Université de Lille 1 SCIENCES ET TECHNOLOGIES

ufr d'IEEA
Formations en
Informatique de
Lille 1

L1 MASS/SESI/PEIP - 2016–2017

———

Algorithmes et Programmation 1

———

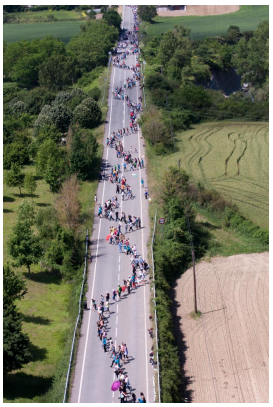
DS2 - 09/05/2017

———

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

Ce devoir contient deux exercices indépendants.
Indiquez sur la copie votre parcours et le numéro de votre groupe.
Sauf mention explicite du contraire, il n'est pas demandé de documentation (docstring) pour les fonctions que vous réaliserez.

Exercice 1 *La chaîne humaine*



Trois millions de personnes, ayant des noms tous distincts, se donnent la main pour former une chaîne humaine reliant New York à San Francisco. Chacune d'elles écrit son nom sur une feuille de papier ainsi que celui de son voisin situé à l'ouest. La personne située au bout de la chaîne à l'ouest, n'ayant pas de voisin à l'ouest et ne sachant pas quoi faire jette sa feuille de papier. Toutes les autres déposent leur feuille de papier dans un (immense) panier. Le contenu du panier est ensuite mélangé. Un informaticien remarque qu'il est possible de reconstituer la chaîne des trois millions de personnes à partir des 2 999 999 feuilles contenues dans le panier.
Saurez-vous reconstituer la chaîne humaine ? [a]

a. Exercice adapté de l'exercice 24 (Second set) du chapitre 5 de *The art of computer programming*, vol 3, de D.E. Knuth.

Dans cet exercice on désigne par n le nombre de personnes dans la chaîne ($n = 3 \cdot 10^6$ dans l'énoncé ci-dessus, mais on considère le problème pour un nombre quelconque de personnes), et on suppose que les noms des personnes sont des nombres entiers naturels tous distincts.

Voici par exemple une chaîne humaine de taille 10 représentée en Python par une liste de 10 entiers : [6, 7, 5, 0, 9, 2, 8, 3, 4, 1]. Dans cette chaîne, la personne la plus à l'est est 6 dont le voisin d'ouest est 7. La personne la plus à l'ouest est 1.

Le panier contenant les $n - 1$ feuilles est représenté en Python par un ensemble de couples d'entiers dont la première composante est le nom d'une personne et la seconde le nom de son voisin d'ouest.

Voici le panier pour la chaîne donnée en exemple : {(8, 3), (6, 7), (9, 2), (2, 8), (7, 5), (5, 0), (0, 9), (3, 4), (4, 1)}. Notez qu'il ne contient que 9 couples.

Question 1 Réalisez une fonction nommée `chaine_en_panier` qui prend une liste ℓ d'entiers tous différents et construit l'ensemble de tous les couples $(\ell[i], \ell[i+1])$ possibles.

On suppose désormais qu'on ne dispose pas de la chaîne humaine, mais uniquement du panier contenant les $n - 1$ feuilles déposées par les participants sauf le plus à l'ouest. On suppose ce panier défini par une constante Python nommée `PANIER`.

On peut reconstruire une partie de la chaîne en triant les feuilles selon le nom des personnes les ayant déposées. Voici les feuilles du panier de notre exemple triées en ordre croissant selon la première composante : [(0, 9), (2, 8), (3, 4), (4, 1), (5, 0), (6, 7), (7, 5), (8, 3), (9, 2)].

Question 2 Avec les données du problème initial ($n = 3 \cdot 10^6$ personnes), pensez-vous raisonnable d'utiliser l'un des algorithmes de tri vus en cours ?

Question 3 En utilisant les fonctionnalités de tri offertes par Python indiquez comment construire la liste des

couples triés selon leur première composante à partir d'un panier. Vous nommerez `FEUILLES_TRIEES` cette liste.

Question 4 Réalisez une fonction nommée `recherche_dicho` prenant en paramètre un entier k et une liste ℓ de couples d'entiers, qui renvoie `None` si aucun couple de ℓ n'a k comme première composante, et qui, dans le cas contraire, renvoie l'indice dans ℓ d'un tel couple. Vous donnerez une documentation (docstring) précise de cette fonction. La recherche doit se faire en suivant l'algorithme dichotomique.

Voici trois exemples d'utilisation de cette fonction avec la liste ci-dessus nommée `FEUILLES_TRIEES`.

```
>>> recherche_dicho(11, FEUILLES_TRIEES)
>>> recherche_dicho(3, FEUILLES_TRIEES)
2
>>> recherche_dicho(7, FEUILLES_TRIEES)
6
```

Question 5 Avec les données du problème initial ($n = 3 \cdot 10^6$ personnes), combien de comparaisons votre fonction effectue-t-elle ? (*Indication :* $2^{20} \approx 10^6$)

Pour reconstruire (une partie de) la chaîne humaine, on peut partir d'une personne quelconque et chercher successivement les personnes voisines en les ajoutant dans une liste. On obtient ainsi une partie de la chaîne humaine : celle qui débute (en parcourant d'est en ouest) à la première personne choisie et qui va jusqu'au bout (la personne la plus à l'ouest). Avec notre exemple, à partir de 0, on obtient la chaîne : [0, 9, 2, 8, 3, 4, 1].

Question 6 Réalisez une fonction nommée `chaine_depuis` qui reconstruit la chaine débutant à la personne donnée en premier paramètre correpondant à la liste triée des feuilles passée en deuxième paramètre.

```
>>> chaine_depuis(0, FEUILLES_TRIEES)
[0, 9, 2, 8, 3, 4, 1]
>>> chaine_depuis(1, FEUILLES_TRIEES)
[1]
```

Maintenant que nous savons reconstruire une chaîne débutant à partir d'une personne donnée, il suffit de trouver la première personne de la chaîne. Cette première personne est la seule qui figure en première composante d'un couple du panier, mais pas en seconde composante.

Pour chercher cette première personne il peut être intéressant de disposer d'un panier organisé de sorte que les couples soient triés dans l'ordre croissant de leur seconde composante. Une façon simple de réaliser cela est de construire une liste contenant tous les couples du panier d'origine, mais avec leurs composantes échangées. Ainsi dans cette liste, la première composante du couple désigne le nom du voisin ouest, et la seconde celui de la personne ayant fourni les renseignements. Voici cette liste correspondant à notre exemple : [(0, 5), (1, 4), (2, 9), (3, 8), (4, 3), (5, 7), (7, 6), (8, 2), (9, 0)].

Question 7 Définissez une constante `FEUILLES_INVERSEES_TRIEES` dont la valeur est un tel panier construit à partir du panier initial (`PANIER`).

La première personne de la chaîne est la seule personne figurant en première composante d'un couple de `FEUILLES_TRIEES`, mais pas en première composante d'un couple de `FEUILLES_INVERSEES_TRIEES`. Dans notre exemple, il s'agit de 6.

Question 8 Réalisez une fonction nommée `premiere_personne` paramétrée par un panier, qui renvoie le « nom » de la première personne de la chaîne humaine correspondant à ce panier.

```
>>> premiere_personne(PANIER)
6
```

Question 9 Réalisez la fonction `panier_en_chaine` paramétrée par un panier qui construit la chaîne humaine correspondante.

```
>>> panier_en_chaine(PANIER)
[6, 7, 5, 0, 9, 2, 8, 3, 4, 1]
```

Exercice 2 *Découper des mots et recoller les morceaux*

L'enseignante de Cours Préparatoire de Barnabé, donne parfois l'exercice suivant :

À partir d'une liste de morceaux de mots coupés en deux parties, il s'agit de retrouver des mots en recollant les morceaux de mots.

Dans cet exercice, vous allez réaliser des outils qui peuvent être utiles à l'enseignante et à l'élève.

1 Outils pour l'enseignant

Question 1 Supposons que la variable `ligne` contienne une chaîne de caractères représentant un mot dont les syllabes sont séparées par des barres verticales (caractère `|`), et se terminant par le caractère `\n`, quelle instruction permet d'affecter à la variable `syllabes` toutes les syllabes de ce mot. Par exemple si `ligne = "sa|cris|tain\n"` alors `syllabes` sera égal à `["sa","cris","tain"]`.

Question 2 Dans la variable `couple_liste`, on dispose d'un tuple de longueur 2 dont les éléments sont des listes homogènes de chaînes de caractères. Dans la variable `res`, on suppose qu'on a une liste de chaine de caractères. Donnez une suite d'instructions PYTHON qui modifie la liste `res` en ajoutant en fin les deux mots obtenus en concaténant les chaînes de chacune des deux listes.

Par exemple si `couple_liste` vaut `(["sa","cris"],["tain"])` et `res` vaut `["re","frain","par","rain"]` après exécution de cette séquence d'instructions `res` vaut `["re","frain","par","rain","sacris","tain"]`.

Question 3 Réalisez une fonction `separe` paramétrée par une liste ℓ de longueur $n \geq 2$ et dont le résultat est un couple de listes (ℓ_1, ℓ_2) tel que
— la concaténation de la liste ℓ_1 avec la liste ℓ_2 donne la liste ℓ ;
— ℓ_1 et ℓ_2 ne sont pas vides ;
— le nombre d'éléments de ℓ_1 est choisi aléatoirement entre 1 et $n-1$ inclus.

```
>>> separe(["sa", "cris", "tain"])
(['sa'], ['cris', 'tain'])
```

Vous documenterez cette fonction.

Question 4 On réalise une fonction nommée `decouper` paramétrée par deux chaînes de caractères :

1. la première désigne un nom de fichier texte. Ce fichier contient une liste de mots à découper à raison d'un mot par ligne. Les positions où il est autorisé de procéder à une découpe (césure) sont marquées par une barre verticale. Cela correspond à un découpage en syllabes du mot. Chaque mot possède au moins quatre lettres et au moins une barre verticale.
2. la seconde désigne le nom du fichier à produire. Ce fichier doit contenir les morceaux de mots. Il y a un morceau par ligne. Chaque mot doit être découpé en exactement deux morceaux. Les deux morceaux obtenus doivent être obtenus en découpant à une position autorisée, les autres marques de découpe ne doivent plus être présentes. Tous les morceaux doivent être mélangés.

Cette fonction ne renvoie aucun resultat, mais elle possède un effet de bord (création ou modification d'un fichier).

Par exemple, si le fichier `mots.txt` contient les mots `re|frain, co|pain, ain|si, par|rain, sa|cris|tain,` à raison d'un mot par ligne, alors l'instruction `decouper("mots.txt","morceaux.txt")` aura pour effet de créer ou d'écraser le fichier `morceaux.txt` dont le contenu pourrait être, par exemple : `sa, frain, rain, cop, ain, par, si, re, ain, cristain,` à raison encore d'un mot par ligne.

On rappelle que dans le module `random`, il y a une procédure `shuffle` dont l'effet de bord est de mélanger une liste passée en paramètre.

Écrivez en PYTHON le code de la fonction `decouper`. Il est imposé de réutiliser les fonctions et les instructions des questions précédentes. Si vous n'avez pas su répondre à certaines questions, ou bien si vous n'avez pas envie de recopier le code vous pouvez juste écrire un commentaire du genre `# ici code de la question <num>`.

---

2 Outils pour l'élève

On dispose d'un lexique des mots français, ce lexique contient l'ensemble de tous les mots du français. Il est implanté par une liste de chaînes de caractères stockée dans la variable `LEXIQUE`. Tous les mots sont en bas de casse (minuscule). Le lexique est dans un ordre quelconque.

Question 5 Réaliser une fonction `lire_morceau` qui permet construire une liste de chaine de caractère `l` contenant l'ensemble des lignes du fichier dont le nom est passé en paramètre et modifie la liste `l` pour que chaque chaine soit nettoyée des espaces, tabulations, ou marque de fin de ligne aux extrémités. et dont le résultat est la liste `l`

On propose l'algorithme suivant :

```
initialiser res à la liste vide
pour prefixe variant dans liste_morceaux
faire
    pour suffixe variant dans liste_morceaux
    faire
        mot = prefixe+suffixe
        si mot est dans le lexique
        alors
           ajouter mot a la fin de la liste res
        finsi
    fait
fait
```

Question 6 Implantez en PYTHON cet algorithme.

Vous utiliserez la fonction à valeurs booléennes `recherche_seq(x, l, a, b)` vue en cours qui effectue une recherche séquentielle de `x` dans la tranche `l[a:b]` pour coder la condition `mot est dans le lexique`.

Question 7 Que contient la variable `res` à la fin de cet algorithme ?

On appelle m le nombre de lignes du fichier `morceaux.txt`, et n la taille du lexique.

Question 8 Quel est le nombre de comparaisons de chaînes de caractères effectuées dans le pire des cas en fonction de n et m.
