Olimpiada Națională de Informatică,
Etapa națională, Clasa a VII-a
Descrierea soluțiilor

Problema 1. Alvn

Propusă de: stud. Marcu Mihai, Delft University of Technology

Soluție de: prof. Panaete Adrian, Colegiul Național "August Treboniu Laurian", Botoșani

Cerința 1.

În rezolvare vom utiliza două matrice: SL, în care calculăm sumele parțiale pe linii, și SC, în care calculăm sumele parțiale pe coloane.

Astfel, $SL[i][j]$ va reține suma elementelor de pe linia i, elemente care se găsesc pe coloanele de la 1 la j, iar $SC[i][j]$ reține suma elementelor de pe coloana j, elemente care se găsesc de la linia 1 la linia i.

Putem observa că, pentru fiecare parcelă, se vor calcula, pentru fiecare poziție la distanța w de parcela curentă, $x[w] \cdot val$. Astfel, pentru toate valorile la distanța w de o anumită parcelă (de pe inelul w, pornind de la parcela noastră), acestea se vor înmulți cu $x[w]$. Vom încerca să calculăm mai rapid suma acestor parcele folosind matricele de sume parțiale definite anterior. Astfel:

- Suma de sus este $SL[i-w+1][j+w-1] - SL[i-w+1][j-w]$;
- Suma din dreapta este $SC[i+w-1][j+w-1] - SC[i-w+1][j+w-1]$;
- Suma de jos este $SL[i+w-1][j+w-2] - SL[i+w-1][j-w]$;
- Suma din stânga este $SC[i+w-2][j-w+1] - SC[i-w+1][j-w+1]$.

Observație: Aceste sume ar putea fi calculate cu modificarea indicilor, dacă aceștia nu se încadrează în intervalele de la 1 la N în cazul liniilor, respectiv de la 1 la M în cazul coloanelor. În particular, sumele vor fi 0 dacă linia sau coloana la care ne referim nu este validă. Sumele se vor calcula pentru $w = \overline{1, K}$, iar complexitatea timp este $O(N \cdot M \cdot K)$.

Cerința 2.

O observație este că doi copaci nu au nicio parcelă în comun dacă diferența dintre liniile lor sau coloanele lor este mai mare sau egală cu $2 \cdot K - 1$.

Pentru soluția eficientă, vom avea nevoie de patru vectori: $maxUD$, $maxLR$, $maxDU$, $maxRL$.

- $maxUD[i]$ reprezintă elementul maxim care are linia mai mică sau egală cu i;

- $maxLR[i]$ reprezintă elementul maxim care are coloana mai mică sau egală cu i;
- $maxDU[i]$ reprezintă elementul maxim care are linia mai mare sau egală cu i;
- $maxRL[i]$ reprezintă elementul maxim care are coloana mai mare sau egală cu i.

Astfel, pentru fiecare element (i, j), parcela maximă al cărei stejar nu are alte parcele în comun cu stejarul de pe parcela (i, j) va fi maximul dintre:

- $maxUD[i - 2 \cdot K + 1]$
- $maxLR[j - 2 \cdot K + 1]$
- $maxDU[i + 2 \cdot K - 1]$
- $maxRL[j + 2 \cdot K - 1]$

Complexitatea în timp este $O(N \cdot M \cdot K)$, deoarece trebuie să calculăm matricea de la cerința 1.

Problema 2. Conturi

Propusă de: prof. Costineanu Veronica-Raluca, Colegiul Național "Ștefan cel Mare", Suceava

Pentru a reține organizat informațiile despre conturile existente, putem folosi o structură `cont`, cu atributele:

- `id`: număr natural, unic pentru fiecare cont;
- `sold`: șir de cifre, reprezentând suma totală depusă în cont;
- `ultimaAdăugat`: număr natural, reprezentând ultima sumă depusă în contul respectiv;

Procesând tranzacțiile pe rând, în ordinea în care apar, avem două cazuri: dacă valoarea citită *val* este pozitivă, vom avea un caz de depunere, iar dacă este negativă, vom avea un caz de retragere.

Putem reține conturile ordonate descrescător după ultima sumă adăugată, iar în caz de egalitate, crescător după `id`. În acest fel, în cazul unei depuneri, vom căuta binar poziția corespunzătoare la care trebuie să facem depunerea (contul cu ultima valoare adăugată maximă, dar strict mai mică decât *val*), iar dacă poziția corespunzătoare este în afara intervalului de indici în care reținem conturile active, vom crea un cont nou.

Pentru operația de retragere, vom reordona conturile descrescător după sold, iar în caz de egalitate, după ultima valoarea adăugată și, în final, crescător după `id`. Putem astfel să realizăm retragerea începând de la contul cu cel mai mic sold (dreapta), spre contul cu cel mai mare sold (stânga). La fiecare pas, vom compara soldul contului curent $cont[i]$ cu val, iar dacă nu există sold suficient, $val = val - cont[i]$ și contul curent dispare. Ne deplasăm atât timp cât $|val| > 0$. După finalizarea extragerii, vom reordona conturile descrescător după ultima valoare adăugată.

În final, vom afișa răspunsul corespunzător cerinței p. Complexitatea timp a soluției este $O(n \cdot k \cdot \log n)$, unde k reprezintă numărul de ștergeri.

Problema 3. Succesori

Propusă de: prof. Panaete Adrian, Colegiul Național "August Treboniu Laurian", Botoșani

O posibilă soluție generează toți succesorii elementelor citite, ordonează crescător șirul format de aceștia și afișează valoarea aferentă. Această soluție este ineficientă, datorită restricțiilor de numere mari.

Având în vedere numărul mare de succesori care vor fi generați, o soluție eficientă nu va memora toți succesorii fiecărui număr. Astfel, vom determina, prima dată, numărul de cifre și cifra dominantă a rezultatului. În acest scop, putem face următoarele observații:

- Pentru fiecare număr de cifre și fiecare cifră dominantă, o valoare poate genera cel mult un succesor;
- Dacă vom genera toți succesorii unui număr, fără a-i memora, putem număra pentru fiecare număr de cifre și pentru fiecare cifră dominantă câți succesori generează toate valorile inițiale;
- Pentru un anumit număr de cifre C, cifra dominantă a primului succesor generat având C cifre nu depinde de valoarea întregului număr, ci doar de cifrele de la pozițiile C, respectiv $C+1$. Mai precis, dacă inițial, cele două cifre sunt x și y, atunci, când y ajunge la valoarea 0, x va ajunge la valoarea $t = (y+10-x)\%10$ și t va fi cifra dominantă a primului succesor generat cu C cifre. În consecință, vor fi generați $10-t$ succesori cu C cifre, iar aceste cifre vor fi $t, t+1, \ldots, 9$.

Notăm cu $sol[C][d]$ numărul total de succesori cu C cifre care au cifra dominantă d. Folosind ultima observație, putem contoriza pentru fiecare valoare și fiecare număr de cifre o unitate doar pentru $sol[C][t]$. Apoi, sumând parțial de la 1 la 9 pe cifre, vom obține pentru fiecare cifră numărul real de succesori care vor fi generați. Această abordare reduce foarte mult timpul de calculare al valorilor $sol[C][d]$.

Numărul de cifre și cifra dominantă a rezultatului pot fi acum determinate parcurgând matricea *sol* în ordinea crescătoare a numărului de cifre și, apoi, în ordinea crescătoare a cifrelor dominante. Sumând toate aceste valori, până în momentul în care suma devine mai mare sau egală decât poziția dorită, determinăm exact cifra dominantă și numărul de cifre al rezultatului. Deoarece matricea *sol* are cel mult 18 linii și 9 coloane, numărul de adunări va fi cel mult 162.

În continuare, bazându-ne pe prima observație și folosind aceeași tehnică, putem genera, pentru fiecare valoare inițială, cel mult un succesor cu numărul de cifre dorit și cifra dominantă dorită. Scăzând din poziția cerută suma numărului de succesori, având strict mai puține cifre sau același număr de cifre, dar cu cifra dominantă strict mai mică, putem determina pe ce poziție, printre succesorii generați, vom găsi rezultatul. Sortând acești succesori, determinăm și care este valoarea cerută. Complexitatea finală a algoritmului este $O(n \cdot maxC + n \log n)$, unde $maxC$ este numărul maxim de cifre pentru valorile din șirul inițial.

De menționat că, pentru a determina al k-lea cel mai mic termen dintr-un șir, se poate folosi algoritmul de partiționare folosit la `quick sort` sau funcția `std::nth_element`, însa nu este necesar pentru obținerea punctajului maxim.

Echipa

Problemele pentru această etapă au fost pregătite de:

- prof. Boca Alina Gabriela, Colegiul Național de Informatică "Tudor Vianu", București
- prof. Costineanu Veronica-Raluca, Colegiul Național "Ștefan cel Mare", Suceava
- stud. Cotoi Rareș-Andrei, Facultatea de Matematică și Informatică, Universitatea Babeș-Bolyai, Cluj-Napoca
- prof. Gorea-Zamfir Claudiu-Cristian, Inspectoratul Școlar Județean, Iași
- stud. Marcu Mihai, Delft University of Technology
- prof. Nicu Vlad-Laurențiu, Liceul Teoretic "Mihail Kogălniceanu", Vaslui
- prof. Panaete Adrian, Colegiul Național "August Treboniu Laurian", Botoșani
- prof. Popa Daniel, Colegiul Național "Aurel Vlaicu", Orăștie
- stud. Pop Ioan-Cristian, Facultatea de Automatică și Calculatoare, Universitatea Națională de Știință și Tehnologie POLITEHNICA București
- prog. Trișcă-Vicol Cezar, 2k Games, Dublin