# Recherche de motifs répétés dans un génome

## Lancement du programme

Le programme **Java** nécessite EXACTEMENT 5 paramètres passés en argument sans quoi ce dernier ne s'exécutera pas, à savoir :

- Le chemin du fichier '.fasta' à lire
- Un entier qui correspond, à la longueur des mots que l'on souhaite rechercher dans le fichier
- 3 Booléens indiquant si oui ou non on comptera comme étant une occurrence d'une séquence
  - sa séquence inverse
  - sa séquence complémentaire
  - sa séquence complémentaire inverse

## Lecture du fichier

La lecture du fichier se fait en utilisant un BufferedReader et un StringBuilder pour réduire le temps de concaténation des lignes de texte. Le programme renvoie le contenu du texte dans un String qui sera parcouru plus tard.

## Calcul des mots de taille N

Ensuite, le programme calcule toutes les mots possibles de taille N en fonction des paramètres (taille N et les 3 booléens).
Pour cela, tous les combinaisons sont calculés, et rajoutés dans une liste si elle, ou une occurrence, n'est pas déjà présente.
Exemple : Pour l'alphabet A = {A, C, G, U},  N = 2 et une occurrence étant uniquement la séquence elle-même et sa séquence complémentaire inverse, on obtiendra la liste suivante :

```
AA, AC, AG, AU, CA, CC, CG, GA, GC, UA
```

Pour nous aider dans notre parcours des combinaisons, nous avons créé un tableau d'entiers de taille N où chaque cellule est un entier entre 0 et 3, que nous transcrivons par la suite sur notre alphabet. Par exemple :

```
combinaisons = {0, 2, 3, 0}  <=>  AGUA
```

Et tant que la combinaison n'est pas la dernière (3, 3, 3, 3), on incrémente le tableau, comme si on comptait en base 4 (taille de l'alphabet).

De cette manière nous pouvons parcourir toutes les combinaisons et vérifier pour chacune d'entre elle si l'on peut la rajouter à la liste ou non (en fonction du type d'occurrences).

## Calcul des occurrences

Une fois tous les mots de taille N calculés, on va rechercher pour chacun d'entre eux leurs occurrences dans le texte et stocker l'index où les occurrences ont été trouvées pour ce mot.

Pour cela, nous avons opté pour l'algorithme de Knuth, Morris et Pratt.

## Algorithme de Knuth, Morris et Pratt

Pour la phase d'initialisation du tableau Next, nous procédons de la manière suivante :

- Pour chaque sous-chaîne du mot allant de 0 à (i-1) inclus, nous énumérons la liste de ses bords.
- Si une sous-chaîne concaténée au caractère du mot à la position i n'est pas préfixe du mot, la sous-chaîne est valide
- Next[i] = la longueur de la sous-chaîne valide la plus grande (-1 si aucune sous-chaîne valide)

Ensuite, nous parcourons pas à pas le mot. Lorsqu'on trouve une différence à la position i du mot, on effectue un

décalage vers la droite de i – Next[i];

## Ecriture du fichier data

Une fois les occurrences calculées, nous allons écrire un fichier data contenant une coordonnée X et une coordonnée Y par ligne correspondant aux « croisement » de tous les index des occurrences d'un mot entre eux.

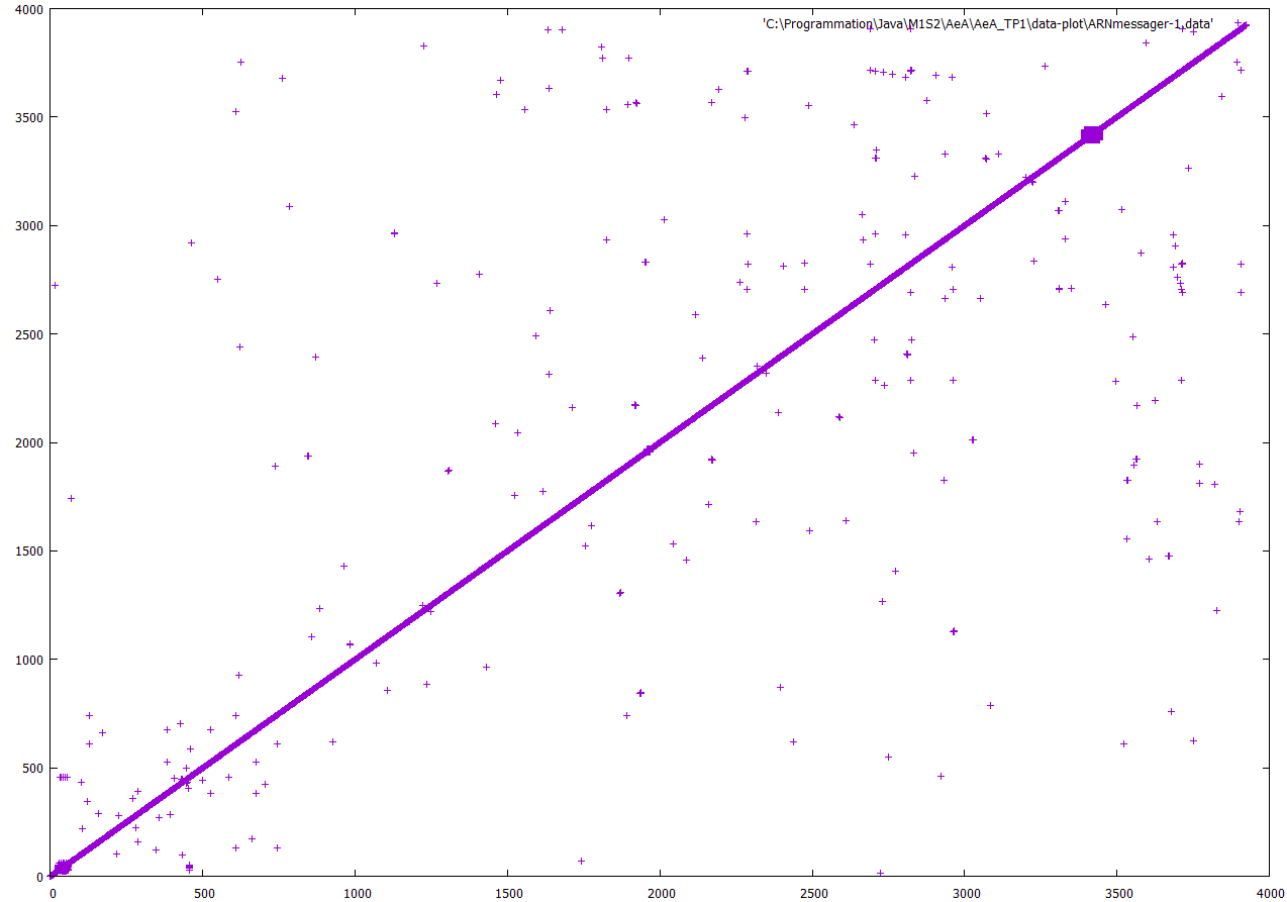
Exemple :

Si le un mot a pour index 3, 5 et 8. Alors on insérera dans le fichier les lignes suivantes :

```
3 3
3 5
3 8
5 3
5 5
5 8
8 3
8 5
8 8
```

## GnuPlot

Une fois le fichier data généré, nous pouvons l'interpréter avec GnuPlot.
Voici un exemple sur le fichier *ARN-messager-1.fasta* pour tous les mots de taille 9 avec pour occurrence la séquence elle-même et sa séquence complémentaire inverse.




Nous pouvons noter une symétrie axiale suite à cette interprétation