

Problema A. Țelină

Fișier de intrare: `telina.in`
Fișier de ieșire: `telina.out`
Limită de timp: 0.5 secunde
Limită de memorie: 16 megabytes

Țelina, sau *apium graveolens*, este o plantă din familia *apiaceae*. Poate atinge o înălțime de până la 1 metru. Frunzele sale sunt mari, penat-lobate. Florile sunt mici, de obicei având culoarea albă. Fructul său este achenă. Poate rezista până la temperaturi aproape de 0 grade Celsius. Perioada de înflorire este la începutul toamnei. Este o plantă hidrofilă. [sursă: wikipedia.ro]

Se dă un șir de N caractere $S_1S_2 \dots S_N$. Țelina citește (poate citi!) acest șir din dreapta spre stânga, începând cu caracterul de pe poziția N și terminând cu cel de pe poziția 1.

Pe măsură ce aceasta parcurge caractere, le introduce într-o stivă T (inițial vidă). În momentul în care citește caracterul A_{N-i+1} (la pasul i) îl introduce în vârful stivei T . Înainte de citi caracterul de la pasul $i + 1$ țelina poate inversa toate caracterele din stivă sau continuă la pasul următor.

Țelina vă roagă să aflați șirul minim lexicografic ce se poate forma în stiva T (primul element fiind în vârful stivei iar ultimul element în capătul stivei) la finalizarea operațiilor de la pasul N .

Date de intrare

În fișierul de intrare `telina.in` se află pe prima linie șirul S .

Date de ieșire

În fișierul de ieșire `telina.out` va conține pe prima linie șirul minim lexicografic ce se poate obține în T aplicând operațiile descrise anterior.

Restricții

$1 \leq N \leq 10^6$.

S conține doar caractere mici ale alfabetului englez.

Exemplu

<code>telina.in</code>	<code>telina.out</code>
<code>telina</code>	<code>anilet</code>

Explicație

La pasul 1, $T = a$. După citirea caracterului n stiva $T = na$. Înainte de a termina pasul 2 țelina întoarce toate elementele din T iar acum $T = an$.

După cei 6 pași $T = anilet$.

Problema B. Coliziune

Fișier de intrare: `coliziune.in`
Fișier de ieșire: `coliziune.out`
Limită de timp: 5 secunde
Limită de memorie: 4 megabytes

Agencia de securitate națională urmărește îndeaproape concursul RoTopCoder. Aceasta a descoperit că în cadrul concursului se află concurenți care au reușit să identifice un algoritm eficient pentru determinarea coliziunilor unei funcții \mathcal{H} . O coliziune reprezintă 2 valori x, y pentru care $\mathcal{H}(x) = \mathcal{H}(y)$.

Este un lucru binecunoscut în criptologie (știință ce se ocupă cu studiul codurilor) că această problemă este una dificilă (pentru o funcție \mathcal{H} bine aleasă).

Fie o funcție \mathcal{H} ce primește ca date intrare șiruri de lungime variabilă conținând doar cifre de 0 și 1 (șir binar). Considerăm un șir binar S de lungime N pentru care primul caracter se află pe poziția 1 (numerotarea cifrelor începe de la 1).

\mathcal{H} este calculată în felul următor:

$$\mathcal{H}(S) = \left(\sum_{i=1}^N (S(i) + 1)B^{N-i} \right) \mod M \quad (1)$$

unde cu $x \mod M$ am notat restul împărțirii lui x la M .

Dându-se 2 numere naturale B și M , se cere să găsiți 2 șiruri binare diferite x, y , astfel încât $\mathcal{H}(x) = \mathcal{H}(y)$.

Date de intrare

În fișierul de intrare `coliziune.in` pe prima linie se vor afla 2 numere întregi M și B (cu semnificația descrisă în enunț) separate prin spațiu. ($2 \leq B \leq M - 2$, $4 \leq M \leq 10^{14}$)

Date de ieșire

În fișierul de ieșire `coliziune.out` se vor afla 2 șiruri binare x, y separate printr-un spațiu astfel încât $\mathcal{H}(x) = \mathcal{H}(y)$.

Precizare

Dacă sunt mai multe soluții atunci puteți afișa oricare dintre ele atât timp cât lungimea șirului x , respectiv lungimea lui y nu depășește 10^6 .

Exemple

<code>coliziune.in</code>	<code>coliziune.out</code>
10 3	11000000 01101110010
4 2	1100 1110

Explicație

Se observă că pentru cel de-al 2-lea exemplu $\mathcal{H}(1100) = 2$ și $\mathcal{H}(1110) = 2$. Deci am găsit 2 șiruri binare diferite 1100, 1110 pentru care funcția întoarce aceleași rezultate.