Olimpiada Națională de Informatică, Etapa județeană, Clasa a IX-a Descrierea soluțiilor

Comisia științifică

March 2, 2026

Problema 1. POSTA

Propusă de: Asistent doctor Alexandru Ioniță, Facultatea de Informatică, Universitatea Alexandru Ioan Cuza din Iași

Subtask 1. (65 puncte)

Se încearcă fiecare x între 1 și valoarea minimă (diferită de 1) din șir. Pentru a verifica dacă un x verifică restricțiile (vizitează casele $a_1, a_2, ..., a_n$) se verifică dacă restul împărțirii fiecarui $a_i - 1$ la x este 0. Complexitate: $O(n \cdot k$, unde k este valoarea minimă diferită de 1.

Subtask 2. (10 puncte)

Pornind de la ideea precedentă, putem observa că ne interesează doar valorile care divid $k = a_1 - 1$. Dacă $a_1 = 1$ atunci vom folosi $k = a_2 - 1$. Astfel, x va lua valori doar dintre divizorii acestui număr. Verificăm pentru fiecare x dintre aceste valori și returnăm valoarea maximă care respectă cerința. Deoarece numarul de divizori ai unui număr k este de ordinul $\sqrt[3]{k}$, complexitatea finala este: $O(n \cdot \sqrt[3]{k} + \sqrt{k}$.

Subtask 3. (25 puncte)

Se observă că x trebuie să dividă toate valorile $a_1 - 1, a_2 - 1, ..., a_n - 1$. Dacă x respectă această condiție, atunci Algorel va vizita toate casele $a_1, a_2, ..., a_n$. Astfel, condiția este necesară și suficientă. Prin urmare, răspunsul problemei este cel mai mare divizor comun al numerelor $a_1 - 1,\ a_2 - 1,\ \ldots,\ a_n - 1$.

Pentru a determina această valoare, calculăm mai întâi

$$d_2 = \gcd(a_1 - 1,\ a_2 - 1).$$

Apoi extindem treptat calculul, determinând

$$d_3 = \gcd(d_2,\ a_3 - 1),$$

și, în general,

$$d_k = \gcd(d_{k-1},\ a_k - 1), \quad \text{pentru } k = 3, \ldots, n.$$

În final, valoarea obținută d_n reprezintă cel mai mare divizor comun al tuturor numerelor $a_1 - 1, a_2 - 1, \ldots, a_n - 1$. Complexitate: $O(n \cdot \log(a_n))$.

Problema 2. CIBERNETICA

Propusă de: student Alex-Matei Ignat, Facultatea de Matematică și Informatică, Universitatea din București

Subtask 1. $K = 1$ (18 puncte)

Toate ferestrele sunt suspecte (toate conțin cel puțin un element distinct) și legitime (toate conțin puțin un element care apare cel puțin o dată), deci răspunsul va fi $\frac{N \cdot (N+1)}{2}$, indiferent de valoarea lui C. Complexitate: O(1)

Subtask 2. $C = 1$, $1 \leq N, K \leq 100$ (17 puncte)

Putem parcurge fiecare fereastră în parte și să stocăm frecvența fiecărui element cu ajutorul unui vector de frecvență, numerele fiind cel mult 10^6. Dacă măcar un element apare de cel puțin K ori, fereastra este suspectă și o contorizăm. Complexitate: O(N^3)

Subtask 3. $C = 1$, $1 \leq N, K \leq 1000$ (12 puncte)

Pentru a nu parcurge fiecare fereastră, putem să trecem ușor la fereastra $(i, j + 1)$ de la (i, j) dacă avem frecvența elementelor din fereastra (i, j). Luăm pe rând fiecare fereastră (i, i). Apoi, pentru fiecare fereastră (i, i), vom trece prin ferestrele (i, i), $(i, i+1)$, $(i, i+2)$, ..., (i, N). Reținem într-o variabilă X câte elemente cu frecvența cel puțin K există, această valoare fiind egală cu 0 inițial. Când trecem la o nouă fereastră, este suficient să adăugăm noua valoare la frecvență și să verificăm doar frecvența noii valori, iar dacă această frecvență este cel puțin K, atunci incrementăm X. Dacă X este nenul, fereastra este suspectă. Complexitate: $O(N^2)$

Subtask 4. $C = 1$, $1 \leq a_i \leq 2, 1 \leq i \leq N$ (7 puncte)

Observație: Dacă fereasta (i, j) este suspectă, atunci și ferestrele (i, k), $j \leq k \leq N$ sunt suspecte.

Pentru a rezolva cerința, pentru fiecare capăt din stânga fixat i, putem **căuta binar** cel mai mic indice $j \geq i$, astfel încât fereastra (i, j) este suspectă. Dacă transformăm elementele egale cu 1 în 0 și cele egale cu 2 în 1, se poate verifica ușor fereastra folosind **sume parțiale**. Fie $s(i, j)$ suma ferestrei (i, j). O fereastră este suspectă dacă $(j - i + 1) - s(i, j) \geq K$ (cel puțin K de 1) sau $s(i, j) \geq K$ (cel puțin K de 2). Complexitate: $O(NlogN)$.

Subtask 5. $C = 1$, fără alte restricții (9 puncte)

Putem folosi tehnica **two pointers**. Reținem doi indici i și j care încep de la 1. La fiecare pas, mutăm unul dintre ei, actualizând frecvența numerelor, folosind ideea de la **Subtask 3**. Dacă fereastra încă nu este suspectă, incrementăm indicele din dreapta j. Dacă fereastra este suspectă,

adunăm la rezultat toate posibilitățile cu indicele stâng i fixat și indicele drept fiind oricare din intervalul $[j, N]$, deci $(N - j + 1)$ posibilități, după care incrementăm indicele stâng i.

La incrementarea indicelui stâng i, dacă frecvența elementului scos din fereastră devine mai mică decât K, îl decrementăm pe X. Complexitate: $O(N)$

Subtask 6. $C = 2$, $1 \leq N, K \leq 100$ (14 puncte)

Folosind ideea de la **Subtask 2**, parcurgem fiecare fereastră în parte și stocăm frecvența fiecărui element cu ajutorul unui vector de frecvență. Dacă există cel puțin K valori cu frecvență nenulă, fereastra este legitimă și o contorizăm. Complexitate: O(N^3)

Subtask 7. $C = 2$, $1 \leq N, K \leq 1000$ (10 puncte)

Folosind ideea de la **Subtask 3**, putem să trecem ușor la fereastra $(i, j + 1)$ de la (i, j) dacă avem frecvența elementelor din fereastra (i, j). De data aceasta, reținem într-o variabilă X câte elemente distince există, această valoare fiind egală cu 0 inițial. Când trecem la o nouă fereastră, adăugăm noua valoare la frecvență, iar dacă această frecvență devine egală cu 1, înseamnă că noul element adăugat nu a mai apărut până atunci, deci vom incrementa X. Dacă $X \geq K$, fereastra este legitimă. Complexitate: $O(N^2)$

Subtask 8. $C = 2$, $1 \leq a_i \leq 2, 1 \leq i \leq N$ (8 puncte)

Observație: Dacă fereasta (i, j) este legitimă, atunci și ferestrele (i, k), $j \leq k \leq N$ sunt legitime.

Folosind ideea de la **Subtask 4**, pentru fiecare capăt din stânga fixat i, putem **căuta binar** cel mai mic indice $j \geq i$, astfel încât fereastra (i, j) este legitimă. Fie $s(i, j)$ suma ferestrei (i, j). Pentru $K = 1$ fereastra este tot timpul legitimă (Subtask 1). Pentru $K = 2$, fereastra este legitimă dacă $0 < s(i, j) < (j - i + 1)$ (să nu fie toate elementele 1 sau toate elementele 2). Pentru $K > 2$, fereastra nu poate fi legitimă (nu există 3 valori distincte în șir). Complexitate: $O(NlogN)$.

Subtask 9. $C = 2$, fără alte restricții (5 puncte)

Folosim ideea de la **Subtask 5**. Dacă fereastra încă nu este legitimă, incrementăm indicele din dreapta j. Dacă fereastra este legitimă, adunăm la rezultat toate posibilitățile cu indicele stâng i fixat și indicele drept fiind oricare din intervalul $[j, N]$, deci $(N - j + 1)$ posibilități, după care incrementăm indicele stâng i.

La incrementarea indicelui stâng i, dacă frecvența elementului scos din fereastră devine egală cu 0, îl decrementăm pe X. Complexitate: $O(N)$

Problema 3. COLLATZ

Propusă de: student Andrei Lețu, Facultatea de Matematică și Informatică, Universitatea "Babeș-Bolyai" Cluj-Napoca

Subtask 1 (63 de puncte)

Putem parcurge numerele de la s_i la d_i și să calculăm pentru fiecare numărul de operații. Pentru fiecare număr este nevoie de maxim aproximativ 3*log operații pentru a fi redus la 1 (numerele pare sunt înjumătățite iar un număr impar x se poate transforma în $x+1$ prin 2 operații succesive, deci în maxim 3 operații un număr devine aproximativ jumătatea sa).
Complexitatea este O($T \cdot N \cdot \log(N)$), unde N este limita capetelor drepte.

Subtask 2 (16 de puncte)

Putem precalcula numărul de operații pentru numerele de la 1 la N și să răspundem la întrebări în O(1) folosind sume de prefixe.
Complexitate O($T + N \cdot \log(N)$).

Subtask 3 (12 de puncte)

Putem folosi observația legată de numerele impare pentru a precalcula mai rapid vectorul parcurgând elementele într-o ordine par, impar, par, impar: 4, 3, 6, 5, 8, 7 etc. și folosind numerele deja calculate: prec[2k+2] va fi egal cu 1+prec[k+1] iar prec[2k+1] = 2+prec[2k+2].
Complexitatea este O($T + N$).

Subtask 4 (9 de puncte)

Putem calcula rezultatul fără precalculări, lucrând cu tot intervalul deodată. Ne dorim ca primul număr din interval să fie impar iar ultimul să fie par (dacă asta nu se întâmplă calculăm separat valoarea pentru primul și/sau ultimul element din interval pentru a-l aduce la forma dorită).

9	10	11	12	...	197	198	199	200

Aplicând 2 operații pe numerele impare obținem un vector format din numere pare consecutive, fiecare apărând de câte 2 ori.

10	10	12	12	...	198	198	200	200

Aplicând operația de înjumătățire pe acest vector obținem un vector ce conține numerele de la $\frac{s_i+1}{2}$ până la $\frac{d_i}{2}$.

5	6	7	...	98	99	100
5	6	7	...	98	99	100

Pașii se pot repeta, ținând într-o variabilă multiplicitatea numerelor.

3	4	5	...	48	49	50
3	4	5	...	48	49	50
3	4	5	...	48	49	50
3	4	5	...	48	49	50

etc.
Complexitatea este O($T \cdot \log(N)$).

Echipa

Problemele pentru această etapă au fost pregătite de:

- Student **Alex-Matei Ignat**, Universitatea din București, Facultatea de Matematică și Informatică, București
- Programator **Alexandra Nicola**, Google, Bucuresti
- Asistent doctor **Alexandru Ioniță**, Universitatea "Alexandru Ioan Cuza", Facultatea de Informatică, Iași
- Student **Alexandru-Mihai Lușcan**, Universitatea "Babeș-Bolyai", Facultatea de Matematică și Informatică, Cluj-Napoca
- Profesor **Alice Georgescu**, Colegiul Național "Mihai Viteazul", Ploiești
- Student **Andrei-Cristian Ivan**, Universitatea Națională de Științe și Tehnologie Politehnica, Facultatea de Automatică și Calculatoare, București
- Student **Andrei Lețu**, Universitatea "Babeș-Bolyai", Facultatea de Matematică și Informatică, Cluj-Napoca
- Profesor **Cristina Anton**, Colegiul Național "Gheorghe Munteanu Murgoci", Brăila
- Student **David Vișan**, Southern University of Denmark, Sønderborg
- Programator **Mihai Gheorghe**, Academia de Studii Economice, Facultatea de Cibernetică, București
- Student **Mircea Măierean**, ETH, Zurich
- Lector doctor **Paul Diac**, Universitatea "Alexandru Ioan Cuza", Facultatea de Informatică, Iași
- Student **Raul Ardelean**, Universitatea "Babeș-Bolyai", Facultatea de Matematică și Informatică, Cluj-Napoca
- Student **Robert Moldovan**, Universitatea "Babeș-Bolyai", Facultatea de Matematică și Informatică, Cluj-Napoca
- Student **Sebastian Predescu**, Universitatea Națională de Științe și Tehnologie Politehnica, Facultatea de Automatică și Calculatoare, București
- Student **Victor Botnaru**, Universitatea Națională de Științe și Tehnologie Politehnica, Facultatea de Automatică și Calculatoare, București