OLIMPIADA NAȚIONALĂ DE INFORMATICĂ, ETAPA JUDEȚEANĂ
CLASA A IX-A
DESCRIEREA SOLUȚIILOR

COMISIA ȘTIINȚIFICĂ

Problema 1: MACARIE

Propusă de: prof. Daniela Lica, Centrul Județean de Excelență Prahova

Subtask 1: Complexitatea $O(Valmax \times N)$, unde $Valmax$ este valoarea maximă din șirul A. Pentru fiecare valoare x, în ordine de la 1 la $Valmax$, se parcurge șirul A și se crează lista D, introducând x de fiecare dată când acesta divide un element al șirului. Răspunsul la fiecare din cele Q întrebări este realizat în $O(1)$.

Subtask 2: Complexitatea $O(Valmax + N + Q)$. Întrucât valorile din fișier sunt numere prime, lista D conține N valori egale cu 1, și valorile șirului A, în ordine crescătoare. Cum acestea sunt mai mici decât 1000000 se pot reține într-un vector de frecvență. Pe baza acestuia se poate construi șirul D și răspunde în $O(1)$ la cele Q întrebări.

Subtask 3: Complexitatea $O(N \times \sqrt{Valmax} + PozMax)$. Pentru fiecare număr din șirul A, vom determina în $O(\sqrt{Valmax})$ divizorii acestuia, marcându-i întru-un vector de frecvență. Pe baza acestuia se va construi șirul D și răspunde în $O(1)$ la cele Q întrebări.

Subtask 4: Complexitatea $O(Valmax \times log(Valmax) + Q)$. Se garantează că pozițiile din șirul D pentru care trebuie să identificăm divizorii sunt mai mici decât 2000000. Vom construi un vector $NrMultipli$ în care se va reține, pentru fiecare valoare x de la 1 la $ValMax$, câte numere din șirul A se divid cu x. Construcția se face în complexitate $Valmax \times log(Valmax)$, similar cu Ciurul lui Eratostene. Pe baza acestui vector se va construi șirul D și răspunde în $O(1)$ la cele Q întrebări.

Subtask 5: Complexitatea $O(ValMax \times log(ValMax) + Q \times log(ValMax))$ Neavând restricții suplimentare, vom contrui vectorul $NrMultipli$ la fel ca la subtaskul 4. Șirul D îl vom construi memorând $(divizor, frecventa)$. Folosind sumele parțiale, putem reține în $D[x].frecventa$ ultima poziție pe care apare x în lista divizorilor. Cum șirul acestor poziții este crescător, pentru a răspunde la cele Q întrebări, vom folosi căutarea binară.

Problema 2: SANTINELE

Propusă de: Cristian Frâncu, clubul Nerdvana România

Subtask 1: Santinela trebuie amplasată în primele $K+1$ înălțimi. Fie h_i cel mai din dreapta maxim din primele $K+1$ înălțimi. Atunci, i este cea mai din dreapta poziție în care putem plasa santinela. Dacă am plasa-o într-o poziție j mai la dreapta, atunci $h_j < h_i$, deci santinela nu va acoperi h_i. Vom calcula numărul maxim de înălțimi ce pot fi vegheate pornind spre dreapta de la poziția i până ce dăm de o înălțime strict mai mare, fie depășim poziția $2K+1$.

Complexitate: $O(K)$ timp, $O(1)$ memorie.

Subtask 2: Fie H maximul dintre înălțimi. Avem două situații:

(1) $h_{N/2} = H$ sau $h_{N/2+1} = H$. Avem nevoie de o singură santinelă ce poate fi amplasată la una din acele poziții și va veghea tot muntele.
(2) $h_{N/2} \neq H$ și $h_{N/2+1} \neq H$. Avem nevoie de două santinele, una pentru primele $K+1$ înălțimi și una pentru ultimele K. Cele două santinele vor fi amplasate în maximele zonelor vegheate de ele.

Complexitate: $O(N)$ timp, $O(1)$ memorie.

Subtask 3: Înălțimile fiind strict crescătoare putem plasa santinele pe pozițiile $K+1$, $2K+2$, $3K+3$, etc. Numărul minim de santinele este $N/(K+1)$ dacă N se divide cu $K+1$, sau $N/(K+1)+1$ în caz contrar. Expresia C/C++ ce calculează acest număr este $(N+K)/(K+1)$.

Complexitate: $O(1)$ timp și memorie.

Subtask 4: Acest subtask este similar cu cel anterior. Vom calcula maximul. Să presupunem că se află pe poziția i. Împărțim logic șirul în două subșiruri: elementele de la 1 la i (inclusiv) și cele de la i la N (inclusiv). Pentru fiecare din subșiruri este necesar să amplasăm un număr de santinele calculabil cu formula anterioară. Pentru primul subșir numărul de santinele este $(i+K)/(K+1)$. Pentru al doilea subșir numărul de santinele este $(N-i+1+K)/(K+1)$ care este totuna cu $(N-i)/(K+1)$. Suma acestor expresii este numărul total de santinele necesar, cu o ajustare: santinela plasată în poziția i este comună ambelor subșiruri, deci vom scădea 1 din expresia finală.

Numărul minim de santinele este $(i+K)/(K+1)+(N-i)/(K+1)-1$.

Complexitate: $O(N)$ timp, $O(1)$ memorie.

Subtask 5: Vom construi un vector $v[]$ încercând să folosim cât mai puține santinele pentru a veghea muntele până la h_i. Pentru aceasta vom încerca să plasăm o santinelă pe fiecare h_i, variind i de la 1 la N. Pentru o poziție i vom parcurge înălțimile la stânga și la dreapta pentru a determina limitele st și dr până unde veghează santinela plasată pe înălțimea h_i. Apoi vom testa ce se întâmplă dacă plasăm santinela la această poziție: știm că pozițiile până la $st-1$ sunt deja vegheate și știm că avem nevoie de $v[st-1]$ santinele. Vom parcurge pozițiile de la st la dr și ne vom întreba dacă nu cumva numărul de santinele necesare la acele poziții j se poate îmbunătăți cu santinela curentă. Cu alte cuvinte dacă $v[j] > v[st-1]+1$, actualizăm $v[j]$ la $v[st-1]+1$. La final vom afișa $v[N]$.

Complexitate: $O(N \times K)$ timp și $O(N)$ memorie.

Subtask 6: Soluția 1

La subtaskul 1 am arătat că cea mai la dreapta poziție pe care putem plasa prima santinelă este poziția celui mai din dreapta maxim din primele $K+1$ înălțimi. Aceasta ne duce la un algoritm tip Greedy: găsim h_i, ultimul maxim din primele $K+1$ și plasăm prima santinela pe poziția i. Apoi verificăm spre dreapta santinelei până unde veghează ea. Fie h_j prima înălțime care nu este vegheată de prima santinelă. Rezultă că avem, acum, un nou subșir, format din înălțimile h_j, h_{j+1}, h_{j+2},..., h_N. Acest subșir trebuie vegheat în întregime, folosind cât mai puține santinele. Avem, deci, o problema identică cu cea originală, dar cu un șir mai mic. Vom relua, deci, calculul, până ce întregul munte este vegheat.

Complexitate: $O(N)$ ca timp și memorie.

Soluția 2

Să presupunem că amplasăm câte o santinelă pe fiecare înălțime. Fiecare din santinele va avea câte un interval stânga-dreapta pe care îl va veghea. Problema ne cere să selectăm un număr minim de intervale care să acopere indicii de la 1 până la N. Aceasta este o problemă cunoscută, care se rezolvă prin parcurgerea intervalelor în ordinea crescătoare a deschiderii intervalului. La fiecare pas vom selecta intervalul care începe în zonă acoperită și se termină cât mai departe spre dreapta. Capetele intervalelor sunt indici între 1 și N. Un interval $[i,j]$ poate fi stocat într-un vector în care setăm $sfarsit[i] = j$. Dacă avem mai multe intervale ce încep în același punct i îl păstrăm pe cel care are j maxim. În acest fel nu avem nevoie să sortăm intervalele.

Rămâne să stabilim cum determinăm acele intervale. Pentru aceasta vom folosi altă problemă cunoscută: pentru fiecare indice i din vectorul de înălțimi dorim să calculăm primul element la stânga strict mai mare decât h_i. Acesta va fi capătul stânga al intervalului, dacă el se află la distanță mai mică sau egală cu K. În caz contrar vom seta $i - K$ drept capătul stânga al intervalului. Pentru a rezolva găsirea maximelor vom ne vom folosi de faptul că avem deja calculat răspunsul pentru indicii anteriori. Pentru i ne vom întreba dacă $h_{i-1} > h_i$. Dacă da, am găsit raspunsul, dacă nu, vom avansa pe indicele primului maxim la stânga al lui h_{i-1}. Vom continua așa până găsim maximul dorit.

Vom proceda similar și pentru primul maxim la dreapta.

Complexitate: $O(N)$ atât ca timp cât și ca memorie.

Problema 3: TRAFALET

Propusă de: stud. Toma Ariciu, Universitatea Politehnica București

Subtask 1: Considerăm fiecare submatrice a matricei inițiale. Punctajul acesteia este reprezentat de diferența în modul dintre suma celulelor albe și suma celulelor negre. Aceste două sume le putem calcula prin parcurgerea submatricei. Rezultatul final este maximul dintre toate punctajele considerate.

Complexitate: $O(N^2 \times M^2)$ pentru fixarea submatricei, $O(N \times M)$ pentru calcularea sumelor. În total, $O(N^3 \times M^3)$.

Subtask 2: Soluția este aceeași ca la subtaskul precedent, doar că vom optimiza aflarea celor două sume. Astfel, vom precalcula sume parțiale 2D și vom obține punctajul fiecărei submatrice cu ajutorul acestora.

Complexitate: $O(N^2 \times M^2)$ pentru fixarea submatricei, $O(1)$ pentru calcularea sumelor. În total, $O(N^2 \times M^2)$.

Subtask 3: Pentru început, vom înmulți toate celulele uneia dintre culori cu -1. Acum, în loc să calculăm diferența între cele două sume, avem doar de calculat suma în modul. Pentru a rezolva asta, vom fixa două linii, reprezentând latura superioară, respectiv inferioară a submatricei. Am rămas cu un vector, unde elementul de pe poziția i reprezintă suma elementelor de pe coloana i, cuprinse între cele două linii fixate anterior.

Problema s-a redus la a calcula secvența de sumă maximă în modul. Aceasta este data de diferența dintre suma parțială maximă și cea minimă. Fie pozițiile acestora $Pmax$ și $Pmin$. Dacă $Pmax > Pmin$, atunci răspunsul îl reprezintă suma intervalului dat de acestea. Dacă $Pmin > Pmax$, atunci intervalul dintre cele două este cel de sumă minimă, care, după aplicarea modulului, se transformă în suma maximă.

Complexitate: $O(N^2 \times M)$.

Echipa

Problemele pentru această etapă au fost pregătite de:

- Prof. Anton Cristina, Colegiul Național "Gheorghe Munteanu Murgoci", Brăila
- Stud. Ariciu Toma, Universitatea Politehnica București
- Stud. Ciuleanu Vlad, ETH Zurich
- Inst. Dăscălescu Ștefan-Cosmin, RoAlgo
- Lect. dr. Diac Paul, Facultatea de Informatică Iași, Universitatea "A.I. Cuza"
- Inst. Frâncu Cristian, Clubul Nerdvana, România
- Stud. Ion Andrei-Robert, TU Delft
- prof. Lica Daniela, Centrul Județean de Excelență Prahova, Ploiești
- Stud. Onuț Andrei, Yale University
- Prof. Pascu Olivia-Cătălina, Colegiul Național "Nichita Stănescu", Ploiești
- Stud. Udrişţoiu Alexandra, Facultatea de Matematică-Informatică, Univ. Bucureşti
- Drd. Nakajima Tamio-Vesa, Department of Computer Science, University of Oxford