Rectangles

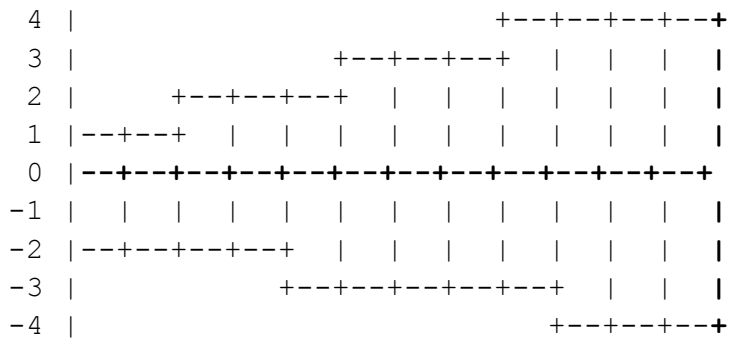
2022, Baraj 3, Seniori

Problema are foarte multe subtaskuri, iar fiecare subtask are o soluție diferită. Printre acestea se numără, pe lângă soluția oficială: mai multe soluții de tip brute-force în care se fixează diferite combinații de borduri, principiul includerii și excluderii în mai multe complexități fie luând toate subseturile posibile de puncte blocate, fie polinomial, fixând doar bounding boxul subseturilor respective, divide în care se taie planul alternant pe orizontală și verticală.

Soluția oficială este una de tip divide et impera: tăiem planul in două jumătăți egale după linia y = m, numărăm câte dreptunghiuri goale există care să treacă prin linia respectivă, după care rezolvăm recursiv pe cele două jumătăți.

Dacă ne fixăm bordura dreaptă a unui dreptunghi și după iterăm cu bordura stângă a dreptunghiului, noi vom adăuga la răspuns produsul dintre numărul de borduri superioare și inferioare valide. Astfel, noi trebuie să menținem următoarea structură de date:

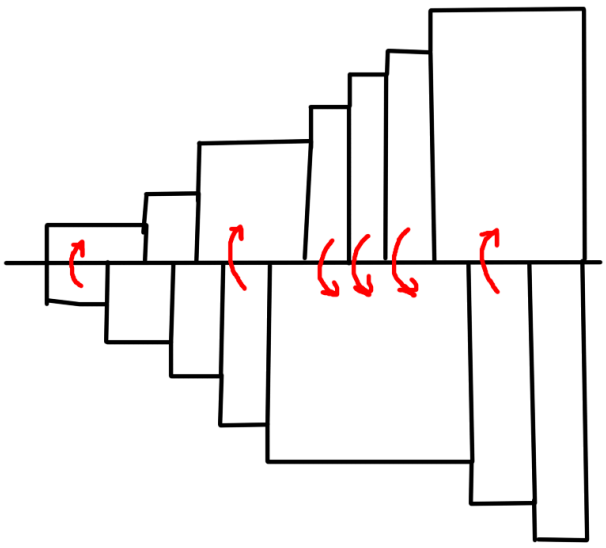
avem două skyline-uri, cel de deasupra liniei de tăietură și cel de sub linia de tăietură, și cumva trebuie menținut la fiecare moment de timp suma pe produse pe fiecare fâșie de latură 1 din partea de sus și partea de jos. Cele două skyline-uri reprezintă regiunea "vizibilă" de pe fiecare parte.



În particular, pentru momentul în care bordura dreaptă a dreptunghiului se află pe desenul de mai sus pe linia verticală cu **bold**, trebuie menținută suma 4 * 4 + 4 * 4 + 4 * 4 + 4 * 3 + 3 * 3 + 3 * 3 + 3 * 3 + 2 * 3 + 2 * 2 + 2 * 2 + 1 * 2 + 1 * 2. Așadar, pentru fiecare bordură dreaptă fixată, se adună suma aceea la răspuns.

Dacă iterăm cu bordura dreaptă de la stânga la dreapta, trebuie să vedem cum se schimbă suma menținută. Când adăugăm o fâșie nouă, cele din stânga se vor uni într-un dreptunghi mai mare, așa că putem menține două stive cu dreptunghiurile vizibile, iar pe lângă să folosim un arbore de intervale ca să menținem suma aceea de produse pentru complexitate O(Nlog^2) .

Pentru a optimiza unul din logaritmi, renunțăm la arborele de intervale și menținem o singură stivă cu dreptunghiuri și de deasupra și de sub linia de tăietură.



Ideea principală pe care se bazează soluția este că odată ce un dreptunghi este "mâncat" de un dreptunghi de pe partea opusă, nu ne mai pasă de el. Tot ce ne pasă ar fi doar pentru dreptunghiul mai mare care e suma ariilor dreptunghiurilor mâncate de pe cealaltă parte. Așadar, pentru fiecare dreptunghi care se află în stivă, deci nu este "mâncat" de alt dreptunghi, trebuie să menținem suma ariilor dreptunghiurilor "mâncate", iar contribuția la sumă generată de două dreptunghiuri care se suprapun parțial se poate face cu brut (sunt doar două astfel de dreptunghiuri). Pe desenul de mai sus, dreptunghiurile din care pleacă săgeți roșii nu mai sunt ținute în memorie, iar dreptunghiul în care se duc săgețile trebuie să țină minte aria respectivă. Cel mai din dreapta dreptunghi de pe partea de jos poate sau nu să fie mâncat de dreptunghiul de sus în funcție de implementare.

Un dreptunghi este "mâncat" dacă proiecția lui pe axa de tăietură este complet inclusă în proiecția altui dreptunghi.

Astfel, dacă normalizăm toate coordonatele și menținem skyline-urile combinate de mai sus și sumele de care avem nevoie, complexitatea algoritmului este O(Nlog).