# Option Bioinformatique L3
# TP4

*2016*

Le support de cours peut être consulté ici.

Le code source réalisé pendant ce TP ainsi que les TPs suivants est à rendre, le **18/03**, à l'adresse suivante:

**rayan.chikhi@univ-lille1.fr**

avec le sujet de mail suivant (remplacer Nom1, Nom2 par chaque nom de famille; monomes autorisés):

**[BI] [TP] Nom1 Nom2**

Il est demandé de, soit, créer une archive .zip ou .tar.gz se décompressant dans un répertoire "BI-Nom1-Nom2", et d'attacher cette archive en pièce jointe à l'email; soit de créer un repository Github (au nom de votre programme principal, par exemple "bioseq") et d'envoyer le lien par email en précisant les noms du binome/monome.

Dans ce répertoire (ou dans le repository), un fichier script nommé **execute-tp4** exécutera (et auparavant, compilera si besoin) tout ce qui est demandé dans les sections de ce TP nommées "Pour le compte-rendu [..]". De telle manière qu'exécuter:

./**execute-tp4**

dans le répertoire BI-Nom1-Nom2 suffira à l'évaluateur du TP pour tester tous vos programmes. **Ce script (ainsi que le code source de votre programme) fera office de compte-rendu du TP4.** Il est demandé que les programmes affichent exactement les résultats aux formats proposés par les énoncés des TPs.

**Note: ne pas rendre ce TP4 seul. L'ensemble des TP4,5,6,7 sera à rendre, d'un seul bloc, à la date indiquée ci-dessus.**

Les langages acceptés sont: **Python, Java, C, C++.** *Points supplémentaires à ceux qui s'initieront à Python.*
Pour d'autres langages, demander à l'instructeur.

# Partie 1: Recherche naïve dans un texte

Ecrire une fonctionnalité **search-fasta-naive** s'exécutant de la manière suivante:

```
./bioseq search-fasta-naive <nom du fichier de génome g>.fasta
                            <séquence de la lecture r>
```

qui implémente l'algorithme naif de recherche d'une sous-chaîne r dans la séquence d'un fichier FASTA g (on suppose que g ne contient qu'une seule séquence):

pour chaque position i de g:
  si g[i..i+|r|] == r alors:
    afficher i

Pour les besoins de l'exercice, il est demandé d'implémenter l'algorithme avec une boucle, explicitement, et de ne pas utiliser une fonction de recherche de sous-chaine fournie par le langage de programmation (ex: "find", "search").

search-fasta-naive affichera chaque position séparée par un espace.

Exemple: "`./bioseq search-fasta-naive`" appliqué à un fichier FASTA contenant la séquence ACTTGATT et recherchant la séquence TT affichera:

2 6

**Pour le compte-rendu**, télécharger le virus lambda phage (au format FASTA, disponible ici: http://www.ncbi.nlm.nih.gov/nuccore/215104?report=fasta). Appliquer search-fasta-naive à la recherche de deux séquences dans le virus lambda phage:
"GGCCATCCTTCCTGACCATTTCCATCATTCCAGTCGAACT"
et
"ATAGTG".

Optionnel: afficher le temps d'exécution de chaque recherche.

# Partie 2: calcul du tableau de suffixes

Ecrire une fonctionnalité **suffix-array** s'éxécutant de la manière suivante:

```
./bioseq suffix-array [nom du génome g].fasta
```

qui calcule le tableau de suffixes du génome g.

Le tableau des suffixes d'une chaîne de caractères s est un tableau de nombres entiers. Chaque nombre entier correspond à la position d'un suffixe dans la chaîne s. Pour construire le tableau, les suffixes sont au préalables triés dans l'ordre lexicographique. L'élément à la position ***i*** du tableau des suffixes correspond à la position dans **s** du ***i***$^{eme}$ suffixe dans la liste des suffixes **triés** de s.

Le programme affichera chaque valeur du tableau séparée par un espace. Vous pouvez utiliser une méthode de type "sort" de votre langage de programmation pour trier les suffixes -- inutile de réimplémenter une fonction de tri.

Exemple: suffix-array appliqué à un fichier FASTA contenant la séquence TCGA affichera:

3 1 2 0

car les suffixes de TCGA ainsi que leur positions respectives dans la chaîne originale, sont:

0: TCGA
1: CGA
2: GA
3: A

en triant les suffixes par ordre lexicographique (=alphabétique), on obtient les valeurs du tableau de suffixes en lisant de haut en bas les positions des suffixes triés:

**3**: A
**1**: CGA
**2**: GA
**0**: TCGA

**Pour le compte-rendu**, appliquer suffix-array à la séquence "GGCCATCCTTCCTGACCATTTCCATCATTCCAGTCGAACT" ainsi qu'au virus lambda phage.
Note: si votre implémentation dépasse la mémoire disponible de la machine, vous avez la possibilité de remplacer le virus lambda phage par le virus Ebola disponible à l'adresse suivante: http://www.ncbi.nlm.nih.gov/nuccore/743615855?report=fasta
Un point "bonus" sera décerné si votre implémentation en Java est capable de construire le tableau des suffixes pour le virus lambda phage sur les ordinateurs de la salle de TP (4 GB de

mémoire). Pour ceux qui font le TP dans un autre langage, vous n'aurez pas ce point bonus car bcp d'autres langages sont apparemment capables de calculer le tableau de suffixes pour le virus lambda phage (mais il n'est pas trop tard pour avoir le point bonus Python).

Optionnel: afficher les temps d'exécution.

## Partie 3: recherche avec tableau de suffixes (optionnel)

Ecrire une fonctionnalité **search-fasta-sa** s'exécutant de la manière suivante:

**`./bioseq search-fasta-sa`** `<genome>.fasta <séquence de la lecture r>`

qui implémente un algorithme de recherche dichotomique d'une sous-chaîne *r* dans la séquence d'un fichier FASTA *genome* (on suppose que *genome* ne contient qu'une seule séquence) à l'aide d'un tableau des suffixes. En construisant au préalable un tableau des suffixes du génome, l'algorithme recherchera la sous-chaîne par une recherche dichotomique aidée par le tableau des suffixes du génome. L'implémentation est libre du moment que la recherche se fait en temps O(log(|genome|)) (en considèrant qu'une comparaison de chaînes prend un temps O(1), ce qui n'est pas tout à fait vrai ni en théorie ni en pratique).

search-fasta-sa affichera chaque position séparée par un espace.

Exemple: "`./bioseq search-fasta-sa`" appliqué à un fichier FASTA contenant la séquence ACTTGATT et recherchant la séquence TT affichera:

2 6

**Pour le compte-rendu**, appliquer search-fasta-sa à la recherche de deux séquences dans le virus lambda phage: "GGCCATCCTTCCTGACCATTTCCATCATTCCAGTCGAACT" et "ATAGTG".

Si votre implémentation ne permet pas de construire le tableau des suffixes du virus lambda phage, recherchez les séquences "AACCATTTCAACATGCGAACACAACGTGT" et "ATAGTG" dans le virus Ebola.

Optionnel: afficher le temps d'exécution de chaque recherche.