Université de Lille 1 SCIENCES ET TECHNOLOGIES

ufr d'IEEA
Formations en
Informatique de
Lille 1

L1 MASS/SESI/PEIP - 2015–2016

——

Algorithmes et Programmation 1

——

DS2 - 17/05/2016

——

durée 2h - documents non autorisés
(sauf memento Python3)

Ce devoir contient trois exercices indépendants.

Sauf mention explicite du contraire, toutes les fonctions que vous réaliserez devront comporter une documentation avec contraintes et exemples d'utilisation.

Le barême est donné à titre indicatif.

Exercice 1 *Recherche dichotomique* (4 pts)

Question 1 Sous forme d'une fonction PYTHON à valeurs booléennes, rappelez l'algorithme de recherche dichotomique d'un nombre dans une liste de nombres qui renvoie `True` si l'élément est présent dans cette liste, et `False` sinon. Il n'est pas demandé de documentation pour cette fonction mais vous préciserez les contraintes d'utilisation.

Soit $\ell = [0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22]$ une liste d'entiers.

Question 2 Donnez la suite des tranches de cette liste parcourue lors de la recherche dichotomique de $x = 11$ dans la liste ℓ. Combien de comparaisons d'éléments sont-elles effectuées ?

Question 3 Même question avec $x = 12$.

Exercice 2 *Transformation de Burrows-Wheeler* (8 pts)

La *transformation de Burrows-Wheeler* est une technique utilisée en compression de données. Il ne s'agit pas à proprement parler d'un algorithme de compression, car aucune réduction de taille n'est effectuée, mais d'une préparation à la compression.

Voici une présentation de cette transformation illustrée avec la chaîne de caractères `TEXTE` en exemple.

Étape 1 : On construit la liste de toutes les permutations circulaires successives de la chaîne à transformer. Avec notre exemple qui est une chaîne de longueur 5, cela donne la liste des cinq chaînes de longueur 5 :

```
TEXTE
ETEXT
TETEX
XTETE
EXTET
```

Étape 2 : Puis on trie les chaînes de cette liste par ordre alphabétique. Avec notre exemple, cela donne :

```
ETEXT
EXTET
TETEX
TEXTE
XTETE
```

Étape 3 : On construit la chaîne constituée des derniers caractères de chacune des chaînes de la liste triée. Avec notre exemple, cela donne `TTXEE`.

Étape 4 : On repère le numéro d'emplacement de la chaîne de départ dans la liste triée. Avec notre exemple, la chaîne `TEXTE` se trouve en quatrième position.

Tout ceci étant réalisé, la transformée de Burrows-Wheeler de la chaîne est le couple (n, s), n étant le numéro de ligne trouvé à la quatrième étape, et s la chaîne construite à la troisième. Avec notre exemple, ce couple est $(4, \texttt{TTXEE})$. En notant `bwt` cette transformation, on peut aussi écrire

$$\texttt{bwt}(\texttt{TEXTE}) = (4, \texttt{TTXEE}).$$

Question 1 Calculez la transformée de la chaîne `CHAT`. Vous montrerez les résultats intermédiaires obtenus aux étapes 1 et 2.

Les deux questions qui suivent ont pour but la programmation de cette transformation.

Question 2 Réalisez une fonction nommée `permute_circulaire` paramétrée par une chaîne de caractères `s` et un entier `k` qui doit être compris entre 0 et la longueur de la chaîne, qui renvoie la chaîne obtenue en permutant circulairement de `k` positions vers la droite tous les caractères de la chaîne `s`.

```
>>> permute_circulaire ('TEXTE', 0)
'TEXTE'
>>> permute_circulaire ('TEXTE', 1)
'ETEXT'
>>> permute_circulaire ('TEXTE', 3)
'XTETE'
>>> permute_circulaire ('TEXTE', 5)
'TEXTE'
```

Question 3 Réalisez une fonction nommée `bwt` paramétrée par une chaîne qui renvoie la transformée de Burrows-Wheeler de cette chaîne sous forme d'un couple.

```
>>> bwt ('TEXTE')
(4, 'TTXEE')
```

Nous allons maintenant étudier la transformation inverse qui, à partir d'un couple (n, s) supposé être la transformée de Burrows-Wheeler d'une certaine chaîne, permet de retrouver cette chaîne.

Étape 1 : La première étape consiste à partir des caractères de s de reconstruire la liste triée de toutes les permutations circulaires de la chaîne à retrouver. Pour cela, en partant d'une liste contenant autant de chaînes vides que de caractères dans s, on effectue une alternance de *collages* et de tris comme le montre le schéma suivant. Le nombre de collages/tris est égal à la longueur de la chaîne s.

```
''          T        E          TE       ET          TET       ETE          TETE
'' collage  T   tri  E collage  TE  tri  EX collage  TEX  tri  EXT collage  TEXT
'' -------> X  ----> T -------> XT ----> TE -------> XTE ----> TET -------> XTET
''          E        T          ET       TE          ETE       TEX          ETEX
''          E        X          EX       XT          EXT       XTE          EXTE

      ETEX          TETEX       ETEXT
 tri  EXTE collage  TEXTE  tri  EXTET
----> TETE -------> XTETE ----> TETEX
      TEXT          ETEXT       TEXTE
      XTET          EXTET       XTETE
```

Une étape de collage consiste à ajouter en tête de chaque chaîne de la liste un caractère de la chaîne s. (Remarquez que cette situation finale est la même que celle atteinte en fin de deuxième étape de la transformation directe.)

Étape 2 : le nombre n donne la position de la chaîne à retrouver dans la liste. Avec $n = 4$ on obtient la chaîne `TEXTE`.

Question 4 Connaissant `bwt`$(s) = (5, \texttt{SISASIE})$, retrouvez la chaîne s. Mettez en évidence les différentes étapes de cette construction.

Les deux dernières questions ont pour objectif de programmer la transformation de Burrows-Wheeler inverse.

Question 5 Réalisez une fonction nommée `collage` paramétrée par une chaîne de caractères `s` et une liste de chaînes, `l`, supposée de même longueur que la chaîne `s`, qui renvoie la liste des chaînes obtenues en concaténant chacun des caractères de `s` à chacune des chaînes de `l`.

```
>>> collage ('TTXEE', ['E', 'E', 'T', 'T', 'X'])
['TE', 'TE', 'XT', 'ET', 'EX']
>>> collage ('TTXEE', ['ET', 'EX', 'TE', 'TE', 'XT'])
['TET', 'TEX', 'XTE', 'ETE', 'EXT']
```

Question 6 Réalisez une fonction nommée `inv_bwt` paramétrée par un couple `(n,s)` supposé être la transformée de Burrows-Wheeler d'une certaine chaîne et qui renvoie cette chaîne.

```
>>> inv_bwt ((4, 'TTXEE'))
'TEXTE'
```

Exercice 3 *Autour du* Sudoku (8 pts)

Les documentations de fonction de cet exercice ne sont pas exigées.

Le jeu du Sudoku est classiquement constitué d'une grille carrée de taille 9×9 dont certaines cases sont remplies de nombres compris entre 1 et 9, les autres étant vides. Le but du jeu consiste à remplir les cases vides avec les nombres de 1 à 9 de telle sorte que

— chaque ligne de la grille contienne tous les nombres de 1 à 9 ;
— ainsi que chaque colonne ;
— et de même pour chacun des neuf carrés 3×3.

La figure 1 montre un problème de Sudoku, accompagné de sa solution.

		1		2		7		
	5						9	
			4					
	8				5			
	9							
				6				2
		2						
		6						5
					9		8	3

3	6	1	9	2	8	7	5	4
4	5	8	6	3	7	2	9	1
7	2	9	4	5	1	8	3	6
2	8	4	1	9	5	3	6	7
6	9	3	7	4	2	5	1	8
5	1	7	8	6	3	9	4	2
8	3	2	5	1	6	4	7	9
9	7	6	3	8	4	1	2	5
1	4	5	2	7	9	6	8	3

Figure 1 – Une grille de Sudoku et sa solution

Dans cet exercice, une grille de Sudoku est représentée par un dictionnaire dont les clés sont les coordonnées des cases de la grille et les valeurs associées le chiffre (caractère) qu'elle contient, le caractère `'0'`

désignant une case vide. Ainsi en supposant que la variable `sudoku` représente la grille de gauche de la figure 1, on doit avoir

```
>>> type (sudoku)
<class 'dict'>
>>> len (sudoku)
81
>>> {sudoku[k] for k in sudoku}.issubset (set ('0123456789'))
True
>>> sudoku[(2,0)]
'1'
```

Commençons par la réalisation d'une fonction qui renvoie une grille de SUDOKU lue dans un fichier texte constituée de 9 lignes de 9 chiffres. Voici un exemple de contenu d'un tel fichier (ce fichier décrit la grille de gauche de la figure 1) :

```
001020700
050000090
000400000
080005000
090000000
000060002
002000000
006000005
000009083
```

Question 1 Citez trois causes qui peuvent faire que la lecture d'une grille déclenche une exception.

Question 2 Réalisez la fonction de lecture `lire_sudoku` qui renvoie la grille de SUDOKU décrite dans le fichier dont le nom est passé en paramètre. Votre fonction doit déclencher une exception pour chacune des causes trouvées dans la question précédente, et vous préciserez quelle est l'exception déclenchée.

Passons maintenant à l'enregistrement d'une grille de SUDOKU dans un fichier texte.

Question 3 Réalisez une fonction nommée `ecrire_sudoku` paramétrée par un SUDOKU et un nom de fichier, qui écrit la grille sous le format texte présenté ci-dessus.

On suppose dans la question suivante qu'une procédure de résolution de problème de SUDOKU est réalisée : `resoudre_sudoku`. Cette procédure est paramétrée par un SUDOKU et à l'issue d'un appel à cette procédure, le SUDOKU passé en paramètre est résolu (il est donc modifié). (On fait ici l'hypothèse que toute grille de SUDOKU proposée à cette procédure possède une solution.)

On suppose aussi que dans le répertoire courant de travail, il y a 100 problèmes de SUDOKU décrits dans des fichiers nommés `grille-1.txt`, `grille-2.txt`, ... `grille-100.txt`, tous ces fichiers étant conformes à la description donnée plus haut.

Question 4 Écrivez une fonction non paramétrée qui permet de produire 100 fichiers nommés `grille-1-sol.txt`, `grille-2-sol.txt`, ..., `grille-100-sol.txt` contenant la solution des 100 problèmes de SUDOKU. Ces fichiers ont le même format que les fichiers initiaux.

Supposons maintenant qu'il peut arriver qu'un ou plusieurs des fichiers de description de grilles ne soient pas conformes.

Question 5 Modifiez la fonction de la question précédente afin qu'une exception déclenchée par la lecture d'un ou plusieurs des fichiers n'interrompe pas brutalement son exécution, et que le traitement des fichiers suivants puisse être effectué. Le nom des fichiers non conformes est imprimé sur la sortie standard.
