DESCRIEREA SOLUȚIILOR, OLIMPIADA JUDEȚEANĂ DE INFORMATICĂ, CLASA A IX-A

COMISIA ȘTIINȚIFICĂ

Cufere

Propusă de: Prof. Liliana Șchiopu - Colegiul Național Frații Buzești, Craiova

Se vor reține într-un tablou unidimensional *fr* numărul de obiecte întâlnite pentru fiecare tip de obiect (etichetă). Acesta se poate construi în timpul citirii datelor de intrare. Când citim numărul *xxyy* ce codifică o celulă dintr-un cufăr, incrementăm numărul de obiecte întâlnite cu eticheta *yy* cu *xx*, adică $fr[xx] = fr[xx] + yy$.

Tabloul va avea lungime maxim 100, întrucât etichetele iau valori în intervalul $[10, 99]$.

Cerința 1. Se afișează în ordinea crescătoare a etichetelor din intervalul $[10, 99]$ toate perechile $x, fr[x]$ pentru care $fr[x] > 0$.

Cerința 2. Se parcurge tabloul fr în ordinea crescătoare a etichetelor, de la 10, până la 99 inclusiv. Pentru fiecare etichetă x se verifică primalitatea acesteia: dacă x este număr prim, se setează dimensiunea maximă a unui grup $g = 16$, altfel se setează $g = 64$.

Cât timp încă există obiecte cu eticheta curentă ($fr[x] > 0$), se adaugă pe următoarea celulă liberă din cele n cufere un nou grup de obiecte cu eticheta x de dimensiune $\min(g, fr[x])$ și se decrementează $fr[x]$ cu g (sau se setează cu 0 în cazul în care $fr[x] < g$). Dacă mai rămân celule libere în cufere, se completează cu 0.

Verificarea primalității unui număr poate fi precalculată într-un tablou unidimensional $prim[i] = 1$ dacă i este prim și 0 altfel, și poate fi calculată în diverse modalități.

Complexitatea temporală este: $O(N)$.

Complexitatea spațială este: $O(1)$, întrucât dimensiunea tablourilor *fr* și *prim* este constantă.

Partitură

Propusă de: Stud. Mihaela Cismaru - Universitatea din Craiova, NetRom Software

Toate demonstrațiile lemelor folosite în demonstrarea corectitudinii soluției acestei probleme se pot găsi la finalul acestui articol în secțiunea Demonstrații problema partitură.

Cazul n = 4 și x = 1. Să presupunem că avem 4 note muzicale de înălțimi $a \geq b \geq c \geq d$ și durata $\frac{1}{2}$.

Lema 1. *Suma maximă se va obține prin împerecherea notelor astfel:* (a, b) *și* (c, d).

Cazul x = 1. Extindem soluția anterioară și considerăm că avem șirul de note muzicale cu înălțimi $a_1 \geq a_2 \geq a_3 \geq \cdots \geq a_n$, toate având durata $\frac{1}{2}$.

Lema 2. *Suma maximă se va obține prin împerecherea notelor astfel:*

$$(a_1, a_2), (a_3, a_4), \ldots, (a_{2i+1}, a_{2i+2}), \ldots, (a_{n-1}, a_n)$$

Acest caz se poate rezolva prin sortarea șirului de note muzicale descrescător în funcție de înălțime și însumând pătratul sumei perechilor de două câte două elemente consecutive.

Complexitate temporală: $O(n \log(n))$.

Complexitate spațială: $O(n)$.

Caz general.

Lema 3. *Se împerechează mai întâi toate notele muzicale cu durată* $\frac{1}{2^{x_{\max}}}$ *conform Lema 2, apoi se împerechează toate notele cu durată* $\frac{1}{2^{x_{\max-1}}}$ *ș.a.m.d. până se consumă toate notele.*

Acest caz se poate implementa prin sortarea notelor muzicale crescător după durata $\frac{1}{2^x}$, iar la egalitate după înălțimea y. Se împarte acest șir în $x_{\max}$ șiruri în funcție de durată și se împerechează toate notele de pe fiecare strat în ordine descrescătoare a duratelor (de la $\frac{1}{2^{x_{\max}}}$ la $\frac{1}{2}$).

Pentru a implementa eficient, este necesară o singură sortare inițială, ulterior adăugarea notelor de durată $\frac{1}{2^{x-1}}$ obținute prin împerecherea celor de durată $\frac{1}{2^x}$ la cele inițiale de durată $\frac{1}{2^{x-1}}$ putând fi făcută prin interclasare.

Complexitate temporală: $O(n\log(n))$.

Complexitate spațială: $O(n)$.

Fibosnek

Propusă de: Gheorghe-Eugen Nodea - Dir. Centrul Județean de Excelență Gorj

Observații.

(1) Având în vedere restricțiile problemei vom memora într-un tablou unidimensional fib doar primii $nr_F = 46$ termeni ai șirului Fibonacci, ultimul termen fiind $fib[46] = 1836311903$.

(2) Nu este necesară reținerea valorilor matricei inițiale, matricea de `int`-uri depășind memoria maximă alocată (6MB). Se vor reține indicii din șirul Fibonacci corespunzători valorilor citite. În cazul valorilor care nu aparțin șirului Fibonacci se poate reține indicele celui mai apropiat număr din șir, cu semn negativ. Așadar 8 va fi reținut ca 5, 13 ca 6, 11 ca -6, iar 4 ca -3.

Astfel, se poate reduce semnificativ memoria, deoarece poate fi folosită o matrice de `char` întrucât valorile din noua matrice sunt intre -46 și 46.

(3) Matricea inițială se poate *liniariza*, adică se poate memora într-un tablou unidimensional prin parcurgerea ***snek***. Elementul de pe celula (i, j) din matrice va apărea în tablou pe poziția $(j-1)\cdot n+i$, pentru oricare $1 \le i \le n, 1 \le j \le m$.

Astfel, în urma acestor transformări asupra problemei, se reduce la determinarea lungimi maxime a cel mult trei secvențe consecutive, compacte, ce alternează *fibosnek*, *non-fibosnek*, *fibosnek*.

Cazul c = 1 și n, m ≤ 1 000. Se vor precalcula primii 46 de termeni ai șirului Fibonacci în tabloul *fib*. Pentru fiecare valoare citită din matricea inițială se va verifica dacă aceasta se află în vectorul fib utilizând căutare binară și se vor număra câte numere Fibonacci au fost întâlnite.

Complexitate temporală: $O(n \cdot m \cdot \log(nr_F) + nr_F)$.

Complexitate spațială: $O(nr_F)$.

Cazul c = 2 și n, m ≤ 100. Se parcurge matricea pe coloane, conform parcurgerii *snek* și pentru fiecare element al matricei se determină dacă este sau nu număr Fibonacci parcurgând secvențial tabloul *fib*.

Pentru fiecare triplet de secvențe alternante (*fibosnek*, *non-fibosnek*, *fibosnek*) se va calcula răspunsul astfel:

- S_1 — suma primei secvențe din triplet *fibosnek* ce are n_1 termeni;
- S_2 — suma secvenței din mijloc *non-fibosnek* ce are n_2 termeni;
- S_3 — suma ultimei secvențe *fibosnek* ce are n_3 termeni.

Se pot întâlni următoarele cazuri particulare:

(1) $n_1 = 0$ – secvența *fibosnek* curentă este prima din parcurgere
(2) $n_1 = 0$, $n_3 = 0$ – nu există nicio secvență *fibosnek* în parcurgere
(3) $n_3 = 0$ – parcurgerea se termină într-o secvență *non-fibosnek*

Se reține suma $S_1 + S_2 + S_3$ a primului triplet întâlnit de lungime maximă $n1 + n2 + n3$.
Complexitate temporală: $O(n \cdot m \cdot nr_F + nr_F)$.
Complexitate spațială: $O(n \cdot m)$.

Cazul c = 2 și n, m ≤ 1 000. Pentru acest caz este necesară îmbunătățirea algoritmului anterior prezentat. Putem verifica dacă un număr este sau nu Fibonacci căutând binar în loc de secvențial acel număr în tabloul *fib*, astfel, reducând complexitatea pentru verificarea fiecărui număr de la $O(nr_F)$ la $O(\log(nr_F) + nr_F)$.
Complexitate temporală: $O(n \cdot m \cdot \ log(nr_F))$.

Soluția finală. Pentru a obține punctajul maxim, este necesară implementarea artificiului de memorare a indiciilor și nu a termenilor *Fibonacci* propriu-ziși pentru a micșora memoria utilizată.

Echipa

Problemele pentru această etapă au fost pregătite de:

- Conf. univ. Doru Popescu Anastasiu - Universitatea din Pitesti, Departamentul de Matematica-Informatica
- Lect. univ. Csaba György Pătcaş - Universitatea Babes-Bolyai, Cluj-Napoca
- Prof. Gheorghe-Eugen Nodea - Centrul Județean de Excelență Gorj
- Prof. Liliana Șchiopu - Colegiul Național Frații Buzești, Craiova
- Prof. Petru Simion Opriță - Liceul "Regina Maria", Dorohoi
- Stud. Mihaela Cismaru - Universitatea din Craiova, NetRom Software
- Stud. Theodor-Gabriel Tulbă-Lecu - Universitatea Politehnica din București
- Stud. Ioan-Cristian Pop - Universitatea Politehnica din București
- Stud. Vlad-Alexandru Gavrila-Ionescu - University of Cambridge
- Stud. Bogdan Iordache - Universitatea din Bucuresti, Facultatea de Matematica si Informatica
- Stud. Denis Andrei Banu - Facultatea de Informatica, Universitatea "Alexandru Ioan Cuza", Iasi
- Stud. Mihnea-Vicențiu Bucă - Universitatea din Bucuresti, Facultatea de Matematica si Informatica

Appendix A. Demonstrații problema partitură

Lema 1. Vrem să demonstrăm ca pentru patru numere $a \geq b \geq c \geq d$, suma pătratelor sumei a două câte două numere este maximizată de soluția: $(a+b)^2 + (c+d)^2$.

Demonstrație: Vom începe prin a calcula cele trei sume posibile:

(1) $(a+b)^2 + (c+d)^2 = a^2 + b^2 + c^2 + d^2 + 2ab + 2cd$
(2) $(a+c)^2 + (b+d)^2 = a^2 + b^2 + c^2 + d^2 + 2ac + 2bd$
(3) $(a+d)^2 + (b+c)^2 = a^2 + b^2 + c^2 + d^2 + 2ad + 2bc$

Vom arăta că diferența (1) - (2) este pozitivă:

$$a^2 + b^2 + c^2 + d^2 + 2ab + 2cd - a^2 - b^2 - c^2 - d^2 - 2ac - 2bd$$
$$2ab + 2cd - 2ac - 2bd$$
$$ab + cd - ac - bd$$
$$a(b-c) - d(b-c)$$
$$(b-c) \cdot (a-d)$$

Cum $b \geq c$ și $a \geq d$, rezultă ca diferența (1) - (2) este pozitivă. Similar se demonstrează că diferența (1) - (3) este și ea pozitivă.

Astfel, dintre toate cele trei combinații suma $(a+b)^2 + (c+d)^2$ este cea mai mare. □

Definiție 1. Numim *inversiune*, oricare 2 perechi de numere $(x,z),(y,t)$ sau $(x,t),(y,z)$, unde $x \geq y \geq z \geq t$.

Lema 2. Vrem să demonstrăm ca pentru un șir de numere $a_1 \geq a_2 \geq \dots a_n$, suma pătratelor sumei a două câte două numere este maximizată de soluția $(a_1,a_2),(a_3,a_4),\dots,(a_{2i+1},a_{2i+2}),\dots,(a_{n-1},a_n)$.

Demonstrație:

Presupunem că soluția ce maximizează suma pătratelor sumei de două câte două numere conține cel puțin o inversiune. Conform Lema 1, suma cu care contribuie această inversiune la suma totală nu este optimă, ci poate fi îmbunătățită prin eliminarea inversiunii.

Astfel, soluția nu este optimă, deci am ajuns la o contradicție – o soluție nu poate conține nicio inversiune. Singura configurație ce nu conține nicio inversiune este:

$$(a_1,a_2),(a_3,a_4),\dots,(a_{2i+1},a_{2i+2}),\dots,(a_{n-1},a_n)$$

□

Lema 3. Vrem să demonstrăm că soluția optimă se obține prin rezolvarea optimă a fiecărui grup de note de durate egale, începând cu cele cu durate cât mai mari.

Demonstrație:

Vom considera o soluție optimă în care știm grupele de note de durată 1.

Considerăm toate cele k grupe din această soluție ce conțin cel puțin o notă de durată minimă ordonate după suma $S_1 \geq S_2 \geq S_3 \geq \dots \geq S_k$.

Fie o notă de durată minimă de înălțime x din grupa de sumă $S = a + x$ și o notă de durată minimă de înălțime y din altă grupă $S' = b + y$. Dacă $x \leq y$, atunci soluția nu este optimă deoarece dacă am face o interschimbare între x și y atunci suma se va îmbunătății cu:

$$\begin{aligned}
&(a+x)^2 + (b+y)^2 - (a+y)^2 - (b+x)^2 \\
=&a^2 + 2ax + x^2 + b^2 + 2by + y^2 - a^2 - 2ay - y^2 - b^2 - 2bx - x^2 \\
=&2ax + 2by - 2ay - 2bx \\
=&2(ax + by - ay - bx) \\
=&2(x-y)(b-a)
\end{aligned}$$

Care este un număr pozitiv.

Astfel, grupa cu cea mai mare sumă ce conține note de durată minimă va conține sufixul de note de durată minimă ordonate după înălțimea maximă. Deci, toate notele de durată minimă vor fi selectate în perechi de două câte două în ordine descrescătoare.

În continuare, putem crește durata minimă a unei note considerând aceste note împerecheate ca fiind o singură notă de lungime dublă și deci reducem problema de la o durată minimă $\frac{1}{2^x}$ la $\frac{1}{2^{x-1}}$. Acest proces se aplică recursiv până când obținem doar grupe de durată 1.