

**OLIMPIADA NAȚIONALĂ DE INFORMATICĂ, ETAPA NAȚIONALĂ
CLASA A VIII-A
DESCRIEREA SOLUȚIILOR**

COMISIA ȘTIINȚIFICĂ

PROBLEMA 1: CASTEL

Propusă de: stud. Sebastian Popa, Universitatea din București

Problema se rezumă la a afla dacă, din $N + 1$ dreptunghiuri date, putem alege N care să aibă intersecția nevidă.

Subtask 1: Se folosește o matrice pentru a simula intersecțiile dreptunghiurilor în complexitate $\mathcal{O}(N * VAL_MAX^2)$, unde VAL_MAX este maximul coordonatelor dreptunghiurilor.

Subtask 2: Se elimină, pe rând, fiecare dreptunghi, și se face intersecția celor rămase. Complexitatea obținută este $\mathcal{O}(N^2)$.

Soluția oficială: Se observă că intersecția a două dreptunghiuri ce au laturile paralele cu axele de coordonate este fie un dreptunghi, fie mulțimea vidă. Astfel, intersecția a două dreptunghiuri (x_1, y_1, x_2, y_2) și (x_3, y_3, x_4, y_4) este un dreptunghi (x_5, y_5, x_6, y_6) cu $x_5 = \max(x_1, x_3)$, $y_5 = \min(y_1, y_3)$, $x_6 = \min(x_2, x_4)$, $y_6 = \max(y_2, y_4)$. Dacă $x_5 > x_6$ sau $y_6 > y_5$, intersecția celor două dreptunghiuri este, de fapt, mulțimea vidă.

Observăm că intersecția dreptunghiurilor este comutativă și asociativă. Construim doi vectori auxiliari pre și suf astfel: $pre[i]$ este egal cu intersecția dreptunghiurilor $1, 2, \dots, i$, iar $suf[i]$ este egal cu intersecția dreptunghiurilor $i, i + 1, \dots, N + 1$. Dacă v_i este al i -lea dreptunghi, $pre[1] = v_1$ și $pre[i]$ este egal cu intersecția dintre $pre[i - 1]$ și v_i pentru $i > 1$. Analog se calculează suf .

Așadar, pentru a găsi punctele ce aparțin intersecției a cel puțin N dreptunghiuri excludem pe rând câte un dreptunghi și calculăm, în timp constant, intersecția celor rămase. Pentru $1 < i < N + 1$, intersecția obținută prin eliminarea dreptunghiului v_i este egală cu intersecția dintre $pre[i - 1]$ și $suf[i + 1]$. Se procedează asemănător și pentru $i = 1$ și $i = N + 1$. Pentru fiecare intersecție nevidă găsită alegem punctul din stânga-jos al dreptunghiului obținut (deoarece toate coordonatele sunt pozitive, deci acela este cel mai apropiat de origine) și comparăm cu un minim global.

Complexitatea obținută este $\mathcal{O}(N)$.

Soluție alternativă: O altă soluție, tot în $\mathcal{O}(N)$, pornește de la următorul raționament. Intersecția tuturor dreptunghiurilor, dacă există, este un dreptunghi de coordonate $(a, b) - (c, d)$. În particular,

$$a = \max\{a_k | 1 \leq k \leq N + 1\}$$

Cum se modifică această coordonată dacă eliminăm al i -lea dreptunghi? Dacă $a_i \neq a$, atunci a nu se modifică. În schimb, dacă $a_i = a$, atunci a capătă valoarea celui de-al doilea maxim al mulțimii $\{a_k\}$. Similar raționăm și pentru ceilalți trei parametri ai intersecției, b, c și d .

Algoritmul calculează primele două maxime sau minime pentru fiecare dintre cei patru parametri: $\max\{a_k\}$, $\min\{b_k\}$, $\min\{c_k\}$, $\max\{d_k\}$. Apoi elimină fiecare dreptunghi pe rând și determină în $\mathcal{O}(1)$ intersecția (vidă sau nu).

PROBLEMA 2: KTH

Propusă de: stud. Andrei Onuț, Universitatea Yale, S.U.A.

Subtask 3: Fie: Max = valoarea cea mai mare a unui element din șirul V dat = $\max(V_i)$, unde: $1 \leq i \leq N$. Din restricția $1 \leq V_i \leq 2000$, pentru fiecare i : $1 \leq i \leq N$, deducem că $1 \leq Max \leq 2000$.

Fiecare întrebare dintre cele Q va fi rezolvată imediat după citirea valorii poz corespunzătoare, astfel:

- (1) Se inițializează un tablou unidimensional de frecvență/numărare $cnt[1, \dots, Max]$. La început, $cnt[x] = 0$, pentru fiecare x : $1 \leq x \leq Max$.
- (2) Se iterează cu ajutorul unui indice j (unde: $poz \leq j \leq poz + L - 1$) prin secvența de L elemente pentru care trebuie să dăm răspunsul la întrebare. Când se ajunge în dreptul valorii V_j , vom marca corespunzător o nouă apariție în tabloul cnt , astfel: $cnt[V_j] = cnt[V_j] + 1$.
- (3) După această traversare (prin exact L elemente): $cnt[x] = [\text{de câte ori se găsește valoarea } x \text{ în secvența } V_{poz}, \dots, V_{poz+L-1}]$. Acest număr este egal cu 0 în cazul în care nu există niciun indice y : $poz \leq y \leq poz + L - 1$, pentru care $V_y = x$.
- (4) Pentru a găsi răspunsul la întrebare, dorim să găsim cea mai mică valoare val , pentru care se întâmplă: $cnt[val] > 0$ (ne asigurăm că val există în secvența de lungime L corespunzătoare întrebării curente) și: $cnt[1] + cnt[2] + \dots + cnt[val] \geq K$. Acest pas poate fi efectuat printr-o parcurgere liniară a tabloului de frecvență cnt .

Complexitatea totală a algoritmului este: $O(N + Q \times (L + Max))$.

Subtask 5: Întrucât $1 \leq V_i \leq 23$, pentru fiecare i : $1 \leq i \leq N$, înseamnă că, pentru fiecare dintre cele Q întrebări răspunsul este un număr din mulțimea: $\{1, 2, 3, \dots, 22, 23\}$.

Astfel, putem considera următorul tablou bidimensional (*de sume parțiale*): $num[x][i] = [\text{câte elemente din mulțimea } \{V_1, V_2, \dots, V_{i-1}, V_i\}, \text{ adică din prefixul } [1, i] \text{ din șir, sunt egale cu numărul întreg } x]$, pentru fiecare x : $1 \leq x \leq 23$ și i : $1 \leq i \leq N$. (Se consideră: $num[x][0] = 0$.)

Prin urmare, pentru a răspunde la o întrebare, trebuie, din nou, să determinăm cea mai mică valoare val ($1 \leq val \leq 23$), pentru care: $(num[val][poz + L - 1] - num[val][poz - 1]) > 0$ și:

$$\sum_{i=1}^{val} (num[i][poz + L - 1] - num[i][poz - 1]) \geq K.$$

Această metodă poate fi implementată *testând*, pe rând, fiecare valoare posibilă pe care val o poate lua; sunt cel mult 23 de astfel de valori.

Complexitatea totală a algoritmului este: $O(N \times 23 + Q \times 23) = O(23 \times (N + Q))$.

Subtask 6: O soluție care garantează trecerea cu succes a tuturor testelor, de exemplu, se poate baza pe metoda *Împărțirii în bucăți de mărime \sqrt{n}* (*Sqrt Decomposition* în Engleză); în România, această tehnică mai este cunoscută și sub numele de *Șmenul lui Bogdan Batog* și puteți citi mai multe pe infoarena.ro. Tehnica aceasta va fi utilizată pentru o mai eficientă procesare a, în esență, unor tablouri unidimensionale de frecvență/numărare.

De asemenea, vom alege să *pre-procesăm* (înainte de a citi numărul Q din fișierul de intrare) răspunsul pentru fiecare poziție de început posibilă poz a unei secvențe de lungime L . Sunt exact $N - L + 1$ astfel de poziții de început.

De remarcat! Pentru a evita procesarea de numere/valori negative, putem crește valoarea fiecărui element din șirul V cu un număr întreg constant; de exemplu, putem crește cu 50001 fiecare element: $V_i = V_i + 50001$, pentru fiecare i : $1 \leq i \leq N$. Acum, elementele din șirul inițial

pot avea doar valori întregi pozitive cuprinse în intervalul $[1, 100001]$. Când se va efectua afișarea unui răspuns, trebuie să avem grijă, la final, să scădem din el această constantă 50001.

Să notăm: $vMax$ = valoare maximă din șirul V , după ce fiecare element a fost crescut cu valoarea 50001. De vreme ce $\lceil \sqrt{vMax} \rceil \leq \lceil \sqrt{100001} \rceil = 317$, putem considera următorul tablou unidimensional cu 316 elemente, numerotate începând de la 1: $t[i] =$ [câte valori cuprinse între $((i-1) \times 317 + 1)$ și $\min((i \times 317), vMax)$ există în șirul V în cadrul secvenței de lungime L : $V_{poz}, \dots, V_{poz+L-1}$], pentru fiecare i : $1 \leq i \leq 316$; cu alte cuvinte, elementul $t[i]$ (sau *bucket-ul* i) reține informație despre valorile: $((i-1) \times 317 + 1), \dots, \min((i \times 317), vMax)$. Variabila poz reprezintă *indicele* cu ajutorul căruia efectuăm *pre-procesarea*: $1 \leq poz \leq N - L + 1$.

Când se realizează trecerea de la poz la $poz + 1$, avem grijă să **ștergem** o copie a valorii V_{poz} și să **adăugăm** o copie a valorii V_{poz+L} în structura reprezentată de t . Ambele operații, atât de ștergere, cât și de adăugare a unei valori x , pot fi realizate în $O(1)$, prin modificarea elementului corespunzător: $t[\lfloor \frac{x+316}{317} \rfloor]$.

Din nou, pentru a afla răspunsul pentru secvența curentă (din cadrul *pre-procesării*) trebuie determinată cea mai mică valoare val ($1 \leq val \leq 100001$), pentru care $t[j] > 0$ și: $t[1] + t[2] + \dots + t[j-1] + t[j] \geq K$, unde: $j = \lfloor \frac{val+316}{317} \rfloor$. Această operație poate fi efectuată în $O(\sqrt{vMax})$.

Complexitatea totală a algoritmului este: $O(N \times \sqrt{vMax} + Q)$.

Soluție alternativă (Prof. Ciurea Stelian): Iată și o soluție în $O(N \times \log(L) + Q)$. Ea depinde de existența unei structuri de date care să ne ofere operații de inserare, de ștergere și de aflare a maximumului și a minimumului în timp logaritm. Precalculăm într-un tablou unidimensional B răspunsurile la toate întrebările posibile. Astfel, $B[i]$ va reține răspunsul pentru secvența $V[i \dots i + L - 1]$ (pentru i : $1 \leq i \leq N - L + 1$).

Vom procesa toate secvențele de lungime L de la stânga la dreapta. Vom menține elementele secvenței curente în două grupe: în grupa 1 cele mai mici K elemente, iar în grupa 2 cele mai mari $L - K$ elemente. În acest caz, răspunsul la întrebare este valoarea maximă din grupa 1. Acum, presupunând că am calculat aceste două grupe pentru secvența care începe la poziția i , vom determina cele două grupe pentru secvența care începe la poziția $i + 1$. Pentru aceasta, observăm că secvența care începe la $i + 1$ conține în locul valorii $V[i]$ valoarea $V[i + L]$. În concluzie, vom șterge valoarea $V[i]$ din grupa în care se află și vom insera în grupa potrivită valoarea $V[i + L]$. Pentru a determina grupele în care facem ștergerea, respectiv inserarea, vom compara fiecare dintre cele două valori cu maximumul din grupa 1; dacă sunt mai mici, ștergerea și inserarea se vor face în grupa 1, altfel în grupa 2. Observație: dacă ștergerea și inserarea se fac în grupe diferite, atunci mărimile celor două seturi vor devia de la cele dorite (K și $L - K$). Putem reechilibra grupele transferând, după caz, maximumul grupei 1 în grupa 2 sau minimumul grupei 2 în grupa 1.

Pentru prima secvență, cele două grupe trebuie construite în mod diferit. O variantă este să inserăm primele K valori în grupa 1, apoi pe celelalte $L - K$ să le inserăm cu reechilibrarea grupelor. O altă variantă este să inserăm pur și simplu toate valorile în grupa 1 și să facem reechilibrarea grupelor doar în momentele în care notăm răspunsurile la întrebări.

O structură de date care oferă aceste operații este *multiset*, declarată în biblioteca `<set>`. O altă structură este un *heap*, care însă trebuie implementat cu grijă, căci varianta de bază nu oferă ștergerea unui element arbitrar, ci numai a maximumului/minimumului.

PROBLEMA 3: STRUGURI

Propusă de: prof. Marinela Șerban, Colegiul Național "Emil Racoviță", Iași

Se utilizează principiul **cutiei lui Dirichlet** la fiecare tură (pentru grămezile rămase). O demonstrație a acestuia se găsește, de exemplu, pe Wikipedia.

Pentru a determina o secvență cu proprietatea din enunț vom considera sumele parțiale:

$$S_1 = a_1$$

$$S_2 = a_1 + a_2$$

...

$$S_i = a_1 + a_2 + \dots + a_i (i : 1 \leq i \leq n)$$

...

$$S_n = a_1 + a_2 + \dots + a_n.$$

Avem două cazuri:

- există k cu S_k divizibil cu n , caz în care secvența căutată este a_1, \dots, a_k ;
- nu există sume care să fie divizibile cu n . În acest caz sumele dau la împărțirea la n resturi ce fac parte din mulțimea $\{1, 2, \dots, n-1\}$. Cum sunt n sume și $n-1$ resturi posibile, conform principiului cutiei lui Dirichlet, rezultă faptul că există două sume S_p și S_q ($p < q$) care la împărțirea la n dau același rest. Deci $S_q - S_p$ este divizibil cu n și putem lua secvența a_{p+1}, \dots, a_q

În funcție de implementare se pot obține timpi diferiți:

- (1) implementat cu Dirichlet deștept (cel mai rapid) – se calculează sumele la citire/refacere și la detectarea unui rest 0 sau a unui rest care a mai apărut se oprește procesul.
- (2) implementat cu căutare completă 0 în resturi apoi, dacă nu există un rest 0, căutarea a două resturi egale.
- (3) implementat cu cea mai lungă secvență detectată la fiecare pas (cele mai puține ture).
- (4) implementat cu cea mai scurtă secvență detectată la fiecare pas (cele mai multe ture).

Desigur, și aceste implementări pot fi îmbunătățite. De exemplu, la implementările (2), (3), (4), după detectarea a două resturi egale procesul poate fi oprit.

Având în vedere că la cerința 1 se cere **secvența** cea mai lungă, se poate deduce, dacă nu se cunoaște principiul cutiei lui Dirichlet, că **există o secvență** de N numere a cărei sumă este divizibilă cu N . Utilizând această idee se obțin pe rând secvențe cu această proprietate (pentru valori diferite ale lui N). Căutarea unei secvențe se poate face elementar, luând pe rând intervale $[st, dr]$ de lungimi $N, N-1, N-2, \dots$. În funcție de implementare, se pot obține punctaje diferite.

ECHIPA

Problemele pentru această etapă au fost pregătite de:

- Prof. Șerban Marinell, Colegiul Național "Emil Racoviță", Iași
- Prof. Ciurea Stelian, Colegiul Național "Samuel von Brukenthal", Sibiu
- Prof. Gorea-Zamfir Claudiu, Inspectoratul Școlar Județean Iași
- Prof. Coman Isabela, Colegiul Național de Informatică "Tudor Vianu", București
- Prof. Manolache Gheorghe, Colegiul Național de Informatică, Piatra Neamț
- Prof. Anton Cristina, Colegiul Național "Gheorghe Munteanu Murgoci", Brăila
- Stud. Popa Bogdan Ioan, Facultatea de Matematică și Informatică, Universitatea București
- Stud. Popa Sebastian, Facultatea de Matematică și Informatică, Universitatea București
- Stud. Onuț Andrei, Universitatea Yale, S.U.A.
- Prof. Lica Daniela, Centrul Județean de Excelență Prahova