DS

MPSI 2

Python

Si, au cours de l'épreuve, un candidat repère ce qui lui semble être une erreur d'énoncé, il le signale sur sa copie et poursuit sa composition en expliquant les raisons des initiatives qu'il est amené à prendre.

Trafic routier

Ce sujet concerne la conception d'un logiciel d'étude de trafic routier. On modélise le déplacement d'un ensemble de voitures sur des files à sens unique (voir Figure 1(a) et 1(b)). C'est un schéma simple qui peut permettre de comprendre l'apparition d'embouteillages et de concevoir des solutions pour fluidifier le trafic.

Notations et consignes

Le sujet comporte des questions de programmation. Le langage à utiliser est PYTHON. Lorsqu'une fonction est demandée par l'énoncé, même si vous ne proposez pas de code pour la définir, vous pouvez supposer qu'elle est définie et vous en servir dans les questions suivantes.

Soit `L` une liste,

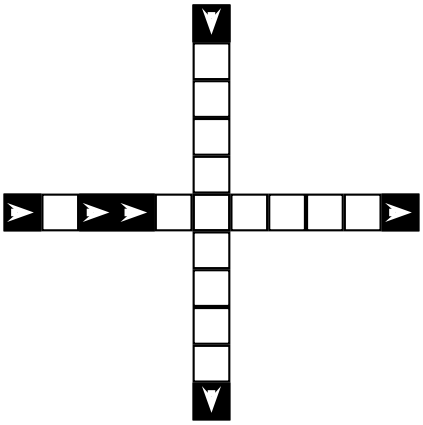
- on note `len(L)` sa longueur ;
- pour i entier, $0 \leq i < len(L)$, l'élément de la liste d'indice i est noté `L[i]` ;
- `p*L`, avec p entier, est la liste obtenue en concaténant p copies de L. Par exemple, `3*[0]` est la liste `[0,0,0]`.
- On rappelle que l'opérateur `+` permet la concaténation de 2 listes, par exemple `[1,2,3] + [4,5]` correspond à la liste `[1,2,3,4,5]`.
- `append` est la seule méthode autorisée pour modifier une liste.

Dans tout le sujet, le sclicing n'est pas autorisé.

1. Préliminaires

Dans un premier temps, on considère le cas d'une seule file, illustré par la Figure 1(a). Une file de longueur n est représentée par n cases. Une case peut contenir au plus une voiture. Les voitures présentes dans une file circulent toutes dans la même direction (sens des indices croissants, désigné par les flèches sur la Figure 1(a)) et sont indifférenciées.

a) Représentation d'une file de longueur onze comprenant quatre voitures, situées respectivement sur les cases d'indices 0, 2, 3 et 10.



(b) Configuration représentant deux files de circulation à sens unique se croisant en une case. Les voitures sont représentées par un carré noir.

Figure 1 – Files de circulation

Q1 – Une file de voitures est représentée à l'aide d'une liste de n objets `True` ou `False`. Pour n case, on prend une liste *file* de taille n où l'instruction `file[i] = True` indique qu'une voiture est sur la case d'indice i (`False` signifiant le contraire).
Quel est le type PYTHON des objets `True` et `False` ?

Q2 – Donner une ou plusieurs instructions PYTHON permettant de définir une liste A représentant la file de voitures illustrée par la Figure 1(a).

Q3 – Soit `file` une liste représentant une file de longueur n et i un entier tel que $0 \leq i < n$, définir en PYTHON la fonction `occupe(file:list, i:int)` qui renvoie `True` lorsque la case d'indice i de la file est occupée par une voiture et `False` sinon.

Q4 – Combien existe-t-il de files différentes de longueur n ? Justifier votre réponse.

Q5 – Écrire une fonction `egal(file1, file2)` retournant un booléen permettant de savoir si deux files représentées par les listes `file1` et `file2` sont égales. La fonction proposée doit utiliser un parcours séquentiel.

Q6 – Que peut-on dire de la complexité de cette fonction dans le meilleur puis dans le pire des cas ?

Q7 – Proposer deux versions de la fonction `copie(file:list)` qui renvoie une copie de la liste donnée en argument. La première version utilise la création de liste par compréhension et la seconde manière s'effectue avec la méthode `append`.

Q8 – Rédiger une fonction `comptage(file)` renvoyant le nombre voitures dans la file. Préciser la complexité de ce code.

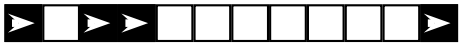
2. Déplacement de voitures dans la file

On identifie désormais une file de voitures à une liste. On considère les schémas de la Figure 2 représentant des exemples de files. Une étape de simulation pour une file consiste à déplacer les voitures de la file, à tour de rôle, en commençant par la voiture la plus à droite, d'après les règles suivantes :

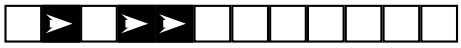
- une voiture se trouvant sur la case la plus à droite de la file sort de la file ;

- une voiture peut avancer d'une case vers la droite si elle arrive sur une case inoccupée ;
- une case libérée par une voiture devient inoccupée ;
- la case la plus à gauche peut devenir occupée ou non, selon le cas considéré.

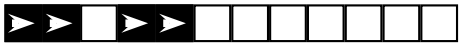
Q9 – Écrire en PYTHON la fonction `avancer(file:list, gauche:bool)` prenant en paramètres une liste de départ codant la file de voitures, un booléen indiquant si la case la plus à gauche doit devenir occupée lors de l'étape de simulation, et renvoyant la liste obtenue par une étape de simulation.
Par exemple, l'application de cette fonction à la liste illustrée par la Figure 2(a) permet d'obtenir soit la liste illustrée par la Figure 2(b) lorsque l'on considère qu'aucune voiture nouvelle n'est introduite, soit la liste illustrée par la Figure 2(c) lorsque l'on considère qu'une voiture nouvelle est introduite.



(a) Liste initiale A



(b) `B = avancer(A, False)`



(c) `C = avancer(A, True)`

Figure 2 – Étape de simulation

Q10 – Étant donnée A la liste définie à la question 2, que renvoie l'instruction suivante : `avancer(avancer(A, False), True)` ?

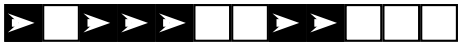
Q11 – On considère une liste `file` et m l'indice d'une case de cette liste ($0 \leq m < len(L)$). On s'intéresse à une étape partielle où seules les voitures situées sur la case d'indice m ou à droite de cette case peuvent avancer normalement, les autres voitures ne se déplaçant pas.
Définir en PYTHON la fonction `avancer_fin(file:list, indice:int)` qui réalise cette étape partielle de déplacement et renvoie le résultat dans une nouvelle liste sans modifier la file donnée en argument.
L'application de cette fonction transforme la liste ci-dessous :

m

en la file suivante :

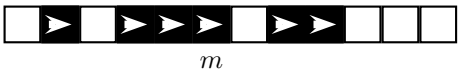
m

Q12 – Soient `file` une liste, `gauche` un booléen et m l'indice d'une case inoccupée de cette liste. On considère une étape partielle où seules les voitures situées à gauche de la case d'indice m se déplacent, les autres voitures ne se déplacent pas. Le booléen `gauche` indique si une nouvelle voiture est introduite sur la case la plus à gauche.
Par exemple la file ci-dessous :



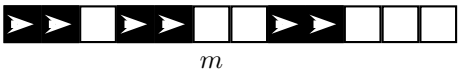
m

devient la file suivante lorsqu'aucune nouvelle voiture n'est introduite.

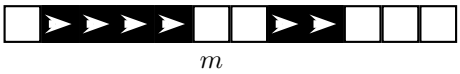


Définir en PYTHON la fonction `avancer_debut(file:list, gauche:bool, indice:int)` qui réalise cette étape partielle de déplacement et renvoie le résultat dans une nouvelle liste sans modifier celle d'origine.

Q13 – On considère une liste `file` dont la case d'indice $m > 0$ est temporairement inaccessible et bloque l'avancée des voitures. Une voiture située immédiatement à gauche de la case d'indice m ne peut pas avancer. Les voitures situées sur les cases plus à gauche peuvent avancer, à moins d'être bloquées par une case occupée, les autres voitures ne se déplacent pas. Un booléen `gauche` indique si une nouvelle voiture est introduite lorsque cela est possible.
Par exemple, la file



devient la file suivante lorsque aucune nouvelle voiture n'est introduite.



Définir en `Python` la fonction `avancer_debut_bloque(file, gauche, indice)` qui réalise cette étape partielle de déplacement et renvoie le résultat dans une nouvelle liste.

On considère dorénavant deux files `file1` (horizontale sur le schéma) et `file2` (verticale sur le schéma) de même longueur impaire se croisant en leur milieu ; on note m l'indice de la case du milieu. La file `file1` est toujours prioritaire sur la file `file2`. Les voitures ne peuvent pas quitter leur file et la case de croisement ne peut être occupée que par une seule voiture. Les voitures de la file `file2` ne peuvent accéder au croisement que si une voiture de la file `file1` ne s'apprête pas à y accéder. Une étape de simulation à deux files se déroule en deux temps.

Dans un premier temps, on déplace toutes les voitures situées sur le croisement ou après. Dans un second temps, les voitures situées avant le croisement sont déplacées en respectant la priorité. Par exemple, partant d'une configuration donnée par la Figure 3(a), les configurations successives sont données par les Figures 3(b), 3(c), 3(d), 3(e) et 3(f) en considérant qu'aucune nouvelle voiture n'est introduite.

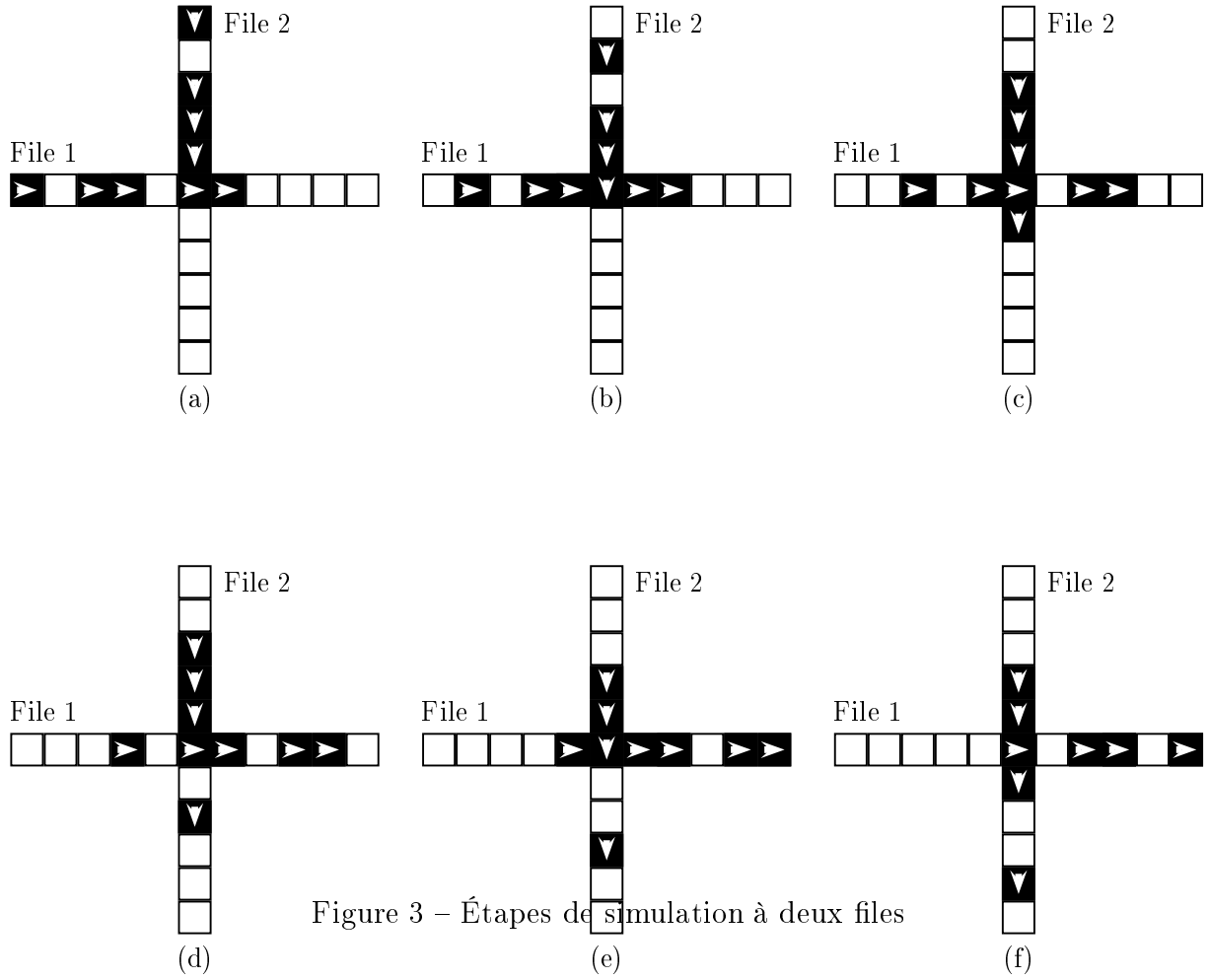


Figure 3 – Étapes de simulation à deux files

3. Simulation à deux files

L'objectif de cette partie est de définir en PYTHON l'algorithme permettant d'effectuer une étape de simulation pour ce système à deux files.

Q14 – En utilisant le langage PYTHON, définir la fonction `avancer_files(file1, gauche1:bool, file2, haut2:bool)` qui renvoie le résultat d'une étape de simulation sous la forme d'une liste de deux éléments notée `[result1, result2]` sans changer les listes données en argument. Les booléens `gauche1` et `haut2` indiquent respectivement si une nouvelle voiture est introduite dans les files `file1` et `file2`. Les listes `result1` et `result2` correspondent aux listes après déplacement.

Q15 – On considère les listes `file1 = [False, True, False, True, False]` et `file2 = [False, True, True, False, False]`. Que renvoie l'appel `avancer_files(file1, False, file2, False)` ?

Q16 – Écrire les instructions pour importer la fonction `random` depuis le module `random`.

On dispose à présent d'une fonction `random()` dont l'appel renvoie un flottant aléatoire, choisi selon une loi uniforme, sur l'intervalle $[0; 1]$.

Q17 – Proposer une fonction `voiture()` qui renvoie `True` ou `False` de façon équiprobable.

Dans toute la suite, on peut utiliser la fonction `voiture()` pour générer un booléen de façon aléatoire.

Q18 – En déduire une fonction `simulation(file1, file2, duree:int)` qui renvoie le résultat d'une simulation après un nombre `duree` d'étape sous la forme d'une liste de deux listes sans changer les listes données en argument. Les nouvelles voitures introduites à gauche verticalement ou en haut horizontalement suivent une probabilité de 1/2.

Q19 – Afin de réaliser la simulation graphique en PYTHON (figure 4), il faut pouvoir convertir les listes de booléens codant les files en liste de couleurs. On considère le blanc (pas de voiture) codé par `'w'` (pour white), le noir (une voiture sur la file horizontale) codé par `'k'` (pour black) et le rouge (une voiture sur la file verticale) codé par `'r'` (pour red).
Proposer une fonction `couleur(file:list, color:str)` renvoyant la liste des couleurs pour une file donnée. Par exemple si on prend `file = [True, False, False]` alors l'appel de `couleur(file, 'k')` renvoie `['k', 'w', 'w']`.

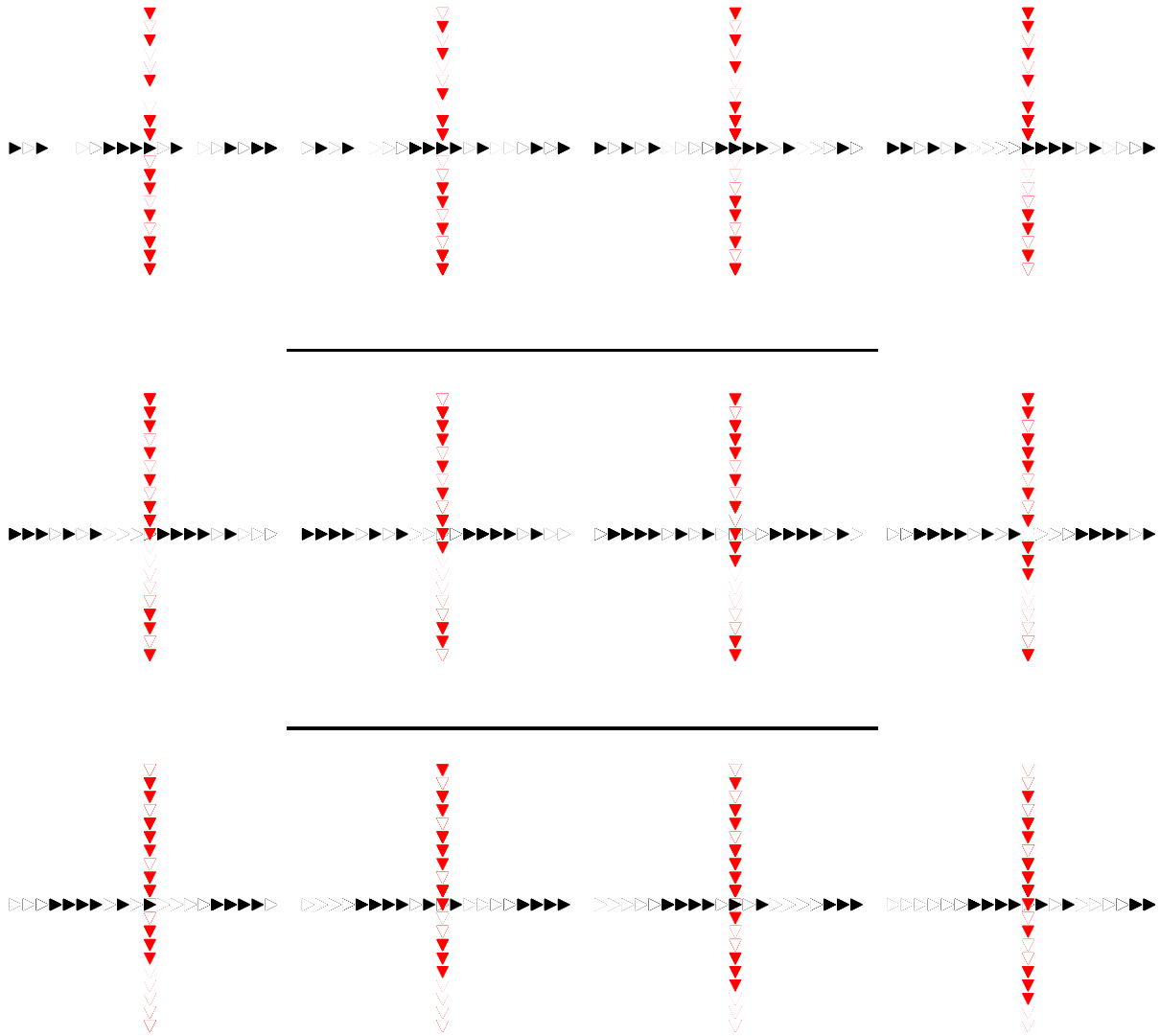


Figure 4 – Exemple de simulation PYTHON