Mugur

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

Cerința problemei este simplă: având la fiecare pas un șir de operații de *push* și *pop* pe o **stivă**, care este elementul de pe vârful **stivei**? La fiecare pas, apare o nouă operație în acest șir, însă poate să apară și în **interiorul** șirului, deci va trebui să abordăm anumite strategii cu ajutorul cărora să determinăm eficient rezultatul la fiecare pas, fără a relua operațiile de la început.

În continuare, vom aborda strategiile necesare fiecărui subtask.

Subtask 1. Indicii operațiilor sunt în ordine crescătoare.

Având indicii în ordine crescătoare, putem pur și simplu să executăm operațiile în ordinea în care ne sunt date și verificăm ceea ce avem în vârful stivei sau dacă este goală.

Complexitate: $O(N)$

Subtask 2. Toate operațiile sunt de tip *push*.

Din moment ce toate operațiile sunt de tip *push*, singurul lucru care ne interesează este operația cu indicele maxim, aceea fiind cea care determină vârful stivei. La fiecare pas actualizăm indicele maxim și totodată reținem elementul acelei operații.

Complexitate: $O(N)$

Subtask 3. $N \leq 5000$.

Numărul de operații fiind mic, putem să reținem operațiile în ordine și să le executăm pe stivă. Pentru a nu le sorta la fiecare pas, putem să reținem indicii cu ajutorul unui *set* din *STL*, la fiecare pas inserând noul indice, iar apoi parcurgând indicii care vor fi deja în ordine crescătoare.

Complexitate: $O(N^2)$

Subtask 4. Toate operațiile *push* apar înaintea operațiilor *pop*.

Operațiile *push* sunt simplu de abordat: la fiecare pas ne interesează care este operația cu indicele maxim. Observația cheie pentru operațiile *pop* este că fiecare astfel de operație anulează exact o operație *push*. Deși anularea nu este concretă (poate în final nu anulează fix acea operație *push*), ideea principală este că un șir de K operații *pop* consecutive anulează primele K operații *push* din stânga lor. Deci, pentru fiecare operație *pop*, este suficient să eliminăm operația *push* cu indice maxim mai mic decât indicele operației *pop*.

Acest lucru poate fi rezolvat cu ajutorul unui *set* din *STL* în care reținem indicii operațiilor *push*. La apariția unei operații *push*, îi introducem indicele în *set*. La apariția unei operații *pop*, căutăm dacă există în *set* un indice mai mic decât cel curent și îl eliminăm în caz afirmativ (putem folosi funcția *upper_bound* care ne determină primul element mai mare și apoi putem obține adresa din stânga sa, fiind un *iterator*, pe care mai apoi îl ștergem). Pentru a determina rezultatul, trebuie să găsim indicele maxim, pe care îl putem accesa din finalul *set-ului*.

Complexitate: $O(N \cdot log\, N)$

Subtask 5. Fără restricții suplimentare.

Pentru acest subtask, trebuie să abordăm o strategie mai eficientă la fiecare pas. Vom face câteva observații cheie care ne vor ajuta în demonstrarea soluției:

Observație 1. Fiecare indice de la **1** la N apare exact o singură dată în șirul de operații.

Considerăm pentru moment fiecare operație *push* ca fiind $+$**1** și fiecare operație *pop* ca fiind **-1**. Pentru a nu ne încurca, pe indicii operațiilor care încă nu apar vom avea valoarea **0**.

Observație 2. Indicele de început pentru cel mai scurt sufix de sumă **1** (dacă există) reprezintă operația care determină vârful stivei.

Pe scurt, cel mai mare indice i astfel încât suma din intervalul $[i, N]$ este **1** reprezintă indicele operației care determină vârful stivei.

Demonstrație: Folosind idea prezentată anterior, dacă luăm de la dreapta la stânga operațiile *pop*, fiecare va anula prima operație *push* din stânga. În momentul în care dăm de o operație *push* pe care practic nu am anulat-o, avem vârful stivei. Deci, putem parcurge operațiile de la dreapta la stânga reținând suma la fiecare pas (unde operația *push* este $+$**1** și operația *pop* este **-1**), iar când suma devine **1**, avem operația necesară pentru rezultat.

Pentru a face acest lucru, putem folosi un **arbore de intervale**, care reține suma elementelor din interval și sufixul de sumă maximă.

Actualizarea se face simplu asupra elementului dat de indicele operației, cu ± 1 în funcție de tipul operației.

Fie m mijlocul intervalului $[i, j]$. Actualizarea unui interval este făcută după următoarea formulă:

$$suma_{[i,j]} = suma_{[i,m-1]} + suma_{[m,j]}$$
$$maxsuf[i,j] = max(maxsuf_{[m,j]}\,,\, maxsuf_{[i,m-1]} + suma_{[m,j]})$$

Interogarea este făcută sub forma unei căutări binare în care determinăm sufixul. Vom crea o funcție care determină cel mai din dreapta sufix dintr-un interval [i, j] cu o anumită sumă S, inițial apelând-o pentru suma $S = 1$ și intervalul $[i, j] = [1, N]$.

Fie m mijlocul intervalului $[i, j]$. Dacă intervalul din dreapta [m, j] are sufixul de sumă maximă mai mare sau egal decât suma pe care o căutăm, ne mutăm în acel interval (deoarece ne așteptăm să avem și un sufix mai mic sau egal - cel pe care îl căutăm - în acel inteval). Dacă este mai mic, înseamnă că sufixul nostru începe din intervalul din stânga, deci vom căuta în acest interval un sufix dar cu suma $S - \sum[m, j]$ (unde $\sum[m, j]$ reprezintă suma elementelor din șir pe acel interval), deoarece știm că sufixul pe care îl căutăm conține tot intervalul din dreapta, deci ne mai interesează să găsim restul sufixului în intervalul din stânga astfel încât să ajungem la suma cerută. Când ajungem la un interval format dintr-un singur element, înseamnă că am găsit indicele cerut pe care îl putem returna.

Dacă sufixul de sumă maximă pe intervalul $[1, N]$ este mai mic decât **1**, înseamnă că stiva este goală și vom afișa **-1**.

Complexitate: $O(N \cdot log\, N)$

Notă.

Există și alte modalități de a implementa o soluție de punctaj maxim, unele însă sunt puțin mai ineficiente. O altă abordare la *Subtask 5* ar fi să căutăm binar răspunsul, iar la fiecare pas să interogăm arborele obținând o complexitate $O(N \cdot (log\, N)^2)$, însă pentru restricțiile date această idee s-ar putea să nu obțină punctaj maxim.

Pentru *Subtask 3* o idee mai *„brute force"* ar fi să sortăm la fiecare pas toate operațiile obținând o complexitate $O(N^2 \cdot log\, N)$, însă din cauza restricțiilor această idee s-ar putea să nu obțină punctele pentru acel subtask.

Mugur vă mulțumește că i-ați fost alături de ziua sa și că nu l-ați lăsat să se plictisească, promite că de ziua voastră vă va aduce cadou o **coadă** ca să nu vă mai stresați cu **stive**.