k的一块相等。而前面一块又和前前面一块相等，以此类推。

反正法可以证明这是最优解，以及k不整除n时答案是n。

5 建一个前缀自动机

考虑1中说的，“由于$\pi[j]$往前递推的范围不会超过s.length（），所以只要存s的前缀数组即可从当前的推到下一个。”有如下状态转移：

$$(\text{old\_}\pi, c) \to \text{new\_}\pi$$

于是可以建一个自动机（也可以理解打表存下来，O（1）转移）。

这个自动机的应用就是处理一个很长的文本串，（存不下来，需要生成的那种）可以达到100^{100}数量级

比如：给出 $k \le 10^5$ 和串s $|s| \le 10^5$,求第k个gray码中s的出现次数。

gray码定义：

$$g_1 =” \ a \ ”$$

$$g_2 =” \ aba \ ”$$

$$g_3 =” \ abacaba \ ”$$

$$g_4 =” \ abacabadabacaba \ ”$$

```
//cpp prefix_automaton_slow
void compute_automaton(string s, vector<vector<int>>& aut) {
    s += '#';
    int n = s.size();
    vector<int> pi = prefix_function(s);
    aut.assign(n, vector<int>(26));
    for (int i = 0; i < n; i++) {
        for (int c = 0; c < 26; c++) {
            int j = i;
            while (j > 0 && 'a' + c != s[j])
                j = pi[j-1];
            if ('a' + c == s[j])
                j++;
            aut[i][c] = j;
        }
    }
}

//cpp prefix_automaton_fast O(26n)
void compute_automaton(string s, vector<vector<int>>& aut) {
    s += '#';
    int n = s.size();
    vector<int> pi = prefix_function(s);
    aut.assign(n, vector<int>(26));
    for (int i = 0; i < n; i++) {
        for (int c = 0; c < 26; c++) {
            if (i > 0 && 'a' + c != s[i])
                aut[i][c] = aut[pi[i-1]][c];
            else
                aut[i][c] = i + ('a' + c == s[i]);
        }
```

```
    }
}
```