Exercices de programmation des oraux Agroveto

Julien Reichert

Session 2017

Ce document regroupe les exercices que j'ai donnés aux oraux d'Agroveto pour la session 2017.

Par rapport à la version 2016 (document encore en ligne), les exercices non posés ont été retirés.

Le nombre d'étoiles est un témoin de difficulté, et le nombre entre parenthèses est le nombre de fois que l'exercice a été posé sur un total de 203 candidats.

Algorithmes de recherche dans une liste / chaîne de caractères / matrice

* Exercice 1 (2) : Écrire en Python une fonction qui détermine la position d'un minimum local dans une matrice composée uniquement d'entiers de `0` à un `k` en paramètre.

On rappelle qu'un minimum local a tous ses voisins (latéraux, pour ne pas alourdir) supérieurs ou égaux à lui-même.

** Exercice 1 bis (5) : Écrire en Python une fonction qui détermine la position du minimum dans une matrice de nombres ayant la propriété que seul ce minimum soit un minimum local.

Bien entendu, on peut se contenter de rechercher le minimum global. On notera que sans propriété supplémentaire, la complexité dans le pire des cas est de l'ordre du nombre d'éléments dans la matrice.

Pour les deux versions de l'exercice 1, il est possible d'ajouter un bord de nombres « très grands » à la matrice, voire de considérer que ce bord existe.

** Exercice 2 (4) : Écrire en Python une fonction qui détermine si deux listes ont les mêmes éléments, peu importe l'ordre et le nombre d'occurrences.

** Exercice 2 bis (4) : Écrire en Python une fonction qui détermine si deux listes ont les mêmes éléments en même nombre, donc sont des permutations l'une de l'autre.

** Exercice 3 (16) : Écrire en Python une fonction qui retourne la position de départ et la longueur de la plus longue suite de caractères identiques dans une chaîne de caractères.

*** Exercice 4 (2) : Écrire en Python une fonction qui détermine si tous les `True` d'une liste de booléens sont côte à côte.

** Exercice 5 (26) : Écrire en Python une fonction prenant en entrée une liste de nombres entre 0 et k et qui retourne le nombre le plus fréquent.

*** Exercice 5 bis (2) : Écrire en Python une fonction prenant en entrée une chaîne de caractères (dont on supposera qu'ils sont tous issus de la table ASCII non étendue) et qui retourne le caractère le plus fréquent.

*** Exercice 5 ter (1) : Écrire en Python une fonction prenant en entrée une liste quelconque et qui retourne l'élément le plus fréquent.

Pour les trois versions de l'exercice 5, le comportement est au choix en cas d'égalité.

**** Exercice 6 (8) : Écrire en Python une fonction qui recherche un motif dans une matrice. Le motif en question est représenté par une matrice contenant éventuellement des None là où l'élément de la matrice où la recherche se fait n'a pas d'importance.

***** Exercice 6 bis (1) : Écrire en Python une fonction qui recherche un mot dans une matrice de lettres, à la manière des mots mêlés.

***** Exercice 7 (15) : Écrire en Python une fonction qui détermine si tous les `0` de la matrice en entrée sont ensemble, c'est-à-dire sur une même composante connexe.

Algorithmes de calcul sur une liste de nombres

** Exercice 8 (4) : Écrire en Python une fonction qui retourne, à partir d'une liste de couples d'entiers représentant respectivement une position et une vitesse (positive), la liste des couples d'entiers obtenue en ajoutant à chaque position la vitesse, à la restriction près qu'un dépassement est interdit (la position devient alors un de moins que la nouvelle position de l'élément qui aurait été dépassé).

**** Exercice 9 (1) : Écrire en Python une fonction qui prend en entrée une liste et un nombre et retourne une liste de listes contenant des éléments de la liste en entrée dans l'ordre, avec les propriétés que la somme de toutes les listes (sauf éventuellement la première) soit le nombre en argument et qu'en cas de besoin des éléments de la liste peuvent être cassés en deux. [1]

** Exercice 10 (9) : Écrire en Python une fonction qui prend en entrée une liste et qui la transforme de sorte que chaque élément devienne la moyenne de lui-même avec ses voisins, ou son voisin pour les éléments du bord. On pourra se limiter à écrire une fonction qui retourne la version transformée de la liste sans effet de bord.

*** Exercice 10 bis (1) : Même exercice mais en dimension deux. Les voisins sont les cases en contact latéral ou diagonal.

** Exercice 11 (30) : Écrire en Python une fonction qui détermine la position de l'élément d'une liste de couples `(x,y)` le plus proche d'un point donné avec la même syntaxe.

On pourra utiliser la norme 1 ou la norme 2 (respectivement somme des distances en abscisse et en ordonnée ou distance euclidienne), voire la norme infinie (maximum des distances en abscisses ou en ordonnée).

** Exercice 12 (1) : Écrire en Python une fonction qui détermine si une liste de nombres en entrée correspond à une suite géométrique (la valeur de retour doit être un booléen).

*** Exercice 12 bis (2) : Écrire en Python une fonction qui détermine si une liste de nombres en entrée correspond à une suite arithmétique ou géométrique (la valeur de retour doit être une chaîne de caractères répondant aux deux questions).

*** Exercice 13 (4) : Écrire en Python une fonction qui détermine si une liste est triée par ordre croissant ou décroissant (la valeur de retour doit être une chaîne de caractères répondant aux deux questions).

*** Exercice 13 bis (16) : Écrire en Python une fonction qui vérifie si une liste est d'abord croissante puis décroissante.

1. Cet exercice consiste à ajouter des mesures à une portée, la liste contenant simplement des durées de notes. On tient compte de l'existence de liaisons de prolongation. La volonté de ne pas renuméroter tous les exercices alors que celui-ci est une nouveauté de 2017 a conduit à le rapprocher de l'exercice 10...

Algorithmes divers

*** Exercice 14 (14) : Écrire en Python une fonction prenant en entrée une matrice carrée et renvoyant la matrice obtenue en faisant une rotation d'un quart de tour dans le sens horaire de la matrice de départ.

*** Exercice 15 (16) : Écrire en Python une fonction qui simule une promenade aléatoire de `n` pas sur une grille de dimensions `(2*L,2*l)` centrée en `(0,0)`, qui est le point de départ. Le programme retourne un code indiquant si on est resté sur la grille ou de quel côté on est sorti (ce qui aurait interrompu la promenade).

**** Exercice 15 bis (4) : Écrire en Python une fonction qui simule une promenade aléatoire de `n` pas sur un graphe donné en tant que liste de listes d'adjacence.

**** Exercice 16 (2) : Écrire en Python une fonction prenant en entrée quatre couples de flottants `a`, `b`, `c` et `d` et déterminant si les segments entre les points de coordonnées `a` et `b` d'une part et `c` et `d` d'autre part se chevauchent.

**** Exercice 17 (4) : Écrire en Python une fonction qui vérifie si un cercle de centre `(x,y)` et de rayon `r` touche un rectangle parmi ceux donnés dans une liste en tant que couples de couples `((xg,yg),(xd,yd))`, où `(xg,yg)` sont les coordonnées du coin en bas à gauche et `(xd,yd)` sont les coordonnées du coin en haut à droite. On pourra faciliter l'exercice en considérant un rectangle et des rectangles, voire un cercle et des cercles.

**** Exercice 18 (4) : Écrire en Python une fonction déterminant le nombre de façons de tracer une ligne de taille `n` (en paramètre) avec des traits de taille `2` ou `3`, en tenant compte de l'ordre des traits.

***** Exercice 19 (3) : Écrire en Python une fonction qui fait deviner un nombre de `1` à `1000` (ou un seuil en paramètre) à l'utilisateur ou par l'ordinateur (suivant un paramètre en entrée) à l'aide d'un protocole de communication où celui qui doit deviner le nombre entend si sa proposition est supérieure ou inférieure à la réponse ou correcte.

*** Exercice 20 (1) : Écrire en Python une fonction qui simule l'addition de deux fractions en travaillant sur deux couples d'entiers. Les arguments comme la valeur de retour correspondront à des fractions simplifiées de dénominateur strictement positif.