Formularea temei

Conway's Game of Life este un *zero-player game* bidimensional, inventat de matematicianul John Horton Conway in anul 1970. Scopul acestui joc este de a observa evolutia unui sistem de celule, pornind de la o configuratie initiala, introducand reguli referitoare la moartea, respectiv crearea unei noi celule in sistem. Acest sistem evolutiv este *Turing-complete*.

Starea unui sistem este descrisa de starea cumulata a celulelor componente, iar pentru acestea avem urmatoarele reguli:

1. **Subpopulare.** Fiecare celula (care este in viata in generatia curenta) cu mai putin de doi vecini in viata, moare in generatia urmatoare.
2. **Continuitate celule vii.** Fiecare celula (care este in viata in generatia curenta), cu doi sau trei vecini in viata, va exista si in generatia urmatoare.
3. **Ultrapopulare.** Fiecare celula (care este in viata in generatia curenta), care are mai mult de trei vecini in viata, moare in generatia urmatoare.
4. **Creare.** O celula moarta care are exact trei vecini in viata, va fi creata in generatia urmatoare.
5. **Continuitate celule moarte**. Orice alta celula moarta, care nu se incadreaza in regula de creare, ramane o celula moarta.

Vecinii unei celule se considera urmatorii 8, intr-o matrice bidimensionala:

a_{00}	a_{01}	a_{02}
a_{10}	celula curenta	a_{12}
a_{20}	a_{21}	a_{22}

Definim starea unui sistem la generatia n ca fiind o matrice $\mathcal{S}_n \in \mathcal{M}_{\mathtt{m}\times\mathtt{n}}(\{0,1\})$ (`m` - numarul de linii, respectiv `n` - numarul de coloane), unde elementul 0 reprezinta o celula moarta, respectiv 1 reprezinta o celula in viata (in generatia curenta).

Definim o k-evolutie ($k \geq 0$) a sistemului o iteratie $\mathcal{S}_0 \to \mathcal{S}_1 \to \cdots \to \mathcal{S}_k$, unde fiecare $\mathcal{S}_{i+1}$ se obtine din $\mathcal{S}_i$, aplicand cele cinci reguli enuntate mai sus.

Observatie. Pentru celulele aflate pe prima linie, prima coloana, ultima linie, respectiv ultima coloana, se considera extinderea la 8 vecini, prin considerarea celor care **nu** se afla in matrice ca fiind celule moarte.

Exemplificare. Fie urmatoarea configuratie initiala $\mathcal{S}_0$:

0	1	1	0
1	0	0	0
0	0	1	1

In primul rand, vom considera extinderea acestei matrice $\mathcal{S}_0$ de dimensiuni `3` $\times$ `4` intr-o matrice extinsa $\overline{\mathcal{S}_0}$ de dimensiuni `5` $\times$ `6`, astfel:

0	0	0	0	0	0
0	0	1	1	0	0
0	1	0	0	0	0
0	0	0	1	1	0
0	0	0	0	0	0

In cele ce urmeaza, vom lucra doar in interiorul matricei principale, dar considerand extinderea pentru procesarea corecta a vecinilor. Vom parcurge fiecare element, si vom vedea ce regula evolutiva putem aplica. De exemplu, pentru elementul de pe pozitia `(0,0)` in matricea initiala, vom aplica regula de continuitate a celulelor moarte, deoarece este o celula moarta si nu are exact trei vecini in viata.

Urmatoarea celula este in viata, si are exact doi vecini in viata, astfel ca se aplica regula continuitatii celulelor in viata.

Pentru celula de pe pozitia `(0,2)` in $\mathcal{S}_0$, observam ca are un singur vecin, astfel ca se aplica regula de subpopulare - celula va muri in generatia urmatoare.

Urmand acelasi rationament pentru toate celulele, configuratia sistemului intr-o iteratie (in $\overline{\mathcal{S}_1}$) va fi:

0	0	0	0	0	0
0	0	1	0	0	0
0	0	0	0	1	0
0	0	0	0	0	0
0	0	0	0	0	0

Schema de criptare simetrica. Definim o cheie de criptare (pornind de la o configuratie initiala $\mathcal{S}_0$ si o k-evolutie) ca fiind operatia $< \mathcal{S}_0, k >$, care reprezinta tabloul **unidimensional** de date (inteles ca sir de biti) obtinut in urma concatenarii liniilor din matrice din matricea extinsa obtinuta, $\overline{\mathcal{S}_k}$.

De exemplu, pornind de la configuratia anterioara $\mathcal{S}_0$, si aplicand doar o 1-evolutie, se obtine matricea extinsa $\overline{\mathcal{S}_1}$ descrisa anterior, care va avea ca efect al aplicarii operatiei $< \mathcal{S}_0, 1 >$ obtinerea urmatorului tablou unidimensional (inteles ca sir de biti):

```
0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
```

Consideram m un mesaj in clar (un sir de caractere fara spatii). Criptarea $\{m\}_{<\mathcal{S}_0,k>}$ va insemna XOR-area mesajului in clar m cu rezultatul dat de $< \mathcal{S}_0, k >$. Sunt urmatoarele cazuri:

- daca mesajul si cheia au aceeasi lungime, se XOR-eaza element cu element, pana se obtine rezultatul;
- daca mesajul este mai scurt decat cheia, se foloseste doar prima parte din cheie, corespunzatoare lungimii mesajului;
- daca mesajul este mai lung decat cheia, se considera replicarea cheii de oricate ori este nevoie pentru a cripta intreg mesajul.

Consideram ca $m =$ **parola**, si utilizam drept cheie $< \mathcal{S}_0, 1 >$, unde $\mathcal{S}_0$ este configuratia initiala descrisa anterior. Am vazut ca rezultatul obtinut este sirul de biti:

```
0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
```

pe care il vom considera fara spatii:

```
000000001000000010000000000000
```

Pentru a efectua criptarea, trebuie sa analizam sirul de criptat, si anume **parola**. Vom vedea care este codificarea ASCII (binara) a fiecarui caracter din acest sir:

p	01110000
a	01100001
r	01110010
o	01101111
l	01101100
a	01100001

Sirul **parola** va fi, astfel, sirul binar

```
011100000110000101110010011011110110110001100001
```

Observam, in acest caz, ca sirul de criptat este mai lung decat cheia de criptare, astfel ca daca incercam acum o XOR-are, am avea urmatoarea situatie:

```
mesaj = 011100000110000101110010011011110110110001100001
cheie = 000000001000000010000000000000
```

Vom considera, in acest caz, ca vom concatena iar cheia la cheia initiala:

```
mesaj = 011100000110000101110010011011110110110001100001
cheie = 000000001000000010000000000000000000001000000010000000000000
```

Iar apoi vom pastra din noua cheie doar cat ne este suficient pentru a cripta mesajul:

```
mesaj = 011100000110000101110010011011110110110001100001
cheie = 000000001000000010000000000000000000001000000010
```

Mesajul criptat se va obtine prin XOR-are element cu element, stiind ca `0 XOR 0 = 1 XOR 1 = 0`, respectiv `0 XOR 1 = 1 XOR 0 = 1`. In acest caz,

```
mesaj = 011100000110000101110010011011110110110001100001
cheie = 000000001000000010000000000000000000001000000010
cript = 011100001110000111110010011011110110111001100011
```

Mesajul criptat afisat va fi in hexadecimal (pentru a nu fi probleme de afisare a caracterelor), iar in acest caz vom avea:

```
cript = 0111 0000 1110 0001 1111 0010 0110 1111 0110 1110 0110 0011
      =   7    0    E    1    F    2    6    F    6    E    6    3
      = 0x70E1F26F6E63
```

Pentru decriptare se aplica acelasi mecanism, mesajul decriptat se va XOR-a cu cheia calculata, si vom avea in final `m XOR k XOR k = m`. (`k XOR k = 0`, iar `m XOR 0 = m`, din asociativitatea lui XOR, respectiv din regulile de calcul). La decriptare, mesajul nu va fi afisat in hexadecimal, ci in clar.

Cerinta

Se citesc de la tastatura (**STDIN**) numarul de linii `m`, numarul de coloane `n`, numarul de celule vii `p`, pozitiile celulelor vii din matrice, respectiv un numar intreg k. **Atentie!** In citirea inputului se considera matricea initiala, **neextinsa**: se citeste configuratia initiala $\mathcal{S}_0$, si **NU** $\overline{\mathcal{S}_0}$! De exemplu, pentru matricea din prezentarea cerintei, inputul ar fi urmatorul:

```
3                   // m - numarul de linii
4                   // n - numarul de coloane
5                   // p - numarul celulelor vii
0
1                   // prima celula vie este in (0,1)
0
2                   // a doua celula vie este in (0,2)
1
0                   // a treia celula vie este in (1,0)
2
2                   // a patra celula vie este in (2,2)
2
3                   // a cincea celula vie este in (2,3)
5                   // numarul intreg k
```

Se cere, la acest pas, afisarea la **STDOUT** a configuratiei sistemului dupa o k-evolutie. **Atentie!** Se va afisa starea sistemului $\mathcal{S}_k$ si **NU** matricea extinsa $\overline{\mathcal{S}_k}$!

Matricea va fi afisata uzual, iar in acest caz, rezultatul este:

```
0 0 0 0
0 0 0 0
0 0 0 0
```

(toate celulele mor dupa cea de-a doua iteratie).

Observatie: Elementele de pe linie vor fi afisate cu un spatiu intre ele, iar la finalul fiecarei linii, veti afisa un caracter \n. **Si dupa ultima linie veti afisa acel caracter \n!**