I. Préliminaires

Définition :
Une file est une structure de données de type FIFO (First In First Out) : le premier qui est rentré sera le premier à sortir.

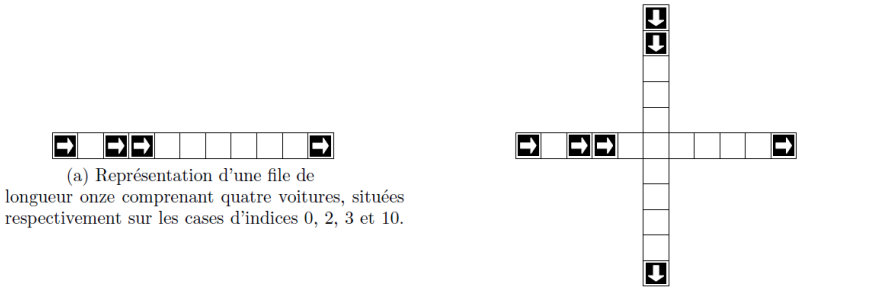
En python, les listes peuvent être utilisées comme des files avec les méthodes suivantes :

— ajout d'un élément à la fin d'une liste : `.append()`.

— suppression du dernier élément et renvoie de sa valeur : `.pop())`

— suppression du premier élément et renvoie de sa valeur : `.pop(0))`

On s'interesse ici au déplacement de voitures sur une route. 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

Q 1 - Donner une ou plusieurs instructions Python permettant de définir une liste A représentant la file de voitures illustrée par la Figure 1.(a) à l'aide d'une liste de booléens.

Q 2 - 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.

Q 3 - Écrire une fonction `egal(L1, L2)` retournant un booléen permettant de savoir si deux listes `L1` et `L2` sont égales.

Q 4 - On cherche à créer une liste de taille n avec p voitures réparties aléatoirement. Définir une fonction `init_file` de paramètres `n` et `p` réalisant une telle répartition.

Q 5 - Pour initialiser la liste, on peut aussi envisager une situation aléatoire où l'existence d'une voiture à une position donnée est conditionnée par une probabilité P. Proposer une fonction `ini_file_V2` de paramètres `n` et `P` réalisant une répartition de voiture, si le paramètre `P` vaut 1 alors la liste est entièrement remplie.

II. Déplacement de voitures dans la file

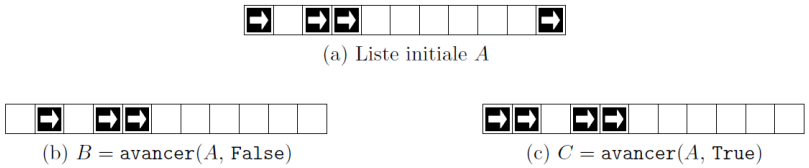
On considère les schémas de la Figure 2 représentant des exemples de files. Une étape de simulation pour une file consiste à déplacer les voitures de la file, à tour de rôle, en commençant par la voiture la plus à droite, d'après les règles suivantes :

— 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.



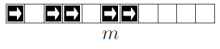
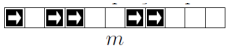
(a) Liste initiale A

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

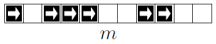
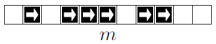
(c) C = `avancer`(A, `True`)

FIGURE 2 – Etape de simulation

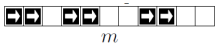
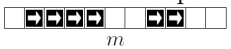
Q 6 - Écrire en Python la fonction `avancer` prenant en paramètres une liste de départ, un booléen indiquant si la case la plus à gauche doit devenir occupée lors de l'étape de simulation, et renvoyant la liste obtenue par une étape de simulation. Par exemple, l'application de cette fonction à la liste illustrée par la Figure 2 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.

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 

Q 7 - 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`.

Q 8 - 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 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`.

Q 9 - 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.

III. Atteignabilité

Certaines configurations peuvent être néfastes pour la fluidité du trafic. Une fois ces configurations identifiées, il est intéressant de savoir si elles peuvent apparaître. Lorsque c'est le cas, on dit qu'une telle configuration est atteignable. Pour savoir si une configuration est atteignable à partir d'une configuration initiale, on a écrit le code incomplet donné en annexe. Le langage Python sait comparer deux listes de booléens à l'aide de l'opérateur usuel ¡, on peut ainsi utiliser la méthode sort pour trier une liste de listes de booléens.

Q 10 - écrire en langage Python une fonction `elim_double(L)` non récursive, de complexité linéaire en la taille de `L`, qui élimine les éléments apparaissant plusieurs fois dans une liste triée `L` et renvoie la liste triée obtenue. Par exemple `elim_double([1, 1, 3, 3, 3, 7])` doit renvoyer la liste `[1, 3, 7]`.

Q 11 - La fonction `recherche` ci-dessous permet d'établir si la configuration correspondant à but est atteignable en partant de l'état `init`. Préciser le type de retour de la fonction `recherche`, le type des variables `but` et `espace`, ainsi que le type de retour de la fonction `successeurs`.

```
1  def recherche(but, init):
2      espace = [init]
3      stop = False
4      while not stop:
5          ancien = espace
6          espace = espace + successeurs(espace)
7          espace.sort() # tri par ordre croissant
8          espace = elim_double(espace)
9          stop = egal(ancien,espace)
10         if but in espace:
11             return True
12     return False
```

```
1  def successeurs(L):
2      res = []
3      for x in L:
4          L1 = x[0]
5          L2 = x[1]
6          res.append( avancer_files(L1, False, L2, False) )
7          res.append( avancer_files(L1, False, L2, True) )
8          res.append( avancer_files(L1, True, L2, False) )
9          res.append( avancer_files(L1, True, L2, True) )
10     return res
```

Q 12 - Rédiger une fonction `recherche_dicho( x ,L )` qui détermine par dichotomie la position de la valeur `x` dans la liste `L` triée. Cette fonction renvoie l'indice de `x` dans `L`.

Q 13 - Afin de comparer plus efficacement les files représentées par des listes de booléens on remarque que ces listes représentent un codage binaire où `True` correspond à 1 et `False` à 0. écrire la fonction `versEntier(L)` prenant une liste de booléens en paramètre et renvoyant l'entier correspondant. Par exemple, l'appel `versEntier([True, False, False])` renverra 4.

Q 14 - On veut écrire la fonction inverse de versEntier, transformant un entier en une liste de booléens. Que doit être au minimum la valeur de `taille` pour que le codage obtenu soit satisfaisant ? On suppose que la valeur de taille est suffisante. Quelle condition booléenne faut-il écrire en ligne 4 du code ci-dessous ?

```python
def versFile(n, taille):
    res = taille * [False]
    i = taille - 1
    while ...:
        if (n % 2) != 0: # res[i] = True
        n = n // 2 # // est la division entière
        i = i - 1
    return res
```

Q 15 - Montrer qu'un appel à la fonction recherche de l'annexe se termine toujours.

Opérations et fonctions Python disponibles

Fonctions

— `random()` renvoie un nombre flottant tiré aléatoirement dans [0,1[suivant une distribution uniforme

— `randint(0,n)` renvoie un entier aléatoire en 0 et n inclus

Opérations sur les listes

— `len(u)` donne le nombre d'éléments de la liste `u` :

— `u + v` construit une liste constituée de la concaténation des listes `u` et `v` :

— `u.append(e)` ajoute l'élément `e` à la fin de la liste `u` (similaire à `u = u + [e]`)

— `del(u[i])` supprime de la liste `u` son élément d'indice `i`

— `u.insert(i, e)` insère l'élément `e` à la position d'indice `i` dans la liste `u` (en décalant les éléments suivants) ; si `i >= len(u)`, `e` est ajouté en fin de liste

— `u[i], u[j] = u[j], u[i]` permute les éléments d'indice `i` et `j` dans la liste `u`