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

COMISIA ȘTIINȚIFICĂ

Problema 1: Aprogressive

Propusă de: prof. Nodea Gheorghe-Eugen, Centrul Județean de Excelență Gorj

. Cerința 1

Pentru fiecare linie a matricei T se calculează suma pe linie prin adunarea elementelor aflate pe aceasta. Sumele obținute pentru fiecare dintre liniile acestei matrice formează termenii unui șir, numit șirul sumelor pe linii S_i cu $1 \leq i \leq n$.

Se determină pentru șirul S valoarea maximă.

La final, se afisează în ordine strict crescătoare indicii liniilor matricei pentru care suma S_i este maximă.

. Cerința 2

Vom privi linia unei matrici ca un șir (vector) cu m elemente.

Cum verificăm dacă elementele șirului pot fi reanjate astfel încât să formeze o progresie aritmetică?

Soluția 1, brute-force $O(m^2)$

Se determină cele mai mici două elemente din șir Min_1, respectiv Min_2, cu $Min_1 \leq Min_2$.

Dacă șirul poate fi rearanjat pentru a forma o progresie aritmetică, atunci rația progresiei este $r = Min_2 - Min_1$.

Dacă rația este nenulă, atunci se caută a treia cea ma mică valoare, a patra cea mai mică, și așa mai departe. Pentru cea mai mică a i-a valoare din șir se verifică dacă diferența între valoarea minimă curentă și valoarea minimă determinată anterior este egală cu rația.

În caz contrar, șirul nu poate fi reanjat astfel încât să formeze o progresie aritmetică.

Soluția 2 $O(mlogm)$

Se sortează elementele șirului crescător. Rația progresiei este dată de diferența dintre al doilea element și primul element al șirului sortat. Se verifică dacă diferențele dintre elementele consecutive din șir se păstrează și este egală cu rația.

În funcție de metoda de sortare folosită se pot obține punctaje diferite.

Se afisează în ordine strict crescătoare indicii liniilor matricei pentru care elementele pot fi rearanjate astfel încât să formeze pe linia respectivă o progresie aritmetică de rație nenulă.

Soluția 3 - $O(m)$ Se determină cele mai mici două elemente din șir Min_1, respectiv Min_2, cu $Min_1 \leq Min_2$. Calculăm rația progresiei $r = Min_2 - Min_1$. Se scade din toate elementele șirului valoarea Min_1. Acum ar trebui ca toate elementele obținute să se se dividă cu rația r, caz în care vom simplifica fiecare element cu r.

Se obține prin simplificare un șir cu valori distincte din mulțimea $\{0, .., m-1\}$ pentru a putea fi reanjat să formeze o progresie aritmetică. Putem verifica această condiție folosind un vector de apariții.

. Cerința 3

Pentru împărțirea în submatrici se folosește un algoritm recursiv (divide et impera).

Fie submatricea R cu colțul stânga-sus (x_1, y_1) și colțul dreapta-jos (x_2, y_2).

Submatricea A are colțul stânga-sus în (x_1, y_1), iar colțul dreapta-jos în $((x_1 + x_2)/2, (y_1 + y_2)/2)$.

Submatricea B are colțul stânga-sus în $(x_1, (y_1 + y_2)/2 + 1)$, iar colțul dreapta-jos în $((x_1 + x_2)/2, y_2)$.

Submatricea C are colțul stânga-sus în $((x_1 + x_2)/2 + 1, y_1)$, iar colțul dreapta-jos în $(x_2, (y_1 + y_2)/2)$.

Submatricea B are colțul stânga-sus în $((x_1 + x_2)/2 + 1, (y_1 + y_2)/2 + 1)$, iar colțul dreapta-jos în (x_2, y_2).

Pentru determinarea șirului S al unei submatrice R se precalculează o matrice a sumelor parțiale pe linie.

Se procedează similar ca la cerința 2 pentru a verifica dacă o submatrice R este aprogressive.

Problema 2: Opșir

Propusă de: stud. Popescu Ștefan-Alexandru, Facultatea de Matematică-Informatică, Universitatea București

Cerința 1. Subtask 1
Pentru fiecare șir se construiește un vector de frecvență ce memorează numărul de apariții al fiecărei litere. Pe baza acestora se determină apoi câte caractere distincte există în cele 2 șiruri. Pentru fiecare caracter distinct ce apare se determină în care dintre șiruri apare de mai multe ori folosindu-se vectorii de frecvență.

Cerința 2. Subtask 2 - S e sortat
În această situație putem aplica o operație de tip 2 pe tot șirul T, eliminând apoi literele care apar în plus. Abordarea e echivalentă cu a verifica dacă, pentru fiecare caracter distinct, frecvența sa în T e mai mare decât frecvența sa în S.

Subtask 3 - S poate fi obținut doar prin operații de tip 1
Problema e echivalentă cu a verifica daca S e subșir al lui T, ceea ce se poate face cu un algoritm Greedy.

Subtask 4 - fără restricții
Să presupunem că există o succesiune de lungime L de operații de tip 1 și 2 care pot transforma șirul T în șirul S (adica e validă ca soluție): op_1, op_2, op_3 ... op_L. Dacă încercăm să interschimbăm 2 operații consecutive din această succesiune (fie ele op_i și op_{i+1}, $1 \leq i < L$) pentru a obține tot o succesiune validă, putem observa următoarele cazuri:

(1) op_i și op_{i+1} sunt operații de tip 1: echivalent cu a șterge 2 poziții din șirul T. Acestea pot fi șterse în orice ordine (cu o eventuală recalculare de indecși)
(2) op_i și op_{i+1} sunt operații de tip 2: din criteriul colorării din cerință știm că intervalele sunt disjuncte, deci nu contează ordinea în care se realizează aceste operații
(3) op_i și op_{i+1} sunt operații de tipuri diferite: echivalent cu a șterge o poziție și a sorta un interval, operații care se pot realiza în orice ordine (cu eventuale recalculări de indecși)

Aceste observații ne arată faptul că putem interschimba 2 operații consecutive dintr-o succesiune validă (cu anumite modificări) pentru a obține tot o succesiune validă. Considerăm în continuare că "am adus", prin interschimbări, operațiile de tip 1 pe primele poziții din succesiune. De asemenea, considerăm că am efectuat deja aceste operații de tip 1. Astfel, obținem din șirul T un șir de lungime n și o mulțime de intervale disjuncte ce corespund sortărilor ce urmează a fi efectuate.

Să considerăm partiționarea șirului S în subsecvențe crescătoare maximale. Fie p numărul subsecvențelor din această partiționare. În acest caz, orice interval din mulțimea precizată anterior

va avea capetele în aceeași subsecvență din această partiționare. Observăm că, dacă înlocuim toate operațiile de tip 2 rămase cu câte o operație de tip 2 pentru fiecare subsecvență din partiționarea șirului S obținem același rezultat.
Astfel, folosindu-ne din nou de interschimbări, putem obține o succesiune de operații în care au loc mai întâi toate sortările, iar fiecare element din șirul T este inclus în cel puțin o sortare.
Problema se reduce, în această situație, la a verifica dacă există o partiționare a șirului T în p subsecvențe, astfel încât frecvențele literelor din fiecare subsecvență a lui T să fie mai mari sau egale decât cele din subsecvența corespondentă din S (din partiționarea de mai sus).
Verificarea se poate face prin următorul algoritm Greedy: pentru fiecare subsecvență din partiționarea lui S (subsecvențele fiind luate de la stânga la dreapta) se determina prefixul de lungime minimă a lui T pentru care frecvențele literelor sale sunt mai mare sau egale decât cele din subsecvența actuală a lui S. Se elimină acest prefix și se continuă cu următoarea subsecvență din S. Dacă pentru fiecare subsecvență a lui S se găsește un astfel de prefix, atunci răspunsul este DA, altfel răspunsul este NU.
Complexitate timp: O(n * dimensiune_alfabet).

Problema 3: Poseidon

Propusă de: stud. Gabor Ioana, Universitatea Babeș-Bolyai, Cluj-Napoca

Cerința 1. Subtask 1. Cazul unei singure insule.
Se va afișa numărul de valori diferite de 0 și -1 din matrice.
Subtask 2. Cazul $N = 1$.
Pentru acest caz, se poate parcurge șirul către stânga, începând cu celula de start, până la întâlnirea primei valori de -1, apoi către dreapta, într-un mod similar. Se vor număra valorile diferite de 0 și -1.
Subtask 3. Restul punctajului.
Pentru restul punctajului, se va aplica algoritmul de flood fill, pornind din celula de start.

Cerința 2. Se vor procesa individual toate insulele din matrice, folosind algoritmul de flood fill, și se vor număra comorile din fiecare. Notăm cu $D(n)$ numărul de permutări de n elemente, fără puncte fixe. Rezultatul va consta în produsul valorilor $D(x_i)$, unde x_i este numărul de comori de pe o insulă. Diferențele de punctaj sunt date de modul de calcul al valorilor $D(n)$. Notăm cu n_c numărul maxim de comori dintr-o insulă.
Subtask 4. Cazul $n_c \leq 4$.
Se pot calcula "pe hârtie" valorile $D(x_i)$, fiind numere mici.
Subtask 5. Cazul $n_c \leq 8$.
Se pot genera toate permutările de câte k elemente, $k \leq n_c$ care respectă condiția dată, cu ajutorul metodei backtracking.
Subtask 6. Restul punctajului.
Este necesară calcularea șirului D într-un mod mai eficient. Pentru ilustrarea metodei, ne imaginăm că vrem să "dezordonăm" șirul format din elementele $1, 2, 3, \ldots, n$, în această ordine. Pe fiecare poziție, se poate amplasa orice număr diferit de numărul de ordine al poziției respective. Notăm cu k numărul care se va amplasa pe poziția 1. Problema se împarte în două cazuri, în funcție de elementul de pe poziția k.

(1) Amplasăm numărul 1 pe poziția k. Problema se reduce la a calcula $D(n-2)$, pentru că pe poziția 1 avem valoarea k, iar pe poziția k avem valoare 1 și restul de $n-2$ valori se pot permuta independent de cele două cu respectarea cerinței inițiale.
(2) Amplasăm un număr diferit de 1 pe poziția k. Pentru fiecare poziție din cele cuprinse între 2 și n, există $n-2$ elemente care pot fi amplasate pe poziția respecitvă. Pentru pozițiile diferite de k, nu se pot amplasa k (pentru că apare deja pe prima poziție) sau elementul cu indicele poziției respective. Pentru poziția k, nu se pot amplasa k sau 1 (pentru că suntem în cazul 2). Problema se reduce la a calcula $D(n-1)$, fiindcă trebuie să amplasăm valorile pe pozițiile cuprinse între 2 și n (adică $n-1$ poziții în total) și

avem $n-2$ variante pentru fiecare. Datorită faptului că numărul k se poate alege în $n-1$ moduri, formula finală este:

$$D(n) = (n-1) \times (D(n-2) + D(n-1)).$$

Calculele se vor efectua modulo 10^9+7.

O altă variantă de abordare este să observăm, că $D(n)$ este egal cu numărul total de permutări de n elemente, minus numărul de permutări care conțin puncte fixe. Să notăm cu $F(n,k)$ numărul de permutări de n elemente, care conțin cel puțin k puncte fixe. Cele k poziții se pot alege în C_n^k feluri, iar restul elementelor se pot permuta oricum, așadar: $F(n,k) = C_n^k \cdot (n-k)!$

Aplicând principiul includerii și excluderii avem:

$$D(n) = n! - F(n,1) + F(n,2) - F(n,3) + \ldots F(n,n)$$

Precalculând valorile factorialelor și a inverșilor modulari a factorialelor modulo 10^9+7 până la 10^6 (valoarea maximă pentru n_c), putem obține fiecare termen al formulei de mai sus în $O(1)$.

Echipa

Problemele pentru această etapă au fost pregătite de:

- Prof. Nodea Gheorghe-Eugen, Centrul Județean de Excelență Gorj
- Lect. univ. Pătcaș Csaba, Facultatea de Matematică și Informatică, Universitatea Babeș-Bolyai, Cluj-Napoca
- Prof. Pițˇ-Rada Ionel-Vasile, Colegiul Național Traian, Drobeta-Turnu Severin
- Prof. Moț Nistor, Școala "Dr.Luca", Brăila
- Stud. Gavrilă-Ionescu Vlad-Alexandru, Zurich, Elveţia
- Stud. Apostol Ilie-Daniel, Facultatea de Matematică-Informatică, Universitatea București
- Stud. Gabor Ioana, Universitatea Babeș-Bolyai, Cluj-Napoca
- Stud. Oprea Mihai-Adrian, Facultatea de Matematică-Informatică, Universitatea București
- Stud. Pagu Tudor Ștefan, Delft University of Technology Delft, Olanda
- Stud. Popa Sebastian, Facultatea de Matematică-Informatică, Universitatea București
- Stud. Popescu Ioan, Facultatea de Automatică și Calculatoare, Universitatea Politehnică București
- Stud. Popescu Ștefan-Alexandru, Facultatea de Matematică-Informatică, Universitatea București