Grăunte

Autor: stud. Lauran David, Universitatea Babeș-Bolyai, Cluj-Napoca
Contribuitor: stud. Bogdan-Alexandru Miron, Universitatea Babeș-Bolyai, Cluj-Napoca

Soluție pentru 20 de puncte

Complexitate: $O(Q_1 \cdot (V \cdot log(V) + N^2) + Q_2 \cdot N^2)$, *unde* V=*max(v)*

Remarcăm că, pentru orice $x < y$, x și y sunt coprime dacă și numai dacă $y - x$ și y sunt coprime. Ținem o matrice de N linii și N coloane care reprezeintă câmpul fermierului. Pentru fiecare dintre cele Q_1 actualizări, calculăm w=suma numerelor coprime cu v astfel: pentru fiecare număr x mai mic decât $v/2$, verificăm, folosind algoritmul lui Euclid cu împărțiri, dacă acest număr este coprim cu v. În caz afirmativ, adăugăm la w pe x și, dacă $2 \cdot x \neq v$, pe $v - x$. După ce o calculăm, adăugăm valoarea w în mod direct fiecărui element din porțiunea dată.

Calculăm suma fiecărei linii. Pe urmă, pentru cele Q_2 interogări, încercăm să alegem ca capete toate perechile de linii. O alegem pe cea mai lungă dintre cele care au suma totală cel mult k.

Soluție pentru 40 de puncte

Complexitate: $O(V^2 \cdot log(V) + Q_1 + Q_2 \cdot N^2)$

Prima diferenta față de soluția anterioară este folosirea tehnicii *difference arrays (șmenul lui Mars)*. Mai exact, pentru fiecare plantare, în loc să adăugăm w fiecărui element, adăugăm w pe pozițiile $(a,\ b)$ si $(c+1,\ d+1)$ si scadem w de pe pozitiile $(a,\ d+1)$ si $(c+1,\ b)$. După ce efectuăm aceste modificări pentru toate plantările, calculăm sumele pe prefixe pentru a obține valorile reale ale matricei.

A doua diferență constă în precalcularea lui w pentru fiecare v de la 1 la $V = 5000$.

Soluție pentru 70 de puncte

Complexitate: $O(V \cdot log(V) + Q_1 + N^2 + Q_2 \cdot N)$

Se pot obține 70 de puncte cu o soluție asemănătoare soluției de 100 de puncte prezentate mai jos. Variantei de 70 de puncte îi lipsește doar observația că matricea poate fi tratată ca un șir ale cărui elemente reprezintă suma elementelor de pe o linie.

Soluție alternativă pentru 70 de puncte

Complexitate: $O(V \cdot log(V) + Q_1 \cdot N + Q_2 \cdot N)$

Se preproceseaza functia lui Euler $\varphi(v)$ pentru toate valorile posibile ale lui v pana la 10^6. Functia calculeaza numarul de valori coprime cu v. Suma valorilor coprime se determina prin:

$$S = \frac{v \cdot \varphi(v)}{2} \mod 1789$$

Acest rezultat provine din simetria numerelor coprime cu v. Concret, se poate observa că numerele coprime cu v sunt situate simetric în jurul valorii $v/2$.

Ținem suma pentru fiecare linie. Pentru fiecare operație de tip 1, se adaugă direct valoarea corespunzătoare fiecărei linii din intervalul $[a, c]$. Această metodă introduce o complexitate $O(N)$ pentru fiecare astfel de operație.

În cazul interogărilor de tip 2, se utilizează *two pointers* pentru a găsi cea mai lungă secvență de linii consecutive a căror sumă este cel mult k. Se menține o sumă cumulată și se elimină liniile din partea stângă atunci când depășim k, astfel încât fiecare linie este procesată de cel mult două ori, ceea ce conduce la o complexitate $O(N)$ per interogare.

Această implementare obține 70 de puncte deoarece adăugarea sumelor pe linii este realizată într-un mod ineficient. Fiecare actualizare este efectuată individual pentru fiecare linie în intervalul specificat, ceea ce conduce la o complexitate $O(n)$ per actualizare. Astfel, în cel mai rău caz, cu Q_1 interogări și valori mari pentru n, complexitatea totală poate deveni $O(Q_1 \cdot N + Q_2 \cdot N)$, ceea ce este prea mare pentru a trece toate testele de timp.

O îmbunătățire esențială pentru a obține punctaj maxim este utilizarea unui difference array, astfel încât fiecare actualizare să fie realizată în $O(1)$, iar sumele finale pentru linii să fie calculate într-o singură trecere în $O(N)$.

Soluție pentru 100 de puncte

Complexitate: $O(V \cdot log(V) + Q_1 + Q_2 \cdot N)$

Pentru fiecare dintre cele Q_1 interogări se actualizează un interval de linii al matricei folosind valoarea w calculată din numărul v. Această valoare este determinată folosind funcția totient (indicatorul lui Euler) și se adaugă la fiecare linie din interval. După procesarea tuturor actualizărilor, se primesc interogări ce cer identificarea celei mai lungi secvențe de linii a căror sumă nu depășește o valoare dată k.

Se precomputează valorile funcției totient $\phi(n)$ pentru toate numerele $1 \leq n \leq \texttt{V}$. Inițial, se setează pentru fiecare i valoarea $\phi(i) = i$, iar apoi se aplică:

Pentru fiecare i de la 2 la V, dacă $\phi(i) = i$ (adică i este prim), atunci pentru toți multiplii j ai lui i se actualizează: $\phi(j) = \phi(j) \cdot \frac{i-1}{i}$.

Se calculează suma numerelor coprime cu v folosind formula:

$$w = \frac{v \cdot \phi(v)}{2} \mod \texttt{1789}$$

Folosind difference arrays, se adaugă valoarea `plusLinie`, definită mai jos, la poziția a și se scade aceeași valoare de la poziția $c+1$. După procesarea tuturor actualizărilor, se calculează suma prefix, astfel încât fiecare linie să aibă valoarea actualizată corespunzătoare.

Valoarea adăugată fiecărei linii este calculată ca:

$$\texttt{plusLinie} = (\texttt{d} - \texttt{b} + 1) \times w,$$

unde $\texttt{d}-\texttt{b}+1$ reprezintă numărul de coloane afectate și w este suma numerelor coprime calculată anterior.

Se utilizează two pointers pentru a găsi secvența de linii cu suma totală mai mică sau egală cu k.