

CNER CODE, IAȘI
CLASA A X-A
DESCRIEREA SOLUȚIILOR

COMISIA ȘTIINȚIFICĂ

PROBLEMA A: SEGALT

Propusă de: Andrei Boacă, Colegiul Național "Emil Racoviță", Iași

Observație generală.

Date fiind condițiile în care se efectuează operațiile, putem concluziona că orice operație este reversibilă (orice adăugare poate fi imediat ștersă și invers), și mai mult de atât, orice operație efectuată pe un șir își are un echivalent pe șirul celălalt. Cunoscând acest fapt, putem considera că pentru primele 3 subtask-uri vom efectua doar adăugări.

Subtask-ul 1. $S \leq 3000$ și dacă perechea este egalabilă, există o transformare cu cel mult o operație:

Pentru rezolvarea acestui subtask este suficient să încercăm fiecare operație posibilă, apoi să verificăm dacă cele două șiruri sunt egale.

Subtask-ul 2. $S > 3000$ și dacă perechea este egalabilă, există o transformare cu cel mult o operație:

Acest subtask se rezolvă în mod similar doar că, pentru a verifica eficient dacă două șiruri sunt egale în urma unei operații, va trebui să precalculăm cel mai lung prefix comun, respectiv cel mai lung sufix comun al șirurilor inițiale, pentru a nu fi nevoie să parcurgem sufixele și prefixele de fiecare dată, ci doar cele 3 litere adăugate.

Subtask-ul 3. $N_{max} \leq 8, S \leq 800$ și dacă perechea este egalabilă, există o transformare cu cel mult două operații:

Pentru rezolvarea acestui subtask se poate aplica un algoritm de tip succesor care generează toate posibilitățile de efectuare a celor două operații.

Subtask-ul 4. fără restricții suplimentare:

Fie S_A, S_B, S_C numărul de A-uri, de B-uri, respectiv de C-uri din șirul S , iar T_A, T_B, T_C frecvențele pentru șirul T . Considerăm că în urma efectuării unei operații asupra șirului S vom obține un șir P , ale cărui respective frecvențe vor fi P_A, P_B, P_C . Vom demonstra că $S_A - P_A \equiv S_B - P_B \equiv S_C - P_C \pmod{3}$:

- Dacă operația este de adăugare sau ștergere a șirului AAA , atunci $P_A = S_A \pm 3, P_B = S_B$ și $P_C = S_C$, deci $S_A - P_A \equiv S_B - P_B \equiv S_C - P_C \equiv 0 \pmod{3}$. Acest lucru se întâmplă și în cazul șirurilor BBB și CCC .
- Dacă operația este de adăugare sau ștergere a șirului ABC sau BAC , atunci $P_A = S_A \pm 1, P_B = S_B \pm 1, P_C = S_C \pm 1$, deci $S_A - P_A \equiv S_B - P_B \equiv S_C - P_C \equiv \pm 1 \pmod{3}$.

Așadar, o condiție necesară este ca $S_A - T_A \equiv S_B - T_B \equiv S_C - T_C \pmod{3}$. Vom demonstra că această condiție este și suficientă prin construirea unei succesiuni de operații care să egaleze orice două șiruri ce respectă proprietatea menționată. Pentru a reuși acest lucru, vom folosi o serie de transformări dintre cele care urmează:

- (1) $C \leftrightarrow CCCC \leftrightarrow ABCCCC \leftrightarrow ABCC \leftrightarrow ABABCC \leftrightarrow ABAB$
- (2) $AB \leftrightarrow ABCCC \leftrightarrow CC \leftrightarrow BACCC \leftrightarrow BA$

Prin urmare, putem să transformăm orice C în $ABAB$, apoi să aducem toate A -urile în fața tuturor B -urilor și să le eliminăm câte 3 cât de mult se poate. Deci S_A va deveni $S_A + 2 \cdot S_C$, S_B va deveni $S_B + 2 \cdot S_C$, T_A va fi $T_A + 2 \cdot T_C$, iar T_B va fi $T_B + 2 \cdot T_C$. Deci, dacă reușim să obținem $S_A + 2 \cdot S_C \equiv T_A + 2 \cdot T_C \pmod{3}$, respectiv $S_B + 2 \cdot S_C \equiv T_B + 2 \cdot T_C \pmod{3}$, am găsit un mod de a egala șirurile. Cunoaștem faptul că $S_A - T_A \equiv S_B - T_B \equiv S_C - T_C \equiv K \pmod{3}$ (K este o notație).

$$\implies (S_A - T_A) + 2 \cdot (S_C - T_C) \equiv 3 \cdot K \equiv 0 \pmod{3} \implies S_A + 2 \cdot S_C - T_A - 2 \cdot T_C \equiv 0 \pmod{3} \implies S_A + 2 \cdot S_C \equiv T_A + 2 \cdot T_C \pmod{3}. \quad (1)$$

$$\implies (S_B - T_B) + 2 \cdot (S_C - T_C) \equiv 3 \cdot K \equiv 0 \pmod{3} \implies S_B + 2 \cdot S_C - T_B - 2 \cdot T_C \equiv 0 \pmod{3} \implies S_B + 2 \cdot S_C \equiv T_B + 2 \cdot T_C \pmod{3}. \quad (2)$$

Din (1) și (2), putem concluziona că două șiruri ce respectă condiția impusă vor putea fi mereu egale, deci tot ce trebuie să verificăm în program este ca $S_A - T_A \equiv S_B - T_B \equiv S_C - T_C \pmod{3}$.

PROBLEMA B: SOS

Propusă de: Răileanu Alin-Gabriel, Colegiul Național "Emil Racoviță", Iași

Subtask-ul 1. $1 \leq N \leq 100$

Pentru rezolvarea acestui subtask este suficientă alegerea tuturor indicilor posibili i, j ($i < j$) și verificarea în timp liniar a compatibilității fabricilor din subsecvența $[i, j]$.

Verificarea se poate face atât prin utilizarea unui vector de frecvență, cât și utilizând următoarea observație:

- Cu siguranță ingredientul care va da compatibilitatea fabricilor din subsecvență este unul dintre ingredientele fabricii i . Așadar, putem verifica compatibilitatea în funcție de oricare dintre aceste ingrediente, complexitatea fiind $O(\lg \cdot N)$ (\lg - cardinalul mulțimii de ingrediente ale fabricii i), aspect care nu schimbă semnificativ timpul de execuție, deoarece \lg are valori foarte mici.

Complexitate teoretică:

- Spațiu: $O(n)$
- Timp: $O(\lg \cdot n^3)$

Subtask-ul 2. $100 < N \leq 1000$

Pentru rezolvarea acestui subtask se poate utiliza soluția de la subtask-ul precedent, cu observația că pentru un i setat, numărul de indici j ($i < j$) pentru care subsecvența $[i, j]$ este compatibilă, se poate determina în timp ce se face și verificarea.

De această dată se recomandă utilizarea vectorilor de frecvență. Vom calcula:

- $f[x]$ -frecvența ingredientului x în subsecvența $[i, j - 1]$

Când se adaugă fabrica j (implicit se va incrementa frecvența fiecărui ingredient deținut de aceasta), verificarea constă în găsirea unui ingredient din mulțimea fabricii j care are frecvența egală cu $(j - i + 1)$.

Complexitate teoretică:

- Spațiu: $O(n + m)$
- Timp: $O(\lg \cdot n^2)$

Subtask-ul 3. $1000 < N \leq 70000$

Observând că N -ul a crescut substanțial, o abordare a problemei în complexitate pătratică nu este suficient de bună. Pentru a rezolva acest subtask poate fi utilizată tehnica "Two Pointers". Vom pleca inițial cu i și j de la 1. Pe parcurs vom crește j -ul cât timp secvența formată de indicii $[i, j]$ este compatibilă. Când proprietatea nu se mai respectă, vom crește i -ul, nu înainte de a contoriza subsecvențele de tipul $[i, k]$ ($i < k < j$), care sunt în număr de $(j - i - 1)$.

Verificarea se va face similar cu cea de la subtask-ul 2, cu mențiunea că înainte de fiecare

creștere a i -ului, trebuie eliminate aparițiile ingredientelor fabricii i din vectorul de frecvență. Se observă că pentru fiecare fabrică, contorizarea ingredientelor acesteia se va face o singură dată.

Complexitate teoretică:

- Spațiu: $O(n + m)$
- Timp: $O(lg \cdot n)$

PROBLEMA C: MATRIX

Propusă de: Andrei Boacă, Colegiul Național "Emil Racoviță", Iași

Subtask-urile 1 și 2. $1 \leq N, M \leq 200$

Pentru a rezolva acest subtask, se pot simula operațiile parcurgând de fiecare dată întreaga matrice. Trebuie însă să avem grijă ca la operația 2 să reducem K la restul său la împărțirea cu 4, deoarece 4 rotații nu vor afecta cu nimic matricea.

Complexitate timp: $O(N^2 \cdot M)$

Subtask-ul 3. există doar operații de tip 1

Având în vedere că sunt doar operații de adunare pe submatrice, acest subtask se poate rezolva folosind Șmenul lui Mars pe matrice.

Complexitate timp: $O(N^2 + M)$

Subtask-ul 4. există doar operații de tip 1, 3 sau 4

După fiecare moment, se poate reține dacă liniile au fost oglindite sau nu, respectiv dacă coloanele au fost oglindite, reușind astfel să actualizăm dreptunghiurile corespunzătoare în matricea inițială și oglindind tabloul doar la finalul celor M operații, dacă este cazul.

Complexitate timp: $O(N^2 + M)$

Subtask-ul 5. fără restricții suplimentare

Aceeași soluție de la subtask-ul 3 ar putea fi extinsă pentru a rezolva întreaga problemă, însă faptul că aceste operații sunt efectuate în ordine arbitrară face ca implementarea să nu fie trivială. Vom defini astfel funcțiile $L(i, j)$ și $C(i, j)$, ambele de forma $a \cdot i + b \cdot j + c$, parametrii i și j fiind coordonatele unei celule din matricea inițială, $L(i, j)$ fiind linia la care se află celula în urma unor operații, iar $C(i, j)$ fiind coloana pe care se află respectiva celulă în urma acelor operații. Inițial, $L(i, j) = i$ și $C(i, j) = j$. Operațiile vor avea loc după cum urmează:

- Considerăm funcția $f(i, j) = f_a \cdot i + f_b \cdot j + f_c$, care se va compune cu funcțiile $f_i(i, j) = f_{i_a} \cdot i + f_{i_b} \cdot j + f_{i_c}$ și $f_j(i, j) = f_{j_a} \cdot i + f_{j_b} \cdot j + f_{j_c}$, obținând funcția $F(i, j)$ în următorul mod: $F(i, j) = f(f_i, f_j) = F_a \cdot i + F_b \cdot j + F_c$, deci $F(i, j) = f_a \cdot (f_{i_a} \cdot i + f_{i_b} \cdot j + f_{i_c}) + f_b \cdot (f_{j_a} \cdot i + f_{j_b} \cdot j + f_{j_c}) + f_c$. Prin urmare $F_a = f_a \cdot f_{i_a} + f_b \cdot f_{j_a}$, $F_b = f_a \cdot f_{i_b} + f_b \cdot f_{j_b}$ și $F_c = f_a \cdot f_{i_c} + f_b \cdot f_{j_c} + f_c$.
- Dacă operația este oglindire de linii, atunci $L(i, j)$ devine $L(N - i + 1, j)$.
- Dacă operația este oglindire de coloane, atunci $L(i, j)$ devine $L(i, N - j + 1)$.
- Dacă operația este rotire spre stânga, atunci $L(i, j)$ devine $L(j, N - i + 1)$.

Operațiile se vor realiza folosind un struct cu 3 câmpuri (a , b și c) și utilizând legea de compunere prezentată mai sus.

Complexitate timp: $O(N^2 + M)$