后缀数组

2020年8月13日 星期四 上午8:35

```
#include <iostream>
#include <algorithm>
#include <cstdio>
#include <cstring>
using namespace std;
const int N = 50010;
int wa[N], wb[N], wv[N], wss[N];
int n, T, rak[N], height[N], cal[N], sa[N];
char s[N];
bool cmp(int *r, int a, int b, int l)
{
    return r[a] == r[b] && r[a + l] == r[b + l];
}
void da(int *r, int *sa, int n, int m)
{
    int i, j, p, *x = wa, *y = wb;
    for (i = 0; i < m; i++) wss[i] = 0;
    for (i = 0; i < n; i++) wss[x[i] = r[i]]++;
    for (i = 1; i < m; i++) wss[i] += wss[i - 1];
    for (i = n - 1; i >= 0; i--) sa[--wss[x[i]]] = i;
    for (j = 1, p = 1; p < n; j *= 2, m = p) {
        for (p = 0, i = n - j; i < n; i++) y[p++] = i;
        for (i = 0; i < n; i++)
            if (sa[i] >= j) y[p++] = sa[i] - j;
        for (i = 0; i < n; i++) wv[i] = x[y[i]];
        for (i = 0; i < m; i++) wss[i] = 0;
        for (i = 0; i < n; i++) wss[wv[i]]++;
        for (i = 1; i < m; i++) wss[i] += wss[i - 1];
        for (i = n - 1; i >= 0; i--) sa[--wss[wv[i]]] = y[i];
        for (swap(x, y), p = 1, x[sa[0]] = 0, i = 1; i < n; i++) {
            if (cmp(y, sa[i - 1], sa[i], j)) x[sa[i]] = p - 1;
            else x[sa[i]] = p++;
        }
    }
}
void calc(int *r, int *sa, int n)
{
    int i, j, k = 0;
    for (i = 1; i <= n; i++) rak[sa[i]] = i;
    for (i = 0; i < n; height[rak[i++]] = k)
        for (k ? k-- : 0, j = sa[rak[i] - 1]; r[i + k] == r[j + k]; k++);
    for (i = n; i; i--) {
        rak[i] = rak[i - 1];
        sa[i]++;
    }
}
int main()
{
    // freopen("in.txt", "r", stdin);
    // freopen("out.txt", "w", stdout);
    scanf("%d", &T);
    while (T--) {
        scanf("%s", s + 1);
        int n = strlen(s + 1);
        for (int i = 1; i <= n; i++) cal[i] = s[i];
        cal[n + 1] = 0;
        da(cal + 1, sa, n + 1, 150);
        calc(cal + 1, sa, n);
        int res = 0;
        for (int i = 1; i <= n; i++)
            res = res + n - sa[i] + 1 - height[i];
        printf("%d\n", res);
    }
    return 0;
}
```

后缀数组：字符串问题

ababc.

子串：[i, j]　　[1,3] → aba.
　　　　　　　　[2,4] → bab.

后缀：ababc.　[i, n]

suff(1) = ababc.
suff(2) = babc.
suff(3) = abc.
suff(4) = bc.
suff(5) = c.

height[1] = 0
height[2] = (a, aba) = 1.
height[3] = (aba, ba) = 0.

aba → {aba, ba, a} —sort→ {a, aba, ba}　排名 1, 2, 3.

sa[1] = 3.
sa[2] = 1.
sa[3] = 2.

rank[1] = 2.
rank[2] = 3.
rank[3] = 1.

sa[ ]：排名为 i 的后缀的起始位置
rank[ ]：起始位置 i 的后缀的排名.

height[ ]：height[i] 表示 LCP(i, i-1)

注：lcp(i, j) 表示 suff(sa[i]) 和 suff(sa[j]) 的最长公共前缀
排名为 i 的和排名为 j 的后缀的公共前缀.
lcp(i, j) = lcp(j, i)
lcp(i, i) = len(suff(sa[i])) = n - sa[i] + 1.

height[i] 表示排名为 i 的与排名 i-1 的最长公共前缀

ababca → {ababca, babca, abca, bca, ca, a} —sort→ a, ababca, abca, babca, bca, ca.　排名 1, 2, 3, 4, 5, 6

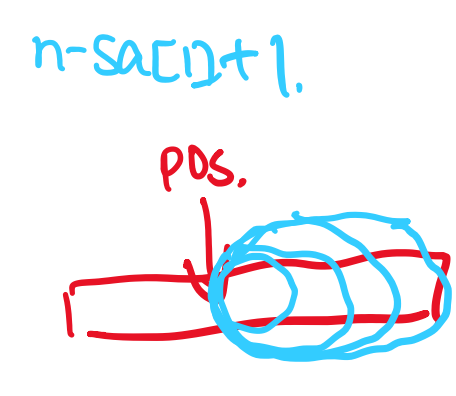
sa[1] = 6.　rak[1] = 2.　h[1] = 0
sa[2] = 1.　rak[2] = 4　h[2] = 1
sa[3] = 3.　rak[3] = 3　h[3] = 2
sa[4] = 2.　rak[4] = 5.　h[4] = 0
sa[5] = 4　rak[5] = 6　h[5] = 1
sa[6] = 5　rak[6] = 1　h[6] = 0

DC3　O(n).　代码长
DA.　O(nlogn).　代码短.

1. 求一个字符串有的多不同的子串.

ababca → {ababca, babca, abca, bca, ca, a} → a, ababca, abca, babca, bca, ca.　排名 1, 2, 3, 4, 5, 6

n-sa[i]+1.



子串：必然是一个后缀的前缀.
对一个后缀来说前缀有 n-sa[i]+1，i 是排名.
(不考虑重复).

a
ababca.
a
ab
aba
abab.
ababc
ababca.

n-sa[1]+1 = 1
n-sa[2]+1 = 6.　height[i]
n-sa[i]+1 − 和前面串最长的公共前缀

$$\sum_{i=1}^{n} n-sa[i]+1-height[i].$$

2. 给你一个字符 x，求串中包含 x 的不同子串.

ababca.

x='b'　ababca.

a x
ab ←
aba
abab
ababc
ababca.

pos[i]：i 后面第 1 次出现 x 的位置.

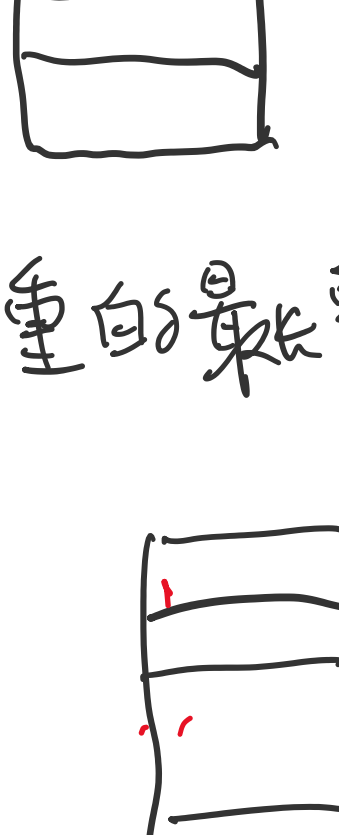
pos[1] = 2.
pos[2] = 2.
pos[3] = 4
⋮

i：n-sa[i]+1
max(height[i], pos[sa[i]] - sa[i])

$$\sum_{i=1}^{n} n-sa[i]+1-\max(height[i],\ pos[sa[i]]-sa[i]).$$

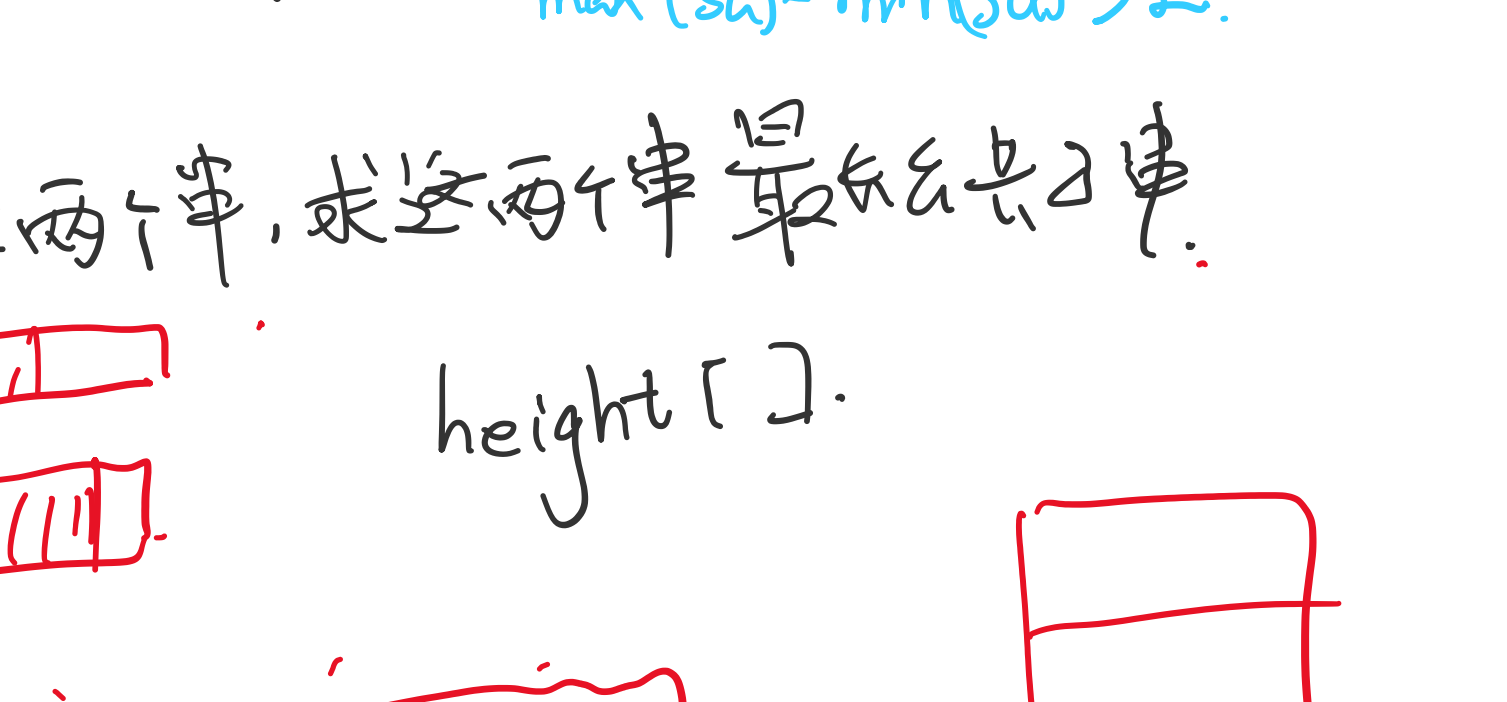
3. 可重叠 K 次的最长重复子串

二分一个 L：(子串的长度).



所有的 h ≥ L.
这一块的公共前缀长度 ≥ L.
判断这块内元素个数是不是 ≥ K.

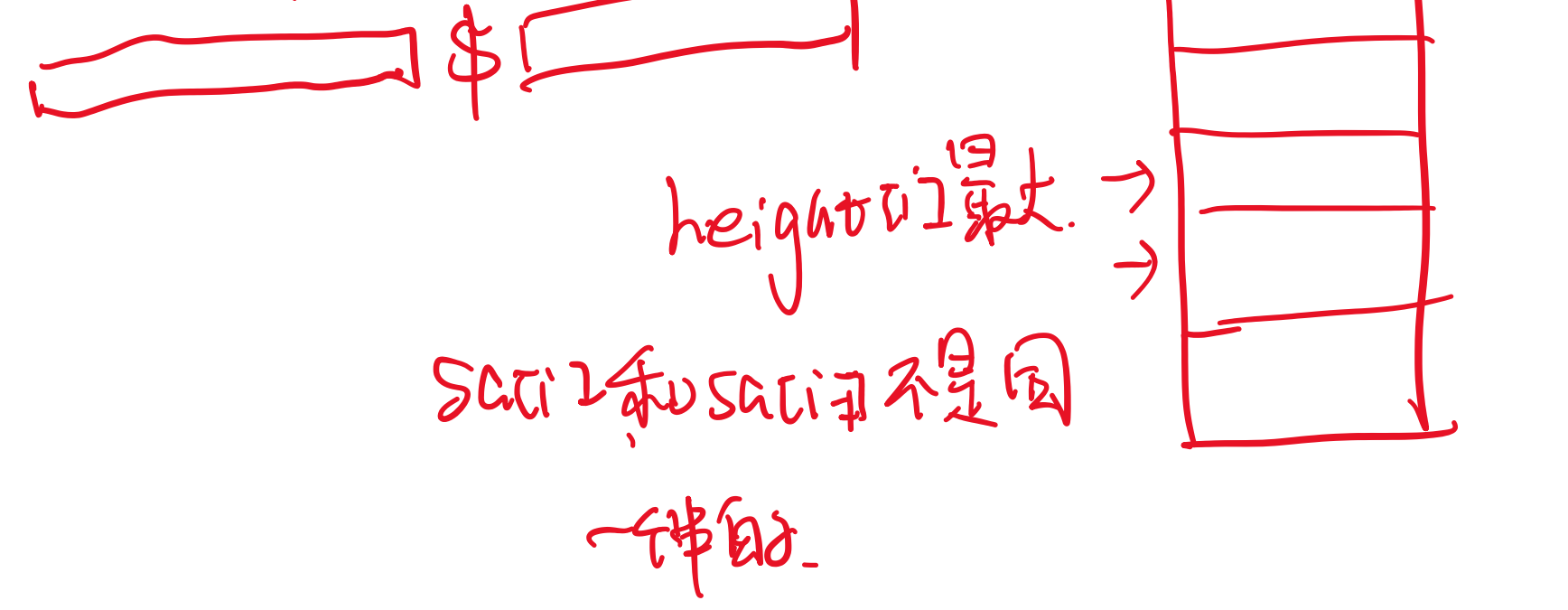
4. 不可重叠的最长重复子串.(重复 2 次).
二分 L.



min(sa).
max(sa).
max(sa) - min(sa) > L.

5：给你两个串，求这两个串最长公共子串.

height[ ].



height[i] 最大.
sa[i] 和 sa[i-1] 不是同一个串的.