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

Comisia științifică

February 28, 2026

Problema 1. Algocoin

Propusă de: șef lucrări dr. Mihai NAN, Universitatea Națională de Științe și Tehnologie Politehnica, Bucureşti, Facultatea de Automatică și Calculatoare

Cerința 1 – Subtask1

Pentru a determina numărul de monede primite de fiecare elev, trebuie mai întâi să ordonăm punctajele crescător.

După sortarea punctajelor, parcurgem vectorul și atribuim pozițiile în clasament astfel:

- primul elev (cel cu scor minim) primește poziția 1;
- pentru orice $i > 1$, dacă $scor_i = scor_{i-1}$, atunci poziția rămâne aceeași;
- în caz contrar, poziția devine i (indexare de la 1).

Pentru fiecare elev calculăm: $monede_i = scor_i \times poziție_i$.

Complexitatea este determinată de sortare, deci $O(N^2)$, dacă se utilizează Bubble Sort, Selection Sort sau Insertion Sort. Pentru punctaj maxim pentru acest subtask este necesară utilizarea unui algoritm de sortare eficient. Utilizând funcția `sort` din STL realizăm sortarea în complexitatea $O(N \log N)$. Atribuirea pozițiilor și calcularea numărului de monede primite de fiecare elev se realizează printr-o singură parcurgere liniară a vectorului.

Cerința 2 – Subtask 2

O soluție directă este generarea tuturor perechilor de elevi. Parcurgem toate perechile (i, j), cu $1 \leq i < j \leq N$, și verificăm dacă: $(monede_i + monede_j)\%K = 0$. Pentru fiecare pereche care respectă condiția, incrementăm un contor. Această metodă verifică explicit toate cele $\frac{N(N-1)}{2}$ perechi posibile.

Complexitatea este O(N^2).

Cerința 2 – Subtask 3

Observăm că este suficient să lucrăm cu resturile la împărțirea la K. Pentru fiecare elev calculăm: $r_i = monede_i \% K$. Construim un vector de frecvență $freq[0 \dots K-1]$, unde $freq[r] =$ numărul de elevi cu restul r. Două resturi r și s formează o pereche validă dacă: $(r+s)\%K = 0$.

Numărul total de perechi se calculează analizând cazurile:

- **Cazul** $r = 0$ Elevii cu rest 0 pot forma perechi între ei. Numărul de perechi este:
$$\frac{freq[0] \cdot (freq[0]-1)}{2}.$$
- **Cazul general** $1 \le r < \left\lceil \frac{K}{2} \right\rceil$ Elevii cu rest r pot fi combinați cu cei cu rest $c = K - r$. Numărul de perechi este: $freq[r] \cdot freq[c]$.

  Observație: Am considerat doar valorile $r < \left\lceil \frac{K}{2} \right\rceil$ pentru a nu dubla numărarea.
- **Cazul special** K **par și** $r = K/2$ În acest caz, $r = c$, deci elevii cu rest $K/2$ pot forma perechi doar între ei. Numărul de perechi este:
$$\frac{freq[K/2] \cdot (freq[K/2]-1)}{2}.$$

Complexitatea este $O(N+K)$, deoarece calcularea resturilor se face în $O(N)$, iar parcurgerea vectorului de frecvență în $O(K)$.

Cerința 3 – $O(N \log M)$

Notăm cu *monede* vectorul obținut la cerința 1 și îl considerăm sortat crescător (avem nevoie oricum de sortare pentru a determina numărul de monede pentru fiecare elev).

Dorim să determinăm cel mai mare multiplu X al lui K pentru care putem forma cel puțin P perechi disjuncte (fiecare elev utilizat în cel mult o pereche), astfel încât: $monede_i + monede_j > X$. Observăm că dacă pentru o valoare X putem forma cel puțin P perechi, atunci pentru orice $X' < X$ proprietatea rămâne adevărată. Prin urmare, putem aplica o căutare binară asupra valorilor posibile ale lui X (multipli ai lui K).

O modalitate de a verifica dacă pentru o valoare candidat X se pot obține cel puțin P perechi în condițiile din enunț este de a utiliza tehnica celor doi pointeri:

- $st = 1, dr = N$;
- dacă $monede[st] + monede[dr] > X$, formăm o pereche și o contorizăm, apoi deplasăm incrementăm st și decrementăm dr;
- în caz contrar, incrementăm doar st, deoarece este necesară o valoare mai mare în pereche.

Dacă putem forma cel puțin P perechi, valoarea X este validă. Pentru fiecare pas al căutării binare, verificarea se face în $O(N)$, iar numărul de pași este $O(\log M)$, unde M este suma maximă posibilă a două valori din vector. Astfel, complexitatea totală este $O(N \log M)$. Metoda prezentată este suficient de eficientă pentru limitele impuse.

Pentru subtaskul 4, se poate realiza o căutare secvențială a valorii X.

Cerința 3 – $O(P)$ – Eduard Pîrțac

Pentru a putea forma P perechi disjuncte, este suficient să alegem cele mai mari $2P$ valori din vector. Dacă există o soluție cu P perechi, atunci putem presupune că aceasta utilizează doar elemente dintre cele mai mari $2P$, deoarece înlocuirea unui element cu unul mai mare nu poate micșora suma perechii.

Prin urmare, vom lucra doar cu elementele: $monede[N-2P+1], monede[N-2P+2], \ldots, monede[N]$.

Pentru a maximiza valoarea minimă a sumei dintre cele P perechi, le vom forma astfel:

$$monede[N-2P+1] \text{ cu } monede[N],$$
$$monede[N-2P+2] \text{ cu } monede[N-1],$$
$$\ldots$$
$$monede[N-P] \text{ cu } monede[N-P+1].$$

Demonstrație pentru strategia optimă

În continuare, vom demonstra de ce este optimă această strategie.

Fie a, b, c, d patru numere oarecare din ultimele $2P$ monede, în ordinea în care apar ele în șir (deci $a \leq b \leq c \leq d$).

Trebuie să formăm două perechi de monede cu acestea. Vom demonstra în continuare că împerecherea $[a, d], [b, c]$ este mereu optimă. Vom analiza cele două împerecheri distincte posibile.

Cazul 1

Presupunem că facem împerecherea $[a, c], [b, d]$ și că am găsit X maxim pentru care $\min\{a + c, b + d\} > X$.

- Știm că $a + c > X$, dar combinând cu faptul că $b \geq a$, obținem că $b + c > X$.
- Știm tot că $a + c > X$, dar combinând cu faptul că $d \geq c$, obținem că $a + d > X$.

Astfel, am obținut că $\min\{b + c, a + d\} > X$, adică împerecherea $[a, d], [b, c]$ este cel puțin la fel de bună în acest caz.

Cazul 2

Presupunem că facem împerecherea $[a, b], [c, d]$ și că am găsit X maxim pentru care $\min\{a + b, c + d\} > X$.

- Știm că $a + b > X$, dar combinând cu faptul că $d \geq b$, obținem că $a + d > X$.
- Știm tot că $a + b > X$, dar combinând cu faptul că $c \geq a$, obținem că $c + b > X$.

Astfel, am obținut că $\min\{b + c, a + d\} > X$, adică împerecherea $[a, d], [b, c]$ este cel puțin la fel de bună în acest caz.

Am demonstrat că, alegând oricare patru numere a, b, c, d din ultimele $2P$, este mereu optim să împerechem a cu d și b cu c. Această demonstrație se poate scala la mai multe perechi, nu

doar la două. Astfel, obținem că răspunsul corect poate fi determinat de împerecherea dintre cel mai mic număr cu cel mai mare până epuizăm toate $2P$ numerele.

Acum că am demonstrat care este varianta optimă în care putem grupa ultimii $2P$ elevi, trebuie să calculăm $mn = \min_{0 \le j < P} (monede[N - j] + monede[N - 2P + 1 + j])$. Valoarea mn reprezintă cea mai mică sumă dintre cele P perechi formate optim. Pentru ca toate cele P perechi să satisfacă $monede[i] + monede[j] > X$, trebuie să avem $mn > X$. Prin urmare, cel mai mare multiplu de K strict mai mic decât mn este: $X = \left\lfloor \frac{mn-1}{K} \right\rfloor \cdot K$.

Dacă $mn = 0$, atunci nu există niciun multiplu nenegativ X pentru care condiția strictă $mn > X$ să fie satisfăcută, deci nu există soluție validă.

După ce avem vectorul sortat, calcularea valorii mn poate fi făcută în $O(P)$.

Problema 2. Pisicesc

Propusă de: prof. Emanuela Cerchez, Colegiul Național "Emil Racoviță" Iași

Cerința 1.

Vom parcurge şirul de caractere care reprezintă fraza în limbaj pisicesc şi vom contoriza numărul de apariţii ale fiecărei litere într-un vector de frecvenţă. Vom determina apoi pentru câte vocale frecvenţa este nenulă.

Cerinţa 2.

Vom căuta aparițiile secvenței `"mau"` în frază apelând succesiv funcția `strstr()`. O altă variantă care nu necesită lucrul cu funcții pe șiruri de caractere este de a parcurge șirul și de a verifica dacă fiecare secvență de 3 litere consecutive coincide cu secvența `"mau"`.

Cerinţa 3 - Subtask 3

Vom construi toate secvenţele din frază şi vom determina pentru fiecare secvenţă numărul de apariţii. Comparăm numărul de apariţii cu numărul maxim de apariţii şi vom reţine în soluţie secvenţa curentă dacă are un număr de apariţii mai mare decât numărul maxim de apariţii curent, sau dacă are acelaşi număr maxim de apariţii dar o lungime mai mare sau dacă are acelaşi număr maxim de apariţii, aceeaşi lungime, dar este mai mică din punct de vedere lexicografic decât soluţia curentă.

Cerinţa 3 - Subtask 4

Soluţia poate fi optimizată analizând secvenţele în ordinea crescătoare a lungimii lor.
Observaţie: orice secvenţă din soluţie va avea acelaşi număr de apariţii cu soluţia. Prin urmare, dacă numărul maxim de apariţii pentru o secvenţă de lungime lg este mai mic decât numărul maxim de apariţii pentru secvenţele de lungime $lg - 1$, atunci analizarea secvenţelor de lungime mai mare decât lg nu poate conduce la o soluţie (numărul de apariţii ale acestora va fi mai mic decât maximul curent).

Cerința 3 - Soluția 2

stud. Cristian Luchian, Universitatea Alexandru Ioan Cuza, Facultatea de Informatică, Iași

Putem considera fiecare secvență de două caractere din alfabetul `"aeiuom"`, acestea fiind singurele care pot apărea la începutul unui cuvânt și verificăm cât de mult le putem extinde. Pentru a verifica cât de mult poate fi extinsă o astfel de secvență, mai întâi reținem într-un vector v toate pozițiile pe care apare în frază secvența (să spunem că avem n astfel de poziții). Presupunând că am ajuns la lungimea L, putem ajunge la lungimea $L + 1$ doar dacă pentru orice $1 \leq i < n$ avem că $sir[v[i] + L] = sir[v[i + 1] + L]$. Dacă verificăm secvențele de două caractere în ordine lexicografică atunci răspunsul căutat va fi prima pereche pentru care avem frecvență maximă și în caz de egalitate lungime maximă. Acest algoritm are complexitatea O(n) deoarece orice secvență de două caractere am alege inițial este imposibil ca secvența inițială să apară de două ori în mesaj, deci nu va exista nici un caracter care va fi verificat de două ori.

Echipa

Problemele pentru această etapă au fost pregătite de:

- prof. Emanuela Cerchez, Colegiul Național "Emil Racoviță" Iași
- șef lucrări dr. Mihai Nan, Facultatea de Automatică și Calculatoare, Universitatea Națională de Științe și Tehnologie Politehnica Bucureşti
- prof. Ionel-Vasile Piț-Rada, Colegiul Național Traian, Drobeta Turnu Severin
- prof. Sandor Lukacs, Colegiul Național "Onisifor Ghibu", Oradea
- prof. Filonela Bălașa, Colegiul Național "Grigore Moisil", București
- prof. Alina Gabriela Boca, Colegiul Național de Informatică Tudor Vianu, București
- prof. Vlad-Laurențiu Nicu, Liceul Teoretic Mihail Kogălniceanu, Vaslui
- stud. Eduard-Lucian Pîrțac, Facultatea de Informatică, Universitatea "Al. I. Cuza" Iași
- stud. Alin Gabriel Răileanu, Facultatea de Informatică, Universitatea "Al. I. Cuza" Iași
- stud. Matei Mocanu, Facultatea de Automatică și Calculatoare, Universitatea Tehnică "Gheorghe Asachi" Iaşi
- stud. Dumitru Ilie, Facultatea de Matematică-Informatică, Universitatea București
- stud. Mihnea-Alexandru Turcu, Facultatea de Matematică și Informatică, Universitatea "Babeș-Bolyai" Cluj
- stud. Cristian Luchian, Facultatea de Informatică, Universitatea "Al. I. Cuza" Iași
- stud. Daniel Gheorghe, Facultatea de Matematică-Informatică, Universitatea București