Problema - xor

Adrian Panaete,

Colegiul Național “A. T. Laurian” Botoșani

Toate soluțiile se referă la rezolvarea unui singur query.

Solutia 1. Complexitate O(m^2 + q)

Se precalculează matricea până la linia m, coloana m. Apoi se răspunde la fiecare întrebare în timp constant.

Solutia 2. Complexitate O(q * m) .

Urmărim de câte ori apare fiecare număr de la 1 la j în rezultat. Să luăm de exemplu i=3 , j=5 .
Pentru 1 avem 0 apariții
0 1 0 0 0 0
0 1 1 0 0 0
0 1 2 1 0 0
0 1 3 3 1 **0**.
Pentru 2 avem o apariție deci 2 apare o data în rezultat
0 0 1 0 0 0
0 0 1 1 0 0
0 0 1 2 1 0
0 0 1 3 3 **1** .
Pentru 3 avem 3 apariții deci 3 apare de 3 ori în rezultat
0 0 0 1 0 0
0 0 0 1 1 0
0 0 0 1 2 1
0 0 0 1 3 **3** .
Pentru 4 avem 3 apariții deci 4 apare de 3 ori în rezultat
0 0 0 0 1 0
0 0 0 0 1 1
0 0 0 0 1 2
0 0 0 0 1 **3**
Pentru 5 avem o apariție deci 2 apare o data în rezultat
0 0 0 0 0 1
0 0 0 0 0 1
0 0 0 0 0 1
0 0 0 0 0 1 .
Folosind proprietatea $a\ xor\ a = 0$ ne interesează doar restul modulo 2 al numărului de apariții .
In cazul nostrum rezultatul este 2 xor 3 xor 4 xor 5 = 0
În general numărul de apariții a unei valori k pe linia i coloana j este C_i^{j-k} care se ia 0 dacă $j - k > i$.
Ne intereseaza doar paritatea acestei combinări deci este suficient să parcurgem valorile de la j la 1 si să actualizăm numărul de factori 2 care apar la numitor și la numărător pentru trecerea de la o valoare la valoarea imediat mai mică. Se obtine un algoritm de complexitate O(j) pentru fiecare întrebare.

Solutia de 100 p. Complexitate O(q*log m) sau O(q * $\log^2$ m)

Folosim ideea de la soluția de 60 de puncte dar facem observația suplimentară că rezultatul poate fi calculat la nivelul fiecărui bit (ceea ce adaugă un factor log m la fiecare întrebare) . Se obsevă că prezența sau nu în rezultatul final a fiecărui bit respectă o regulă foarte clară – mai précis numărul de apariții se dovedește a fi tot o combinare modulo 2. Calculul combinărilor modulo 2 poate fi realizat cu diverse complexități (patratic, liniar, logarithmic, constant). Cele mai rapide metode de calcul vor fi prezentate în continuare.

Detalii:

Fie matricea infinita $s[i,j] = C_{i+j}^{j}\ modulo\ 2$ care poate fi construită cu formula de recurență:

$$s[i,j] = s[i-1,j]^\wedge s[i,j-1]$$
$$s[i,0] = s[0,j] = 1$$

Matricea este foarte des întâlnită în probleme şi aplicații de informatică sub denumirea de "triunghiul Serpinski". Dacă vom construi parțial matricea vom obține:

1111111111111111…L0
1010101010101010…L1
1100110011001100…L2
1000100010001000
1111000011110000…L4
1010000010100000
1100000011000000
1000000010000000
1111111100000000…L8
1010101000000000
1100110000000000
1000100000000000
1111000000000000
1010000000000000
1100000000000000
1000000000000000
1111111111111111…L16

Proprietăți ale "triunghiului Serpinski"

1. $s[i][j] = C_{i+j}^{i}\ mod\ 2$
2. $s[i][j] = s[j][i] pentru\ i,j \geq 0$
3. $s[i][j] = s[i-1][j] + s[i][j-1]\ pentru\ i,j \geq 1$
4. $s[i][j] = s[i-2^p][j] pentru\ i \in [2^p, 2^{p+1})$ și $j \in [0, 2^p)$
5. $s[i][j] = 0\ pentru\ i\&j \neq 0$
6. $s[i][j] = 1\ pentru\ i\&j = 0$
7. $s[i][j] = 1\ daca\ există\ p\ astfel$ încât $i + j = 2^p - 1$
8. $s[2^p + j][2^p] = s[2^p][2^p + j] = 0\ pentru\ \ j \in [0, 2^p)$
9. $s[i][j] = 0\ pentru\ i,j \in [2^p, 2^{p+1})$

Toate aceste proprietăți pot fi deduse fie din proprietățile generale ale combinărilor fie direct analizând puterea la care apare factorul prim 2 în combinări.
Proptietatea 3. ne permite să calculăm valorile matricei in timp pătratic.
Daca folosim formula combinărilor un element oarecare se poate calcula liniar folosind inversul modular.
Proprietațile 3. și 8. ne permit să calculăm un element in timp logaritmic.
Propietățile 4. și 5. ne permit să calculăm un element în timp constant.

Să notăm cu $bp[\,][\,]$ tabloul definit astfel.
$bp[i][j] = 1$ dacă soluția problemei pentru poziția (i,j) conține bitul p (mai precis conține valoarea 2^pîn descompunerea unică în sumă de puteri ale lui 2)
$bp[i][j] = 0$ dacă soluția problemei pentru poziția (i,j) nu conține bitul p.
În particular $bp[0][j]$ este 1/0 după cum j conține / nu conține bitul p

Din modul cum e definit tabloul din enunț se deduce că:

$$bp[i][j] = bp[i-1][j]^\wedge bp[i][j-1]$$

adică respectă aceeași relație de recurență ca matricea s.
Este foarte ușor de observat că un număr j conține bitul p dacă și numai dacă $j \pm 2^p$ nu conține bitul p.
De aceea avem $bp[0][j] = 1 \Leftrightarrow j$ conține bitul $p \Leftrightarrow j \pm 2^p$ nu conține bitul $p \Leftrightarrow 2^p\ \&(j \pm 2^p)$=0$\Leftrightarrow s[2^p][j \pm 2^p] = 1$
Analog $bp[0][j] = 0 \Leftrightarrow s[2^p][j \pm 2^p] = 0$

Concluzia este că $bp[0][j] = s[2^p][j \pm 2^p] = 0$.
Pentru $j < 2^p$ avem evident $bp[i][j] = 0$.
Pentru $j \geq 2^p$ folosind recurențele se va deduce în final $bp[i][j] = s[i + 2^p][j - 2^p]$

Acum avem o metodă prin care putem calcula bit cu bit soluția pentru orice pozitie folosind „triunghiul Serpinski”.

Soluția va fi <u>sau pe biti</u> din valorile $2^p * s[i + 2^p][j - 2^p]$ pentru toate valorile lui p cu $2^p \leq j$