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

Comisia științifică

22 martie 2026

## Problema 1. Operatii

*Propusă de: prof. Beiland Arnold, Liceul Teoretic, Carei*

### Subtaskurile 1 și 2

Putem simula operațiile descrise în enunț și după fiecare dintre acestea determinăm răspunsul prin construirea unui vector de frecvență și căutarea frecvenței maxime. Complexitatea acestei soluții este $O(N^2)$.

### Subtaskurile 3 și 4

Pe parcursul procesării operațiilor vom reține frecvența fiecărei valori (frecv$[v]$, $0 \leq v \leq 200\,000$), frecvența maximă ($fmax$), suma valorilor care au o anumită frecvență (sum$[f]$, $1 \leq f \leq N$), respectiv câte elemente au o anumită frecvență (cnt$[f]$, $1 \leq f \leq N$).

Actualizările acestora se pot face în O(1) la ambele tipuri de operații. Când frecvența elementului $x$ se schimbă din f1 = frecv$[x]$ în f2 = f1 ± 1, vom efectua actualizările:
`sum[f1] -= x; cnt[f1]--; sum[f2] += x; cnt[f2]++`.

Observăm că frecvența maximă poate să crească sau să scadă cu o singură unitate la ambele tipuri de operații, deci și $fmax$ se poate actualiza în timp constant. Complexitatea soluției este $O(N)$.

Pentru a folosi mai puțină memorie, în locul vectorilor `sum` și `cnt` putem reține aceste informattii doar pentru valorile $fmax$ și $fmax - 1$ (dar acest lucru nu este necesar pentru obținerea punctajului maxim).

# Problema 2. Pinball

*Propusă de: stud. Antonie Aureliu Valentin, Facultatea de Automatică și Calculatoare,*
*Universitatea Națională de Știință și Tehnologie Politehnica București*

## Varianta 1 de rezolvare - stud. Antonie Aureliu Valentin

Considerăm Max ca fiind valoarea maximă posibilă a lui $v_i$ în subtaskul curent.

• $C = 1,\ 1 \leq n \leq 1\,000,\ 1 \leq v_i \leq 1\,000$

Pentru acest subtask, este suficient să parcurgem punctajele participanților și să calculăm, pentru fiecare, numărul de divizori, printr-o parcurgere de la 1 la $v_i$. Apoi, scădem din $b$ suma divizorilor pentru a afla răspunsul la prima cerință. Complexitate $O(n \cdot \text{Max})$.

• $C = 1$ **Fără restricții suplimentare**

Pentru abordarea de punctaj maxim, modificăm ciurul lui Eratostene și ținem minte, pentru fiecare număr non-prim, cel mai mic / mare număr prim care îl divide (SPF / LPF). Complexitatea precalculării este $O(Max \log\log(Max))$. Folosind această precalculare, putem găsi numărul de divizori al unui număr $v_i$ în $O(\log(v_i))$. Astfel, complexitatea primei cerințe este $O(Max \log\log(Max) + n \cdot \log(Max))$.

O altă abordare pentru prima cerință este de a ne folosi de vectorul de numere prime până la $\sqrt{\text{Max}}$ în momentul în care factorizăm numerele din șir. Astfel, putem calcula numărul de divizori ai lui $v_i$ în complexitate $O(\pi(\sqrt{v_i}))$, unde $\pi(v_i)$ = numărul de numere prime $\leq v_i$. Această abordare nu obține punctaj maxim. Complexitatea este de $O(\text{Max} \log\log \text{Max} + n \cdot \pi(\sqrt{\text{Max}}))$.

• $C = 2, 1 \leq n \leq 50\,000$ **și** $1 \leq v_i \leq 100\,000$.
**Pentru toate testele din acest subtask, există soluție** $P \leq 100\,000$.

Putem parcurge șirul de numere și calcula numărul de divizori al fiecărui număr din șir $v_i$, folosind algoritmul clasic în $O(\sqrt{v_i})$. Pentru a găsi numărul de bomboane rămas, scădem din numărul total de bomboane numărul de divizori al fiecărui element al șirului. Apoi, iterăm prin toate numerele de la 1 la Max și verificăm dacă numărul curent are numărul de divizori găsit. Complexitate temporală $O((n + \text{Max}) \cdot \sqrt{\text{Max}})$.

• $C = 2$, $1 \leq v_i \leq 1\,000\,000$.
**Pentru toate testele din acest subtask, există soluție** $P \leq 1\,000\,000$.

Pentru acest subtask, putem modifica ciurul lui Eratostene, pentru a precalcula numărul de divizori al tuturor numerelor până la Max în $O(Max \log(Max))$, ținându-le minte într-un tablou unidimensional. Pentru a găsi numărul de bomboane rămas, scădem din bugetul $b$, numărul de divizori al fiecărui element al șirului. După aceea, parcurgem toate numerele de la 1 la Max și vedem dacă numărul curent are numărul de divizori găsit. Complexitate temporală $O(Max \log(Max) + n)$.

• $C = 2$ **Fără restricții suplimentare**

Pentru abordarea de 100 de puncte, plecăm de la soluția enunțată la ultimul subtask al cerinței 1 (SPF / LPF). Pe urmă, ne folosim de formula de la descompunerea în factori primi. Știm că $P = p^a q^b r^c \cdots \iff d(P) = (a+1)(b+1)(c+1)\ldots$. Cunoscând $d(P) = \text{nr}$, îl descompunem în factori primi, apoi, folosindu-ne de vectorul de numere prime până la $\sqrt{Max}$, obținut folosind ciurul lui Eratostene, putem genera punctajul cerut, astfel:

$$P = p^a q^b r^c \ldots$$

unde $p, q, r, \ldots$ reprezintă numere prime consecutive $p < q < r \ldots$, iar $a+1, b+1, c+1, \ldots$ reprezintă factori primi din descompunerea lui $d(P)$, $a+1 \geq b+1 \geq c+1 \ldots$.

Complexitate temporală $O(Max \log\log(Max) + n \cdot \log(Max))$.

*Observație*: Cerința 2 poate fi de asemenea rezolvată și folosind metoda programării dinamice, aflând astfel numărul minim cu nr divizori, dar această metodă nu este în programa clasei a VI-a.

# Problema 3. Snake

*Propusă de: stud. Boacă Andrei, Facultatea de Informatică, Universitatea "Alexandru Ioan Cuza", Iași*

## Varianta 1 de rezolvare - stud. Boacă Andrei

### Subtask 1.

Pentru acest subtask este suficient să verificăm fiecare secvență în timp $O(N)$, compresând elementele egale și făcând verificările necesare conform enunțului. Complexitatea totală a acestei soluții este $O(N^3)$.

### Observații.

Vom numi vecin distinct la stânga al poziției $i$ cea mai mare poziție $j < i$ cu proprietatea că $A_i \neq A_j$. De asemenea, numim vecin distinct la dreapta al poziției $i$ cea mai mică poziție $j > i$ cu proprietatea că $A_i \neq A_j$.

O secvență este *snake* dacă orice poziție cu vecin distinct atât la stânga cât și la dreapta este fie mai mare, fie mai mică decât ambii săi vecini distincți.

Observăm că pentru un $L$ setat, există un $R_{max}$ cu proprietatea că orice pereche $(L, R)$ cu $L < R \leq R_{max}$ este *snake* și orice pereche $(L, R)$ cu $R > R_{max}$ nu este *snake*. Pentru subtask-urile 2, 3 și 4 ne propunem să găsim $R_{max}$ pentru fiecare $L$, urmând ca răspunsul să fie dat de suma dintre $R_{max} - L$ pentru $L$ de la 1 la $N - 1$.

### Subtask 2.

Pentru un $L$ setat, putem parcurge de la stânga la dreapta pozițiile $L, L + 1, L + 2$ ș.a.m.d. păstrând mereu în memorie ultimele două numere distincte întâlnite pe parcurs (fie acestea $a$ și $b$, unde $a$ este penultimul număr, iar $b$ este ultimul). Atunci când suntem la o poziție $i$, trebuie să analizăm următoarele cazuri:

- Dacă $A_i = b$, continuăm către $i + 1$ fără a schimba nimic, deoarece nu avem un nou număr distinct la final.
- Dacă $a < b < A_i$ sau $a > b > A_i$ setăm $R_{max} = i - 1$ și ne oprim din parcurgerea pentru $L$-ul curent, fiindcă secvența nu va mai fi *snake* începând cu poziția $i$.
- Altfel, setăm $a = b$ și $b = A_i$ și continuăm către $i + 1$.

Această soluție face $O(N)$ pași pentru fiecare $L$, deci are o complexitate totală $O(N^2)$.

### Subtask 3.

Vom parcurge elementele șirului $A$ de la dreapta la stânga, modificând la fiecare pas în $O(1)$ valoarea $R_{max}$. Inițial avem $R_{max} = N$, iar pe măsură ce întâlnim o nouă poziție $L$, se va modifica în felul următor:

- Dacă $L + 2 \leq N$ și $A_L < A_{L+1} < A_{L+2}$ sau $A_L > A_{L+1} > A_{L+2}$ atunci $R_{max}$ devine $L + 1$, deoarece începând cu poziția $L + 2$ șirul nu va mai fi *snake*.
- În caz contrar, $R_{max}$ rămâne neschimbat.

Complexitatea acestei soluții este $O(N)$.

**Subtask 4.**

Pentru a obține soluția finală, ne rămâne să generalizăm soluția de la subtask-ul 3. Vom merge tot de la dreapta la stânga, cu $L$ de la $N-1$ până la 1. Vom considera $a$ și $b$ ca fiind ultimele două numere distincte întâlnite ($a$ este ultimul, iar $b$ este penultimul). De asemenea, $last_a$ este cea mai mare poziție cu proprietatea că orice poziție între $L$ și $last_a$ are valoarea $a$, iar $last_b$ este cea mai mare poziție cu proprietatea că orice poziție între $last_a + 1$ și $last_b$ are valoarea $b$. Înainte de a începe parcurgerea vom inițializa prin convenție $a = b = A_N$ și $R_{max} = last_a = last_b = N$. Atunci când întâlnim o nouă poziție $L$, vom analiza următoarele cazuri:

- Dacă $A_L = a$ nu schimbăm nimic.
- Dacă $A_L \neq a$ și $A_L < a < b$ sau $A_L > a > b$ atunci setăm $R_{max} = last_a$.
- Indiferent dacă cazul precedent se aplică sau nu, atunci când avem $A_L \neq a$ trebuie să setăm $b = a$, $last_b = last_a$, $a = A_L$ și $last_a = L$.

Complexitatea soluției finale este $O(N)$.

### Varianta a 2-a de rezolvare - prof. Carmen Mincă

Complexitate: $O(N)$.

Se compactează șirul eliminând valorile consecutive egale și reținând frecvențele lor. Problema se reduce la a număra subsecvențele din șirul compactat în care direcțiile (creștere/scădere) alternează.

Parcurgem șirul (de la stânga la dreapta) și menținem un segment maximal de tip snake. Fiecare element $a[i]$ din șirul comprimat, care face parte din segmentul curent de tip snake, mărește lungimea acestui segment cu $fr[i]$.

Pentru fiecare segment de lungime totală lg (ținând cont de frecvențe), numărul de subsecvențe (implicit de perechi de poziții) este: $lg \cdot (lg - 1)/2$. Pentru a evita dubla numărare la trecerea între segmente, scădem contribuția blocului comun: $scad = fr[i] \cdot (fr[i] - 1)/2$.

## Echipa

Problemele pentru această etapă au fost pregătite de:

- stud. Antonie Aureliu Valentin, Facultatea de Automatică și Calculatoare, Universitatea Națională de Știință și Tehnologie Politehnica București
- prof. Beiland Arnold, Liceul Teoretic, Carei

- stud. Boacă Andrei, Facultatea de Informatică, Universitatea "Alexandru Ioan Cuza", Iași
- prof. Gorea-Zamfir Claudiu-Cristian, Liceul Teoretic de Informatică "Grigore Moisil", Iași
- prof. Mincă Carmen-Nicoleta, Colegiul Național de Informatică ”Tudor Vianu”, București
- prof. Nodea Gheorghe-Eugen, Centrul Județean de Excelență, Gorj
- stud. Oancea Teodora, Facultatea de Matematică și Informatică, Universitatea "Babeș-Bolyai", Cluj-Napoca
- prof. Opriță Petru Simion, Liceul "Regina Maria", Dorohoi
- prof. Pintea Adrian Doru, Inspectoratul Școlar Județean, Cluj-Napoca
- stud. Porumb Valentin, Facultatea de Matematică și Informatică, Universitatea București
- prof. Pracsiu Dan, Liceul Teoretic "Emil Racoviță", Vaslui
- prof. Șandor Nicoleta-Lenuța, Colegiul Național "Mihai Eminescu", Satu Mare
- prof. Șerban Marinel Paul, Centrul Județean de Excelență, Iași