Clădiri - soluţie

Fie `A[0..n-1][0..m-1]` matricea citită de la intrare. Este trivial de găsit in timp O(`N^2`) o matrice `B[0..n-1][0..m-1]`, astfel încât `B[i][j] = 1` doar dacă putem plasa un dreptunghi de `w` coloane pe `h` linii începând cu linia `i` şi coloana `j`.

Folosim algoritmul de determinare a dreptunghiului de dimensiune maximă, determinăm în O(`N^2`) pentru fiecare linie şi pentru fiecare coloană care este cel mai mare dreptunghi care se găseşte înainte/respectiv după linia/coloana respectivă.

Acum ne plimbăm în matrice prin toate poziţiile în care putem aşeza dreptunghiul (O(`N^2`)) de `w*h` şi putem determina în O(`1`) care este cel mai mare dreptunghi rămas liber astfel:

- presupunem dreptunghiul de `w*h` plasat pe liniile `i`, `i+1`, ..., `i+h-1` şi pe coloanele `j`, `j+1`, ..., `j+w-1`.
- luăm dimensiunea maximă de deasupra liniei `i`, de sub linia `i+h-1`, din stânga coloanei `j`, din dreapta coloanei `j+w-1` şi calculăm maximul dintre aceastea.

Alternativ (Ionuţ Fechete), putem plasa dreptunghiul de `w*h` în patru poziţii: cât mai în stânga, cât mai în dreapta, cât mai sus, cât mai jos, şi putem rula apoi algoritmul pentru determinarea dreptunghiului maxim pentru fiecare din cele 4 plasări posibile. Dintre acestea, alegem maximul. Se observă că acest că acest algoritm este corect (din algoritmul anterior).

Mesaj - soluţie

Problema se poate rezolva utilizand programarea dinamică în O($N*2^K*K$) unde `k` este numărul maxim de mesageri care trec printr-un nod.

Definim o stare `S` într-un nod `I` un număr în baza doi de lungime `K` (numărul de mesageri), un bit de `1` în `S` pe poziţia `x` sugerează ca al `x`-lea mesager care trece prin `I` este angajat, iar un bit de `0` pe poziţia `x` sugerează că al `x`-lea mesager care trece prin `I` nu este angajat.

Calculăm `TA(I,S)` ca fiind costul minim pentru a acoperi toate nodurile din subarborele cu rădăcina în `I` cu starea `S`.

Folosim următoarea recurenţă :

$$TA(I,S) = \sum_J \underset{T}{MIN}(TA(J,T))$$

Unde `J` este un fiu al lui `I`, `S` este starea în `I`, iar `T` este orice stare în `J` care este compatibila cu `S` (dacă `S` spune că mesagerul `x` este angajat atunci `T` trebuie sa spună acelaşi lucru).

Pentru a calcula în O(`1`) $\underset{T}{MIN}(TA(J,T))$ pentru orice `T` compatibila cu `S` reţinem un alt tablou `TB(J,P)` unde `P` este o stare a mesagerilor care trec prin `J` şi prin tatăl lui `J`, care poate fi calculat în acelaşi timp cu tabloul `TA`.

$TB(J,P) = \underset{S}{MIN}(TA(J,S))$ unde `S` este o stare a mesagerilor care trec prin `J` (nu neapărat şi prin tatăl lui) compatibilă cu `P`. Recurenţa de mai sus devine:

$TA(I,S) = \sum_J TB(J,P)$ `P` este unică şi poate fi calculată în O(`K`) în funcţie de `S`.

Zuzu - soluţie

Pentru fiecare axă, vom considera toate coordonatele punctelor de pe cele `n` paralelipipede dreptunghice. Vom elimina din numerele de pe axa Ox pe cele care se repetă, de pe axa Oy pe cele care se repetă, de pe axa Oz pe cele care se repetă. In felul acesta obţinem trei vectori: `x`, cu `mx` elemente, `y` cu `my` elemente şi `z` cu `mz` elemente.

Ducând plane paralele cu cele trei plane ale sistemului de coordonate Oxy, Oxz, Oyz, prin punctele date de cei trei vectori `x`, `y`, `z` se obţine spaţiul partajat în paralelipipede dreptunghice care conţin cei `n` nori. Folosind acest lucru putem construi un tablou `a` tridimensional cu elemente `0` şi `1`.
`a[i,j,k]=1` dacă paralelipipedul corespunzător (dat prin intermediul segmentelor `x[i+1]-x[i]`, `y[i+1]-y[i]`, `z[i+1]-z[i]`) este inclus măcar într-un nor, respectiv `0` contrar.

Determinăm volumul formaţiunilor noroase folosind un algoritm de tip fill.

Operaţiile de mai sus le efectuăm după fiecare unitate de timp, actualizând coordonatele conform direcţiilor de deplasare.

Observaţie:
Dacă se considerau coordonate întregi problema era mult mai directă. Dacă se înmulţesc coordonatele cu `100000`, se pot utiliza numere întregi însă nu există memorie suficientă pentru a memora tablouri tridimensionale necesare la aplicarea unui algoritm fill.