

### Exercice 1

Vous devez gérer un tableau contenant les informations ci-dessous. Ecrire le programme qui permet cette création puis l'affichage correspondant à ce que vous voyez :

—	—	—	—	—	—
—	0	0	0	0	—
—	1	1	1	1	—
—	0	0	0	0	—
—	1	1	1	1	—
—	0	0	0	0	—
—	1	1	1	1	—
—	—	—	—	—	—

### Exercice 2

1. Corrigez le programme ci-dessous en respectant les règles du langage Python et donnez ce qui s'affiche à l'écran (après correction du programme) si la variable info est égale à 8 .

```
def molecule(counter):
    n=s=1
    pas=3
    for n in range (pas,Counter,pas)
        if (n%2 != 0):
            s = s-n
            print s
    print 'n =',n
    return s

print 'Hello'
print 'give a' value'
info=input()
total=molecule[info]
print total
```

### Exercice 3

Ecrire un programme qui demande à l'utilisateur des nombres jusqu'à ce qu'il donne le nombre 0. Ces nombres seront sauvés dans une liste ordonnée (ordre croissant). Après la saisie vous afficherez à l'écran tous les nombres supérieurs au dernier nombre saisi (avant 0) par l'utilisateur.

#### 4 Problème

Vous étudiez la cascade *Map Kinase* des espèces *C. Elegans*, *E. Coli*, *Arabidopsis Thaliana* et *B. Subtilis* à partir des bases de données internationales. Vous avez décidé de réaliser un programme Python pour gérer les informations recueillies. Vous **choisissez de stocker ces informations dans une structure de dictionnaire**. Ces informations sont :

- un identifiant,
- une séquence ADN,
- la taille de la séquence,
- la liste des gènes potentiellement identifiés dans cette séquence.

Votre programme doit permettre :

1. la saisie des données dans le dictionnaire,
2. l'affichage des informations concernant une espèce donnée par l'utilisateur,
3. l'affichage de la liste des gènes et les séquences associées, vous prévoyez qu'un gène puisse apparaître dans plusieurs séquences.
4. Vous donnerez aussi le programme principal qui appelle toutes ces fonctions.

Après quelques semaines d'utilisation de votre programme, vous vous apercevez qu'il serait utile d'ajouter le nom de la base de laquelle provient la séquence.

Proposez une modification qui permettrait de réaliser cet ajout sans bouleverser tout votre programme.

# DS - Algorithmique - Master 1 Bio-informatique

19 décembre 2017 - Durée : 1h30 minutes

Les documents, les ordinateurs et les téléphones ne sont pas autorisés.

## 1 Pile et File

### Exercice 1

Donnez la définition d'une file. **Dessinez un exemple avec une pile d'entiers** qui illustre votre définition.

### Exercice 2

Dans cet exercice, vous devez utiliser uniquement les piles données à l'aide des fonctions primitives suivantes :

```
def creer_pile():
    # renvoie une nouvelle pile vide

def empiler( p, e ):
    # empile dans la pile p l'élément e.

def depiler( p ):
    # dépile l'élément situé en haut de la pile p et le renvoie.

def taille( p ):
    # renvoie le nombre d'éléments contenus dans la pile p.
```

Dans cet exercice, il n'est pas autorisé d'utiliser des listes, tuples et dictionnaires du langage Python.

1. Proposez un algorithme *multiplication(p)* qui prend en paramètre une pile *p* remplie de réels et qui renvoie le produit de tous les réels contenus dans la pile. La pile *p*, à la fin du calcul ne doit pas avoir été changée.
2. Proposez un algorithme *separation(p1, p2)* qui prend en paramètres deux piles *p1* et *p2* contenant toutes deux des entiers et qui met **dans p1 tous les entiers pairs** (se trouvant initialement dans *p1* et *p2*), et **dans p2 tous les entiers impairs** (se trouvant initialement dans *p1* et *p2*). Pour cet algorithme, vous n'avez **pas le droit de créer de nouvelles piles**. Vous devez vous contenter d'utiliser uniquement *p1* et *p2*.

## 2 Racine carré

Les deux exercices de cette section sont indépendants et peuvent être traités séparément. Il s'agit de deux méthodes pour calculer la racine carré d'un nombre.

La racine carré d'un réel  $r$  positif ou nul est notée  $\sqrt{r}$  et est l'unique réel positif tel que  $\sqrt{r} \times \sqrt{r} = r$ .

Par exemple, la racine carré de 25 est 5 car  $5^2 = 5 \times 5 = 25$ .

La racine carrée de 3 est proche de 1.7320508 car  $1.7320508 \times 1.7320508 \approx 3$ .

### Exercice 3

L'objectif de cet exercice est de calculer la racine carrée d'un nombre à l'aide de la méthode de Newton.

Pour calculer la racine carré de  $r$ , Newton propose de calculer la suite  $u_n$  défini par :

$$\begin{cases} u_0 = r \\ u_{n+1} = \frac{u_n + \frac{r}{u_n}}{2} \end{cases} \text{ pour tout entier } n \geq 0$$

On peut démontrer que  $(u_n)_{n \geq 0}$  est une suite de réels qui s'approche aussi près que l'on veut de  $\sqrt{r}$  quand  $n$  grandit.

Par exemple, si  $r = 3$ , alors  $u_0 = 3, u_1 = 2, u_2 = 1.75, u_3 = 1.7321, u_4 = 1.7320\dots$ , qui, quand  $n$  est grand devient aussi proche que l'on veut de  $\sqrt{3}$ .

1. Proposez un programme `def suite_newton(r, n)`, qui prend en paramètres un réel  $r$  et un entier  $n$  et qui renvoie le terme  $u_n$  de la suite de Newton défini plus haut.
2. Proposez un programme, `def sqrt_newton(r, erreur)` qui prend en paramètres deux réels  $r$  et  $erreur$  et qui renvoie une approximation  $e$  de  $\sqrt{r}$  de sorte à ce que la différence entre  $r$  et  $e \times e$  soit comprise entre  $-erreur$  et  $+erreur$ .

Par exemple, l'appel à `sqrt_newton(3, 0.01)` pourrait renvoyer la valeur 1.7320, car  $1.7320^2 = 2.999824$  et la différence  $3 - 2.999824 = 0.000176$  est bien dans l'intervalle  $[-0.01, 0.01]$ .

### Exercice 4

L'objectif de cet exercice est de calculer la racine carrée d'un nombre à l'aide d'un algorithme dichotomique.

La racine carré est une fonction qui vérifie la propriété suivante :

$$\begin{cases} \text{si } a^2 < r \text{ alors } a < \sqrt{r}, \\ \text{si } a^2 > r \text{ alors } a > \sqrt{r}. \end{cases}$$

Par exemple, on peut utiliser cette propriété pour trouver un approximation de la racine carré de 3.

Supposons que nous souhaitions trouver la racine carré de 3. Essayons la valeur 2. Comme  $2 \times 2 \neq 3$ , on sait que 2 n'est pas le bon résultat. Cependant,  $2 \times 2 = 4$  est plus grand que 3, donc la racine carré de 3 est plus petite que 2.

Essayons la valeur 1.5. Comme  $1.5 \times 1.5 \neq 3$ , on sait que 1.5 n'est pas le bon résultat. Cependant,  $1.5 \times 1.5 = 2.25$  est plus petit que 3, donc la racine carré de 3 est plus grande que 1.5.

On vient de déduire que  $\sqrt{3}$  appartient à l'intervalle  $[1.5, 2]$ .

Proposez un algorithme dichotomique qui prend en paramètres deux réels  $r$  et  $erreur$  et qui renvoie une approximation  $e$  de  $\sqrt{r}$  de sorte à ce que la différence entre  $r$  et  $e \times e$  soit comprise entre  $-erreur$  et  $+erreur$ .

## 1. Traitement et analyse d'image

On se propose de traiter et d'analyser une image avec des dessins de flèches comme celle présentée dans la Fig. 1A avec le logiciel ImageJ.

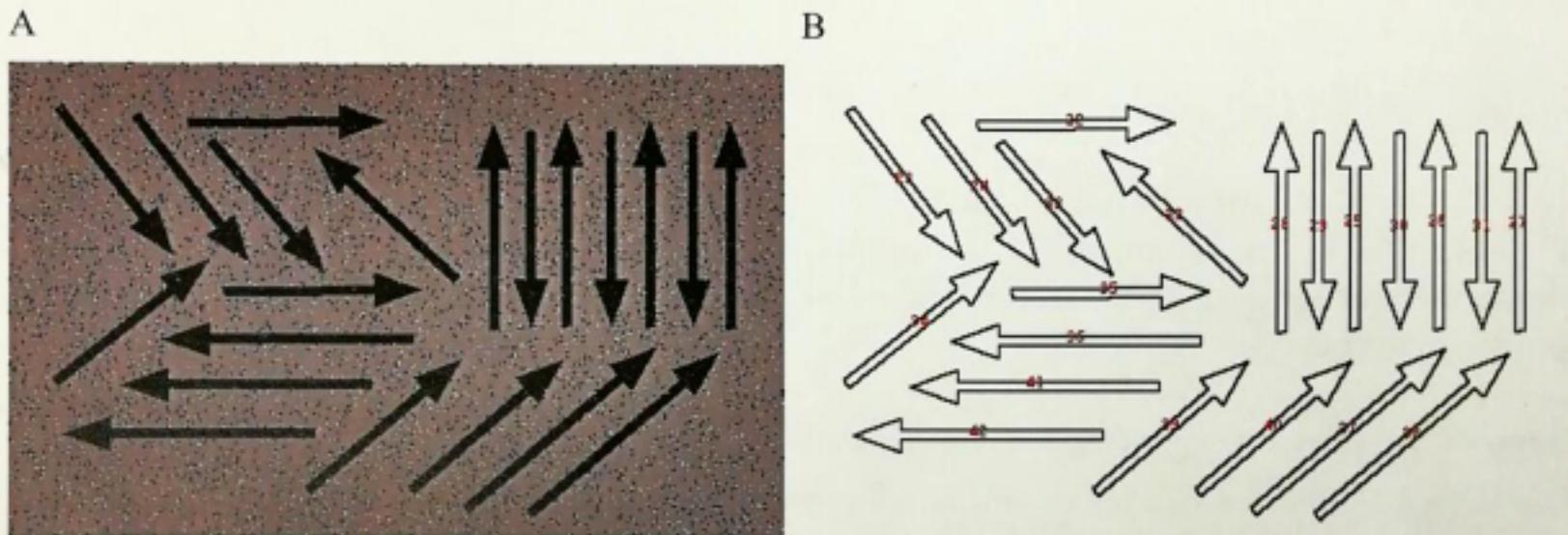


Fig.1 : Image de flèches. A) Image originale 16-bit de 487 x 300 pixels. B) Image après analyse.

- Question 1.1 : Que signifie le terme "16-bit " dans la légende de la Fig.1.
- Question 1.2 : Quelle est la taille de l'image en octets ? En kilo-octets ? Détaillez les calculs.
- Question 1.3 : Comptage des flèches. Indiquez un protocole pour calculer le nombre de flèches à partir de l'image de la Fig.1A pour obtenir le résultat de la Fig.1B. Pour chacune des étapes, justifiez le choix d'une technique et décrire succinctement quel est le principe de la fonction utilisée.
- Question 1.4 : Quel est le critère de mesure nécessaire pour calculer la longueur de chacune des flèches et l'angle entre la flèche et un axe horizontal ? Décrire le principe de calcul de ce critère de mesure ?

## 2. Détermination de l' orientation des flèches

On souhaite déterminer dans quelle direction pointent les flèches en différenciant leurs deux extrémités (la pointe et sa base). Pour cela, après squelettisation (Fig. 2A), on extrait les extrémités et le « carrefour » (intersection avec les lignes) par morphologie mathématique (Fig. 2B). On considère alors que le centroïde de la pointe de flèche est la moyenne du point « carrefour » et des trois extrémités les plus proches.

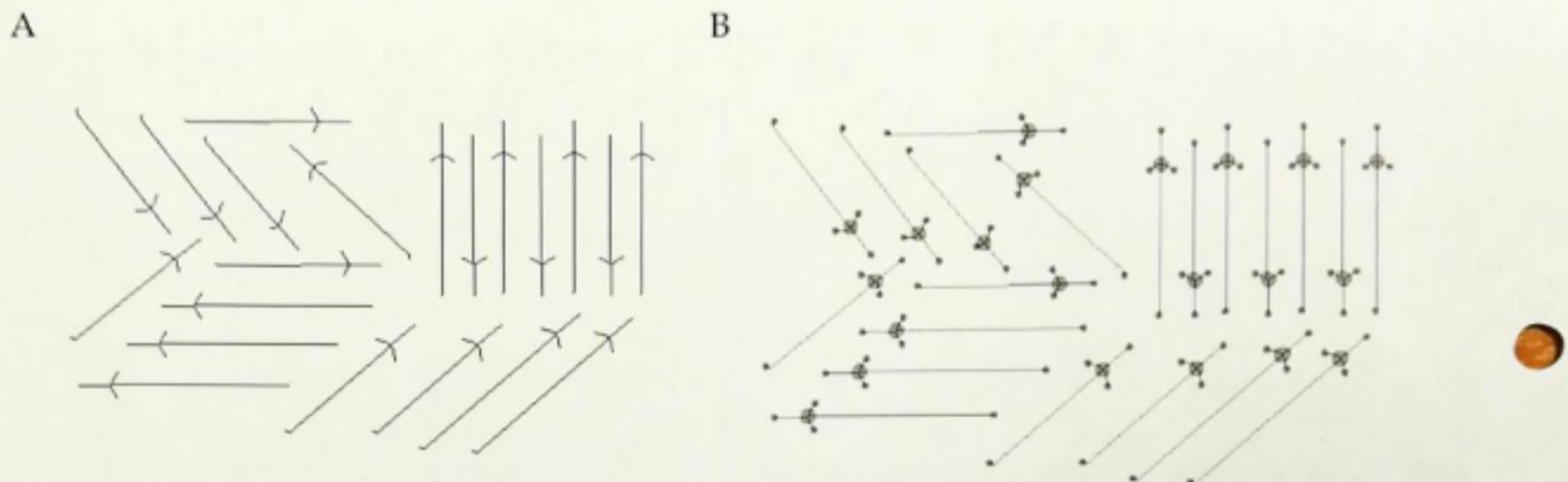


Fig.2 : A) Squelettisation de l'image de la Fig. 1A après traitement. B) Résultat des commandes de morphologie mathématique. Les points de diamètre supérieur correspondent aux intersections et les autres aux extrémités du squelette. On cherche à calculer le centroïde des pointes de flèche.

- **Question 2.1 :** Décrire le principe de la squelettisation?
- **Question 2.2 :** Quelles sont les commandes de morphologie mathématique utilisées pour extraire les extrémités et les intersections ? Quel est leur principe ?
- **Question 2.3 :** Pour calculer le centroïde de chacune des pointes de flèches, on doit calculer la moyenne des quatre points composant la pointe de flèche : le point d'intersection et les trois extrémités les plus proches. Ecrire en JavaScript une fonction **moyenne(x,y)** qui prend en argument deux tableaux x et y contenant les coordonnées respectives en X et en Y des quatre points et retourne un tableau contenant les coordonnées X et Y de la moyenne des quatre points.

**Note :** Pour l'implantation, on utilisera absolument des boucles **for**.

Exemple d'utilisation:

```
let x = [1,2,3,4]
let y = [6,7,8,9]
let resultat = moyenne(x,y) ;
console.log(resultat); // ← [2.5,7.5]
```

- **Question 2.4 :** Proposez un protocole différent et plus simple que celui décrit précédemment permettant d'extraire le centroïde de chaque pointe de flèche directement par traitement et analyse de l'image.

**Exercice 1 :**

Grâce à des nouvelles technologies dites de séquençage aujourd'hui on peut lire le génome d'un individu rapidement et à moindre coût. Le séquençage produit un grand nombre de petits segments du génome appelés lectures (*reads* en anglais). Un *read* est donc une chaîne de bases nucléiques (soit de caractères parmi les quatre : 'A', 'T', 'G', 'C'). Par exemple un *read* peut être défini par la séquence « AGGGTCCCGAGAT ». Le but de ces exercices est de manipuler un ensemble de N *reads* modélisé par une liste de N chaînes de caractères.

Note 1: Pensez à réutiliser vos fonctions.

Note 2: Seules les fonctions et méthodes Python suivantes sont autorisées: List.append(), len(), print(), range()

Note 3: Pour ajouter un élément à la fin d'une liste, on utilise la méthode append().

Exemple: my\_liste.append('ACGT') ajoute à la fin de la liste my\_liste, l'élément 'ACGT'.

1.1. Écrire une fonction **average(reads)** qui calcule la longueur moyenne des reads.

Exemple : **average(['AGGCT', 'GGAT', 'GGCAAA'])** renverra le résultat 5

1.2. Écrire une fonction **threshold(reads)** qui prend en argument un ensemble de N *reads* passé en paramètre de la fonction et qui retourne ceux qui ont une longueur supérieure ou égale à la longueur moyenne des N *reads*.

Exemple : **threshold(['AGGCT', 'GGAT', 'GGCAAA'])** renverra ['AGGCT', 'GGCAAA']

1.3. Écrire une fonction **countNucl(seq,symbol)** qui compte le nombre de nucléotides 'symbol' dans la sequence 'seq'.

Exemple : **countNuc('AGGCT', 'G')** renverra 2

1.4. Écrire une fonction **ratioGC(reads)** qui, pour un ensemble de N *reads* passés en paramètre de la fonction, calcule le taux moyen de GC (proportion de G et C dans la chaîne de caractères) des N *reads*.

Exemple : **ratioGC(['AGGCT', 'GGAT', 'GGCAAA'])** renverra 0.53333 équivalent à ((3/5 +2/4 +3/6)/3)

1.5. Écrire une fonction **match(seq1,seq2)** prenant en arguments deux séquences de même longueur et retournant un score de matchs (1 point si identiques et 0 si différents).

Exemple : **match('AGGCA', 'GGCAA')** renverra 2

1.6. Écrire une fonction **removeEnds(reads, adaptor)** qui, pour un ensemble de N *reads* et un **adaptator** (adaptateur : courte chaîne de bases nucléiques) passés en paramètre de la fonction, filtre enlève les extrémités des *reads* correspondants à l'adaptateur. Ici on s'intéresse uniquement aux matchs parfaits entre les extrémités des *reads* et la séquence de l'adaptateur. La fonction devra retourner le nouvel ensemble de *reads*

Exemple : **removeEnds(['TTTCAGGC', 'GGATTTTC', 'GGCAAATTTC'], 'TTTC')**  
renverra ['AGGC', 'GGAT', 'GGCAA']

## Exercice 2

Une étude a été menée par Zeng et al. en 2016 ([doi:10.1038/srep33031](https://doi.org/10.1038/srep33031)) sur la diversité et présence de certains gènes bactériens, impliqués dans la **dégradation du sulfure**, dans les fjords des régions de l'Arctique. Les échantillons recueillis ont été ensuite séquencé en utilisant des primers spécifiques des gènes **Ddd** (DddL, DddA, DddP ...). A noter ces séquences n'étaient pas, au début de l'étude, **présentes dans les bases de données du NCBI**. On s'intéressera dans les extraits suivants exclusivement aux gènes **DddP**.

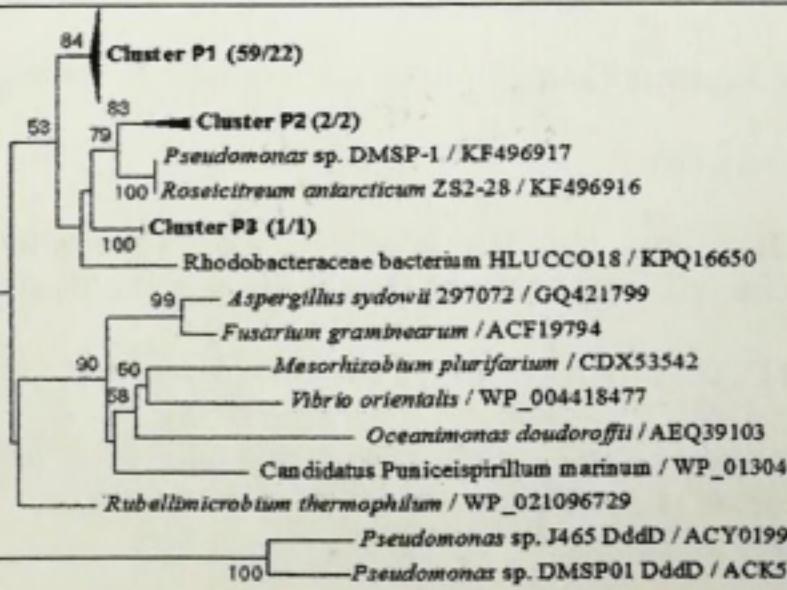
2.1 Expliquez en vous appuyant sur l'extrait suivant de la méthode analytique mise en œuvre pour construire l'arbre les étapes clés et quels outils ont été utilisés pour l'obtenir?

### Data analysis

As the known *DddP* genes with a similarity >81% from different bacteria belong to the same genus, sequences showing more than 81% amino acid identity with each other were grouped in the same cluster. ... using a BLASTX search against NCBI databases (<http://www.ncbi.nlm.nih.gov>).... Sequence alignment and phylogenetic tree building were completed using respectively MUSCLE and Neighbour-joining method. Bootstrap tests of phylogeny were run with 1000 replicates.

2.2- A partir de l'extrait suivant, et en vous appuyant sur la figure de l'arbre, vous devrez décrire l'arbre obtenu et expliquer sur quelles observations s'appuient l'hypothèse d'un **transfert horizontal** (les clusters représentent un groupe de gènes identifiés dans les zones de prélèvements (3 stations : K1, K2 et K3). Les autres gènes sont issus des bases de données du NCBI) :

The *dddP* gene is one of the most frequently detected *ddd* genes in marine bacteria and is mainly found in the Roseobacter and SAR116 clades of Alphaproteobacteria<sup>18,20,43</sup>. However, evidence for horizontal gene transfer of *dddP* to some Gammaproteobacteria and fungal species has been reported. In the present study, the *DddP* sequence of *Pseudomonas* sp. DMSP-1...), was found to be closely related to Cluster P2 and *Roseicitreum antarcticum* ZS2-28 (Fig. 4), suggesting a possible inter-class horizontal gene transfer of *dddP* between Alpha and Gammaproteobacteria. In addition, comparing with the absence of the genus *Roseovarius*, which dominated *dddP* genotypes in this study, the genera *Sulfitobacter* and *Loktanella* were the dominant members of the Alphaproteobacteria in bacterioplankton community in Kongsfjorden<sup>31</sup>, suggesting a possible inter-genus horizontal gene transfer of *dddP* in Alphaproteobacteria.....



**Figure 4 : Phylogenetic tree of deduced *DddP* sequences from two clone libraries of seawater in Kongsfjorden plus those from known bacterial species available in NCBI.**

*DddD* sequences from *Pseudomonas* species were used as an outgroup.

Bootstrap values of <50 have been removed for clarity.

Numbers in parentheses following cluster names indicate the number of sequences found in stations K1 and K3, respectively. The scale bar indicates evolutionary distance.

2.3- Expliquez à quoi correspondent dans la légende *outgroup* et *bootstrap*.