Soluţie pătrate

Pentru început determinăm un patrat de latura x având colţul din stânga-sus în origine care cuprinde toate pătrăţelele negre şi în care aria ocupată de pătrăţelele negre este mai mică decât S/k, unde S este aria pătratului de latura x (x^2).
Împărţim acest pătrat în patru pătrate egale disjuncte de latura x/2 (prin cele două linii mediane - ca la algoritmul quadro) şi în aceste 4 pătrate aria punctelor negre nu va depăşi 4S/k (deoarece aria celor 4 pătrate este de patru ori mai mică decît aria pătratului iniţial).
Pentru fiecare dintre aceste patru pătrate avem unul din următoarele 3 cazuri:

- el nu conţine nici un pătrăţel negru; în acest caz, pătratul respective nu mai prezintă nici un interes
- el conţine un număr de pătrăţele negre a căror arie însumată se încadrează între limitele impuse prin enunţ: în acest caz el va face parte din soluţie;
- aria pătrăţelelor negre este mai mică decât (aria lui)/k; în acest caz îl divizăm în patru pătrate disjuncte egale ş.a.m.d.

În cel mai defavorabil caz obţinem pătrate de latură egală cu 2 care conţin un singur pătrăţel negru. Acestea îndeplinesc condiţiile din enunţ (1/k< 1/4!).

Având în vedere limitele pentru coordonatele pătrăţelelor negre, aria pătratului iniţial nu va depăşi 2048, deci numărul maxim de divizări va fi $(2048/2)^2=2^{10}$.

Pentru a determina eficient numărul de pătrăţele negre dintr-un pătrat, am folosit o matrice care permite acest calculul în O(1) bazându-ne pe faptul că pătratele care rezultă în timpul divizării au coordonate fixe indiferent de dispunerea pătrăţelelor negre.

Actualizarea matricei se face la citirea coordonatelor pentru fiecare pătrăţel negru într-un număr de paşi mai mic sau cel mult egal cu 10.

Soluţie - treinr

Să notăm cu `try(a,b,c)` răspunsul pentru întrebarea
`ASK a b c`
Cu maxim `n*n` întrebări putem găsi un triplet (a,b,c) pentru care `try(a,b,c)=1`, apoi cu trei întrebari determinăm care e perechea de numere corectă şi încă maxim `n` întrebări ne dau şi ultimul număr.
Chiar se poate demonstra că numărul de întrebări necesar e O(n*n), dar cu o constanta subunitară. Algoritmul ales determina tripletul corect din maxim `n*n/2+n+3` întrebări:
Împărţim mulţimea {`1`, `2`, ..., `n`} în două submulţimi de cardinal `n/2` şi respectiv `n-n/2`: {`1`, `2`, ..., `n/2`} şi {`n/2+1`, ..., `n`} Astfel suntem siguri că cel puţin două dintre numerele tripletului se vor gasi în aceeaşi submulţime. Acum cu maxim `(n/2)*(n/2)` întrebări găsim o pereche "bună" şi apoi tripletul căutat.
Cum procedăm cu cele două submulţimi? Scăzând cel mai mic număr le aducem la forma {`0`, `1`, `2`, ..., `k-1`} Iar pentru a găsi două numere corecte din mulţimea {`0`, `1`, `2`, ..., `k-1`} putem alege tripletele (`a`, `b`, `c`) cu `a+b+c mod k = 0`, pentru oricare două dintre numerele `a`, `b`, `c` al treilea rezultă în mod unic, deci sunt `k*k` astfel de triplete.

Soluţie - turism

Problema cere să se determine un set minim de arce care adaugate in graf formeaza un graf eulerian. Vom construi doi vectori, `IN` si `OUT` cu noduri in care mai trebuie sa intre arce si respectiv noduri din care trebuie sa mai iasa arce. Un nod poate aparea de mai multe ori intr-un vector. Pentru a calcula acesti vectori procedam astfel pentru fiecare nod `i`:

- daca `grad_int[i] > grad_ext[i]` atunci se insereaza nodul `i` in vectorul `OUT` de `grad_int[i]-grad_ext[i]` ori
- daca `grad_int[i] < grad_ext[i]` atunci se insereaza nodul `i` in vectorul `IN` de `grad_ext[i]-grad_int[i]` ori

De asemenea, exista posiblitatea sa existe mai multe componente conexe in graf. In acest caz, poate exista o componenta conexa pentru care `grad_int[] = grad_ext[]` pentru orice nod din aceea componenta, deci noduri din componenta respectiva nu au fost adaugate in `IN` si `OUT`. Aceasta situatie constituie o problema deoarece graful la final nu va fi tare conex, chiar daca se respecta conditia `grad_int[] = grad_ext[]` pentru fiecare nod. Astfel, pentru fiecare astfel de componenta se insereaza cel mai mic nod (ca numar) o data in vectorul `IN` si odata in vectorul `OUT`. Dupa ce s-au calculat, vectorii `IN` si `OUT` se sorteaza si se introduc arce in ordinea (`OUT[i]`, `IN[i]`) cu `i` crescator. In cazul in care muchia (`OUT[i], IN[i]`) uneste doua noduri din aceeasi componenta conexa (din nou privind graful ca unul neorientat nu, orientat) iar acestea sunt ultimele doua noduri din aceea componenta care mai exista in vectorii `IN` si `OUT`, atunci inserarea acestei muchii ar duce din nou la obtinerea unui graf care nu este tare conex. Pentru a rezolva aceasta situatie se interschimba `IN[i]` cu `IN[i+1]` si se continua procesul.