

DESCRIEREA SOLUȚIILOR, OLIMPIADA NAȚIONALĂ DE INFORMATICĂ, LOT SENIORI 1

COMISIA ȘTIINȚIFICĂ

PROBLEMA : BENZI

Autor: Stud. Liviu-Mihai Sillion

Subtask 1 (7 puncte). Pentru $N, Q \leq 5\,000$ putem face brute force pentru fiecare query.

Subtask 2 (9 puncte). Putem construi un arbore de intervale unde fiecare nod va ține numărul de benzi, lungimea și culoare pentru cea mai lungă bandă și pentru prefix/sufix. Avem nevoie să implementăm doar funcție pentru Query pentru arborele de intervale.

Subtask 3 (5 puncte). În acest caz putem actualiza unul câte unul fiecare element al șirului implementând funcția Update pentru o poziție pentru arborele de intervale.

Subtask 4 (10 puncte). Funcția de Update trebuie modificată astfel încât să suporte operația clasică de Lazy Propagation cu o singură valoare.

Subtask 5 (11 puncte). Pentru fiecare valoare lazy vom ține numărul patternului și offsetul de la care începe intervalul față de pattern. Astfel, putem calcula valoare propriu-zisă dintr-un nod lazy dacă un pattern se repetă de mai multe ori în $\mathcal{O}(1)$ combinând tot patternul repetat de mai multe ori în $\mathcal{O}(1)$, un prefix și un sufix al patternului. În cazul în care patternul este mai mic decât intervalul desemnat de nodul lazy, ne va trebui răspunsul pentru o secvență care nu este nici prefix și nici sufix. Acest lucru poate fi făcut în timp constant cu precalculare brută în $\mathcal{O}(M^2)$ pentru acest subtask. În total avem $\mathcal{O}(N + M^2 + Q \cdot \log N)$

Subtask 6 (27 puncte). Vom folosi un alt arbore de intervale pentru a afla răspunsul pe subsecvențe ale patternurilor. Complexitatea va deveni $\mathcal{O}(N + M + Q \cdot \log N \cdot \log M)$. Soluțiile în $\mathcal{O}(Q \cdot \sqrt{N})$ sau cu treapuri vor trece cel mult acest subtask.

Subtask 7 (31 puncte). De fapt ceea ce ne dorim este $\mathcal{O}(1)$ pentru fiecare query pe patternuri, pentru că avem $\mathcal{O}(Q \cdot \log N)$ query-uri. Dacă operația de combinare a două stări ar fi fost idempotentă, am fi putut înlocui arborele de intervale cu RMQ.

Varianta 1 (Tamio-Vesa Nakajima). Redimensionăm șirul a astfel încât să fie putere de 2. Atunci, fiecare nod din arborele de intervale va corespunde unui interval cu lungime putere de 2. Deci, mereu când vom avea nevoie de query pe o subsecvență dintr-un pattern, lungimea va fi putere de 2 și va fi de ajuns să precalculeăm din fiecare poziție răspunsul pentru puteri de 2.

Varianta 2. O variantă ar fi să folosim structuri de date cu query în timp constant ce funcționează cu orice operație asociativă. Două astfel de structuri sunt Disjoint Sparse Table (cu $\mathcal{O}(1)$ per query și build în $\mathcal{O}(M \cdot \log M)$) și Sqrt Tree (cu $\mathcal{O}(1)$ per query și build în $\mathcal{O}(M \cdot \log \cdot \log M)$).

PROBLEMA : APA

Autor: Tamio-Vesa Nakajima

Subtask 1 (8 puncte). Putem folosi un algoritm de tip brute-force. Pentru un lanț de la u la v , luăm toate nodurile individuale și pentru fiecare nod de pe lanț putem vedea câte noduri vor avea aceeași greutate.

Vom considera nodul 1 ca fiind rădăcina arborelui. Ne calculăm s_i ca fiind mărimea subarborelui în jos pentru nodul i . Astfel, vom avea mai multe cazuri în funcție de poziția nodului:

- (1) dacă un nod nu are niciun fiu care să facă parte din lanț, atunci greutatea acestuia va fi s_i ;
- (2) dacă un nod are un fiu care se află în lanț și un tată care face parte tot din lanț, atunci greutatea acestuia va fi $s_i - s_{fiu}$;
- (3) dacă un nod are oricâți fii care să facă parte din lanț și nu are un tată care să facă parte din lanț, atunci greutatea acestuia va fi $N - S$ unde S este suma de s_{fiu} pentru fiecare fiu al nodului i care face parte din lanț.

Având greutatea calculată pentru fiecare nod, putem să le înmulțim cu înălțimile corespunzătoare și să facem suma. Această soluție are complexitatea $\mathcal{O}(N \cdot Q)$

Subtask 2 (6 puncte). Ca și la soluția anterioară, ne vom construi răspunsul folosind șirul s_i .

Fie x , y și h unul dintre query-uri. Ca să satisfacem primul nod din lanț, în cazul nostru rădăcina, vom considera că în tot arborele turnăm h litri de apă. Suma va fi inițializată astfel cu $N \cdot h$.

Continuând cu următorul nod de pe lanț, x' , vom vedea că acesta are h litri când ar trebui să aiba turnați doar $h - 1$. Așadar, din sumă vom scădea $s_{x'}$. Practic, din tot subarborele lui x' , vom scoate un litru, iar nodurile din influența lui x' vor avea acum $h - 1$ litri. Continuând cu următorul nod de pe lanț, x'' , acesta va avea volumul $h - 1$ când ar trebui să fie $h - 2$, deci din subarborele acestuia vom mai scădea $s_{x''}$ și tot așa.

Putem deduce astfel următoarea formulă:

$$N * (h + 1) - \sum s_i \text{ unde } i \text{ este fiecare nod de pe lanțul de la } x \text{ la } y.$$

A se nota că am început prin a turna $h + 1$ litri în toate nodurile pentru a avea o formulă în care nu excludem primul nod din lanț. Informația esențială pe care ne-o oferă această formulă este faptul că putem construi fragmente de lanț care să respecte cerința problemei folosind suma de s_i pe lanț. În particular, $\sum s_i$ va reprezenta răspunsul problemei cu query-ul de la x la y unde turnăm 1 litru de apă în nodul x , 2 litri de apă în nodul x' , 3 litri de apă în nodul x'' ș.a.m.d. Dacă punem minus în fața sumei, atunci vom avea o progresie descrescătoare. Pentru a ne ajusta suma să înceapă cu o anumită constantă, putem adăuga $N \cdot x$.

Un alt lucru important este faptul că trebuie să ne mutăm capătul y al lanțului în sus, astfel încât să nu ajungem să turnăm capacități negative de apă în noduri. Pentru asta, putem folosi un algoritm dfs în care menținem o stivă cu toate nodurile de la rădăcină până la nodul curent, iar dacă h este mai mic decât mărimea stivei, atunci ne vom folosi de stivă ca să aflăm ușor cum să mutăm capătul y .

Această soluție are complexitatea $\mathcal{O}(N + Q)$.

Subtask 3 (12 puncte). Ne vom calcula, ca și la subtaskul anterior, acea sumă pe lanțurile de la rădăcină până la fiecare nod pe care o vom nota cu S_{nod} . Astfel, dacă vrem să facem rapid suma de s_{nod} pe un anumit lanț de la x până la y (vom considera pentru acest subtask că nodul x este un descendent de-al lui y), vom face $S_x - S_{y'}$ unde y' este tatăl lui y .

Volumele de la x la y vor fi în ordine $l, l - 1, l - 2, \dots, 1$, unde l este lungimea lanțului. Trebuie să mai facem niște ajustări manuale, deoarece momentan avem apă doar în subarborele lui y , dar toate nodurile din afara acestui subarbore au volumul 0. Pentru a egaliza cantitatea de apă dintre nodul y și celelalte noduri din afara acelui subarbore, putem scădea 1 din toate nodurile din subarborele lui y . Astfel, suma pe care o avem până în acest moment este $S_x - S_{y'} - s_y = S_x - S_y$. În acest moment, lanțul are o progresie aritmetică de la $l - 1$ până la 0, iar toate nodurile din

afara lanțului vor avea aceleași volume ca și nodurile asociate lor din lanț. Trebuie doar să aflăm ce volum ar trebui să aibă nodul y , fie acesta H , putem ajusta progresia aritmetică, iar răspunsul va fi $S_x - S_y + N \cdot (H)$.

Ca și la soluția precedentă, trebuie să avem grijă să mutăm capătul y în caz că h este mai mic decât lungimea lanțului.

Subtask 4 (61 puncte). Pentru acest subtask, vom folosi același proces ca și la următorul subtask, doar că vom folosi structuri de date sau alți algoritmi precum Heavy-light Decomposition, sau Centroid Decomposition în complexitate $\mathcal{O}(N \cdot \log N \cdot \log N)$

O altă soluție posibilă ar fi folosirea algoritmului lui Mo pe arbore cu o complexitate $\mathcal{O}(N \cdot \sqrt{Q})$. Această soluție nu are nevoie de formule, dar este semnificativ mai complicată decât soluția de 100 de puncte. O problemă a acestei soluții este faptul că într-un query, structura de date pe care o folosim pentru a calcula răspunsul exclude și strămoșul comun, deci acesta va trebui să fie adăugat artificial.

Subtask 5 (100 puncte). Folosind subtaskurile 2 și 3, putem calcula orice lanț unde unul din noduri este strămoș al celuilalt nod. Rămâne să vedem cum se transformă formulele atunci când avem un lanț diferit de cele din subtaskurile 2 și 3.

Vom împărți lanțul în două secțiuni: de la nodul x până la L și de la L până la y , unde nodul L este strămoșul comun al nodurilor x și y . Ce dorim acum este să găsim o formulă astfel încât să producem un lanț ce are o progresie aritmetică.

Pentru lanțul de la x la L , putem aplica formula $S_x - S_L$. Aceasta va modela o jumătate de lanț cu volumele de apă $lx - 1, lx - 2, \dots, 0$ (prin lx am notat lungimea lanțului de la x la L).

Pentru lanțul de la L la y , asemănător, putem folosi $S_y - S_L$ și vom avea o progresie aritmetică de la 0 până la $ly - 1$ (prin ly am notat lungimea lanțului de la L la y). Pentru a avea valori negative, putem să adăugăm un minus în fața formulei: $S_L - S_y$.

Dacă însumăm cele două valori, practic o să avem lanțul cu volumele de apă $lx - 1, lx - 2, \dots, 1, 0, -1, -2, \dots, -(ly - 1)$. Putem observa că cei doi subarbori care se întâlnesc la L sunt total independenți, deci formula este corectă. Pentru nodul L , volumul acestuia va fi 0, iar toate celelalte noduri care nu se află în niciun subarbor sunt inițializate cu 0, deci formula este corectă. Suma celor două formule va fi $(S_x - S_L) - (S_L - S_y) = S_x - S_y$. Ca și la soluțiile anterioare, este necesar să aflăm care este volumul pe care trebuie să îl aibă unul dintre noduri și să adăugăm $d \cdot N$ unde d este diferența de volume. De asemenea, trebuie să mutăm capătul y în caz că h este mai mare decât lungimea lanțului.

Pentru a ne putea deplasa ușor pe arbore, putem folosi tehnica de binary lifting (cunoscut și sub numele de "șmenul de la strămoși") pentru a calcula strămoșul comun a două noduri sau pentru a afla al k -lea strămoș al unui nod. Complexitatea acestei soluții este $\mathcal{O}((N + Q) \cdot \log N)$.

Putem folosi și tehnici offline de a calcula strămoșii comuni ai unui nod, ceea ce ar duce la o complexitate $\mathcal{O}(Q \cdot \log * N)$.

PROBLEMA : SIRDACIC

Autor: Stud. Liviu-Mihai Silion

Problema se reduce la a număra câte șiruri sunt fără 2 elemente adiacente egale, considerând numerele modulo k și rămânând doar de înmulțit cu un factor dat de permutări cu repetiții.

Subtask 3 (51 puncte). Ne vom utiliza de o dinamică pe "componente", construind soluția adăugând pe rând valori noi. În continuare vom defini o componentă ca fiind o subsecvență maximală cu toate elementele egale.

Suntem la o valoare v care apare de cnt ori în șir. Avem:

$dp[i]$ = numărul de șiruri format din valorile curente care au i componente.

$dp'[i]$ definit analog pentru șiruri formate din valorile de până acum, fără v .

La fiecare pas, va trebui să vedem în câte componente j împărțim cele cnt valori (Stars and Bars), în câte moduri putem sparge l componente de până acum, respectiv cum adăugăm $j - l$ componente la restul.

Alternativ, ne putem defini dinamica după numărul de elemente adiacente egale. Atunci am "despărți" elementele adiacente, și am avea recurența (unde n numărul total de elemente ale șirului până acum și n' analog, dar fără valorile v)

$$dp[i] = \sum_j \sum_l dp'[i + (cnt - j) - l] \times \binom{cnt-1}{j-1} \times \binom{i}{l} \times \binom{n'+1-i}{cnt-l}$$

unde i este numărul de valori adiacente egale și cu răspunsul în $dp[0]$ după procesarea tuturor valorilor.

Subtask 4 (100 puncte). Vom adăuga naiv componentele noi, ignorând momentan faptul că ele vor genera conflicte unele cu celelalte. Obținem următoarea recurență, cu mențiunea că $dp[i]$ va număra șirurile cu $\leq i$ componente:

$$dp[i+j] = \sum_j dp'[i] \times \binom{cnt-1}{j-1} \times \binom{i+j}{i}$$

Să presupunem că vrem să numărăm stringurile binare cu 4 de 1 și 4 de 0 care nu au două valori adiacente egale. Notăm cu r_i stringurile care au exact i componente.

	r_8	r_7	r_6	r_5	r_4	r_3	r_2
$dp[8]=$	2	6	18	18	18	6	2
$dp[7]=$	0	6	36	54	72	30	12
$dp[6]=$	0	0	18	54	108	60	30
$dp[5]=$	0	0	0	18	72	60	40
$dp[4]=$	0	0	0	0	18	30	30
$dp[3]=$	0	0	0	0	0	6	12
$dp[2]=$	0	0	0	0	0	0	2

Se observă că însumând alternant $dp[i]$, nu vom număra decât valorile cerute. De fapt, soluția problemei noastre va fi mereu $dp[n] - dp[n-1] + dp[n-2] - \dots$.

Vom arăta acest lucru folosindu-ne de perspectiva numărării elementelor adiacente egale în locul componentelor. Avem $dp[i]$ numărul de șiruri cu $\geq i$ elemente adiacente egale și $r[i]$ numărul de șiruri cu exact i elemente adiacente egale.

Observăm că în $dp[i]$, vom număra $r[j], j \geq i$ de exact $\binom{j}{i}$ ori, pentru că noi în realitate am numărat șirurile privindu-le ca având (alegând) doar i valori adiacente egale dintr-un total de j .

$$dp[i] = \sum_{j \geq i} r[j] \times \binom{j}{i}.$$

$$\sum_{i \geq 0} dp[i] \times (-1)^i = \sum_{i \geq 0} (-1)^i \times \sum_{j \geq i} r[j] \times \binom{j}{i} = \sum_{j \geq 0} r[j] \times \sum_{0 \leq i \leq j} \binom{j}{i} \times (-1)^i = \sum_{j \geq 0} r[j] \times (1 + (-1))^j = r[0] \times 0^0 + r[1] \times 1^0 + \dots = r[0].$$