Lotul Național de Informatică 2024, Orăștie
Seniori, Proba 4 de baraj
Descrierea Soluțiilor

Comisia Științifică

28 mai 2024

Subiectul I („Nervi”)

Problema aceasta este relativ simplă ca idee de soluție, dar implementarea poate fi fie simplă fie mai anevoioasă. Vom descrie mai întâi ideea de bază a soluției, apoi o implementare curată.

Idee soluție

Ne fixăm centrul spiralei și vedem cât de mult ne putem extinde. Putem să ne fixăm sensul de mers și prima direcție pe care o luăm și să ne deplasăm în spirală atâta timp cât valorile descresc. Această soluție are complexitate $O(N^4)$.

Pentru a nu parcurge linii/coloane întregi de elemente atâta timp cât descresc, ne putem precalcula pentru fiecare poziție cât de mult putem merge în sus/jos/dreapta/stânga și să meargă doar descrescător. Această soluție are complexitate $O(N^3)$.

Pentru a nu mai parcurge toate liniile și coloanele, grupăm mai multe linii/coloane în funcție de sensul de mers. Practic, spirala aceea este descompusă în patru cadrane în formă de triunghiuri dreptunghice, iar pentru centrul spiralei, vrem să calculăm cât de mult se pot extinde cadranele respective. Putem calcula acest lucru cu o dinamică de tipul: $dp_{i,j}$ = mărimea maximă a unui triunghi dreptunghic cu vârful în (i, j) și recurența ar fi $dp_{i,j} = 1 + \min dp_{i-1,j-1}, dp_{i-1,j}, dp_{i-1,j+1}$ dacă pe (i, j) este în creștere sau $dp_{i,j} = 0$ în caz contrar (acest caz este pentru un triunghi care se extinde în sus). Această soluție merge în $O(N^2)$.

Implementare cu rotații și oglindiri

Toate soluțiile de mai sus pot fi implementate calculînd și precalculînd datele necesare separat pentru fiecare direcție. Această implementare va fi redundantă și pasibilă de erori de copy-paste. Iată și o implementare diferită.

Numărăm spiralele pornind din centru, pe valori necrescătoare. Le numărăm doar pe cele care pornesc spre est, apoi spre sud (așadar în sens orar). Pentru a le număra pe cele în sens orar, dar în celelalte trei direcții, rotim matricea cu cîte 90°. Pentru a le număra pe cele în sens antiorar, oglindim matricea față de orice axă aleasă și repetăm cele patru rotații.

Pentru spiralele de latură impară, centrul este bine definit. Spirala maximă de mărime impară care pornește dintr-un centru dat este maximul celor opt posibilități de mai sus, pe care îl reținem pe măsură ce iterăm prin rotații și oglindiri.

Pentru spiralele de latură pară este nevoie de un caz particular. Cînd rotim matricea cu 90°, valoarea maximă a laturii unei spirale pare trebuie mutată din vechiul centru într-o celulă vecină (de exemplu, cu o linie mai jos în

cazul rotirii în sens antiorar). Această observație se aplică și la soluția în $O(N^2)$, cu precalculări pe triunghiuri, și la soluția în $O(N^3)$, cu precalculări pe segmente.

Subiectul II („Toni")

Pentru început, vom calcula lis_i = lungimea celui mai lung subșir crescător care începe la poziția i. Dacă j este critic pentru i, atunci orice subșir crescător de lungime maximă care începe în i va avea ca sufix de lungime lis_j un subșir crescător de lungime maximă care începe în j.

Așadar, notăm cu $next_i$ cel mai mic j cu proprietatea că $i < j$ și elementul p_j este critic pentru p_i. Este clar că răspunsul pentru i va fi răspunsul pentru j la care se adaugă 1. Practic, elementele critice pentru indexul i sunt $next_i, next_{next_i}, \ldots$.

Putem privi aceste relația dintre i și $next_i$ ca o muchie într=un arbore T. Este clar că $next_i$ este primul element comun al subșirurilor crescătoare de lungime maximă care încep în j, unde $j > i, lis_i = lis_j - 1$ și $p_j > p_i$. Primul element comun (prin urmare, critic) al tuturor acestor subșiruri este cel mai apropiat strămoș comun al tuturor j-urilor cu proprietatea de mai sus.

Putem rezolva acum problema în $O(N \log^2 N)$ cu o structură de date care acceptă adăugarea unui element pe o poziție în șir și interogare de lca pe sufix. Dacă această soluție este implementată atent, ea poate obține punctaj maxim.

Pentru a rezolva problema cu o complexitate mai bună, putem să facem următoarea observație: fie L_1 subșirul minim lexicografic (din punct de vedere al indicilor) de lungime maximă care pleacă din i și L_2 subșirul maxim lexicografic (din punct de vedere al indicilor) de lungime maximă care pleacă din i, Dacă există un element j care apare atât în L_1, cât și în L_2, atunci j este critic. Folosind această observație, complexitatea se poate reduce la $O(N \log N)$.

Subiectul III („Single Crossing")

În primul rând, vom elimina permutările duplicate (ele pot fi readăugate la răspuns cu ușurință). Acest lucru se poate realiza în $O(NM \log NM)$ sau $O(NM)$ în funcție de implementare.

Să ne imaginăm o instanță a problemei pentru care există soluție, pentru simplitate chiar permutarea identitate $P = [1, 2, \ldots, N]$. Fără a restrânge generalitatea, putem presupune că prima permutare este $X^1 = [1, 2, \ldots, M]$. Altfel, am putea reindexa numerele $1, \ldots M$ pentru a ajunge în acest caz. Acum, proprietatea *single-crossing* se reduce la una mai simplă: pentru orice $1 < i \leq N$, dacă X^i are o inversiune, atunci și toate permutările de după trebuie să o aibă. Aceasta condiție este și suficientă.

Mai mult, dacă notăm cu $\text{inv}(P)$ numărul de inversiuni ale unei permutări P, atunci vom avea că $0 = \text{inv}(X^1) \leq \text{inv}(X^2) \leq \ldots \leq \text{inv}(X^N)$. Cum permutările sunt distincte două câte două, avem mai mult că $0 = \text{inv}(X^1) < \text{inv}(X^2) < \ldots < \text{inv}(X^N)$. Aceasta deja indică o soluție de punctaj parțial: fixăm in $O(N)$ toate variantele pentru P_1, reindexăm numerele de la 1 la M astfel încât X^{P_1} să fie permutarea identitate (cu alte cuvinte, compunem toate cele N permutări cu inversa lui X^{P_1}), apoi calculăm pentru fiecare permutare numărul de inversiuni și le ordonăm crescător după numărul de inversiuni. Numărul de inversiuni ale unei permutări se poate calcula in $O(M \log M)$ cu un arbore indexat binar (AIB) (sau folosind o variație a algoritmului Mergesort). După ce am ordonat permutările astfel, rămâne doar să verificăm dacă permutarea P astfel rezultată face ca proprietatea de single-crossing să fie respectată. Momentan, acest lucru se poate face in $O(NM^2)$ verificând pentru fiecare pereche de elemente (a, b) cu $1 \leq a < b \leq M$ dacă există un 'cross' pe acea pereche în ordinea dată de P. Vom reveni ulterior la cum se poate face această verificare mai eficient. Pentru moment, soluția de față are complexitate $O(N^2 M \log NM + NM^2)$.

Acum, să găsim o metodă mai eficientă de a identifica prima permutare (adică X^{P_1}). Pentru aceasta, alegem arbitrar una dintre permutări, să zicem X^1. În permutarea soluție P, este posibil ca X^1 să nu fie prima. Cu

toate acestea, să reindexăm numerele $1, \dots M$ astfel încât X_1 să fie permutarea identitate. Ca rezultat, dacă măsurăm numărul de inversiuni ale tuturor permutărilor, permutarea (sau permutările) care au număr maxim de inversiuni trebuie să se afle ori la începutul, ori la finalul lui P. Dacă observăm in plus că dacă P este soluție atunci si permutarea $\text{reverse}(P)$, obținută răsturnând șirul P, este soluție, obținem că putem alege arbitrar una dintre permutările cu număr maxim de inversiuni ca fiind prima în soluția P. Astfel, soluția anterioară capătă complexitatea $O(NM \log NM + NM^2)$, deoarece acum calculăm eficient valoarea P_1, dar încă nu știm cum să verificăm eficient dacă permutarea P rezultată în urma sortării este soluție. Descriem în cele ce urmează cum acest lucru se poate realiza în $O(NM \log M)$, acesta fiind și ultimul ingredient necesar pentru rezolvarea de 100 de puncte a problemei.

Să zicem, pentru simplitate, că vrem sa verificăm dacă permutarea $P = [1, 2, \dots, N]$ este soluție. Pentru o descriere mai elegantă a soluției, vom folosi conceptul de spațiu metric. Un *spațiu metric* este o mulțime M înzestrată cu o funcție $d : M^2 \to \mathbb{R}_{\geq 0}$ (numită *metrică* sau funcție de distanță) cu următoarele trei proprietăți:

1. $d(m, m) = 0$ pentru orice $m \in M$;
2. $d(a, b) = d(b, a)$ pentru orice $a, b \in M$;
3. $d(a, b) + d(b, c) \geq d(a, c)$ pentru orice $a, b, c \in M$ („*Inegalitatea Triunghiului*").

Definim un spațiu metric peste mulțimea $M = \{1, \dots, N\}$ cu metrica $d(i, j)$ = numărul de perechi $1 \leq a < b \leq M$ astfel încât a apare înainte de b în X^i și după b în X^j, sau vice versa. Cu alte cuvinte, numărul de inversiuni relative dintre permutările X^i și X^j. Se observă că $d(i, j)$ este, de fapt, chiar numărul de inversiuni ale lui X^j dacă reindexăm ambele permutări astfel încât X^i să fie permutarea identitate. Cu alte cuvinte, numărul de inversiuni ale compunerii dintre inversa lui P și permutarea Q. Pentru a demonstra că aceasta este într-adevăr o metrică, primele două proprietăți reies relativ natural, în timp ce demonstrarea inegalității triunghiului necesită o analiză mai atentă, după cum urmează. Definim, pe aceeași mulțime M, pentru fiecare pereche (a, b) cu $1 \leq a < b \leq M$ câte o metrică auxiliară $d_{a,b}$ astfel încât $d_{a,b}(i, j) = 1$ dacă perechea (a, b) este o inversiune relativă între permutările X^i și X^j, și $d_{a,b}(i, j) = 0$ altfel. Acestea sunt metrici deoarece ne putem imagina că asociem pentru fiecare X^i punctul $p^i_{a,b} = 0 \in \mathbb{R}$ dacă a apare înainte de b în X^i și punctul $p^i_{a,b} = 1 \in \mathbb{R}$ în caz contrar, astfel, metrica $d_{a,b}$ este exact distanța Euclidiana pe $\mathbb{R}$ între puncte din mulțimea $\{0, 1\}$. Observând că metrica noastră d este formata însumând metricile $d_{a,b}$ pentru toate perechile (a, b) cu $1 \leq a < b \leq M$ și folosind faptul ca suma a două metrici este tot o metrică, rezultă că d este o metrică (deci satisface, într-adevăr, și inegalitatea triunghiului).

Acum, să reanalizăm condiția de single-crossing prin prisma metricilor introduse. Pentru o pereche (a, b) cu $1 \leq a < b \leq M$, condiția de verificat se traduce la $p^i_{a,b}$ este 0 pana la un punct, și 1 de atunci încolo, sau vice-versa. Se poate verifica că acest fapt ține dacă și numai dacă pentru oricare $1 \leq i \leq j \leq k \leq N$ avem că $d_{a,b}(i, j) + d_{a,b}(j, k) = d_{a,b}(i, k)$, adică inegalitatea triunghiului ține cu egalitate pentru toate tripletele de permutări. Mai mult, pentru a verifica această condiție este suficient să se verifice doar pentru tripletele (i, j, k) de forma $(1, \ell, \ell + 1)$. Lăsăm motivația tehnică a acestui fapt ca temă cititorului.

Așadar, trebuie să verificăm dacă inegalitatea triunghiului ține cu egalitate pentru fiecare metrica $d_{a,b}$ și fiecare triplet de forma $(1, i, i + 1)$ cu $1 < i < N$. Cum metrica noastră d este formată însumând aceste metrici, este suficient să verificăm dacă $d(1, i) + d(i, i + 1) = d(1, i)$ pentru $1 \leq i < N$. Acest lucru necesită timp total $O(NM \log M)$, finalizând soluția.

Pentru o înțelegere mai profundă a soluției, invităm cititorul să verifice de ce condiția alternativă $d(1, i) + d(i, i + 1) = d(1, i)$ pentru $1 < i < N$ nu ar fi fost, în schimb, suficientă.

Înzestrați fiind cu formalismul metric al problemei, soluția problemei se poate reformula cu un grad mai sporit de eleganță. Mai exact, alegem o permutare arbitrară din cele N, măsurăm distanța față de celelalte $N - 1$ permutări și alegem una dintre cele cu distanță maximă ca fiind prima din soluție. Apoi, sortăm permutările după distanța lor față de prima din soluție. Aceasta constituie obligatoriu o soluție validă, dat fiind că o soluție

ar exista. Rămâne doar de verificat dacă soluția găsită respectă proprietatea. Pentru aceasta, verificăm dacă $d(1, i) + d(i, i+1) = d(1, i)$ pentru $1 \leq i < N$. Observăm similaritatea părții de determinare a soluției cu algoritmul de determinare a diametrului unui arbore. Aceasta asemănare nu este întâmplătoare, lăsăm explicația fenomenului ca temă cititorului.

Soluție alternativă $O(NM \log NM)$ bazată pe algoritmul Quicksort

Soluția are doi pași. Întîi construim un candidat de soluție, care are proprietatea că fie el este *single-crossing*, fie nu există soluție. Apoi verificăm dacă acest candidat este *single-crossing*.

Definim următorul comparator util pentru sortare. Date fiind trei permutări X, Y și Z, fixăm ordinea primelor două. Raportat la acestea există cel mult o poziție corectă pentru Z, din cele trei variante: stînga (ZXY), centru (XZY) și dreapta (XYZ). O găsim astfel. Căutăm prima diferență între X și Y (prima poziție i unde $X_i \neq Y_i$). Fie aceste valori a și b. Atunci a apare înainte de b în X, iar b apare înainte de a în Y. Pentru concizie, notăm aceste proprietăți cu $a <_X b$ și $b <_Y a$. Verificăm cum apar a și b în Z. Iau naștere două cazuri.

1. Dacă $a <_Z b$, atunci ordinea corectă poate fi doar ZXY sau XZY, căci XYZ cauzează o dublă trecere $ab - ba - ab$. Căutăm prima diferență între X și Z, fie $c <_X d$ și $d <_Z c$. Căutăm c și d în Y. Iau naștere două subcazuri:
   - Dacă $c <_Y d$, atunci XZY devine incorectă (dublă trecere $cd - dc - cd$), deci singura ordonare este ZXY (sau niciuna).
   - Dacă $d <_Y c$, atunci ZXY devine incorectă, deci ordonarea este XZY (sau niciuna).
2. Similar pentru $b <_Z a$, unde avem de decis între XZY și XYZ.

Cu acest comparator putem implementa un quicksort. Ca preambul, alegem două permutări la întîmplare, X și Y, și partiționăm restul permutărilor în trei grupe: la stînga lui X, între X și Y, la dreapta lui Y. Pe fiecare grupă apelăm quicksort. Observăm că la fiecare moment în recurență cunoaștem cel puțin o permutare de la marginea intervalului de sortat. Acest detaliu este important, deoarece comparatorul nostru are nevoie de trei permutări ca să funcționeze. De exemplu, pentru grupa stîngă alegem un pivot T și distribuim restul grupei la stînga sau la dreapta lui T în funcție de comparația cu T și cu X.

Complexitatea sortării este $O(NM \log N)$, pentru că sortăm N elemente, iar costul unei comparații este $O(M)$. La finalul sortării obținem un candidat. Prin logica partiționării acest candidat este singurul care respectă toate tripletele comparate. Dar este nevoie să ne asigurăm că le respectă pe toate, deci că este într-adevăr *single-crossing*.

Să presupunem că prima permutare este $(1, 2, \ldots, M)$ (în practică, iterăm prin valorile care ne interesează în ordinea în care apar ele în prima permutare). Procedăm inductiv. Dacă primele k permutări sînt *single-crossing*, dorim să verificăm dacă cea de-a $k+1$-a cauzează vreo dublă trecere. Cu alte cuvinte, vrem să vedem dacă vreo pereche de valori $x < y$, care erau inversate (y apărea înaintea lui x în P_k), devin acum ordonate (x apare înaintea lui y în P_{k+1}). Procesăm valorile de la 1 la M în ordine crescătoare. Dacă valoarea x apare pe poziția i în P_k și pe poziția j în P_{k+1}, dorim să știm: am văzut anterior vreo valoare pe o poziție $i' > i$ în P_k și $j' < j$ în P_{k+1}? Putem afla asta cu un singur AIB, care notează valoarea i la poziția j. Atunci interogarea devine: dă-mi maximul pe pozițiile $[1 \ldots j-1]$. Așadar, folosim un AIB cu modificare pe poziție și maxim pe prefix.