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

Comisia științifică

March 2, 2026

## Problema 1. Bonsai

*Propusă de: stud. Cotoi Rareș-Andrei, Universitatea „Babeș-Bolyai", Cluj-Napoca*

### Cerința 1.

Un bonsai de rază $R = 0$ poate fi plantat pe parcela $A[i][j] > 0$ dacă una dintre următoarele condiții este adevărată:

- $i = 1$ sau $i = N$ sau $j = 1$ sau $j = M$;
- $A[i-1][j] = 0$ sau $A[i][j-1] = 0$ sau $A[i+1][j] = 0$ sau $A[i][j+1] = 0$.

Pentru rezolvarea cerinței 1 este suficient să parcurgem matricea $A$ și să verificăm pentru fiecare $A[i][j]$ dacă una din condițiile de mai sus este adevărată.
Pentru a rezolva eficient cerințele **2** și **3**, este necesar să cunoaștem rapid, pentru orice parcelă $A[i][j]$, cât de mult ne putem extinde în cele patru direcții (Nord, Sud, Est, Vest) fără a întâlni o valoare nulă. Vom construi patru matrice auxiliare, astfel:

- $U[i][j]$: lungimea maximă a brațului în sus pornind din $(i, j)$;
- $D[i][j]$: lungimea maximă a brațului în jos;
- $L[i][j]$: lungimea maximă a brațului la stânga;
- $R[i][j]$: lungimea maximă a brațului la dreapta.

Pentru fiecare element, relațiile de recurență sunt:

- stânga ($L$) și sus ($U$) — se calculează parcurgând matricea de la $(1, 1)$ la $(N, M)$:

$$L[i][j] = \begin{cases} 0, & \text{dacă } A[i][j] = 0 \\ L[i][j-1] + 1, & \text{altfel} \end{cases}$$

$$U[i][j] = \begin{cases} 0, & \text{dacă } A[i][j] = 0 \\ U[i-1][j] + 1, & \text{altfel} \end{cases}$$

- dreapta ($R$) și jos ($D$) — se calculează parcurgând matricea invers, de la $(N, M)$ la $(1, 1)$:

$$R[i][j] = \begin{cases} 0, & \text{dacă } A[i][j] = 0 \\ R[i][j+1] + 1, & \text{altfel} \end{cases}$$

$$D[i][j] = \begin{cases} 0, & \text{dacă } A[i][j] = 0 \\ D[i+1][j] + 1, & \text{altfel} \end{cases}$$

Raza maximă posibilă a unui bonsai cu centrul în $(i, j)$ este minimul dintre extinderile în cele 4 direcții:

$$Raz_{max}[i][j] = \min(U[i][j], D[i][j], L[i][j], R[i][j]) - 1$$

(scădem 1 deoarece toate cele 4 matrice auxiliare includ și centrul $(i, j)$ în lungimea calculată.)

## Cerința 2.

Având calculate valorile $Raz_{max}[i][j]$ pentru toate celulele matricei, soluția constă în parcurgerea acestora pentru a determina valoarea maximă globală $R_{max}$ (dacă $Raz_{max}[i][j] > R_{max}$, actualizăm $R_{max}$).
Complexitatea timp este $O(N \cdot M)$, iar această soluție obține 41 de puncte pentru $C = 2$. O abordare mai ineficientă care obține, de asemenea, 31 de puncte este parcurgerea matricei și verificarea, pentru fiecare element, cât ne putem extinde pe cele 4 direcții, fără a întâlni o valoarea nulă sau a ieși din matrice și reținem valoarea maximă.

## Cerința 3.

Deoarece importanțele parcelelor sunt numere naturale, suma maximă pentru un centru fixat se obține întotdeauna alegând raza maximă posibilă ($Raz_{max}[i][j]$).
O abordare directă presupune ca, pentru fiecare centru $(i, j)$, să parcurgem element cu element brațele bonsaiului de rază $Raz_{max}[i][j]$ pentru a calcula suma. Complexitatea acestei abordări este $O(N \cdot M \cdot \min(N, M))$ și obține 9 puncte pentru $C = 3$.
Pentru a calcula suma în $O(1)$, vom utiliza sume parțiale pe linii și pe coloane:

- $SP\_Lin[i][j]$: suma elementelor de pe linia $i$, de la coloana 1 la $j$;
- $SP\_Col[i][j]$: suma elementelor de pe coloana $j$, de la linia 1 la $i$.

Suma unui bonsai cu centrul în $(i, j)$ și raza $r = Raz_{max}[i][j]$ se calculează astfel:

- Suma pe orizontală ($S_H$): $SP\_Lin[i][j + r] - SP\_Lin[i][j - r - 1]$;
- Suma pe verticală ($S_V$): $SP\_Col[i + r][j] - SP\_Col[i - r - 1][j]$;
- Suma Totală: $S_H + S_V - A[i][j]$ (scădem centrul o dată deoarece a fost inclus în ambele sume).

Vom reține soluția care oferă suma maximă, respectând criteriile de departajare din enunț (raza maximă, apoi linia minimă, apoi coloana minimă). Complexitatea totală este $O(N \cdot M)$.

## Problema 2. Mario

*Propusă de: stud. Mogovan Jonathan, Universitatea „Babeș-Bolyai", Cluj-Napoca*

### Cerința 1.

Se parcurg simultan cele două șiruri și se calculează $|A[i] - B[i]|$ pentru fiecare indice $i$. Se reține maximul curent și indicele primei apariții a acestuia, actualizând indicele doar dacă diferența curentă îl *depășește strict* pe cel reținut (pentru a garanta indicele minim). Complexitatea de timp este $O(N)$.

### Cerința 2.

Trebuie să găsim indicele de început al ferestrei de lungime fixă $K$ pe care Mario consumă energia totală maximă, cu preferință pentru indicele minim în caz de egalitate.
**Soluție naivă — $O(N \cdot K)$.** Pentru fiecare fereastră de lungime $K$, se recalculează suma de la zero. Această abordare obține punctaj parțial.

**Soluție eficientă — Sume prefixate, $O(N)$.** Definim $SP[0] = 0$ și $SP[i] = SP[i-1] + A[i]$ pentru $i \geq 1$. Suma oricărei ferestre de lungime $K$ care începe la indicele $i$ este:

$$S_i = SP[i + K - 1] - SP[i - 1].$$

Se parcurg toate ferestrele posibile calculând $S_i$ în $O(1)$ per pas, menținând maximul și indicele asociat. Complexitatea totală este $O(N)$.

### Cerința 3.

Trebuie să găsim un segment $[st, dr]$ pe care suma energiilor lui Mario este egală cu suma energiilor lui Wario, alegând segmentul cu $st$ minim, iar la egalitate pe cel cu $dr$ minim.
**Reformulare matematică.** Definim șirul diferențelor $\text{diff}[i] = A[i] - B[i]$ și vectorul sumelor prefixate:

$$P[0] = 0, \qquad P[k] = \sum_{i=1}^{k} \text{diff}[i], \quad k \geq 1.$$

Condiția ca energiile totale să fie egale pe segmentul $[st, dr]$ devine:

$$P[dr] - P[st - 1] = 0 \iff P[dr] = P[st - 1].$$

Problema se reduce la *găsirea a doi indici $p < q$ în vectorul $P$ cu $P[p] = P[q]$*, unde segmentul corespunzător este $[p + 1,\ q]$.

**Soluție naivă — $O(N^3)$.** Se verifică toate perechile de indici $(st, dr)$ și se recalculează suma de la zero pentru fiecare segment. Această abordare obține punctaj parțial.

**Soluție îmbunătățită — $O(N^2)$.** Se elimină cel de-al treilea ciclu observând că suma pe segmentul $[st, dr]$ se poate actualiza incremental: pornind de la $st$ fix, suma pentru $[st, dr+1]$ se obține adăugând $\text{diff}[dr+1]$ la suma deja calculată pentru $[st, dr]$. Astfel, pentru fiecare $st$ fix parcurgem $dr$ de la $st$ la $N$, menținând suma curentă și verificând dacă aceasta este 0. Complexitatea este $O(N^2)$ și obține punctaj parțial.

**Soluție intermediară — $O(N \log N)$, nu obține punctajul maxim.** Construim vectorul de perechi $(P[i],\ i)$ pentru $i = 0, 1, \ldots, N$ și îl sortăm crescător după valoare, cu departajare după poziție. După sortare, doi indici cu $P[p] = P[q]$ vor fi vecini în tabloul sortat. Parcurgând tabloul sortat, la fiecare pereche de elemente consecutive cu aceeași valoare identificăm un segment valid $[p+1,\ q]$ (unde $p < q$ sunt pozițiile originale). Se reține segmentul cu $st = p+1$ minim, respectiv $dr = q$ minim la egalitate de $st$. Complexitatea este dominată de sortare: $O(N \log N)$. Această abordare este corectă și se implementează elegant, însă pentru $N = 10^6$ poate depăși limita de timp, neobținând punctajul maxim.

**Soluție optimă — $O(N)$ cu vector de frecvență și offset.** Din restricțiile problemei, valorile $P[k]$ se află în intervalul $[-450\,000,\ 450\,000]$. Adăugând constanta $\text{OFFSET} = 450\,000$, toate valorile devin pozitive, permițând utilizarea unui vector static *firstPos* de dimensiune 900 001, inițializat cu $-1$.
Algoritmul parcurge șirul o singură dată:

- Se inițializează $\textit{firstPos}[\text{OFFSET}] = 0$ (valoarea $P[0] = 0$ apare înaintea primei sărituri).
- La pasul $i$, dacă $\textit{firstPos}[P[i] + \text{OFFSET}] = p \neq -1$, segmentul $[p+1,\ i]$ este valid și se actualizează soluția conform criteriilor de departajare.
- Altfel, se memorează $\textit{firstPos}[P[i] + \text{OFFSET}] = i$.

Dacă nu există nicio pereche cu valori egale în $P$, se afișează $-1\ -1$. Complexitatea timp este $O(N)$.

## Echipa

Problemele pentru această etapă au fost pregătite de:

- prof. Costineanu Raluca Veronica, Colegiul Național „Ștefan cel Mare", Suceava
- prof. Panaete Adrian, Colegiul Național „A.T. Laurian", Botoșani
- prof. Popa Daniel, Colegiul Național „Aurel Vlaicu", Orăștie
- prof. Rotar Dorin-Mircea, Colegiul Național „Samuil Vulcan", Beiuș
- stud. Cotoi Rareș-Andrei, Universitatea „Babeș-Bolyai", Facultatea de Matematică și Informatică, Cluj-Napoca
- stud. Mogovan Jonathan, Universitatea „Babeș-Bolyai", Facultatea de Matematică și Informatică, Cluj-Napoca

- stud. Rotaru Răzvan-Alexandru, Universitatea „Alexandru Ioan Cuza", Facultatea de Informatică, Iași
- stud. Mitri Robert-Cristian, Universitatea Națională de Științe și Tehnologie Politehnica, Facultatea de Automatică și Calculatoare, București
- stud. Borodi Bogdan, Universitatea „Babeș-Bolyai", Facultatea de Matematică și Informatică, Cluj-Napoca