OLIMPIADA NAȚIONALĂ DE INFORMATICĂ, ETAPA JUDEȚEANĂ/A SECTOARELOR MUNICIPIULUI BUCUREȘTI
CLASA A VI-A
DESCRIEREA SOLUȚIILOR

COMISIA ȘTIINȚIFICĂ

Problema 1: AVID

Propusă de: Stud. Buzatu Giulian, Facultatea de Matematică și Informatică, Universitatea București

Cerința 1: Pentru 12 puncte, se citesc numerele din fișierul de intrare într-un vector și numărăm câte triplete respectă condiția din enunț. Vom calcula cel mai mare divizor comun al fiecărui triplet de valori de pe poziții consecutive, după care vom calcula numărul de divizori al acestuia, iar dacă este mai mic sau egal cu p, vom incrementa răspunsul. Vom folosi algoritmul lui Euclid prin împărțiri repetate și descompunerea în factori primi până la radical. Complexitatea este $\mathcal{O}(n(\log \max\{a_i\} + \sqrt{\max\{a_i\}})) = \mathcal{O}(n\sqrt{\max\{a_i\}})$.

Pentru restul de 17 puncte, vom folosi același procedeu pentru a calcula numărul de triplete, dar vom calcula mai eficient numărul de divizori. Vom precalcula numerele prime până la $\sqrt{5\,000\,000} \approx 2\,236$, folosind ciurul lui Eratostene. Astfel, când vom face descompunerea în factori primi până la radicalul numărului nostru, vom folosi doar numerele prime. Complexitatea este $\mathcal{O}(n\sqrt{\max\{a_i\}} + \sqrt{\max\{a_i\}}\log\log\sqrt{\max\{a_i\}})$. Observăm că, deși complexitatea este similară, algoritmul este mult mai rapid în practică.

Cerința 2: Pentru 29 de puncte, se citesc numerele din fișierul de intrare într-un vector. Parcurgem vectorul și verificăm dacă tripletul care se termină pe poziția curentă respectă condiția din enunț, folosind algoritmul lui Euclid prin împărțiri repetate și descompunerea în factori primi până la radical. În caz afirmativ, incrementăm lungimea secvenței curente. Altfel, se încearcă actualizarea maximului, doar dacă lungimea curentă este cel puțin 3, și după se resetează lungimea secvenței curente la valoarea 2. La finalul parcurgerii se va mai încerca o dată actualizarea maximului, cu aceeași condiție ca mai sus, deoarece este posibil ca secvența curentă să fie maximală. Complexitatea este $\mathcal{O}(n(\log \max\{a_i\} + \sqrt{\max\{a_i\}})) = \mathcal{O}(n\sqrt{\max\{a_i\}})$.

Pentru restul de 42 de puncte, se procedează similar, dar vom calcula mai eficient numărul de divizori, folosind modul descris în cel de-al doilea paragraf de la descrierea cerinței 1. Complexitatea este $\mathcal{O}(n\sqrt{\max\{a_i\}} + \sqrt{\max\{a_i\}}\log\log\sqrt{\max\{a_i\}})$. Observăm că, deși complexitatea este similară, algoritmul este mult mai rapid în practică.

Observație: Pentru ambele cerințe putem să rezolvăm problema fără a reține datele citite într-un vector. Totuși, pentru simplitate și pentru că limita de memorie este generoasă, nu este nevoie să facem acest lucru.

Observație: Descompunerea în factori primi a lui n (mergând până la $\sqrt{n}$) este folosită pentru determinarea numărului său de divizori astfel: fie $n = p_1^{e_1} \cdot p_2^{e_2} \cdots p_k^{e_k}$, atunci numărul n are $(e_1+1)(e_2+1)\cdots(e_k+1)$ divizori, conform regulii produsului.

Observație: Pentru soluția fără ciur, numărul de divizori al lui n poate fi calculat în $\mathcal{O}(\sqrt{n})$ și fără formula precedentă. Este suficient să parcurgem divizorii lui n cuprinși între 1 și $\sqrt{n}$. De fiecare dată când găsim un divizor d, deducem că și n/d este un divizor al lui n, așa că adunăm 2 la rezultat. Singura excepție constă în cazul în care $d = n/d$, când se adună doar 1.

Problema 2: PERECHI

Propusă de: Alina-Gabriela Boca, profesor la Colegiul Național de Informatică Tudor Vianu, București

Cerința 1: Pentru 27 de puncte, se citesc numerele din fișierul de intrare într-un vector, se parcurg elementele vectorului, iar pentru fiecare element aflat pe poziția i, cu $1 \leq i \leq n-1$, se verifică toate elementele aflate pe pozițiile j, cu $i+1 \leq j \leq n$, numărându-se toate elementele care sunt egale cu oglinditul lui a_i și formează o pereche oglindită cu a_i. Complexitatea algoritmului este $\mathcal{O}(n^2)$.

Pentru 50 de puncte, se construiește un vector de frecvență pe baza numerelor citite. Se parcurge vectorul de frecvență și, pentru fiecare valoare x care există în acesta, se calculează oglinditul lui x în variabila y. Dacă y există în vectorul de frecvență, este strict mai mare decât x și formează o pereche oglindită cu x, atunci se va aduna la numărul de perechi produsul dintre frecvența lui x și frecvența lui y. Din moment ce n poate fi 100 000, rezultatul poate depăși valoarea maximă reprezentabilă pe tipul de date `int`. Complexitatea algoritmului este $\mathcal{O}(n + \mathsf{max})$, unde max este numărul maxim din șir.

Cerința 2: Pentru 27 de puncte, se citesc numerele din fișierul de intrare într-un vector. Se parcurg elementele vectorului, iar fiecare element aflat pe poziția i, cu $1 \leq i \leq n-1$, se compune cu toate elementele aflate pe pozițiile j, cu $i+1 \leq j \leq n$, verificându-se dacă cele două valori rezultate din compunere sunt numere palindrom. Pe parcurs, se reține valoarea maximă obținută. Complexitatea algoritmului este $\mathcal{O}(n^2)$.

Pentru 50 de puncte, se construiește un vector de frecvență pe baza numerelor citite. Se parcurge vectorul de frecvență și, pentru fiecare valoare x care există în acesta, se calculează oglinditul lui x în variabila y.

Dacă y există în vectorul de frecvență și este diferit de x, atunci se compun cele două numere, rezultând două numere palindrom.

Din numărul y se elimină pe rând câte o cifră. Se verifică dacă numărul rezultat există în vectorul de frecvență și, în caz afirmativ, se alipește la începutul numărului x inițial.

Din numărul x se elimină pe rând câte o cifră. Se verifică dacă numărul rezultat există în vectorul de frecvență și, în caz afirmativ, se alipește la începutul numărului y inițial.

Pe parcurs, dintre toate numerele palindrom obținute în acest proces, se reține maximul.

Complexitatea algoritmului este $\mathcal{O}(n + \mathsf{max})$, unde max este numărul maxim din șir.

Echipa

Problemele pentru această etapă au fost pregătite de:

- Prof. Pintea Adrian Doru, Inspectoratul Școlar Județean Cluj, Cluj-Napoca
- Prof. Pracsiu Dan, Liceul Teoretic "Emil Racoviță", Vaslui
- Prof. Boca Alina Gabriela, Colegiul Național de Informatică "Tudor Vianu", București
- Prof. Iordaiche Eugenia-Cristiana, Liceul Teoretic "Grigore Moisil", Timișoara
- Prof. Tîmplaru Roxana Gabriela, Colegiul "Ștefan Odobleja", Craiova
- Prof. Arișanu Ana Maria, Colegiul Național "Mircea cel Bătrân", Râmnicu Vâlcea
- Prof. Georgescu Camelia Alice, Colegiul Național "Mihai Viteazul", Ploiești
- Prof. Oprița Petru Simion, Liceul "Regina Maria", Dorohoi
- Stud. Buzatu Giulian, Facultatea de Matematică și Informatică, Universitatea București
- Stud. Oleniuc Iulian, Facultatea de Informatică Iași, Universitatea "Alexandru Ioan Cuza"