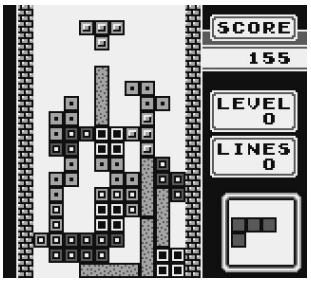
Projet 1

Comparaisons des différents générateurs de pièces au TETRIS

sujet proposé par Lucas Gerin

lucas.gerin@polytechnique.edu

Outils de MAP361 : Conditionnement, Combinatoire, Probabilités discrètes, ...



Dans les versions successives de TETRIS (sur les consoles NES, Gameboy, SuperNES, puis sur PC) les développeurs n'ont étonnamment pas utilisé la même loi pour générer les pièces aléatoires[1]. En particulier, et contrairement à ce qu'on pourrait imaginer, aucune des versions ne renvoie des pièces aléatoires i.i.d. uniformes[2].

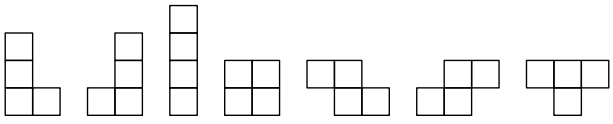
L'idée des développeurs du jeu était d'utiliser des générateurs qui limitent le nombre de pièces successives identiques et inversement d'éviter des trop longues attentes pour une pièce donnée.

1.1 Deux générateurs de pièces

Toutes les versions de TETRIS se jouent sur une grille rectangulaire sur laquelle tombent successivement des pièces choisies dans l'ensemble $\mathcal{P} = \{L, J, I, O, Z, S, T\}$ des 7 pièces suivantes :

[1] Sources : History of Tetris randomizers, Générateur de NES.

[2] Il faut distinguer deux problèmes différents. 1) L'implémentation d'un générateur de nombres *pseudo-aléatoires*. 2) Étant donné ce générateur, avec quelle distribution de probabilités génère-t-on les pièces?
À propos de 1) : Sur les différentes consoles l'aléa est produit par l'horloge interne de la machine (une graine aléatoire est fournie, par exemple, en comptant le nombre de ms depuis que la console est allumée). Dans ce projet on ne va considérer que le problème 2).



(Dans vos simulations vous pourrez identifier chaque pièce à un nombre dans $\{0, 1, \ldots, 6\}$ avec la correspondance ci-dessus.)

Ce qu'on appelle *générateur de pièces* dans ce projet est une loi de probabilité $\mathbf{G}$ sur l'ensemble des suites d'éléments de $\mathcal{P}$. À titre de comparaison avec les autres générateurs, le générateur $\mathbf{G}_{\text{iid}}$ est la loi donnée par une suite de variables aléatoires i.i.d. uniformes dans $\mathcal{P}$. Nous allons commencer par définir les lois de deux générateurs différents : $\mathbf{G}_{\text{NES}}$ et $\mathbf{G}_{\text{bag}}$.

Le générateur $\mathbf{G}_{\text{NES}}$

Sur la console NES (1989) le générateur de pièces $\mathbf{G}_{\text{NES}}$ produisait une suite $(P_k)_{k\geq 1}$ définie de la façon suivante :

- On tire P_1 uniformément au hasard dans $\mathcal{P}$.
- Pour $k \geq 2$ on tire Q uniformément au hasard dans $\mathcal{P}$ indépendamment de tout ce qui précède :
  - Si $Q \neq P_{k-1}$ on pose $P_k = Q$.
  - Si $Q = P_{k-1}$ on tire R uniformément au hasard dans $\mathcal{P}$ indépendamment de tout ce qui précède et quelque soit le résultat on pose $P_k = R$.

Le générateur $\mathbf{G}_{\text{bag}}$

Sur la version PC (2001) on utilise le générateur de pièces $\mathbf{G}_{\text{bag}}$. Tous les 7 tirages on se donne un nouveau "sac" virtuel de 7 pièces dans lequel on pioche jusqu'à épuisement du sac, ce qui fait que l'algorithme est parfois appelé 7-*bag*. Formellement $\mathbf{G}_{\text{bag}}$ produit une suite $(P_k)_{k\geq 1}$ définie de la façon suivante :

- On tire uniformément au hasard et sans remise $P_1, P_2, \ldots, P_7$ dans $\mathcal{P}$ (autrement dit $(P_1, P_2, \ldots, P_7)$ est une permutation aléatoire uniforme de $\mathcal{P}$).
- Indépendamment de $P_1, P_2, \ldots, P_7$ on tire uniformément au hasard et sans remise $P_8, P_9, \ldots, P_{14}$ dans $\mathcal{P}$.
- Indépendamment de $P_1, P_2, \ldots, P_{14}$ on tire uniformément au hasard et sans remise $P_{15}, \ldots, P_{21}$ dans $\mathcal{P}$.
- Et ainsi de suite...

1.2 Premières simulations

On dit qu'un générateur est *équitable* si pour tout $k \geq 1$ et pour tout $p \in \mathcal{P}$ on a $\mathbb{P}(P_k = p) = 1/7$. Tout est symétrique dans les constructions des générateurs, donc il est clair que $\mathbf{G}_{\text{iid}}$, $\mathbf{G}_{\text{bag}}$ et $\mathbf{G}_{\text{NES}}$ sont équitables.

S1. Simuler une suite $(P_k)_{1\leq k\leq K}$ pour $\mathbf{G}_{\text{iid}}$ jusqu'à $K = 10000$. Pour vérifier expérimentalement que le générateur est équitable, afficher avec un histogramme la fréquence de chacune des pièces. Avec `matplotlib` un histogramme des valeurs `Tirages` avec comme classes $\{1, 2, \ldots, 7\}$ s'affiche avec

```
plt.hist(Tirages, np.arange(0.5,8.5), density='true', ec='black')
```

(l'argument `density='true'` sert à normaliser l'histogramme pour avoir des fréquences, et `ec='black'` sert à dessiner les bords des bâtons).

S2. Simuler une suite $(P_k)_{1\leq k\leq K}$ pour $\mathbf{G}_{\text{NES}}$ jusqu'à $K = 10000$. Afficher l'histogramme de la fréquence de chacune des pièces.

S3. Simuler une suite $(P_k)_{1\leq k\leq K}$ pour $\mathbf{G}_{\text{bag}}$ jusqu'à $K = 10000$. Afficher l'histogramme de la fréquence de chacune des pièces.

1.3 Matrices de transition

Pour différencier $\mathbf{G}_{\text{NES}}$ et $\mathbf{G}_{\text{bag}}$ du générateur $\mathbf{G}_{\text{iid}}$ on commence par considérer le nombre de *transitions* entre deux pièces données.

Pour un entier $K \geq 2$ fixé et deux pièces $p, p' \in \mathcal{P}$, on note $N_K^{p',p}$ la variable aléatoire

$$N_K^{p',p} = \sum_{k=1}^{K-1} \mathbf{1}_{P_k=p', P_{k+1}=p}.$$

Une remarque utile pour la suite :

$$\sum_{p,p'\in\mathcal{P}} N_K^{p',p} = \sum_{k=1}^{K-1} \sum_{p,p'\in\mathcal{P}} \mathbf{1}_{P_k=p', P_{k+1}=p} = \sum_{k=1}^{K-1} \mathbf{1} = K - 1.$$

T1. Pour tout $K \geq 2$, pour le générateur $\mathbf{G}_{\text{iid}}$, calculer $\mathbb{E}[N_K^{p',p}]$.

S4. En réutilisant votre code de la question S1, afficher pour $K = 10000$ une réalisation pour $\mathbf{G}_{\text{iid}}$ de la matrice N_K de taille 7×7 définie par

$$N_K = \left(N_K^{p',p}\right)_{p,p'\in\mathcal{P}}.$$

Pour que cela soit plus visuel, afficher une *heat map* de N_K avec les commandes suivantes :

```
plt.imshow(N_K, cmap='hot')
plt.colorbar()
plt.show()
```

T2. Pour $\mathbf{G}_{\text{NES}}$, calculer pour tous $p, p' \in \mathcal{P}$ et $k \geq 2$ la probabilité $\mathbb{P}(P_k = p \mid P_{k-1} = p')$.

T3. En utilisant la question précédente, démontrer que pour tout $K \geq 2$, pour le générateur $\mathbf{G}_{\text{NES}}$,

$$\mathbb{E}[N_K^{p',p}] = \begin{cases} (K-1)\left(\frac{1}{7^2} + \frac{1}{7^3}\right), & \text{si } p' \neq p, \\ (K-1) \times \frac{1}{7^3} & \text{si } p' = p. \end{cases}$$

S5. On souhaite confirmer le résultat de la question T3 avec des simulations. En réutilisant votre code de la question S2, afficher pour $K = 10000$ une réalisation pour $\mathbf{G}_{\mathrm{NES}}$ de la matrice N_K, sous forme de *heat map*.

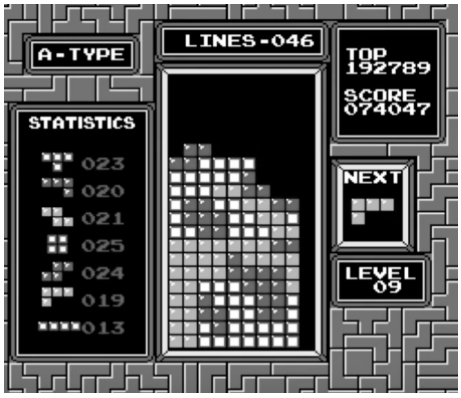
T4. Pour tout $K \geq 2$, pour le générateur $\mathbf{G}_{\mathrm{bag}}$, calculer $\mathbb{E}[N_K^{p',p}]$.
(Indication : Commencer par le cas $p = p'$.)

S6. En réutilisant votre code de la question S3, afficher pour $K = 10000$ une réalisation pour $\mathbf{G}_{\mathrm{bag}}$ de la matrice N_K, sous forme de *heat map*.

S7. Commenter rapidement les résultats graphiques des trois *heat map* obtenues.

1.4 Pénurie de I

L'un des grandes angoisses des joueurs de Tetris est de perdre à cause d'une trop longue attente de la pièce I :



On cherche donc à étudier la loi de l'écart entre deux I successifs. Pour une suite de pièces donnée $(P_k)_{k\geq 1}$ engendrée par un générateur $\mathrm{g} \in \{\mathrm{iid}, \mathrm{bag}, \mathrm{NES}\}$ on note

$$\tau_I^{(1)} = \min\{k \geq 1,\ P_k = I\}, \qquad \tau_I^{(2)} = \min\{k > \tau_I^{(1)},\ P_k = I\}, \qquad E_I^{\mathrm{g}} = \tau_I^{(2)} - \tau_I^{(1)}.$$

La variable aléatoire E_I^{g} représente donc l'écart entre les deux premières pièces I produites par le générateur g.
Remarque. *Cela peut sembler restrictif de n'étudier que l'écart entre le premier et le deuxième I. En fait on peut démontrer que pour les trois générateurs $\mathbf{G}_{\mathrm{iid}}$, $\mathbf{G}_{\mathrm{bag}}$, $\mathbf{G}_{\mathrm{NES}}$, et pour tout $r \geq 1$, $\tau_I^{(r+1)} - \tau_I^{(r)} \overset{\text{(loi)}}{=} \tau_I^{(2)} - \tau_I^{(1)}$, où $\tau_I^{(r)}$ désigne la position de la r-ème pièce I.*

T5. Pour le générateur $\mathbf{G}_{\mathrm{iid}}$, déterminer la loi de E_I^{iid}.

T6. Pour le générateur $\mathbf{G}_{\mathrm{bag}}$, déterminer la loi de E_I^{bag}.

T7. Pour le générateur $\mathbf{G}_{\mathrm{NES}}$, démontrer que

$$\mathbb{P}(E_I^{\mathrm{NES}} = k) = \begin{cases} \alpha & \text{pour } k = 1, \\ (1-\alpha)\beta(1-\beta)^{k-2} & \text{pour tout } k \geq 2, \end{cases}$$

où $\alpha = \frac{1}{49}$ et $\beta = \frac{1}{7} + \frac{1}{49}$.
Indication : On peut décomposer selon la valeur de $\tau_I^{(1)}$ et écrire

$$\mathbb{P}(E_I^{\mathrm{NES}} = k) = \mathbb{P}(\tau_I^{(1)} = 1, \tau_I^{(2)} = 1 + k) + \sum_{\ell \geq 2} \mathbb{P}(\tau_I^{(1)} = \ell, \tau_I^{(2)} = \ell + k)$$

S8. Écrire deux fonctions python `Ecart_bag(k)` et `Ecart_NES(k)` qui renvoient respectivement les valeurs numériques de $\mathbb{P}(E_I^{\mathrm{bag}} = k)$ et $\mathbb{P}(E_I^{\mathrm{NES}} = k)$.

S9. Tracer sur le même graphique les fonctions $k \mapsto \mathbb{P}(E_I^{\mathrm{iid}} = k)$, $k \mapsto \mathbb{P}(E_I^{\mathrm{bag}} = k)$, $k \mapsto \mathbb{P}(E_I^{\mathrm{NES}} = k)$ pour $1 \leq k \leq 25$. Comme il s'agit de fonctions discrètes, ajouter l'argument `'o-'` pour faire apparaître les points. Par exemple,

```
plt.plot(range(1,26),[Ecart_bag(k) for k in range(1,26)],'o-',label='7-bag')
```

1.5 Excès de T

La pièce T est parfois considérée comme la "pire" pièce du TETRIS. Nous allons étudier expérimentalement la longueur de la plus longue séquence de T consécutifs pour les deux générateurs $\mathbf{G}_{\mathrm{iid}}$ et $\mathbf{G}_{\mathrm{NES}}$. (Le problème est bien sûr trivial pour $\mathbf{G}_{\mathrm{bag}}$ car on ne peut pas avoir plus de deux T consécutifs.)

Pour un générateur g fixé on note R_K^{g} la variable aléatoire donnée par le plus grand nombre de T consécutifs dans une suite de K tirages. Formellement,

$$R_K^{\mathrm{g}} = \max\{\ell \geq 1 \mid \exists\, 1 \leq k \leq K - \ell + 1 \text{ tel que } P_k = P_{k+1} = \cdots = P_{k+\ell-1} = T\}\,.$$

(Si jamais $P_k \neq T$ pour tout k alors on pose $R_K^{\mathrm{g}} = 0$.) On va comparer expérimentalement les distributions de R_K^{iid} et R_K^{NES}.

S10. Écrire une fonction `SequenceSuccessive()` qui prend en entrée une liste de pièces `ListePieces` et une pièce `p` $\in \mathcal{P}$ et qui renvoie le plus grand nombre de `p` consécutifs dans `ListePieces`. Par exemple,

```
>ListePieces=[5, 2, 4, 2, 2, 2, 0, 6, 5, 3, 5, 6, 6]
>print(SequenceSuccessive(ListePieces,2))
3
>print(SequenceSuccessive(ListePieces,1))
0
```

S11. En réutilisant les questions S1 et S2, simuler $S = 500$ chacune des variables aléatoires R_K^{iid} et R_K^{NES} pour $K = 1000$. Représenter les deux distributions de valeurs sur le même histogramme : si `Matrice` est une matrice à 2 colonnes, la commande

```
plt.hist(Matrice, np.arange(0.5,8.5), density=True, label=['iid', 'NES'])
```

affiche deux histogrammes sur le même graphique (un pour chaque colonne de la matrice).

T8. On observe expérimentalement que R_K^{NES} atteint très rarement 4. Démontrer que, en effet,

$$\mathbb{P}\left(R_K^{\text{NES}} \geq \ell\right) \leq (K - \ell + 1) \times \frac{1}{7} \times \left(\frac{1}{49}\right)^{\ell-1}.$$

Application numérique : Combien vaut le terme de droite pour $K = 1000$, $\ell = 4$?

1.6 Un cas inéquitable : Le générateur GameBoy

Sur la console GameBoy (également en 1989, comme la NES) la situation était encore différente, et assez complexe[3]. Pour des raisons d'implémentation et d'espace mémoire le générateur $\mathbf{G}_{\text{GB}}$ fonctionnait de la façon suivante. On associe à chacune des 7 pièces un triplet de trois bits :

L	J	I	O	Z	S	T
`[0,0,0]`	`[0,0,1]`	`[0,1,0]`	`[0,1,1]`	`[1,0,0]`	`[1,0,1]`	`[1,1,0]`

On définit alors l'opérateur logique `OU` sur les pièces en faisant le `OU` bit par bit sur sur le triplet correspondant. Par exemple

$$\texttt{OU}(J, S, Z) = \texttt{OU}(001, 101, 100) = 101 = S.$$

Les pièces (P_k) sont générées ainsi :

- P_1, P_2 sont tirées uniformément au hasard dans $\mathcal{P}$, indépendamment.
- Pour $k \geq 3$, on tire U uniformément au hasard dans $\mathcal{P}$, indépendamment de ce qui précède ;
  - Si $\texttt{OU}(P_{k-2}, P_{k-1}, U) \neq P_{k-2}$ on pose $P_k = U$;
  - Sinon on tire V uniformément au hasard dans $\mathcal{P}$;
    - Si $\texttt{OU}(P_{k-2}, P_{k-1}, V) \neq P_{k-2}$ on pose $P_k = V$;
    - Sinon on tire W uniformément au sort dans $\mathcal{P}$ et on accepte dans tous les cas : $P_k = W$.

Comme la situation n'est plus symétrique (car les transitions dépendent du nombre de bits 1 dans le triplet de chaque lettre), le générateur n'est même pas équitable. On va le vérifier expérimentalement.

S12. Écrire une fonction `TirageGB()` qui renvoie un triplet uniforme parmi les 7 triplets `[0,0,0]`, `[0,0,1]`,..., `[1,1,0]`.

[3] Source : `https://harddrop.com/wiki/Tetris_(Game_Boy)#Randomizer`

S13. Écrire une fonction `OU` qui prend en entrée 3 triplets de 3 bits et effectue le `OU` bit par bit. Par exemple

```
> print(OU([0,0,1],[1,0,1],[1,0,0]))
[1,0,1]
> print(OU([1,0,0],[1,0,0],[0,0,0]))
[1,0,0]
```

S14. Simuler une suite $(P_k)_{1\leq k\leq K}$ pour $\mathbf{G}_{\mathrm{GB}}$ jusqu'à $K = 100000$. Afficher l'histogramme des fréquences des pièces. Quelles sont les 3 pièces les plus fréquentes?