Problema – nuz

Descrierea soluției

Autor: prof. Constantin Gălățan
Colegiul Național „Liviu Rebreanu” Bistrița

Soluție O(n*n)

Un cuvânt poate avea cel mult lungimea **`n`**. Caracterele au pozițiile **`[1..n]`**. Dacă am ajuns la poziția **`i`** și dorim să aflăm numărul de cuvinte corect construite începând cu acea poziție, atunci trebuie să știm care este lungimea unei secvențe de litere identice care se termină la poziția **`i`** și dacă până la poziția i a fost întâlnită sau nu o *secvență repetabilă maximală*. Programare dinamică, deci. Fie **`c[i][j][ok]`** numărul de cuvinte ce conțin secvențe maximale începând cu poziția **`i`** (**`1 <= i <= n`**) dacă la poziția i se termină o secvență de lungime **`j`** formată din litere identice și **`ok = true`** dacă până la poziția **`i`** a fost întâlnită o sercvență maximală (de lungime **`k`**).

Recurența este următoarea:

Dacă **`i >= n`** atunci dacă **`ok = true`** sau **`j = k`**
`c[i][j][ok] = 1`
altfel

dacă **`j = k`**
`c[i][j][ok] = 25 * c[i + 1][1][true]` La poziția **`i + 1`** începe o nouă secvență repetabilă. Litera repetabilă trebuie să difere de litera de la poziția i și sunt 25 de litere disponibile.
altfel
`c[i][j][ok] = c[i + 1][ j + 1][ok]` (continuăm cu litera anterioară)
`+ 25 * c(i + 1][1][ok]` (începe o nouă secvență (25 posibilitati pentru prima literă))

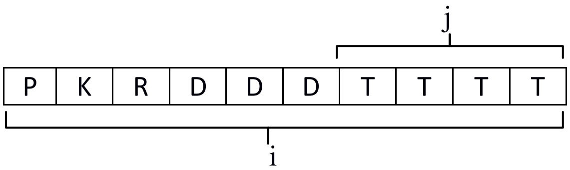
Soluția problemei se obține ca suma:

`Sol = c[1][1][false] + c[i][1][false] + …+ c[n – k + 1][1][false];`

Dezavantaj: Consum mare de spațiu de memorie.

Soluție O(n*n)

Programare dinamică. Definim șirul **`c`**, cu semnificația: **`c[i]`** = nr posibilități de a construi un cuvant de lungime FIX **`i`**, astfel încât în interior nici o secvență formată din litere identice să nu depășească lungimea **`k`**.
Să presupunem că un asemenea cuvânt se termină cu o secvență de lungime **`j`** (**`1 <= j <= k`**) de litere identice. Atunci din fiecare cuvânt de lungime **`i - j`** se pot crea **`25`** de cuvinte de lungime **`i`**.



`c[i]` se obține deci prin însumarea: **`c[i] += 25 * c[i - j]`**,
cu **`1 <= i <= n, 1 <= j <= min(k, i)`**

Însumând **`sum1`** valorile din șirul c, se obțin numărul de cuvintele de orice lungime, având *secvențe repetabile maximale* de lungimi **cel mult** k.
Pentru a determina cuvintele având *secvențe repetabile maximale* **strict mai mici** decat k, se repetă dinamica și algoritmul de mai sus, pentru k = k – 1, obținându-se `sum2` = numărul de cuvintele de orice lungime, având *secvențe repetabile maximale* de lungimi **strict mai mici** decât k.
Soluția finală este: **`sum1 – sum2`**.

Soluție O(n)

Metoda este aceeași cu cea descrisă la soluția anterioară, doar ca în plus, odată cu șirul **`c`** se construiește un alt șir **`sp`**, al sumelor parțiale, cu `sp[i] = c[0] + ... + c[i]`. Aceasta scade complexitatea la O(n).

Algoritmul

```
Pentru fiecare i de la 1 la n
        Dacă i > k
                c[i] += 25 * (sp[i - 1] - sp[i - k - 1])
        altfel
                c[i] += 25 * sp[i - 1]

        sp[i] = sp[i - 1] + c[i]
```