Introduction

Doc. v1 (29/11/19)

Dans ce projet, nous allons faire un programme permettant d'aligner deux séquences de manière **globale** en utilisant l'algorithme de **Needleman-Wunsch**. Vous trouverez des exemples d'alignements dans le cours 2.

Soumission Le projet se fait par groupe de un ou deux et peut être réalisé dans n'importe quel langage. Le rendu se fait par mail à :

pierre1.parutto@supbiotech.fr

avant la date indiquée dans le mail, sous la forme d'une archive nommée :

projet_bioinfo_nom1[_nom2].tar.bz2.

Le format de l'archive doit être le suivant :

```
$ tar -tf projet_bioinfo_parutto.tar.bz2
projet_bioinfo_parutto/
projet_bioinfo_parutto/README
projet_bioinfo_parutto/AUTHORS
projet_bioinfo_parutto/src/
projet_bioinfo_parutto/src/[fichiers de code]
```

où le fichier *AUTHORS* contient le prénom et nom de chaque membre du projet, un par ligne et le fichier *README* indique comment utiliser (et compiler selon le langage) le projet et doit lister **toutes les dépendances**. Le projet sera évalué automatiquement via une moulinette.

N'hésitez pas à poser vos questions par mail à l'adresse ci-dessus.

Programme

Le programme doit se nommer `nwalign` (avec possiblement une extension, ie. *.py*, *.jar*, ...).

Entrées `nwalign` prend trois arguments entrées : `cmd x y` (dans cet ordre), où `cmd` est soit `score` soit `align` et `x`, `y` sont les deux séquences à aligner. Les séquences peuvent être soit de l'ADN (alphabet `ATGC`), soit de l'ARN (alphabet `AUGC`), soit protéique (alphabet `ARNDCQEGHILKMFPSTWYVBZX`), pour pouvoir être aligné, les séquences doivent être du même type.

Option L'option `--gamma=e,o` avec `e` et `o` deux entiers ≤ 0. Cette option permet de spécifier les valeurs des paramètres e et o de la fonction d'évaluation des trous $\gamma(x) = ex + o$. Si l'option n'est pas spécifiée, alors par défaut, $e = -1$ et $o = 0$. Vous n'avez pas à gérer $o \neq 0$ avant le palier 3.

Sortie Si la commande est `score`, le programme doit afficher sur la sortie standard, sur une unique ligne, le score d'alignement entre `x` et `y`. Si la commande est `align`, le programme doit afficher sur la sortie standard l'alignement entre les deux séquences avec x sur la première ligne et y sur la seconde. La commande `align` est à gérer à partir du palier 2. S'il n'y a pas d'erreur, le programme doit retourner 0, sinon 1.

Erreurs Dans le cas où les séquences `x` et `y` ne sont pas du même type ou d'un type non reconnu, ou qu'une autre erreur se produit, le programme doit retourner 1, le message d'erreur est libre.

Matrice de substitution La matrice de substitution Σ, dépend du type de séquence d'entrée :

- Nucléiques (ADN ou ARN) : Σ est de taille 4×4, avec $\sigma_{i,i} = 1$, $\sigma_{i,j} = -1$, $0 \leq i \neq j < 4$.
- Protéiques (acides aminés) : Σ est de taille 23×23 et correspond à la matrice blossum62 (la dernière ligne/colonne correspondant au caractère `*` n'est pas utile).

Remarque sur l'implémentation

Si vous le souhaitez, vous pouvez ignorer les paliers et implémenter directement la version de l'algorithme du palier 3 qui fonctionne aussi quand γ est linéaire. Notez cependant que cette version est plus lente donc pour une utilisation réelle on voudrait séparer les deux implémentations.

Palier 1 (10 points)

But Gérer les entrées, implémenter la commande `score` et la version linéaire de l'algorithme.

Algorithme Dans un premier temps, on veut implémenter l'algorithme de Needleman-Wunsch dans le cas où la fonction d'évaluation des trous γ est linéaire, c'est à dire $\gamma(x) = ex$ où x est la taille d'un trou (nombre de -) et $e < 0$ est un coefficient entier. Pour deux séquences $x = x_1, \ldots, x_M$ et $y = y_1, \ldots, y_N$ de tailles respectives M et N, l'algorithme consiste à remplir la matrice de score S de taille $(M+1) \times (N+1)$ en utilisant la formule suivante :

$$s_{i,j} = \max \begin{cases} \sigma_{x_i, y_j} + s_{i-1,j-1} & \text{(substitution)} \\ e + s_{i-1,j} & \text{(insertion)} \\ e + s_{i,j-1} & \text{(délétion)} \end{cases},$$

pour $0 < i \leq M$, $0 < j \leq N$ et pour les bords,

$$s_{i,0} = i \times e, \quad s_{0,j} = j \times e.$$

Pour ce palier on aura toujours γ linéaire (donc $o = 0$) et il n'y a pas à produire d'alignement, juste le score.

Palier 2 (5 points)

But Implémenter la commande `align`.

Modification de l'algorithme Pour obtenir l'alignement, il faut considérer dans l'algorithme en plus de la matrice de score S, une seconde matrice appelée matrice de retour (*backtrack*) B de taille $(M+1) \times (N+1) \times 2$. B contient pour chaque position (i, j) la position (k, l) de la case utilisée pour remplir la case $s_{i,j}$. On remplit B en même temps que S.

Algorithme de retour Une fois les matrices S et B remplies, il faut faire l'étape de retour pour obtenir l'alignement. On commence à la position de la case bas droite de B (coordonnées $(u = M, v = N)$) et on met à jour u, v en utilisant les coordonnées contenues dans $b_{u,v}$ jusqu'à atteindre la case haut gauche de coordonnées $(u = 0, v = 0)$. A chaque étape, on progresse d'un caractère dans l'alignement en fonction de la valeur de $b_{u,v}$:

$b_{u,v}$	Alignement pour x	Alignement pour y
$(u-1, v-1)$	x_u	y_v
$(u-1, v)$	x_u	-
$(u, v-1)$	-	y_v

Palier 3 (5 points)

But Produire un alignement quand le coefficient o de γ est $\neq 0$.

Nouvel algorithme Dans le cas où $o \neq 0$ dans γ alors on ne peut plus utiliser la formule linéaire du palier 1 pour remplir S. A la place, il faut utiliser la formule suivante :

$$s_{i,j} = \max \begin{cases} \sigma_{x_i,y_j} + s_{i-1,j-1} & \text{(substitution)} \\ \max_k(o + e \times k + s_{i-k,j}), \quad 1 \leq k \leq i & \text{(insertion)} \\ \max_k(o + e \times k + s_{i,j-k}), \quad 1 \leq k \leq j & \text{(délétion)} \end{cases},$$

c'est à dire que dans le cas d'une insertion (resp. délétion), on cherche la case à une distance horizontale (resp. verticale) k de la case actuelle tel que le score de celle-ci $s_{i-k,j}$ (resp. $s_{i,j-k}$) plus le score d'un trou de taille k, $\gamma(k) = ek + o$, soit le plus grand possible. Les bords sont remplis en suivant

$$s_{i,0} = \gamma(i) = ei + o \quad s_{0,j} = \gamma(j) = ej + o.$$

Le remplissage de la matrice de retour B s'ajuste au remplissage de S, dans le cas d'une insertion (resp. délétion) on aura $b_{i,j} = (i-k, j)$ (resp. $b_{i,j} = (i, j-k)$) où k correspond à la taille du trou choisi pour remplir $s_{i,j}$ et $b_{i,0} = b_{0,j} = (0, 0)$.

Nouvel algorithme de retour Lors de l'étape de retour, il faut aussi prendre en compte la taille du trou, de la manière suivante :

$b_{u,v}$	caractère(s) pour x	caractère(s) pour y
$(u-1, v-1)$	x_u	y_v
$(u-k, v)$	$x_{(u-k+1):u}$	$k\times$ -
$(u, v-k)$	$k\times$ -	$y_{(v-k+1):v}$

où $x_{u-k+1:u}$ (resp. $y_{v-k+1:v}$) est la sous-séquence de x (resp. y) contenue entre les positions $u-k+1$ et u (resp. $v-k+1$ et v) inclues. Par exemple $x_{u:u} = x_u$ et $x_{u-1:u} = x_{u-1}x_u$.

Exemples de comportements

```
$ ./nwalign.py score YOYO YOYO
ERROR MSG
$ echo $?
1
$ ./nwalign.py yoyo AAAA AAA
ERROR MSG
$ echo $?
1
$ ./nwalign.py score AAAA AAAA
4.0
$ echo $?
0
$ ./nwalign.py score ATG ACTG
2.0
$ ./nwalign.py score ATG ACTG
A-TG
ACTG
$ ./nwalign.py score TAT ATGAC
-1.0
$ ./nwalign.py align TAT ATGAC
-T-AT
ATGAC
$ ./nwalign.py --gamma=0,-1 score TAT ATGAC
0.0
$ ./nwalign.py --gamma=0,-1 align TAT ATGAC
TAT---
-ATGAC
$ ./nwalign.py --gamma=-1,-1 score SBZSKDAMKLHLILEGSVNGHCFEIHGEGEG SLSKDAMKLHLVNGHCFNIHGRGEG
125.0
$ ./nwalign.py --gamma=-1,-1 align SBZSKDAMKLHLILEGSVNGHCFEIHGEGEG SLSKDAMKLHLVNGHCFNIHGRGEG
SBZSKDAMKLHLILEGSVNGHCFEIHGEGEG
S-LSKDAMKLHL-----VNGHCFNIHGRGEG
```