Olimpiada Națională de Informatică, Etapa județeană, Clasa a x-a Descrierea soluțiilor

Comisia științifică

March 2, 2026

Problema 1. UNUZERO

Propusă de: Prof. Mihai Bunget

Subtask 1.(65 puncte)

La primul subtask se dă o matrice cu o linie (şir) în care două elemente au valoarea egală cu 1. Dacă cele două elemente sunt pe poziţii consecutive, atunci avem un grup-1, în caz contrar avem două grupuri-1.

Complexitatea soluţiei este $O(Q)$.

Subtask 2.(3 puncte)

Într-un şir de lungime N se atribuie valoarea 1 elementelor de pe poziţiile $P_1, P_2, \ldots, P_Q$, apoi se numără secvenţele de elemente egale cu 1, acesta fiind numărul de grupuri-1.

Complexitatea soluţiei este $O(N + Q)$.

Subtask 3.(2 puncte)

Se procedează la fel ca la subtaskul 2 cu menţiunea că, dacă matricea linie se scrie de K ori, atunci numărul de grupuri-1 rămâne acelaşi.

Complexitatea soluţiei este $O(N + Q)$.

Subtask 4.(7 puncte)

Se parcurge şirul P şi pentru fiecare termen P_i se determină linia şi coloana din matrice pentru care elementul va lua valoarea 1, respectiv $l = (P_i - 1)/N + 1, c = (P_i - 1)\%N + 1$. Se aplică apoi algoritmul Fill pentru a determina numărul de grupuri-1.

Complexitatea soluţiei este $O(M \cdot N + Q)$.

Subtask 5.(2 puncte)

Se parcurge şirul P şi se numără secvenţele de numere consecucutive, acesta fiind numărul de grupuri-1.

Complexitatea soluţiei este $O(Q)$.

Subtask 6.(4 puncte)

Se parcurge şirul P şi pentru fiecare termen P_i se determină linia şi coloana din matrice pentru care elementul va lua valoarea 1, respectiv $l = (P_i - 1)/N + 1, c = (P_i - 1)\%N + 1$. Se adaugă valoarea 1 elementelor matricei ce au indicii $(l + M \cdot N \cdot (j-1), c + M \cdot N \cdot (j-1))$, pentru fiecare j de la 1 la K. Se aplică apoi algoritmul Fill in matricea de dimensiuni $M \cdot K$ si N, pentru a determina numărul de grupuri-1.

Complexitatea soluţiei este $O(K \cdot Q + M \cdot K \cdot N)$.

Subtask 7.(5 puncte)

Matricea rezultată fiind de dimensiuni mari se aplică algoritmul Fill folosind şirul P. Pentru fiecare element nevizitat din şirul P vom determina linia şi coloana din matrice, respectiv $l = (P_i - 1)/N + 1, c = (P_i - 1)\%N + 1$, unde elementul are valoarea 1. Vecinii acestui element sunt de coordonate $(l - N, c)$, $(l + N, c)$, $(l, c - 1)$, respectiv $(l, c + 1)$, cu condiţia ca indicele de linie să fie între 1 şi M, iar cel de coloană între 1 şi N. Verificarea dacă un vecin a fost sau nu vizitat se poate face folosind o tabelă hash, structura map din STL, sau o căutare binară în şirul P.

Complexitatea soluţiei este $O(Q)$, respectiv $O(QlogQ)$.

Subtask 8.(6 puncte)

Se parcurge şirul P şi pentru fiecare termen P_i se determină linia şi coloana din matrice pentru care elementul va lua valoarea 1, respectiv $l = (P_i - 1)/N + 1, c = (P_i - 1)\%N + 1$. Se aplică apoi algoritmul Fill pentru a determina numărul A de grupuri-1 din matrice. Se determină numarul B de grupuri-1 corespunzător cazului $K = 2$, iar numărul de grupuri-1 care se pierd la fiecare îmbinare de două matrice este $2 \cdot A - B$. Numărul cerut de grupuri-1 este $A \cdot K - (2 \cdot A - B) \cdot (K - 1)$.

Complexitatea soluţiei este $O(M \cdot N + Q)$.

Subtask 9.(6 puncte)

Se determină numerele A şi B folosind algoritmul de la subtaskul 7. Numărul cerut de grupuri-1 este $A \cdot K - (2 \cdot A - B) \cdot (K - 1)$.

Complexitatea soluţiei este $O(Q)$, respectiv $O(QlogQ)$.

Problema 2. INGHETATA

Propusă de: Stud. Matei Benchea

Subtask 1 (60 puncte)

Pentru rezolvarea primului subtask este suficientă o abordare de tip brute-force. Parcurgem vectorul A, iar pentru fiecare i căutăm capătul stânga și dreapta al grupului său de simpatizanți, oprindu-ne în ambele cazuri la primul element care nu este divizibil cu A_i. Pe măsură ce parcurgem aflăm și lungimea maximă a unei secvențe și afișăm la final poziția la care am găsit maximul.

Complexitate: $O(N^2)$

Subtask 2 (4 puncte)

Pentru rezolvarea subtaskului 2 vom afla pentru fiecare i lungimea grupului de simpatizanți așa cum am făcut la primul subtask. Acum ne vom concentra pe cum putem calcula o singură valoare $v_i(k)$. Numerotăm cele k arome cu $1, 2, \ldots, k$, iar cu x_k notăm de câte ori am achiziționat aroma k. Conform cerințelor din enunț o să avem: $x_1 + x_2 + \cdots + x_k = |G_i|$ și $x_1, x_2, \ldots, x_k > 0$ și trebuie să numărăm câte soluții are această ecuație. Acest număr este dat de tehnica Stars and Bars, care ne spune că avem $C^{k-1}_{|G_i|-1}$ soluții. Iterăm apoi cu i de la st la dr și adunăm la răspuns $C^{i-1}_{|G_i|-1}$. Pentru calculul combinărilor se poate utiliza Triunghiul lui Pascal.

Complexitate: $O(N^2 + 60 \cdot Q)$

Subtask 4 (3 puncte)

Cum toate valorile din vectorul A sunt egale, atunci fiecare grup de simpatizanți are lungime maximă, adică N. Astfel, pentru fiecare interogare, suma pe care va trebui să o calculăm va fi de forma: $C^{st-1}_{N-1} + C^{st}_{N-1} + \cdots + C^{dr-1}_{N-1}$. Pentru a putea răspunde rapid la întrebări, este necesar să precalculăm factorialele până la $N - 1$. Pentru a putea calcula combinările mod $10^9 + 7$ putem să folosim Algoritmul lui Euclid extins, exponențierea rapidă sau chiar putem să precalculăm direct și inversele modulare ale factorialelor la preprocesare. Dacă vrem optimizare în plus, putem să menținem un vector de sume parțiale pentru combinările cu coeficientul inferior $N - 1$ ca să răspundem la întrebări în $O(1)$, deși această optimizare nu este necesară pentru a obține punctajul maxim pentru acest subtask.

Complexitate: $O(N + Q)$

Subtask 5 (9 puncte)

Fiecare element din vectorul A este putere de 2 ($A_i = 2^{p_i}$). Să spunem că A_j este divizor al lui A_i este echivalent cu a spune că $p_j \leq p_i$. Pentru a găsi cel mai îndepărtat element care este divizor al lui A_i putem să aplicăm un algoritm de tip **next greater element** cu ajutorul unei stive. Putem astfel să aflăm pentru fiecare i lungimea grupului său de simpatizanți în complexitate liniară. Pentru a răspunde la eficient la întrebări precalculăm factorialele.

Complexitate: $O(N + 60 \cdot Q)$

Subtask 6 (3 puncte)

Pentru următoarele subtaskuri ne trebuie o metodă mai eficientă de a determina lungimea grupului de simpatizanți pentru fiecare i.

Vom calcula capătul stânga pentru fiecare grup de simpatizanți (pentru capătul dreapta se procedează analog). Parcurgem vectorul A cu i de la stânga la dreapta și pe parcurs ținem minte o stivă în care memorăm lcm-urile distincte pe sufixul lui $[1, 2, \ldots, i]$, precum și cele mai din stânga poziții unde se găsesc aceste valori (avem că $lcm(A_i) \leq lcm(A_i, A_{i-1}) \leq \cdots \leq lcm(A_i, A_{i-1}, \ldots, A_1)$). Cum facem acest lucru? De la stiva procesată la pasul $i-1$ punem în vârful stivei elementul A_i (și menținem și poziția i), iar apoi pentru fiecare element S_j al stivei facem $S_j = lcm(S_j, A_i)$ (în cazul în care valoarea lcm-ului ar depăși $VMAX = 10^9$ atunci lcm devine pur și simplu $VMAX + 1$ căci nu ne interesează valori mai mari de $VMAX$). Acum dacă în stivă există elemente consecutive cu valori egale le eliminăm și păstrăm doar elementul care indică spre cel mai din stânga indice. Acum parcurgem stiva de la vârf în jos și desemnăm drept capătul stânga al secvenței lui i indicele spre care arată ultimul element din stivă S_j care este divizor al lui A_i (acest lucru înseamnă că dacă capătul stânga al secvenței lui i este j este echivalent cu a spune că $lcm(A_i, \ldots, A_j)$ este divizor al lui A_i). În plus, facem observația că la un moment dat, în stiva noastră se vor afla cel mult $O(log(VMAX))$ elemente (cum în stivă ținem mereu valorile distincte pe care le avem pe sufixul curent, atunci în cel mai bun caz două valori de pe stivă vor diferi cu un factor de 2). Complexitatea acestei abordări este $O(N \cdot log^2(VMAX))$, iar în practică această metodă se comportă foarte bine.

Menționăm aici că există și abordări care găsesc capetele stânga și dreapta în complexitate de $O(N)$, însă acestea nu sunt necesare pentru obținerea punctajului maxim.

La întrebări avem restricția că $1 \leq st \leq dr \leq max(2, |G_i|)$, deci în cel mai rău caz trebuie să calculăm $v_i(1) + v_i(2)$. Dacă se vinde o singură aromă de înghețată, atunci există un singur mod de a face cumpărături și anume aceeași aromă pentru fiecare înghețată achiziționată. Deci $v_i(1) = 1$ pentru orice i. Dacă se vând două arome de înghețată, să zicem ciocolată și vanilie, atunci fixăm numărul de înghețate cu ciocolată (să zicem x) și observăm că numărul de înghețate cu vanilie o să fie determinat drept $|G_i| - x$. Cum x poate lua valori de la 1 la $|G_i| - 1$, avem $|G_i| - 1$ moduri de a cumpăra înghețată, așadar $v_i(2) = |G_i| - 1$.

Complexitate: $O(N \cdot log^2(VMAX) + Q)$

Subtask 7 (4 puncte)

Calculăm lungimile grupurilor de simpatizanți pentru fiecare i idem ca la subtaskul 6.

La întrebări avem restricția că $max(1, |G_i| - 1) \leq st \leq dr \leq |G_i|$, deci în cel mai rău caz trebuie să calculăm $v_i(|G_i| - 1) + v_i(|G_i|)$. Dacă se vând $|G_i|$ arome de înghețată, atunci singurul mod de a cumpăra este să luăm una din fiecare (dacă am cumpăra mai mult de una atunci ar fi o aromă pe care nu am cumpăra-o deloc), deci $v_i(|G_i|) = 1$. Dacă se vând $|G_i| - 1$ arome, atunci trebuie să cumpărăm cel puțin una din fiecare și apoi ne mai rămâne o înghețată de cumpărat pe care o putem alege oricum din cele $|G_i| - 1$, așadar $v_i(|G_i| - 1) = |G_i| - 1$.

Complexitate: $O(N \cdot log^2(VMAX) + Q)$

Subtask 8 (13 puncte)

Pentru acest subtask combinăm ideile de la subtaskurile 6 pentru calcularea lungimilor grupurilor de simpatizanți și 2 pentru aflarea răspunsurilor la întrebări.

Complexitate: $O(N \cdot log^2(VMAX) + 60 \cdot Q)$

Problema 3. MISTER

Functia mentine un **deque descrescător după valori**. Pentru fiecare poziție i:

- se elimină din spate toate pozițiile j pentru care $A_i > A_j$;
- se elimină din față poziția $i - K$ dacă iese din fereastra de lungime K;
- se adaugă poziția i în deque;
- se setează B_i = dimensiunea deque-ului.

Subtask 1 (15 puncte)

Restricții: $N = 4$

Pentru acest subtask se pot genera toate șirurile posibile A cu valori între 1 și N. Pentru fiecare șir generat se rulează algoritmul misterios și se verifică dacă șirul obținut coincide cu șirul B dat.

Complexitate: $O(N^N)$, care este acceptabilă pentru $N = 4$.

Subtask 2 (50 puncte)

Restricții: $B_i = 1$ pentru orice i

Observăm că, pentru ca algoritmul să producă $B_i = 1$, condiția din linia 7 nu trebuie să fie niciodată îndeplinită. Un șir simplu care satisface această cerință este:

$$A = (1, 2, 3, \ldots, N)$$

pentru care algoritmul va produce $B_i = 1$ pentru toate pozițiile.

Complexitate: $O(1)$

Solutie oficiala

Complexitate: $O(N)$

Vom simula aceeasi structura de date, insa vom construi sirul A, astfel incat dimensiunea deque-ului este egala cu B_i

Deci, la pasul i:

1. Eliminam elementul din fata daca iese din fereastra de lungime K;
2. Reducem deque-ul din spate pana cand dimensiunea devine $B_i - 1$;

3. Adaugam pozitia i.

Mentinem doua valori:

$$low = 1, \qquad high = N.$$

- Cand eliminam un element din spatele deque-ului, înseamna ca in simulare ar trebui sa existe un A_i mai mare decât valoarea lui. Pentru a face acest lucru posibil, îi atribuim o valoare mica, $A_j = low++$. Astfel, orice valoare viitoare mai mare il va putea elimina.
- Cand un element iese din deque prin fata (deoarece părăsește fereastra de lungime K), Ii atribuim o valoare mare, $A_j = high--$. Astfel ne asiguram ca nu ar fi fost eliminat mai devreme din cauza comparatiilor de valori, ci doar pentru ca a iesit din fereastra.

La final, elementele ramase în deque primesc valori mari, în ordine descrescatoare.

Echipa

Problemele pentru această etapă au fost pregătite de:

- Cercetător ştiinţific Tamio-Vesa Nakajima, Universitatea Marburg, Hesse
- Prof. Mihai Bunget, Colegiul Naţional "Tudor Vladimirescu", Târgu Jiu
- Prof. Ioan-Cristian Pop, Education Beyond Boarders & Nerdvana, Bucureşti
- Stud. Alexandru Lorintz, Universitatea "Babeş-Bolyai", Cluj-Napoca
- Stud. Gheorghe Liviu Armand, Universitatea Bucureşti
- Programator Radu Muntean, TDB, Bucureşti
- Stud. Alexandru Peticaru, Universitatea Politehnică Bucureşti
- Stud. Tiberiu Cozma, Universitatea Alexandru Ioan Cuza, Iaşi
- Programator Răzvan Viorel Uzum, Google, Bucureşti
- Stud. Matei Benchea, Universitatea Bucureşti
- Stud. Cristian-Adrian Blaga, Universitatea "Babeş-Bolyai", Cluj-Napoca
- Stud. Laurian Iacob, Universitatea Bucureşti
- Stud. Raul Mihai Feier, Universitatea Politehnică Bucureşti