Problème (2)

Cet énoncé fait directement suite à la feuille de TD intitulée **Problème (1)**.

Récupérer les fichiers (archive compressée) associés à cette feuille de TD.

1 Applications aux acides aminés ... suite

1.1 EXERCICE Compléter le script `nb-aa-de-type.sh`. Ce script prend deux paramètres, un acide aminé et un fichier de protéine normalisé. Il renvoie le nombre d'occurrences de l'acide aminé (de la protéine) passée en paramètre. Par exemple,

```
$ ./nb-aa-de-type.sh K q9660.norm
2801
```

1.2 EXERCICE Compléter le script `aa-le-plus-present.sh`. Ce script prend en paramètre un fichier de protéine normalisé. Il renvoie l'acide aminé (de la protéine) le plus présent. Par exemple,

```
$ ./aa-le-plus-present.sh q9660.norm
L
```

1.3 EXERCICE Compléter le script `histogramme-proteine.sh`. Ce script prend en paramètre le nom d'un fichier normalisé d'une protéine suivi d'un entier. Cet entier représente le nombre de colonnes qui sera utilisé pour afficher le nombre d'occurrences de l'acide aminé le plus présent dans la protéine. Par exemple,

```
$ ./histogramme-proteine.sh q9661.norm 30
```

produira un histogramme (horizontal) représentant le nombre de chaque acide aminé de la protéine. Sur l'exemple présenté ci-dessous, l'acide aminé le plus présent dans la protéine est la lysine (`K`). Ainsi, la lysine sera représentée par une ligne horizontale composée de 30 étoiles.

```
Protéine: q9660.norm
Largeur max: 30

A ************************
C *****
D ****************
E ************************
F *********
G ******************
H ******
I ******************
K **********************
L ******************************
M ********
N *************
O
P ************
Q ***********
```

```
R *********
S ****************
T **************
U
V ********************
W *
Y ********
```

Pour calculer le nombre d'étoiles devant être affiché pour chaque acide aminé, vous pourrez tirer avantage de la commande expr.

1.4 Exercice Compléter le script `extrait-une-sous-chaine.sh` qui extrait un sous-ensemble d'acide aminé d'une protéine décrit par un fichier normalisé. Votre script prendra deux paramètres. Le premier sera le nom du fichier normalisé de la protéine, le second correspondra à position des acides aminés de la sous-séquence à extraire. L'exemple ci-dessous permet d'extraire du fichier q9660.norm les nucléotides en position $2, 4, 5, 6, 11$:

```
$ ./extrait-une-sous-chaine.sh q9660.norm 2,4-6,11
ISAAL
```

Examinez le script `genere-intervalles.sh`, afin d'en comprendre le fonctionnement.

```
$ cat genere-intervalles.sh
#!/bin/bash

# le premier paramètre est la longueur de la séquence
PAS=$1
# le second paramètre est la valeur à ne pas dépasser
FIN=$2

# initialisation de variables
NBR_DEBUT=1
NBR_FIN=$PAS

# l'instruction -le signifie "inférieur ou égal à"
while [ $NBR_FIN -le $FIN ]
do

 echo $NBR_DEBUT-$NBR_FIN

 # la commande expr permet de faire des calcul sur des entiers
 NBR_DEBUT=$(expr $NBR_DEBUT + 1)
 NBR_FIN=$(expr $NBR_FIN + 1)

done
```

Ce script prend deux paramètres ; le premier est la taille du pas de calcul, le second le nombre maximum à ne pas dépasser. Ce script, avec un pas de 6 et un nombre maximum de 10 produit le résultat suivant :

```
$ ./genere-intervalles.sh 6 10
1-6
2-7
3-8
4-9
5-10
```

1.5 Exercice En réutilisant le script précédent, compléter le script `extrait-toutes-les-sous-chaines.sh` qui prend en paramètre le nom d'un fichier normalisé d'une protéine, et un entier n correspondant à la longueur d'une sous-chaîne. Votre script renverra toutes les sous-chaînes (consécutives) de taille n composant la séquence des acides aminés de la protéine.

Par exemple la recherche de toute les sous-chaînes consécutives de taille 6 contenu dans le fichier `F6S5X4_HORSE.norm` se fera en invoquant la commande :

```
$ ./extrait-toutes-les-sous-chaines.sh F6S5X4_HORSE.norm 6
MITSAA
ITSAAG
TSAAGI
[...]
EPFEYI
PFEYID
FEYIDD
```

1.6 Exercice Invoquer le script précédent de telle sorte que le résultat produit soit sauvegardé dans un fichier `liste-sous-chaines-5-q9660.norm.dat`. Le nom du fichier de sauvegarde sera choisi de la façon suivante : `liste-sous-chaines-`$\prec$*taille_de_la_sous_chaine*$\succ$`-`$\prec$*nom_du_fichier_proteine*$\succ$`.dat`.

1.7 Exercice Compléter le script `sous-chaine-la-plus-frequente.sh`. Celui-ci prend en paramètre le nom d'un fichier .dat contenant la liste de toutes les sous-chaînes consécutives de taille fixée. Ce script renvoie l'une des sous-chaînes apparaissant le plus souvent dans ce fichier. Cette sous-chaîne sera précédé du nombre de fois qu'elle apparaît. Par exemple,

```
$ ./sous-chaine-la-plus-frequente.sh liste-sous-chaines-4-F6S5X4_HORSE.norm.dat
5 EKEK
```

1.8 Exercice Modifier `sous-chaine-la-plus-frequente.sh` afin qu'il fonctionne en consommant ses données depuis son entrée standard dans le cas ou aucun nom de fichier .dat n'est passé en paramètre. Cela signifie que votre script devra aussi fonctionner de la manière suivante :

```
$ cat liste-sous-chaines-4-F6S5X4_HORSE.norm.dat | ./sous-chaine-la-plus-frequente.sh
5 EKEK
```

Pour ce faire, vous pourrez utiliser la commande read [1] qui permet de lire une lire depuis l'entrée standard. Pour vous aidez dans l'utilisation de cette commande, testez et comprenez le fonctionnement de la suite de commandes suivante :

```
$ seq 1 10 | while read UNE_LIGNE; do echo $UNE_LIGNE; done | tr [0-9] [A-J]
```

Lorsqu'un nom de fichier sera passé en paramètre, c'est le comportement de l'Exercice 1.7 qui primera.

1.9 Exercice Compléter le script `plus-petite-chaine-non-repetee.sh` qui prend en paramètre le nom d'un fichier normalisé d'une protéine. Il renvoie la plus petite longueur de sous-chaîne consécutive n'ayant aucune répétition dans la protéine. Par exemple,

```
$ ./plus-petite-chaine-non-repetee.sh F6S5X4_HORSE.norm
8
```

1.10 Exercice Réfléchissez à différents moyens qui permettraient d'accélérer le calcul effectué par le script `plus-petite-chaine-non-repetee.sh`.

1. La documentation de la commande read est dans la page de manuel de bash.