Game of Life

George-Vlad Popescu, Andra Băltoiu

Descrierea proiectului

Game of Life este un 'joc' care demonstrează cum, printr-un set de reguli simple, se pot genera comportamente complexe. Nu este propriu-zis un joc, ci un automat celular, adică un model matematic ce descrie funcționarea sistemelor de calcul (hardware sau software). A fost creat de matematicianul John Conway.

Game of Life se desfășoară pe o grilă de celule pătrate, în care fiecare celulă poate avea două stări: vie sau moartă. Celulele interacționează, de la un moment de timp la altul (denumite generații), cu cele 8 celule vecine, după următoarele reguli:

1. Orice celulă vie cu mai puțin de doi vecini în viață moare (*subpopulare*).
2. Orice celulă vie cu doi sau trei vecini vii trăiește în continuare.
3. Orice celulă vie cu mai mult de trei vecini vii moare (*suprapopulare*).
4. Orice celulă moartă cu exact trei vecini devine o celulă vie (*reproducere*).

Puteți vedea o exemplificare a acestor reguli în secțiunea **Explanation** de aici.

Figura 1 prezintă un exemplu de aplicare a regulilor de mai sus pentru o configurație inițială dată (generația 0).

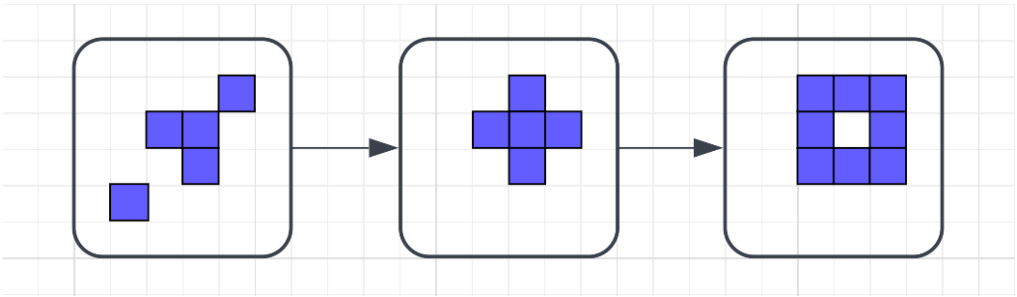

Figura 1: Două generații succesive pornind de la o inițializare în Game of Life

Cu toate că regulile sunt foarte simple, în funcție de configurația inițială a grilei, pot apărea comportamente emergente complexe, spre exemplu structuri care oscilează sau care se mișcă prin grilă (denumite *spaceships*).

Din interacțiunea acestor structuri se pot obține comportamente și mai complexe. Aici este un ceas digital construit în întregime în Game of Life. Iar aici Game of Life e simulat în Game of Life. Datorită proprietăților teoretice ale 'jocului' (e *Turing complete*), se poate construi un calculator cu Game of Life.

Dacă sunteți interasați de aflați mai multe despre structurile ce pot fi create în 'joc', le găsiți în cartea Conway's Game of Life.

În acest proiect veți utiliza structuri de date (liste, stive, arbori, grafuri) pentru a simula Game of Life.

Task 1

Deadline. 30 Martie
Scop. Implementarea regulilor Game of Life

În acest task veți utiliza o matrice de caractere pentru a simula grila din Game of Life și pentru a reține configurația celulelor dintr-o anumită generație.

Celulele vii vor fi reprezentate prin caracterul 'X', iar celulele moarte prin caracterul '+'.

Cerință

Implementați regulile din Game of Life astfel încât, pornind de la o matrice de dimensiune $N \times M$, reprezentând generația inițială, să obțineți următoarele K generații, salvate de asemenea ca matrici.

Matricea inițială va fi citită dintr-un fișier, iar matricile corespunzătoare generațiilor următoare vor fi scrise într-un alt fișier.

Testare

Input

Fișierele cu date de intrare conțin matrici de diferite dimensiuni ce reprezintă generațiile inițiale. Formatul acestor fișiere este următorul:

`T`, numărul taskului
`N`, `M`, dimensiunea grilei
`K`, numărul generațiilor de calculat
matricea conținând configurația inițială a celulelor

Mai jos aveți un exemplu de fișier input, în care dimensiunea grilei este $N = 6$, $M = 6$ și se cer $K = 2$ generații.

Exemplu fișier input

```
1
6 6
2
++++++
++++X+
++XX++
+++X++
+X++++
++++++
```

Output

Scrieți într-un fișier matricile corespunzătoare fiecărei generații dintre cele K cerute, separate printr-o linie goală. Mai jos aveți un exemplu de fișierul de output, ce reprezintă rezolvarea inputului anterior.

Exemplu fișier output

```
++++++
+++X++
++XXX+
+++X++
++++++
++++++

++++++
++XXX+
++X+X+
++XXX+
++++++
++++++
```

Task 2

Deadline. 13 Aprilie
Scop. Stocarea eficientă a diferențelor între generații

În situația în care se dorește stocarea tuturor generațiilor, forma matricială este ineficientă deoarece, în cele mai multe dintre situații, doar o parte din celule își modifică starea de la o generație la alta.

În acest task veți utiliza o structură în care să rețineți doar coordonatele celulelor care se modifică de la o generație la alta.

Cerință

Creați o stivă în care fiecare element corespunde câte unei generații. Fiecare astfel de generație va fi reprezentată printr-o listă de celule ce și-au schimbat starea față de generația anterioară. Prin urmare, veți construi o stivă de liste.

Fiecare element de listă va conține doi întregi, `l`, `c`, reprezentând coordonatele (linia, coloana) celulei care-și schimbă starea.
Liniile sunt numerotate începând cu cea sus, iar coloanele începând cu cea din stânga.
În interiorul unei liste, elementele sunt ordonate întâi după linie, apoi după coloană.
Structura **nu** trebuie să conțină starea curentă a celulei (vie, moartă).

Exemplul de mai jos arată trei generații consecutive și conținutul stivei după fiecare transformare.

Generația 0	Generația 1	Generația 2
+++++ +X+++ +XX++ +++X+ +++++	+++++ +XX++ +XX++ ++X++ +++++	+++++ +XX++ +++X+ +XX++ +++++

Conținut stivă la generația 1

```
1: (1,2) (3,2) (3,3)
```

Conținut stivă la generația 2

```
1: (1,2) (3,2) (3,3)
2: (2,1) (2,2) (2,3) (3,1)
```

Testare

Input

Formatul fișierelor de intrare este același ca la Taskul 1, anume:

`T`, numărul taskului
`N`, `M`, dimensiunea grilei
`K`, numărul generațiilor de calculat
matricea conținând configurația inițială a celulelor

Dându-se o matrice reprezentând inițializarea grilei (generația 0) și un număr de K iterații, scrieți într-un fișier conținutul stivei la generația K.

Output

Conținutul stivei după cele K generații trebuie scris într-un fișier în formatul următor:
$k\ l_i\ c_i$
unde $k \in [1, K]$ reprezintă numărul generației, iar $i \in [0, M_k]$, cu M_k numărul de celule ce și-au modificat starea la respectiva generație.

Așadar, outputul va fi asemănător celui din exemplul de mai sus - conținutului stivei la generația 2.

Bonus

Realizați și operația inversă, utilizând de asemenea, stiva de liste. Dându-se conținutul unei stive pentru K generații și matricea corespunzătoare generației K, scrieți într-un fișier matricea corespunzătoare inițializării (generația 0).

Task 3

Deadline. 11 Mai
Scop. Schimbarea regulilor Game of Life și observarea noii dinamici

În acest task veți crea o alternativă la Game of Life în care există **o singură regulă**, B, și anume: Orice celulă cu exact doi vecini vii devine celulă vie.

Pentru a observa diferențele între Game of Life în varianta originală și cel în care se aplică **doar** noua regulă, B, veți utiliza un arbore binar.

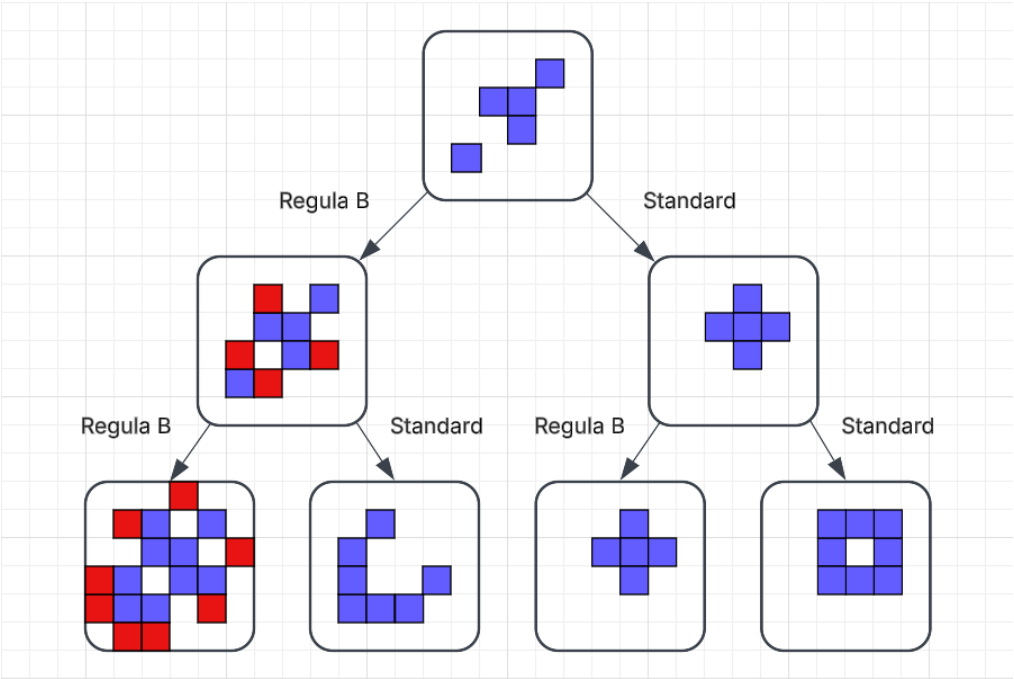

Figura 2: Arborele generațiilor utilizând cele două alternative ale Game of Life

Cerință

Dându-se o inițializare (rădăcina arborelui), creați doi fii: cel din stânga reprezintă generația 1 obținută cu regula nouă, B, iar cel din dreapta generația 1 obținută cu regulile obișnuite. Repetați operația pentru toate nodurile astfel create până la generația K, dată.

Figura 2 ilustrează arborele pentru $K = 2$. Cu roșu sunt semnalate celulele care se nasc la fiecare generație atunci când se utilizează regula B.

Fiecare nod al arborelui va conține, la fel ca la Taskul 2, doar coordonatele celulelor care și-au modificat starea față de generația anterioară, stocate într-o listă. Rădăcina va conține coordonatele celulelor vii din inițializare (generația 0).

Testare

Input

Formatul fișierelor de intrare este același ca la task-urile anterioare, anume:

`T`, numărul testului
`N`, `M`, dimensiunea grilei
`K`, numărul generațiilor de calculat
matricea conținând configurația inițială a celulelor

Output

Parcurgeți în preordine arborele creat și scrieți într-un fișier, pentru fiecare nod din arbore pe care îl parcurgeți, matricea corespunzătoare.

Pentru inputul și $K = 1$ (primele două nivele ale arborelui din Figura 2), outputul va fi:

```
++++++
++++X+
++XX++
+++X++
+X++++
++++++

++++++
++X+X+
++XX++
+X+XX+
+XX+++
++++++

++++++
+++X++
++XXX+
+++X++
++++++
++++++
```

Observați că prima matrice reprezintă inițializarea (generația 0), a doua matrice reprezintă rezultatul aplicării noii reguli (generația 1), iar a treia matrice reprezintă aplicarea regulilor standard Game of Life (generația 1). Ordinea celor trei matrici în fișier corespunde parcurgerii în preordine a arborelui.

Task 4

Deadline. 25 Mai
Scop. Se pot desena structurile din Game of Life fără a 'ridica de pe hârtie creionul' și trecând printr-o celulă o singură dată?

O generație din Game of Life poate fi reprezentată sub forma unui graf, în care celulele vii reprezintă vârfurile grafului. Între două vârfuri există o muchie dacă respectivele celule sunt vecine.
Spre exemplu, generația din Figura 3 poate fi reprezentată ca un graf având matricea de adiacență din Tabela 1.

Observați că generația ilustrată în Figura 3 conține 3 structuri (blocuri de celule vii) separate, ceea ce face ca graful generației să fie neconex. Graful conține 3 componente conexe.

Cerință

Determinați, pentru fiecare nod al arborelui obținut la Taskul 3 până la o generație dată K, cel mai lung lanț Hamiltonian.

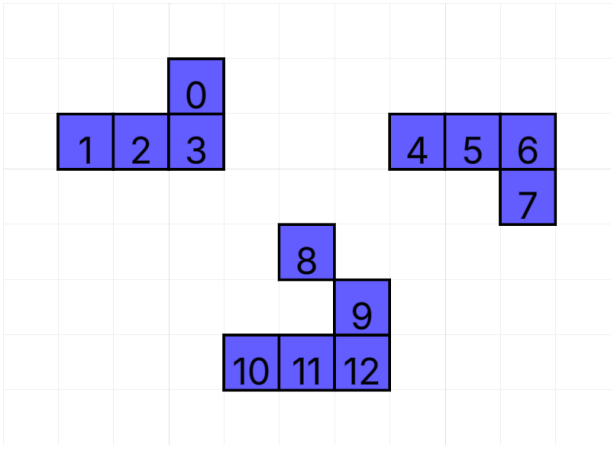

Figura 3: O generație din Game of Life

Nod	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0	0	1	1	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0	0	0	0	0
2	1	1	0	1	0	0	0	0	0	0	0	0	0
3	1	0	1	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	1	0	0	0	0	0	0	0
5	0	0	0	0	1	0	1	1	0	0	0	0	0
6	0	0	0	0	0	1	0	1	0	0	0	0	0
7	0	0	0	0	0	1	1	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	1	0	0	0
9	0	0	0	0	0	0	0	0	1	0	0	1	1
10	0	0	0	0	0	0	0	0	0	0	0	1	0
11	0	0	0	0	0	0	0	0	0	1	1	0	1
12	0	0	0	0	0	0	0	0	0	1	0	1	0

Tabela 1: Matricea de adiacență a grafului generației din Figura 3

Reamintim că un lanț într-un graf este o succesiune de vârfuri cu proprietatea că orice pereche de vârfuri consecutive în această succesiune sunt adiacente.
Un lanț elementar este un lanț în care toate vârfurile sunt distincte.
Un lanț Hamiltonian este un lanț elementar care conține toate vârfurile grafului.

Putem aplica noțiunea de lanț Hamiltonian și la componentele conexe dintr-un graf neconex.

Testare

Input

Formatul fișierelor de intrare este același ca la task-urile anterioare.

Output

Pentru fiecare nod al arborelui, scrieți într-un fișier lungimea celui mai lung lanț Hamiltonian și secvența de noduri prin care se obține, în formatul:

L_i
(l_j, c_j)

unde L reprezintă lungimea lanțului, i reprezintă nodul din arbore pentru care se face afișarea, iar l_j și c_j reprezintă coordonatele (linie și coloană) ale vârfului j din lanț, cu $j \in [1, L+1]$.

- Dacă în graf sau într-una din componentele sale conexe există mai multe lanțuri cu aceeași dimensiune (maximă), se va alege lanțul care pornește dintr-un nod de pe linia cu indicele cel mai mic. Dacă există mai multe lanțuri care pleacă din această linie, se va alege lanțul în care vârful are indicele de coloană cel mai mic. Același principiu se aplică pentru întreaga succesiune de vârfuri din lanț.
- În cazul în care într-un graf există mai multe componente conexe ce conțin lanțuri Hamiltoniene, se va lua în considerare componenta cu lanțul cel mai lung.
- Dacă în graf sau în componentele sale conexe nu există niciun lanț Hamiltonian, afișați lungimea `-1`.

Exemplul 1

Input Exemplul 1

```
4
6 7
1
+++++++
+++X+++
++++X++
++XXX++
+++++++
+++++++
```

Pentru inputul de mai sus, după crearea arborelui de la Taskul 3, obținem următoarele trei noduri (unul pentru generația 0 și două pentru generația 1).

Nodurile nivelului 1

```
+++++++
+++X+++
++++X++
++XXX++
+++++++
+++++++

+++++++
+++XX++
++++XX+
++XXXX+
++X+X++
+++++++

+++++++
+++++++
++X+X++
```

```
+++XX++
+++X+++
+++++++
```

Generația 0. Există două lanțuri Hamiltoniene, ambele de lungime 4:
Un lanț este: `(1,3) (2,4) (3,4) (3,3) (3,2)`.
Alt lanț este: `(3,2) (3,3) (3,4) (2,4) (1,3)`.
Conform regulilor de mai sus, îl vom alege pe cel cu indicele minim al liniei, anume primul din lista de mai sus.

Generația 1. În nodul din stânga al generației 1 (cea de-a doua matrice din exemplul de mai sus), există mai multe lanțuri de lungime 9, între care:
Un lanț este: `(1,3) (1,4) (2,4) (2,5) (3,4) (3,5) (4,4) (3,3) (3,2) (4,2)`.
Alt lanț este: `(1,3) (1,4) (2,4) (3,4) (2,5) (3,5) (4,4) (3,3) (4,2) (3,2)`.

Cele două lanțuri de mai sus sunt ilustrate în Figura 4.
Obervați că din nodul `(1,4)` s-ar fi putut continua atât în `(2,4)`, cât și în `(2,5)`, ambele alegeri conducând la drumuri Hamiltoniene. Nodul `(2,4)` a fost ales deoarece are indicele coloanei mai mic.
În continuare, există iarăși două opțiuni: deplasarea pe aceeași linie (în nodul `(2,5)`) sau pe coloană (nodul `(3,4)`). Conform regulei, alegem prima variantă.

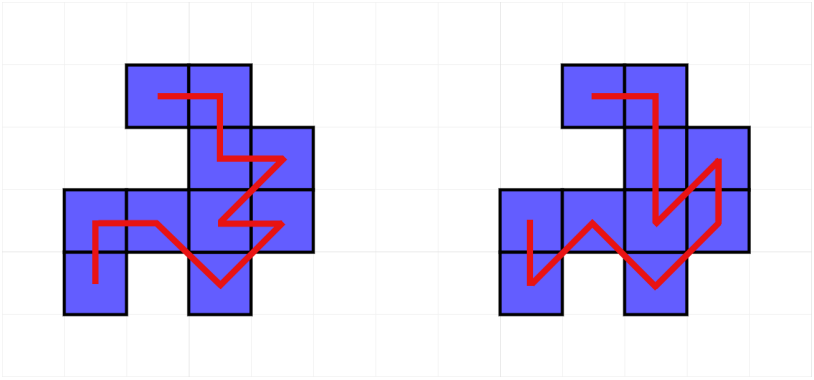

Figura 4: Două posibile lanțuri Hamiltoniene

Aplicând aceleași reguli și pentru nodul din dreapta al generației 1 (cea de-a treia matrice din exemplul de mai sus), obținem lanțul `(2,2) (3,3) (2,4) (3,4) (4,3)`.

Așadar, outputul pentru acest test va fi:

Output Exemplul 1

```
4
(1,3) (2,4) (3,4) (3,3) (3,2)
9
(1,3) (1,4) (2,4) (2,5) (3,4) (3,5) (4,4) (3,3) (3,2) (4,2)
4
(2,2) (3,3) (2,4) (3,4) (4,3)
```

Exemplul 2

Input Exemplul 2

```
4
6 7
0
+++++++
+XXXXX+
+++++++
+++++X+
+++XXX+
+++++++
```

Graful corespunzător inputului de mai sus este neconex și are două componente conexe. Prima componentă conexă conține lanțul Hamiltonian de lungime 4:
`(1,1) (1,2) (1,3) (1,4) (1,5)`.
Cea de-a doua componentă conexă conține mai multe lanțuri Hamiltonian de lungime 3, dintre care cel care respectă ordinea liniilor și coloanelor este: `(3,5) (4,5) (4,4) (4,3)`.
Conform regulilor, se alege componenta conexă cu lanțul cel mai lung, așa că outputul va fi:

Output Exemplul 2

```
4
(1,1) (1,2) (1,3) (1,4) (1,5)
```

Exemplul 3

Input Exemplul 3

```
4
6 7
0
+++++++
+X+X+++
++X+X++
++XXX++
+X+++X+
+++++++
```

Observați că în exemplul de mai sus există trei noduri 'marginale' (noduri cu grad 1, care au câte o singură muchie). Din acest motiv, în graf nu există lanț Hamiltonian.
O interpretare pentru inexistența unui lanț Hamiltonian o reprezintă faptul că nu se poate desena structura fără a ridica de pe hârtie creionul și trecând printr-o celulă o singură dată.

Output Exemplul 3

```
-1
```