Descrierea soluţiei - cmmstr

Autor: Ionel-Vasile Piţ-Rada,
Prof. C.N. "Traian"
Drobeta Turnu Severin

Solutia 1

In sirul final fiecare litera va aparea exact o data.

Prima litera a sirului final va fi selectata de pe cea mai din stanga pozitie a ei (ca sa lase la dreapta cat mai mult spatiu de cautare pentru celelalte), fie j aceasta, si astfel incat la dreapta ei sa se gaseasca fiecare dintre celelalte litere care apar in sirul dat. Pentru a fi siguri ca la dreapta pozitiei j avem toate celelalte litere vom determina mai intai cea mai mare pozitie r astfel incat in sirul a[r]..a[n-1] sa apara toate literele si apoi vom determina 0≤j≤r . Se repeta aceeasi idee si pentru celelalte litere.

Se citeste sirul a[0]..a[n-1] si se determina multimea $M=\{c_1,c_2,...c_m\}$ a cifrelor care apar in sirul citit a.

In m etape, la fiecare etapa vom avea de determinat in sirul a[p]..a[n-1] cea mai mica litera x din multimea de litere M, x avand proprietatea ca exista o pozitie j astfel incat a[j]=x si sirul a[j+1]..a[n-1] contine toate celelalte litere M-{x}. Pozitia j va fi aleasa cat mai mica. Dupa ce pozitia j a fost aleasa litera x va fi afisata si apoi stearsa din multimea M , iar cautarea va continua in sirul a[j+1]..a[n-1].

Fie $M=\{c_1,c_2,...c_m\}$
Pas1. Se porneste cu p=0 apoi
Pas2. Cat timp M este nevida
  Pas2.1 Se determina cel mai mare r pentru care secventa a[r..n-1] contine toate cifrele din M
  Pas2.2 Se determina a[j]=min{a[i] / p≤i≤r si a[i]apartine lui M} astfel incat j sa fie minim
  Pas2.3 Se afiseaza a[j]
  Pas2.4 p=j+1

Complexitatea este O(m·n). Deoarece 0<m≤26 se poate spune ca avem complexitatea O(n) cu constanta m.

Solutia 2

Are la baza aceeasi idee , dar s-a schimbat mecanismul de asigurare a proprietatii ca la dreapta pozitiei j se afla toate literele din M.

Se determina la inceput pentru fiecare i cu 0≤i<n b[i]=pozitia urmatoarei litere egale cu a[i] sau b[i]=-1 daca nu mai exista la dreapta litera egala cu a[i] .

Se mai determina simultan cu valorile b[i] valorile start[j] si stop[j], pentru fiecare j cu 0≤j<26 astfel incat start[j]=prima pozitie din sirul a si respectiv stop[j]=ultima pozitie din sirul a, unde apare caracterul 'a'+j. Initial avem start[i]≥0 pentru orice 0≤i<26.

In sirul M depunem in ordine strict crescatoare toate literele care apar in sirul a, M[0], M[1], ..., M[m]. Vom avea 0<m<26.

In m etape, la fiecare etapa vom determina in sirul a[p]..a[n-1] (la inceput p=0) , in ordine strict crescatoare, prima litera M[i] cu proprietatea ca pentru toate celelalte litere M[j] avem stop[M[j]]>start[M[i]], adica toate celelalte litere M[j] se afla la dreapta literei M[i]. Deoarece litera

M[i] este minima ea este afisata. Bineinteles in continuare vom cauta litere la dreapta pozitiei start[M[i]], adica in cuvantul a[start[M[i]]+1]..a[n-1]. Va trebui sa ne asiguram ca pentru fiecare $j \neq i$ si $0 \leq j < m$ avem start[M[j]]>start[M[i]] si, pentru aceasta, vom folosi valorile b[], repetand convenabil instructiunea start[M[j]]=b[start[M[j]]]. Stergem apoi M[i].
Deoarece sirul dat a[0]..a[n-1] este parcurs de 2 ori in timpul algoritmului avem O(n) la care se adauga numarul total de operatii de determinare a literei minime M[i] care este cel mult egal cu $m(m-1)+((m-1)(m-2)+...+2\cdot 1=(m-1)m(m+1)/3$ adica $O(m^3)$. Deoarece $0<m^3<=5^6=15625$ putem considera ca $O(m^3)=O(1)$.
Avem astfel complexitatea finala O(n) cu constanta 2.
Deci solutia 2 are constanta mai buna.

Solutia 3

Solutia propusa de d-nul prof. Marius Nicoli este asemanatoare cu solutia 2, dar imbunatateste constanta complexitatii algoritmului utilizand cautari binare in sirurile pozitiilor literelor ramase in discutie la un moment dat. In acest caz , intrucat in loc de parcurgere secventiala se utilizeaza cautari binare constanta va fi 1 deoarece in locul unei parcurgeri complete a sirurilor de pozitii se vor efectua $26\cdot\log(n)$ deplasari ($\log(n)<=22$).

Solutia 4

Acest algoritm nu imbunatateste timpul de executie al solutiilor 2 si 3 , dar imbunatateste volumul de memorie rezervata. Se pastreaza intr-un vector, ordonat strict crescator in raport cu valorile nenegative, poz[c] cea mai mica pozitie unde apare in sir litera c. In momentul cand la pozitia i apare un caracter nou c2 (unde aveam poz[c2]==-1) inlocuim poz[c2]=i si la toate pozitiile mai mari ca c2 punem -1. In acelasi timp determinam momentul cand un caracter c3 apare pentru ultima data la pozitia j (s[j]=c3). In acel moment vom afisa din vectorul poz toate caracterele mai mici sau egale cu c3. Algoritmul continua pana cand se vor afisa toate caracterele.