Quack

Autor: stud. Ignat Alex-Matei, Universitatea Babeș-Bolyai, Cluj-Napoca

Problema cere găsirea unui șubșir care să fie prefix, sufix, și în interiorul șirului inițial s. O observație foarte importantă este că sufixul și prefixul nu se consideră în interior, altfel răspunsul ar fi fost unul evident. Din moment ce dimensiunea șirului este una mare, trebuie să căutăm anumite strategii pe baza cărora să determinăm eficient rezultatul, fără prea multă memorie folosită.

În continuare, vom aborda strategiile necesare fiecărui subtask.

Subtask 1. $3 \leq |s| \leq 10^2$.

Pentru acest subtask, putem aborda o strategie *Brute Force*, în care să luăm toate prefixele posibile, le comparăm cu sufixele corespunzătoare și totodată le căutăm în șirul de caractere. Îl reținem pe cel de lungime maximă și îl afișăm, sau mesajul corespunzător dacă nu există.

Complexitate: $O(|s|^3)$

Subtask 2. $3 \leq |s| \leq 10^4$.

Plecând de la ideea anterioară, putem face anumite optimizări, care deși în teorie ar părea că nu optimizează mult algoritmul, în practică se comportă mult mai bine. Începem cu șirurile de lungime maximă, astfel când găsim șirul cerut, ne oprim fără a mai continua căutarea. Totodată, compararea a două șiruri se poate opri în momentul în care găsim două caractere diferite la comparare. Pentru a optimiza memoria, reținem la fiecare pas doar șirurile care ne interesează, fără a stoca toate șirurile posibile. Se pot folosi *String-urile* din *STL* pentru ca aceste operații să fie implementate foarte rapid și eficient. Aceste optimizări ar trebui să fie suficiente pentru a obține punctajul pe aceste teste.

Complexitate: $O(|s|^2)$

Subtask 3. Șirul este format doar din caracterele a și b.

Putem considera șirul ca fiind unul binar, în care caracterul a reprezintă valoarea 0, iar caracterul b reprezintă valoarea 1. Cu această observație, putem așadar crea hash-uri pe șirul s, folosind sume parțiale pentru a determina rapid hash-ul unui subșir. Astfel, verificarea egalității se poate face în timp constant. Pentru a optimiza căutarea șirului de lungime maximă în interior, putem reține toate valorile sumelor parțiale, iar prin căutare binară sau folosirea unor structuri de date specifice putem determina dacă un anumit șir de o anumită lungime se află sau nu în interior (căutând o sumă parțială complementară, din hash-ul căreia să obținem ceea ce căutăm).

Complexitate: $O(|s|)$ sau $O(|s| \cdot log\,|s|)$

Subtask 4. Nu există restricții suplimentare.

Din moment ce dimensiunea șirului este mare, trebuie să folosim un algoritm specific care ar putea determina prefixul de lungime maximă care este și sufix. Un astfel de algoritm este KMP, cu ajutorul căruia putem determina acest șir. Totodată, în construirea tabelului specific algoritmului KMP, valorile din tabel vor reține pentru fiecare poziție din șir, care este lungimea maximă a unui prefix care se încheie cu acea poziție.

După construirea tabelului (fie acesta notat cu p), vom căuta în șir valoarea $p[|s|]$. În cazul în care această valoare nu există, asemănător cu ideea de la KMP, vom căuta în șir valoarea $p[p[|s|]]$. Acest lucru continuă până când ajungem la valoarea 0, caz în care se afișează mesajul corespunzător negăsirii unui șir.

O soluție alternativă ar fi preluarea ideii de la subtask-ul anterior și crearea unor hash-uri. Însă, din moment ce caracterele noastre sunt literele mici ale alfabetului englez, nu putem doar să creem numerele pe baza valorii literii, pentru că acest hash este unul cu multe coliziuni (multe șiruri care de fapt vor fi distincte, vor avea același hash). Pentru a asigura puține coliziuni, putem să lucrăm cu puteri ale unor numere prime în loc de valorile literelor, alegând totodată și un număr prim corespunzător pentru modulo.

Complexitate: $O(|s|)$

Notă.

Deși operațile cu *string-uri* sunt liniare în lungimea acestora, în practică se comportă mult mai bine.

Mugurel vă mulțumește pentru ajutor și promite să vă ofere o rață de aur la următoarea problemă.