Solutie Nakhla

Problema se reduce la jocul NIM jucat pe N gramezi, fiecare gramada fiind formata dintr-un numar de monede egal cu spatiul liber dintre jetonul alb si jetonul negru de pe fiecare rand. Dupa cum se stie, exista strategie de castig pentru o configuratie daca si numai daca suma xor a gramezilor este diferita de 0. Fie X suma xor a celor N gramezi la un moment dat. Daca X este diferit de 0, atunci trebuie sa gasim o gramada G[i] astfel incat X xor G[i] < G[i] pentru a efectua o mutare care aduce suma xor la 0. Pentru a se respecta conditia mentionata trebuie ca G[i] sa aiba setat bitul cel mai semnificativ de 1 din numarul X.

Datorita numarului mare de gramezi aceasta gramda trebuie gasita intr-o complexitate O(lg N). O solutie cu O(N) memorie este urmatoarea:

- se mentine un arbore de intervale care retine in fiecare nod xor-ul pe acel interval
- pentru a gasi o gramada care are setat cel mai semnificativ bit din X se face o parcurgere in arbore, la fiecare pas mergand pe subarborele care are setat acel bit (nu-l pot avea setat ambii subarborii)
- dupa ce se face o mutare se actualizeaza arborele de la frunza care reprezinta gramada respectiva pana la radacina

O alta solutie cu O(lg N) pe mutare, dar cu O(N lg N) memorie este urmatoarea:

- se pastreaza cate o lista pentru fiecare bit de la 0 la 30 cu indicii gramezilor care contin acel bit
- pentru a efectua o mutare trebuie sa ne uitam in lista corespunzatoare bitului cel mai semnificativ al lui X
- dupa ce se face o mutare se sterge elementul din listele in care era inainte si se insereaza in alte liste, conform noii valori
- pentru a implementa usor acest algoritm se poate folosi o tehnica de “lazy deletion”: dupa o mutare nu se sterg elementele din lista; atunci cand vrem sa efectuam o mutare scoatem dintr-o lista pana cand gasim un element care chiar contine acel bit

Soluţie - sir

Se observa faptul ca odata ce am gasit doua perechi (T_i,T_{i+1}) si (T_j,T_{j+1}) egale, cu $i\neq j$ se demonstreaza prin inductie ca $T_{i+k}=T_{j+k}$ pentru orice k. Asadar $T_{i+k}=T_{i+k\%(j-i)}$.

Deoarece exista maxim M^2 perechi posibile lungimea unui ciclu (diferenta j-i) va fi maxim M^2. Determinarea acestui ciclu se face folosind algoritmul Floyd folosind doar O(1) memorie. Incepem cu doi pointeri i si j, pointerul i il deplasam cu un pas, pointerul j il deplasam cu doi pasi. In momentul in care elementele vor fi egale se detecteaza un ciclu(prin elemente ne referim la perechi (T_k,T_{k+1})). Pentru determinarea lungimii ciclului se mai face o parcurgere a lui.

Odata detectat ciclul, rezultatul va fi $T_{i+(n-i)\%l}$

Complexitatea algoritmului este $O(l) = O(M^2)$.

Centru – descrierea solutiei

Problema se rezolva in doua etape. In prima etapa vom calcula pentru fiecare nod al caroiajului distanta pana la cel mai apropiat centru de prim ajutor. In a doua etapa vom determina solutia ceruta bazandu-ne pe informatiile de la primul pas.

Calculul distantelor pana la cel mai apropiat punct de prim ajutor se realizeaza in $O(N^2)$. In acest sens, vom explicita formula distantei Manhattan intre un punct (x, y) si un centru de prim ajutor (x_c, y_c). Din formula $|x-x_c|+|y-y_c|$ se obtin patru cazuri :

1. $x \geq x_c$, $y \geq y_c \rightarrow$ distanta = $x+y-x_c-y_c$

2. $x \geq x_c$, $y < y_c \rightarrow$ distanta = $x-y-x_c-y_c$

3. $x < x_c$, $y \geq y_c \rightarrow$ distanta = $-x+y+x_c-y_c$

4. $x < x_c$, $y < y_c \rightarrow$ distanta = $-x-y+x_c+y_c$

Astfel, pentru un nod oarecare caroiajul se va imparti in patru cadrane, pentru fiecare dintre ele fiind necesar sa se minimizeze una dintre expresiile $-x_c-y_c$, $-x_c-y_c$, x_c-y_c, x_c+y_c. Acest lucru se rezolva in $O(N^2)$ folosind patru parcurgeri. Distantele minime calculate se introduc intr-o matrice M.

Sa consideram ca dupa introducerea noului punct de prim ajutor distanta maxima parcursa in cazul cel mai defavorabil este k. Asta inseamna ca toate nodurile (x, y) pentru care M[x][y] > k sunt *acoperite* de noul centru, altfel spus distanta pana la acesta este mai mica decat k. De aici se contureaza prima idee de rezolvare folosind cautarea binara: se fixeaza un k si se verifica daca toate nodurile pentru care M[x][y] > k pot fi acoperite cu un singur centru. De remarcat ca multimea nodurilor acoperite de un centru (x, y) este inclusa intr-un romb centrat in (x, y). Ramane astfel suficient sa verificam daca rombul minim care contine toate nodurile cu M[x][y] > k pot fi incluse intr-un romb care sa satisfaca o restrictie de dimensiune (din centrul sau sa se poata ajunge in orice al nod inclus in acesta parcurgand o distanta de cel mult k). Acesta se determina in $O(N^2)$ ducand la o complexitate finala de $O(N^2 \lg N)$.

Pentru reducerea complexitatii la $O(N^2)$ se observa ca odata cu decrementarea lui k apar noi puncte care trebuie acoperite, in timp ce dimensiunea rombului care trebuie sa le acopere scade. Ramane astfel sa se considere nodurile caroiajului in ordinea descrescatoare a distantelor calculate in M si la fiecare pas sa se calculeze in O(1) rombul minim care contine toate punctele considerate pana in momentul respectiv. Aceasta se realizeaza mentinand valorile min(x+y), max(x+y), min(x-y), max(x-y) pentru coordonatele nodurilor deja considerate si verificarea corespunzatoare in O(1).

Pentru mai multe detalii consultati sursa oficiala.