UFR de mathématique et d'informatique
Université de Strasbourg

Licence 1 Semestre 1 Automne
Algorithmique et Programmation 1
Contrôle continu
Jeudi 12 décembre 2019
Durée : 1 heure 15
Documents autorisés :
tous cours et fichiers de votre compte
Accès au web interdit

— Dans cette épreuve, le seul module autorisé est le module `graph` disponible sur la plateforme *Moodle*. Vous ne pouvez pas modifier ce module.

— Lorsqu'on spécifie que le paramètre d'une fonction doit satisfaire certaines pré-conditions, vous n'avez pas besoin d'écrire des tests pour vérifier que ces conditions sont effectivement satisfaites.

— Toutes vos fonctions devront être testées. Nous vous invitons à réaliser une batterie de tests pour chaque fonction et à garder trace de ces tests dans votre programme.

— Nous vous demandons d'écrire toutes vos fonctions dans un même fichier.

— À titre indicatif, chacune des 8 questions est évaluée sur 2 points. Les 4 points restants évaluent la lisibilité de votre programme et votre méthodologie de test.

Les deux premières sections sont indépendantes. La troisième fait appel aux résultats des deux premières. Le contexte est celui d'un jeu d'échecs. Vous n'avez pas besoin de savoir jouer aux échecs pour réussir cette épreuve. Sachez simplement que les échecs se jouent sur un plateau appelé *échiquier* (identique à un *damier*) comportant 64 cases noires ou blanches organisées en 8 lignes et 8 colonnes (figure 1). Chaque case est caractérisée par un numéro de ligne et de colonne. À un instant donné, la position de chaque pièce est définie par deux entiers représentant la ligne et la colonne de la case sur laquelle elle se trouve. Le fou est une pièce dont le mouvement peut être seulement en diagonal (jamais horizontal ou vertical) (figure 1). Notre objectif est calculer la liste de toutes les cases que cette pièce peut atteindre à partir d'une position donnée en un mouvement (figure 2).

1 Les cases accessibles

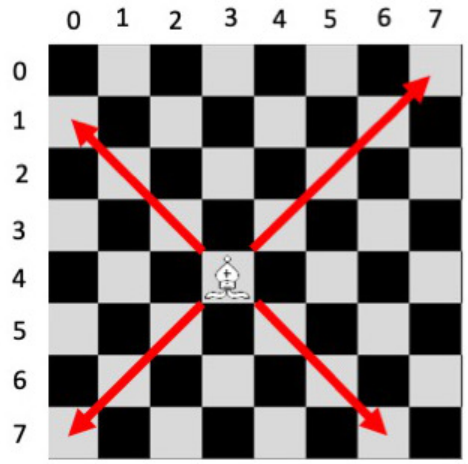

FIGURE 1: Un échiquier avec un fou en (4,3) : un mouvement de cette pièce peut être seulement en diagonal (jamais horizontal ou vertical).

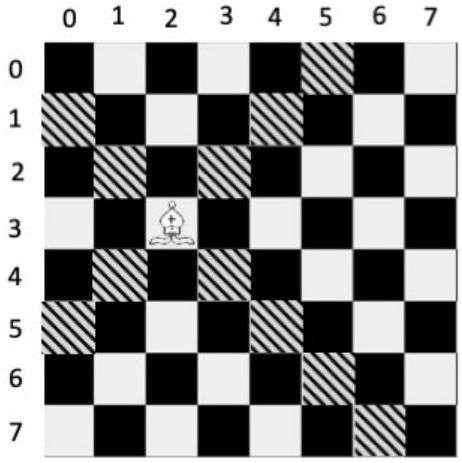

FIGURE 2: Pour une fou situé en (3,2), les cases accessibles ont été hachurées. Notre objectif, dans ce TP, est de calculer, à partir de la position de la fou (ici (3,2)), la liste de ces cases hachurées et de les visualiser en rouge.

Dans cette épreuve, on suppose que deux cases sont voisines si et seulement si elles ontun sommet en commun, mais aucun côté commun

1. Dans un premier temps, nous supposons que l'échiquier est infini et que les numéros de ligne et de colonne sont des entiers quelconques (positifs ou non). Écrire une fonction `un_pas_fou_inf` qui admet deux paramètres entiers `lgn` et `col` représentant une case de l'échiquier. Cette fonction devra retourner une liste de 4 couples d'entiers représentant les cases voisines de `(lgn,col)`. Par exemple l'appel : `un_pas_fou_inf(3,7)` devra retourner la liste `[(2,6),(2,8),(4,6),(4,8)]`. Voir figure 3.
2. Toujours dans le contexte d'un échiquier infini, écrire une fonction `N_pas_fou_inf` qui admet trois paramètres entiers : `lgn` et `col` représentant une case de l'échiquier et `N`, un entier strictement positif. Cette fonction devra retourner une liste de 4 couples d'entiers représentant les cases situées dans la même direction que les voisins de `(lgn,col)`, mais à `N` cases de `(lgn,col)`. Par exemple l'appel : `N_pas_fou_inf(3,7,3)` devra retourner la liste `[(0,4),(0,10),(6,4),(6,10)]`. C'est presque le résultat représenté sur la figure 4. (*presque* car les éléments trop éloignés n'ont pas été représentés). Ainsi, pour $N \geq 8$, la liste retournée est toujours vide, quelque soient `lgn` et `col`.
3. À présent, prenons en compte le fait que l'échiquier est fini et comporte 8 lignes et 8 colonnes numérotées de 0 à 7. Écrire une fonction `est_sur_echiquier` qui admet deux arguments `lgn` et `col` entiers quelconques représentant la position d'une case. Cette fonction devra retourner une valeur logique `True` si la case `(lgn,col)` est sur l'échiquier et `False` dans le cas contraire. Par exemple `est_sur_echiquier(3,2)` vaut `True` mais `est_sur_echiquier(3,8)` vaut `False`.
4. En utilisant les fonctions précédentes, écrire une fonction `N_pas_fou_fini` qui admet trois paramètres entiers : `lgn` et `col` représentant une case de l'échiquier et `N`, un entier strictement positif. Cette fonction devra retourner une liste de couples d'entiers (au nombre de 4 ou moins) représentant les cases situées dans la même direction que les cases voisines (question précédente), mais à `N` cases de `(lgn,col)`. On demande, de plus, à ce que ces cases appartiennent à l'échiquier. Par exemple l'appel : `N_pas_fou_fini(3,7,3)` devra retourner la liste `[(0,4),(6,4)]` comme sur la figure 4.
5. En utilisant la fonction précédente, écrire une fonction `cases_accessibles_fou` qui admet deux paramètres entiers : `lgn` et `col` représentant une case de l'échiquier. Cette fonction devra retourner la liste de toutes les cases accessibles en un mouvement par une fou situé sur la case `(lgn,col)`. Proposition d'algorithme : commencer avec une liste vide et y ajouter les cases accessibles avec 1 pas, les cases accessibles avec 2 pas, et ainsi de suite jusqu'à ce que la liste des cases accessibles avec N pas soit vide. Sur la figure 2 les cases retournées par `cases_accessibles_fou(3,2)` ont été hachurées.

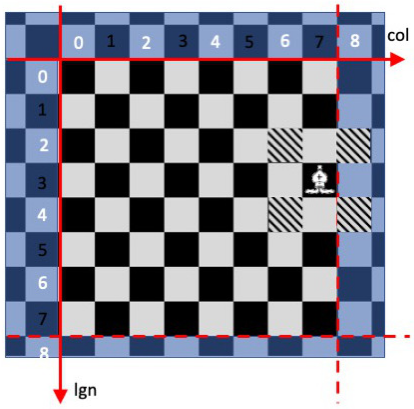

FIGURE 3: La fou est placé en (3,7). Si on suppose que l'échiquier est infini, alors la liste des voisins est la suivante : [(2,6),(2,8),(4,6),(4,8)].

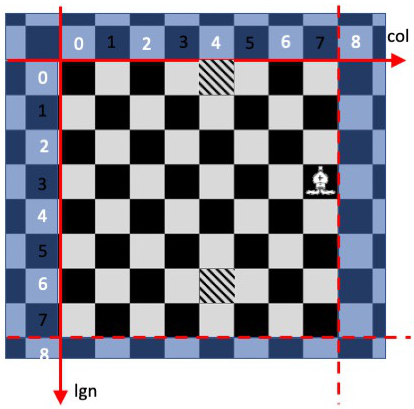

FIGURE 4: La fou est en (3,7). Si on suppose que l'échiquier est infini, alors la liste `N_pas_fou_inf(3,7,3)` est la suivante : [(0,4),(0,10),(6,4),(6,10)].

2 Dessiner des cases

Dans cette section, notre objectif est de dessiner des cases de l'échiquier, en rouge ou en noir. Assurez-vous que le fichier `graph.py` (disponible sur Moodle) est dans le même répertoire que votre programme python.
À titre de rappel, la fonction `graph.plot`, qui dessine un pixel, admet deux paramètres obligatoires représentant la ligne et la colonne de ce pixel. Elle admet aussi un paramètre optionnel de type chaîne de caractères et qui indique la couleur du pixel dessiné. Par exemple `graph.plot(20,10,"red")` colorie en rouge le pixel qui se trouve à la ligne 20, colonne 10.

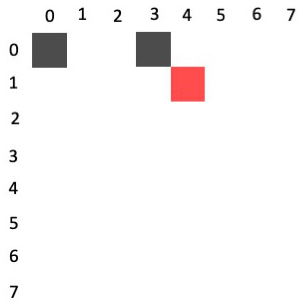

FIGURE 5: La fonction `dessineCase` permet de colorier une case. Les numéros de lignes et de colonnes ont été ajoutés à titre indicatif.

1. Écrire une fonction `dessine_case` qui admet quatre paramètres : `lgn` et `col`, des entiers compris entre 0 et 7 représentant une case, `lcase` un entier représentant la taille de chaque case en pixels, et `color`, une chaîne de caractères représentant une couleur. La valeur de retour devra être `None`. Cette fonction devra colorier en `color` la case `(lgn,col)` de l'échiquier. Par exemple les appels suivants :

   ```
   dessine_case(0,0,20,"black")
   dessine_case(0,3,20,"black")
   dessine_case(1,4,20,"red")
   ```

   produisent le résultat représenté sur la figure 5 où chaque case a une taille de 20 pixels. Indication : vous pouvez utiliser la fonction `rectangle` ou `rectangle_couleur` que vous avez écrit en TP. Pour la fonction `dessine_case`, un seul appel à la fonction `rectangle_couleur` suffit. Il vous faut simplement de calculer les arguments de `rectangle_couleur` en fonction de `lgn`, `col` et `lcase`.
2. En utilisant la fonction précédente, écrire une fonction `echiquier`, qui admet un seul paramètre `lcase` représentant la taille de chaque case en pixels. La valeur de retour de la fonction devra être `None`. Cette fonction devra afficher un échiquier comportant 8 lignes et 8 colonnes comme sur la figure 1. On demande expressément à ce que la case en haut à gauche soit noire.

3 Dessiner les cases accessibles

En utilisant la fonction `cases_accessibles_fou` écrite à la section 1 et la fonction `echiquier` et `dessine_case` écrites à la section 2, écrire une fonction `dessine_cases_accessibles` qui admet trois entiers : `lgn`, `col` et `lcase`.

- `lgn` et `col` représentent, chacun, un entier compris entre 0 et 7 et représentent une case ;
- `lcases` représente un entier strictement positif représentant la taille, en pixels, d'une case.

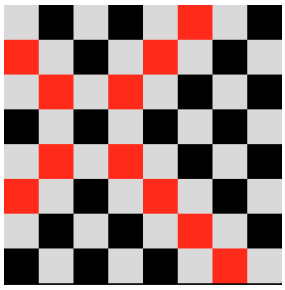

FIGURE 6: Le résultat de l'appel à `cases_accessibles_fou(3,2,20)`

Étant donnés ces paramètres, cette fonction devra afficher une image représentant un échiquier, mais où toutes les cases accessibles, en un mouvement, par un fou situé sur la ligne `lgn` et la colonne `col`, seraient dessinées

en rouge. Par exemple, l'appel à la fonction `cases_accessibles_fou(3,2,20)` produirait une image semblable à celle de la figure 6.

À la fin de cette épreuve, je vous invite à déposer le fichier contenant votre programme dans le lien *Moodle* prévu à cet effet.