Concours Blanc – Informatique

Jeudi 25 Avril 2024

Ce sujet est constitué de deux problèmes complètement indépendants, que vous pourrez traiter dans l'ordre de votre choix.

- La première partie du premier problème présente un algorithme de tri de listes de nombres (attention, c'est un peu subtil), avant de basculer sur des bases de données (rien de bien méchant, par contre).
- Dans la seconde partie, on met en place la méthode d'Euler pour résoudre un système d'équations différentielles (c'est très guidé, il faut compléter des codes).
- Le second problème porte quant à lui uniquement sur des manipulations de listes (assez amusant et sans grosse difficultés ... mais il faut lire attentivement l'énoncé pour bien comprendre les situations).

Voici quelques indications pour vous faire gagner du temps.

Modélisation de la propagation d'une épidémie

Q4. Reprendre et modifier la fonction `Tri` du début de la partie I.

Q8. Commencer par écrire une requête `SQL` permettant de déterminer la valeur maximale du nombre de nouveaux cas de paludisme l'année 2010.

Q11. L'énoncé parle de compléter la ligne 4, mais ce que l'on va mettre peut occuper plusieurs lignes. Ceci est valable pour les autres questions.

De plus, la fonction `f` doit renvoyer un tableau `numpy` et non une liste, afin que les opérations de la ligne 24 aient un sens.

Q13. Dans le code, on considère que $\mathrm{I}(t-\tau)$ – qui vaut aussi $\mathrm{I}(t-p\,\mathrm{d}t)$ – est un paramètre passé à la fonction f sous le nom `Itau`. Il faudra le calculer ligne 28 avant d'invoquer la fonction f.

A part cela, le calcul de f est assez proche de celui mené question 11 : on a juste modifié deux facteurs dans le membre de droite de **(1)**.

Noter que
$$\texttt{Itau}=\begin{cases}\mathrm{I}(0) & \text{si } t < p\ \mathrm{d}t\\ \mathrm{I}(t-p\,\mathrm{d}t) & \text{sinon}\end{cases}$$

si bien que
$$\texttt{Itau}=\begin{cases}\mathrm{I}(0) & \text{si } i < p\\ \mathrm{I}((i-p)\,\mathrm{d}t) & \text{sinon}\end{cases}$$

puisque $t = i\,\mathrm{d}t$ dans la boucle `for`. Enfin, utiliser la liste `XX` pour récupérer $\mathrm{I}(0)$ et $\mathrm{I}((i-p)\,\mathrm{d}t)$.

Étude de trafic routier

Noter que les cases des files sont repérées à l'aide de leurs indices qui commencent à 0, exactement comme pour les éléments d'une liste.

Q1. Les booléens sont les objets `True` et `False`.

Q5. La fonction doit renvoyer `True` si les listes sont égales et `False` sinon. On évitera les tests du genre `L1==L2`, qui ont des effets pervers ... il faut donc examiner les longueurs des listes, puis comparer leurs éléments un à un.

Q9. L'instruction `L[:]` permet de récupérer une copie d'une liste `L` que l'on souhaite éviter de modifier (c'est un bon réflexe !).

Penser à traiter à part la case d'indice `m` dans la nouvelle liste, car elle sera nécessairement inoccupée à l'issue de cette étape.

Q10. Même démarche qu'à la question 9 ... en traitant cette fois à part la case d'indice 0.

Q11. Utiliser la fonction `avancer_debut` avec un indice bien choisi, en réfléchissant à ce qui se passe sur l'exemple.

Q12. Notons que les deux files sont de longueur $2m + 1$.

Les voitures de la file L1 avancent quoiqu'il arrive puisqu'elles sont prioritaires, et celles de L2 avancent si elles le peuvent ! Il faudra utiliser les fonctions `avancer`, `avancer_debut_bloque` et `avancer_fin`.

Modélisation de la propagation d'une épidémie

L'étude de la propagation des épidémies joue un rôle important dans les politiques de santé publique. Les modèles mathématiques ont permis de comprendre pourquoi il a été possible d'éradiquer la variole à la fin des années 1970 et pourquoi il est plus difficile d'éradiquer d'autres maladies comme la poliomyélite ou la rougeole. Ils ont également permis d'expliquer l'apparition d'épidémies de grippe tous les hivers. Aujourd'hui, des modèles de plus en plus complexes et puissants sont développés pour prédire la propagation d'épidémies à l'échelle planétaire telles que le SRAS, le virus H5N1 ou le virus Ebola. Ces prédictions sont utilisées par les organisations internationales pour établir des stratégies de prévention et d'intervention.

Le travail sur ces modèles mathématiques s'articule autour de trois thèmes principaux : traitement de bases de données, simulation numérique (par plusieurs types de méthodes), identification des paramètres intervenant dans les modèles à partir de données expérimentales. Ces trois thèmes sont abordés dans le sujet. *Les parties sont indépendantes.*

Dans tout le problème, on peut utiliser une fonction traitée précédemment. On suppose que les bibliothèques `numpy` et `random` ont été importées par :

```
import numpy as np
import random as rd
```

Partie I. Tri et bases de données

Dans le but ultérieur de réaliser des études statistiques, on souhaite se doter d'une fonction tri. On se donne la fonction `tri` suivante, écrite en `Python` :

```
1  def tri(L):
2      n = len(L)
3      for i in range(1, n):
4          j = i
5          x = L[i]
6          while 0 < j and x < L[j-1]:
7              L[j] = L[j-1]
8              j = j-1
9          L[j] = x
```

❑ **Q1** – Lors de l'appel `tri(L)` lorsque `L` est la liste `[5, 2, 3, 1, 4]`, donner le contenu de la liste `L` à la fin de chaque itération de la boucle `for`. On justifiera soigneusement les réponses.

❑ **Q2** – Soit `L` une liste non vide de nombres flottants. On admet que « la liste `L[0:i+1]` (avec la convention `Python`) est triée par ordre croissant à l'issue de l'itération `i` » est un invariant de boucle. En déduire que `tri(L)` trie la liste `L`.

❑ **Q3** – Évaluer la complexité dans le pire des cas de l'appel `tri(L)` en fonction du nombre `n` d'éléments de `L`.

On souhaite, partant d'une liste constituée de couples (chaîne, entier), trier la liste par ordre croissant de l'entier associé suivant le fonctionnement suivant :

```
>>> L = [['Bresil', 76], ['Kenya', 26017], ['Ouganda', 8431]]
>>> tri_chaine(L)
>>> L
[['Bresil', 76], ['Ouganda', 8431], ['Kenya', 26017]]
```

❑ **Q4** – Écrire en `Python` une fonction `tri_chaine` réalisant cette opération.

Pour suivre la propagation des épidémies, de nombreuses données sont recueillies par les institutions internationales comme l'O.M.S. Par exemple, pour le paludisme, on dispose de deux tables :

– la table `palu` recense le nombre de nouveaux cas confirmés et le nombre de décès liés au paludisme ; certaines lignes de cette table sont données en exemple (on précise que `iso` est un identifiant unique pour chaque pays) :

nom	iso	annee	cas	deces
Bresil	BR	2009	309 316	85
Bresil	BR	2010	334 667	76
Kenya	KE	2010	898 531	26 017
Mali	ML	2011	307 035	2 128
Ouganda	UG	2010	1 581 160	8 431
...				

– la table `demographie` recense la population totale de chaque pays ; certaines lignes de cette table sont données en exemple :

pays	periode	pop
BR	2009	193 020 000
BR	2010	194 946 000
KE	2010	40 909 000
ML	2011	14 417 000
UG	2010	33 987 000
...		

❑ **Q5** – Au vu des données présentées dans la table `palu`, parmi les attributs `nom`, `iso` et `annee`, quels attributs peuvent servir de clé primaire ? Un couple d'attributs pourrait-il servir de clé ? Lequel ?

❑ **Q6** – Écrire une requête en langage SQL qui récupère depuis la table `palu` toutes les données de l'année 2010 qui correspondent à des pays où le nombre de décès dus au paludisme est supérieur ou égal à 1 000.

On appelle *taux d'incidence d'une épidémie* le rapport du nombre de nouveaux cas pendant une période donnée sur la taille de la population-cible pendant la même période. Il s'exprime généralement en « nombre de nouveaux cas pour 100 000 personnes par année ». Il s'agit d'un des critères les plus importants pour évaluer la fréquence et la vitesse d'apparition d'une épidémie.

❑ **Q7** – Écrire une requête en langage SQL qui détermine le taux d'incidence du paludisme en 2011 pour les différents pays de la table `palu`.

❑ **Q8** – Écrire une requête en langage SQL permettant de déterminer le nom du pays ayant eu le plus grand nombre de nouveaux cas de paludisme en 2010 (on pourra supposer qu'il n'y a pas de pays *ex æquo* pour les nombres de cas).

On considère la requête R qui s'écrit dans le langage de l'algèbre relationnelle :

$$R = \pi_{\texttt{nom},\texttt{deces}}\left(\sigma_{\texttt{annee}=2010}(\texttt{palu})\right)$$

On suppose que le résultat de cette requête a été converti en une liste `Python` stockée dans la variable `deces2010` et constituée de couples (chaîne, entier).

❑ **Q9** – Quelle instruction peut-on écrire en `Python` pour trier la liste `deces2010` par ordre croissant du nombre de décès dus au paludisme en 2010 ?

Comment faire cela directement en SQL ?

Partie II. Modèle à compartiments

On s'intéresse ici à une première méthode de simulation numérique.

Les modèles compartimentaux sont des modèles déterministes où la population est divisée en plusieurs catégories selon leurs caractéristiques et leur état par rapport à la maladie. On considère dans cette partie un modèle à quatre compartiments disjoints : sains (S, "susceptible"), infectés (I, "infected"), rétablis (R, "recovered", ils sont immunisés) et décédés (D, "dead"). Le changement d'état des individus est gouverné par un système d'équations différentielles obtenues en supposant que le nombre d'individus nouvellement infectés (c'est-à-dire le nombre de ceux qui quittent le compartiment S) pendant un intervalle de temps donné est proportionnel au produit du nombre d'individus infectés avec le nombre d'individus sains.

En notant $S(t)$, $I(t)$, $R(t)$ et $D(t)$ la fraction de la population appartenant à chacune des quatre catégories à l'instant t, on obtient le système :

$$\left.\begin{aligned} \frac{d}{dt}S(t) &= -r\,S(t)I(t) \\ \frac{d}{dt}I(t) &= r\,S(t)I(t) - (a+b)\,I(t) \\ \frac{d}{dt}R(t) &= a\,I(t) \\ \frac{d}{dt}D(t) &= b\,I(t) \end{aligned}\right\} \tag{1}$$

avec r le taux de contagion, a le taux de guérison et b le taux de mortalité. On suppose qu'à l'instant initial $t = 0$, on a $S(0) = 0{,}95$, $I(0) = 0{,}05$ et $R(0) = D(0) = 0$.

❑ **Q10** – Préciser un vecteur X et une fonction f (en donnant son domaine de définition et son expression) tels que le système différentiel (1) s'écrive sous la forme

$$\frac{d}{dt}X = f(X).$$

❑ **Q11** – Compléter la ligne 4 du code suivant (on précise que `np.array` permet de créer un tableau `numpy` à partir d'une liste donnant ainsi la possibilité d'utiliser les opérateurs algébriques).

```
1  def f(X):
2      """ Fonction definissant l'equation differentielle """
3      global r, a, b
4      # a completer
5
6  # Parametres
7  tmax = 25.
8  r = 1.
9  a = 0.4
10 b = 0.1
11 X0 = np.array([0.95, 0.05, 0., 0.])
12
13 N = 250
14 dt = tmax/N
15
16 t = 0
17 X = X0
18 tt = [t]
19 XX = [X]
```

```
20
21  # Methode d'Euler
22  for i in range(N):
23      t = t + dt
24      X = X + dt * f(X)
25      tt.append(t)
26      XX.append(X)
```

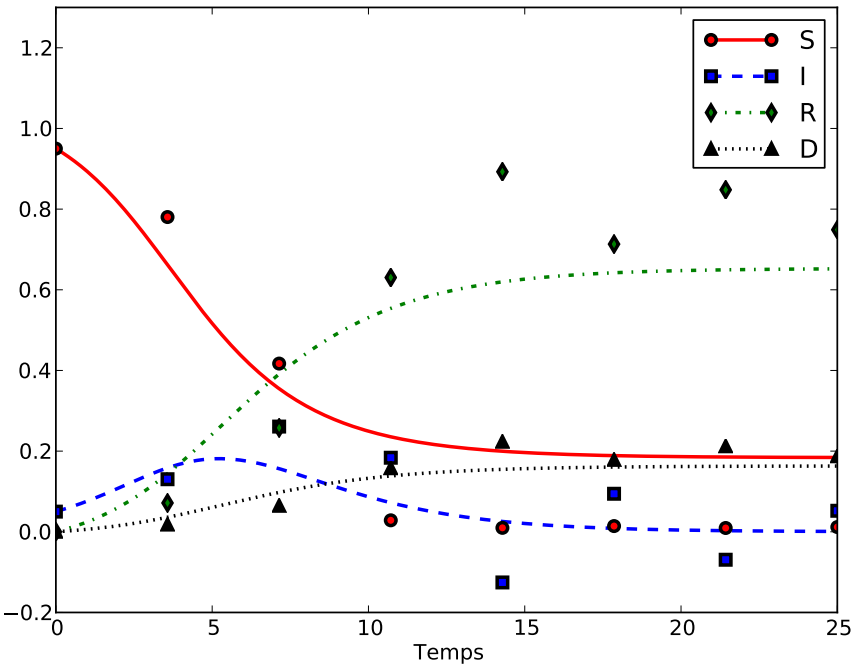


FIGURE 1 – Représentation graphique des quatre catégories S, I, R et D en fonction du temps pour $N = 7$ (points) et $N = 250$ (courbes).

❑ **Q12** – La figure 1 représente les quatre catégories en fonction du temps obtenues en effectuant deux simulations : la première avec $N = 7$ correspond aux points (cercle, carré, losange, triangle) et la seconde avec $N = 250$ correspond aux courbes. Expliquer la différence entre ces deux simulations. Quelle simulation a nécessité le temps de calcul le plus long ?

En pratique, de nombreuses maladies possèdent une phase d'incubation pendant laquelle l'individu est porteur de la maladie mais ne possède pas de symptômes et n'est pas contagieux. On peut prendre en compte cette phase d'incubation à l'aide du système à retard suivant :

$$\begin{cases} \dfrac{d}{dt}S(t) = -r\,S(t)I(t-\tau) \\ \dfrac{d}{dt}I(t) = r\,S(t)I(t-\tau) - (a+b)\,I(t) \\ \dfrac{d}{dt}R(t) = a\,I(t) \\ \dfrac{d}{dt}D(t) = b\,I(t) \end{cases}$$

où τ est le temps d'incubation. On suppose alors que pour tout $t \in [-\tau, 0]$, $S(t) = 0{,}95$, $I(t) = 0{,}05$ et $R(t) = D(t) = 0$.

En notant $tmax$ la durée des mesures et N un entier donnant le nombre de pas, on définit le pas de temps $dt = tmax/N$. On suppose que $\tau = p \times dt$ où p est un entier ; ainsi p est le nombre de pas de retard.

Pour résoudre numériquement ce système d'équations différentielles à retard (avec `tmax = 25`, `N = 250` et `p = 50`), on a écrit le code suivant :

```
1  def f(X, Itau):
2      """
3      Fonction definissant l'equation differentielle
4      Itau est la valeur de I(t - p * dt)
5      """
6      global r, a, b
7      # a completer
8
9  # Parametres
10 r = 1.
11 a = 0.4
12 b = 0.1
13 X0 = np.array([0.95, 0.05, 0., 0.])
14
15 tmax = 25.
16 N = 250
17 dt = tmax/N
18 p = 50
19
20 t = 0
21 X = X0
22 tt = [t]
23 XX = [X]
24
25 # Methode d'Euler
26 for i in range(N):
27     t = t + dt
28     # a completer
29     tt.append(t)
30     XX.append(X)
```

❑ **Q13** – Compléter les lignes 7 et 28 du code précédent (utiliser autant de lignes que nécessaire).

Étude de 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.

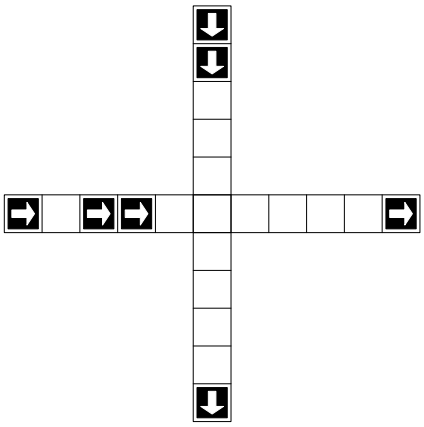
Le sujet comporte des questions de programmation. Le langage à utiliser est `Python`.

Notations

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 $L[i]$;
- pour i et j entiers, $0 \leq i < j \leq len(L)$, $L[i : j]$ est la sous-liste composée des éléments $L[i]$, …, $L[j-1]$;
- $p * L$, avec p entier, est la liste obtenue en concaténant p copies de L. Par exemple, $3 * [0]$ est la liste [0, 0, 0].

(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

Partie I. 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.

❑ **Q1** – Expliquer comment représenter une file de voitures à l'aide d'une liste de booléens.

❑ **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 L une liste représentant une file de longueur n et i un entier tel que $0 \leq i < n$. Définir en `Python` la fonction `occupe(L, i)` 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(L1, L2)` retournant un booléen permettant de savoir si deux listes $L1$ et $L2$ sont égales.

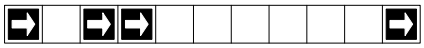
❑ **Q6** – Que peut-on dire de la complexité de cette fonction ?

Partie II. 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.

On suppose avoir écrit en `Python` la fonction `avancer` prenant en paramètres une liste de départ, 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.



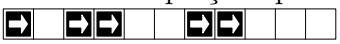
(a) Liste initiale A

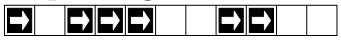
(b) $B =$ `avancer`$(A,$ `False`$)$

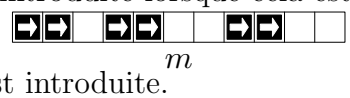
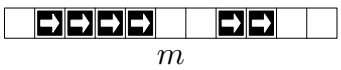
(c) $C =$ `avancer`$(A,$ `True`$)$

FIGURE 2 – Étape de simulation

❑ **Q8** – Étant donnée A la liste définie à la question 2, que renvoie `avancer(avancer(`A`, False), True)` ?

❑ **Q9** – On considère L une liste 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.
Par exemple, la file  devient 
m m
Définir en `Python` la fonction `avancer_fin`$(L,\ m)$ qui réalise cette étape partielle de déplacement et renvoie le résultat dans une nouvelle liste sans modifier L.

❑ **Q10** – Soient L une liste, b 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 b indique si une nouvelle voiture est introduite sur la case la plus à gauche.
Par exemple, la file  devient  lorsque aucune
m m
nouvelle voiture n'est introduite.
Définir en `Python` la fonction `avancer_debut`$(L,\ b,\ m)$ qui réalise cette étape partielle de déplacement et renvoie le résultat dans une nouvelle liste sans modifier L.

❑ **Q11** – On considère une liste L 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 b indique si une nouvelle voiture est introduite lorsque cela est possible.
Par exemple, la file  devient  lorsque aucune nouvelle voiture n'est introduite.

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

On considère dorénavant deux files $L1$ et $L2$ de même longueur impaire se croisant en leur milieu ; on note m l'indice de la case du milieu. La file $L1$ est toujours prioritaire sur la file $L2$. 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 $L2$ ne peuvent accéder au croisement que si une voiture de la file $L1$ 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*.

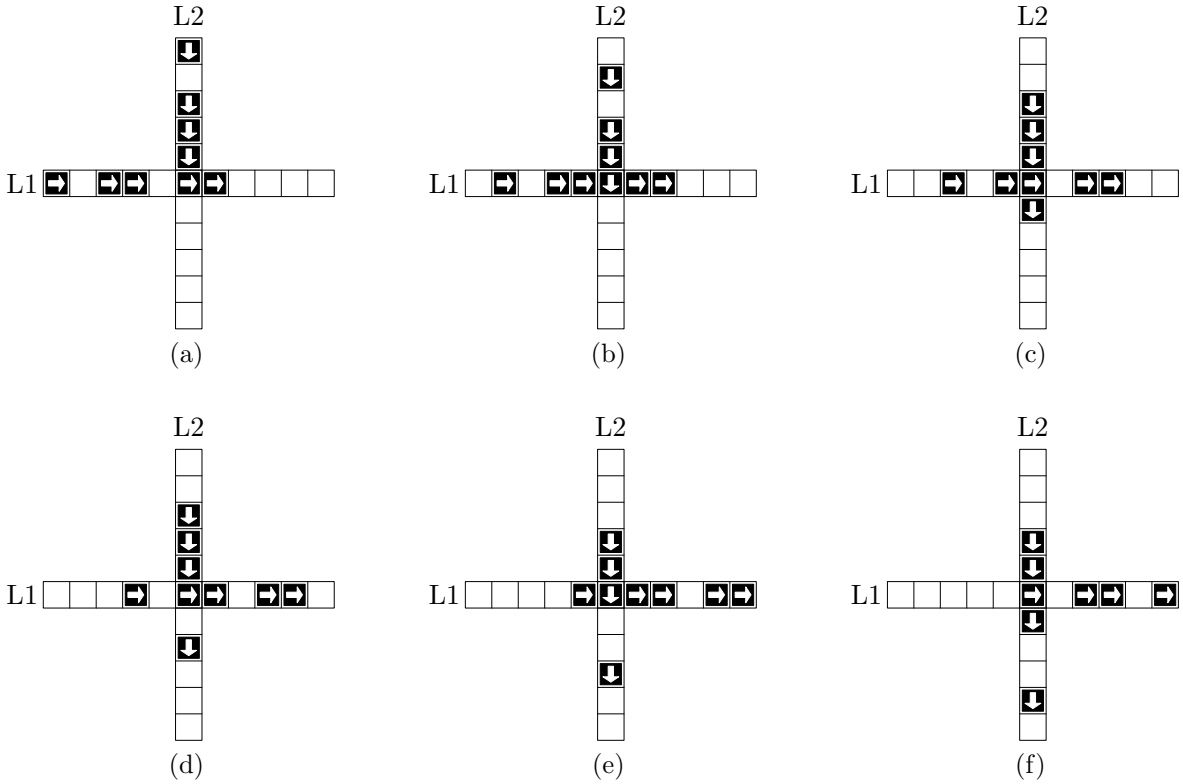


FIGURE 3 – Étapes de simulation à deux files

Partie III. Une étape de 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.

❑ **Q12** – En utilisant le langage `Python`, définir la fonction `avancer_files`($L1$, $b1$, $L2$, $b2$) qui renvoie le résultat d'une étape de simulation sous la forme d'une liste de deux éléments notée [$R1$, $R2$] sans changer les listes $L1$ et $L2$. Les booléens $b1$ et $b2$ indiquent respectivement si une nouvelle voiture est introduite dans les files $L1$ et $L2$. Les listes $R1$ et $R2$ correspondent aux listes après déplacement.

❑ **Q13** – On considère les listes

$$D = [\texttt{ False}, \texttt{True}, \texttt{False}, \texttt{True}, \texttt{False}], \qquad E = [\texttt{False}, \texttt{True}, \texttt{True}, \texttt{False}, \texttt{False}]$$

Que renvoie l'appel avancer_files(D, `False`, E, `False`) ?

Partie IV. Transitions

❑ **Q14** – En considérant que de nouvelles voitures peuvent être introduites sur les premières cases des files lors d'une étape de simulation, décrire une situation où une voiture de la file $L2$ serait indéfiniment bloquée.

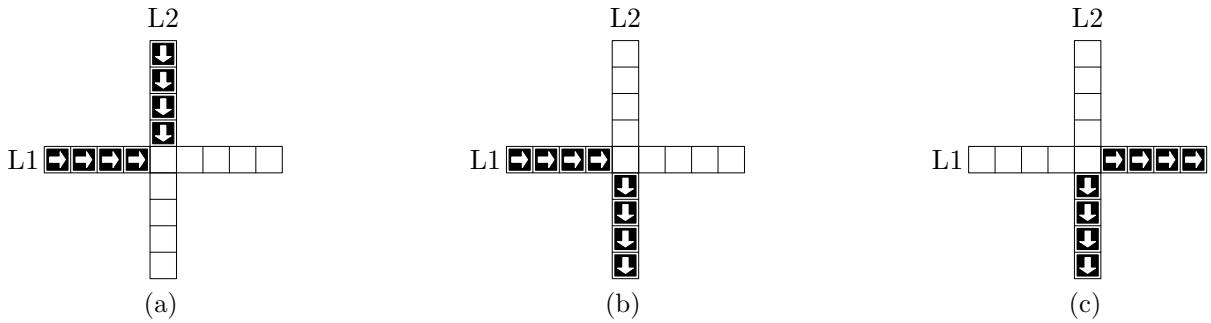


FIGURE 4 – Étude de configurations

❑ **Q15** – Étant données les configurations illustrées par la Figure 4, combien d'étapes sont nécessaires (on demande le nombre minimum) pour passer de la configuration 4(a) à la configuration 4(b) ? Justifier votre réponse.

❑ **Q16** – Peut-on passer de la configuration 4(a) à la configuration 4(c) ? Justifier votre réponse.