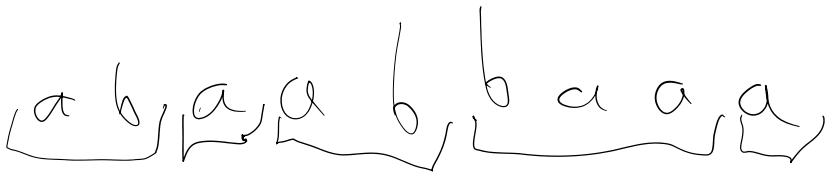
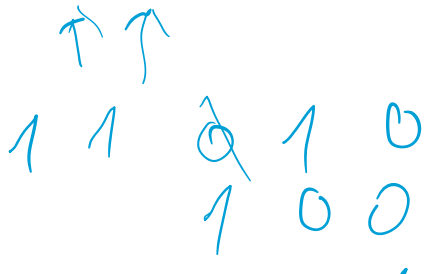
K21 - Omowienia

Friday, 1 April 2022 16:26



a
ab
baa



dp[0]: 1 1

for i:=0 i<=f ++i;

if(dp[i])

for j:0 j<n j++

if da sie pokryć dict[j] poz od i+1

dp[i+dict[j].len]=1

nie wychodzi poza zakres

cout<<(dp[f];

da sie pokryć <=> substringhash(i+1, i+dt)== hash[j]

Substr hash(beg, end)

ret suf[beg] - >suf(end+1)* base^(end-beg+1)

suf[i]= base · suf[i+1] + (t[i]-'a'+1);

b -> bx^0 —

ab -> bx+a

$bx^3 + ax^2 + bx + a$

$x = base$

$ab \rightarrow bx + a$

$bab \rightarrow bx^2 + ax + b$

$abab \rightarrow bx^3 + ax^2 + bx + a$

$ax + b = b_a$

Zadanie przystawka

Zacznijmy od jednego z najpopularniejszych problemów optymalizacyjnych. Mamy dany ciąg (a_i) o długości $n \leq 10^6$ i chcemy znaleźć długość jego najdłuższego podciągu rosnącego. Dla proszczenia załóżmy, że $a_i \leqslant 10^6$, gdyż w innym przypadku możemy bezproblemowo przeskalować wartości. Poprzez **podciąg** ciągu (a_i) rozumiemy taki ciąg liczb, który możemy otrzymać z ciągu (a_i) poprzez usunięcie niektórych jego wyrazów.

$dp[i] = 1 + \max_{j<i : a_i > a_j} dp[j]$

kończy się w i

```
for i = 1  i <= n  i++
    for j = 1  j < i  j++
        if a[i] > a[j]
            tmp = max(tmp, dp[j])
    dp[i] = tmp + 1;
    res = max(res, dp[i]);
```

$O(n^2)$

$j < i \quad a_i > a_j$

```
tree.insert(pos = a[i], val = dp[i])
tree.queryMax(1, a[i] - 1)
for i = 1  i <= n  i++
    dp[i] = 1 + tree.queryMax(1, a[i] - 1)
```

```
for i=1 ; i<=n ; i++
    dp[i] = 1 + t.queryMax(1, a[i]-1)
    res = max(res, dp[i])
    t.insert(a[i], dp[i])        O(n lg n)
```