

Langages et programmation

Mini projet sur les automates cellulaires

Simulation d'un feu de forêt

Objectif

Simuler la propagation d'un feu de forêt au moyen d'un *automate cellulaire* :

- *automate* car le système évolue par transitions successives à partir d'un état initial ;
- *cellulaire* car les états sont enregistrés dans des cellules.



I. Présentation

La parcelle étudiée est représentée sous la forme d'une grille carrée de taille $n \times n$. Autrement dit, elle est formée de n^2 cellules, chaque cellule étant de l'un des quatre types suivants :

Type de cellule	Couleur	Code
Sol nu	ivoire	0
Arbre en vie	verte	1
Arbre en feu	rouge	2
Arbre en cendres	grise	3

Remarque. La propagation d'un feu de forêt dépend essentiellement de la densité des arbres dans la parcelle étudiée.

Si p désigne la proportion (i.e la densité) d'arbres en vie dans la parcelle, alors le nombre N d'arbres initialement (i.e avant le feu de forêt) en vie est donné par la relation suivante :

$$N = \lfloor p \times n^2 \rfloor$$

L'évolution du feu dans la parcelle n'est que le reflet de l'évolution de la grille dont les cellules changent d'état. Autrement dit, chaque cellule passe d'un état *courant* à un état *suitant* conformément aux *règles d'évolution* de l'automate :

- si la cellule est un arbre en feu (code 2), à l'état suivant, elle devient une cellule d'un arbre en cendres (code 3) ;
- si la cellule est un arbre en feu (code 2), et qu'une de ses voisines v dans les quatre directions (haut, bas, gauche et droite) est un arbre en vie (code 1), alors v devient une cellule d'un arbre en feu (code 2) à l'état suivant.

En résumé, l'implémentation de cet automate cellulaire, consiste simplement à coder une succession de grilles jusqu'à ce qu'il n'y ait plus aucune cellule en feu (i.e fin de la propagation de l'incendie dans la parcelle).

II. Automate cellulaire

Consignes.

L'intégralité des codes des fonctions relatives à la mise en œuvre de l'automate cellulaire est à implémenter dans le fichier `feu_foret_automate.py`.

La fonction `construction_grille(p, n)` prend en paramètres la valeur de la densité p d'arbres dans une parcelle ainsi que la taille n de cette dernière, et renvoie une grille de $n \times n$ cellules contenant $N = \lfloor p \times n^2 \rfloor$ arbres en vie placés aléatoirement selon une loi uniforme.

Remarque. Le tirage aléatoire des coordonnées (i, j) , avec i (resp. j) le numéro de ligne (resp. de colonne), des N arbres en vie est réalisé au moyen de la fonction `tirage_aleatoire(p, n)` qui renvoie une liste de tuples de deux valeurs.

La fonction `affichage_grille(grille)` permet d'afficher en console les valeurs des cellules de la grille passée en argument.

Remarque. La variable globale `epoque` est incrémentée à chaque appel de la fonction d'affichage de sorte de pouvoir identifier à quel état de mise à jour de l'automate cellulaire correspond la grille affichée en console.

Le dictionnaire `etats_ac` permet de mémoriser le nombre d'arbres de chacune des catégories (en vie, en feu et en cendres). Les valeurs associées aux clés de ce dictionnaire sont actualisées à chaque mise à jour de l'état de l'automate cellulaire.

A titre d'exemple, l'affichage obtenu en console, après exécution du programme principal du fichier `feu_foret_automate.py`, est le suivant :

```
époque 0: 1 arbres en feu
2 0 1 1 0 0 0 1
1 0 1 1 0 0 1 1
1 1 0 1 1 0 0 0
0 0 1 1 1 1 1 1
1 1 1 0 1 1 1 0
0 0 1 0 0 0 1 1
0 0 0 1 1 1 1 1
1 1 0 0 1 1 0 1
```

1) Écrire une fonction `voisins(n, i, j)` qui :

- prend en paramètres la taille n de la grille représentant la parcelle, et les coordonnées i et j , avec i (resp. j) le numéro de ligne (resp. de colonne), d'une cellule ;
- renvoie la liste des voisins (suivant les quatre directions haut, bas, gauche et droite) de cette cellule sous la forme d'une liste de tuples de deux valeurs.

A titre d'exemple, la trace de l'exécution en console de cette fonction pour différentes valeurs de ses arguments :

```
>>> voisins(8,0,0)
[(0, 1), (1, 0)]
>>> voisins(8,2,2)
[(2, 3), (2, 1), (1, 2), (3, 2)]
```

2) Écrire une fonction `mise_a_jour_grille(grille)` qui :

- prend en paramètre une grille de $n \times n$ cellules représentant la parcelle, et procède à la mise à jour des états des cellules de celle-ci suivant les règles d'évolution de l'automate cellulaire ;
- renvoie le booléen `True` s'il existe encore au moins une cellule de type arbre en feu dans la grille, et `False` sinon.

A titre d'exemple, la trace de l'exécution en console de cette fonction lors de deux appels consécutifs :

```
>>> mise_a_jour_grille(parcelle)
True
>>> affichage_grille(parcelle)
époque 1: 1 arbres en feu
3 0 1 1 0 0 0 1
2 0 1 1 0 0 1 1
1 1 0 1 1 0 0 0
0 0 1 1 1 1 1 1
1 1 1 0 1 1 1 0
0 0 1 0 0 0 1 1
0 0 0 1 1 1 1 1
1 1 0 0 1 1 0 1

>>> mise_a_jour_grille(parcelle)
True
>>> affichage_grille(parcelle)
époque 2: 1 arbres en feu
3 0 1 1 0 0 0 1
3 0 1 1 0 0 1 1
2 1 0 1 1 0 0 0
0 0 1 1 1 1 1 1
1 1 1 0 1 1 1 0
0 0 1 0 0 0 1 1
0 0 0 1 1 1 1 1
1 1 0 0 1 1 0 1
```

- 3) Modifier le corps du programme principal `main` du fichier `feu_foret_automate.py` afin d'appeler en boucle la fonction de mise à jour de l'automate cellulaire tant qu'il existe des modifications de ses états internes (i.e tant que la propagation du feu dans la parcelle se poursuit).
- 4) Ajouter dans la fonction d'affichage toute information qui vous semble pertinente pour évaluer les caractéristiques de l'incendie dans la parcelle étudiée :
 - nombre (ou pourcentage) des arbres en cendres ;
 - nombre d'arbres en feu par unité de temps ;
 - etc.

III. Interface graphique

Consignes.

L'intégralité des codes des fonctions relatives à la mise en œuvre de l'interface graphique est à implémenter dans le fichier `feu_foret_interface.tk.py`.

Remarque. L'interface graphique de l'automate cellulaire est construite et gérée à partir des fonctions du module *Tkinter* de Python dont une documentation (cf fichier *tkinter.pdf*) vous est donnée dans le dossier du mini projet.

Les éléments de l'interface graphique sont les suivants :

- *bouton gauche de la souris* : modification de l'état d'une cellule correspondant initialement à un arbre en vie (code 1) vers un nouvel état correspondant à un arbre en feu (code 2) ;
- *bouton RUN* : lancement de l'automate cellulaire et calcul d'une nouvelle génération toutes les 150 ms ;
- *bouton STOP* : arrêt de l'automate cellulaire ;
- *bouton STEP* : fonctionnement en mode pas à pas de l'automate cellulaire (i.e seule la génération suivante est calculée) ;
- *label Arbres en feu* : affichage du nombre d'arbres en feu de la génération actuelle.

Actuellement, seules les fonctionnalités suivantes sont implémentées :

- initialisation de l'automate cellulaire : fonction principale `main` ;
- affichage de l'état de l'automate cellulaire : fonctions `dessiner_grille`, `dessiner_cellule` et `actualiser_etats` ;
- création d'un départ de feu : fonction `incendier_arbre` ;
- propagation du feu en mode pas à pas : fonctions `executer_pas_a_pas` et `propager_feu`.

- 5) Exécuter le fichier `feu_foret_interface.tk.py`, puis tester les fonctionnalités déjà implémentées de l'interface graphique.
Prendre connaissance du code source du fichier `feu_foret_interface.tk.py` afin de comprendre comment sont codées les fonctionnalités testées.

- 6) Compléter les codes sources des fonctions `lancer_execution` et `propager_incendie` afin d'implémenter la fonctionnalité relative au bouton *RUN* de l'interface graphique.

Remarque (1). La variable (globale) booléenne `run_stop` permet de mémoriser l'état de marche de l'automate cellulaire :

- automate en marche : `run_stop = True` ;
- automate à l'arrêt : `run_stop = False` ;

Remarque (2). Pour implémenter le fonctionnement *automatique* de l'automate cellulaire (i.e le calcul d'une nouvelle génération toutes les 150 ms), consulter le chapitre *Les animations* du document *tkinter.pdf*.

- 7) Compléter le code source de la fonction `arreter_execution` afin d'implémenter la fonctionnalité relative au bouton *STOP* de l'interface graphique.
- 8) Ajouter dans l'interface graphique un nouveau bouton *NEW*, associé à la fonction `reinitialiser_automate`, qui permet de réinitialiser la grille de l'automate cellulaire (i.e créer une nouvelle parcelle sans arbres en feu ou en cendres).
- 9) Ajouter dans l'interface graphique toute information qui vous semble pertinente pour évaluer les caractéristiques de l'incendie dans la parcelle étudiée :
- nombre (ou pourcentage) des arbres en cendre ;
 - nombre d'arbres en feu par unité de temps ;
 - etc.

Remarque. Pour afficher de nouvelles valeurs, il faut au préalable déclarer les labels correspondants dans la fonction principale `main`.

La mise à jour des valeurs associées à ces nouveaux labels peut être réalisée dans le corps de la fonction `actualiser_etats`.