FpA : Informatique

Victor Wetzel

Fondamentaux pour ATIAM

Master 2 SpI, ATIAM
Septembre-Octobre 2017

Table des matières

1 Implémentation de l'algorithme d'alignement de Neddleman-Wunsch

L'objet de cette partie est de donner quelques éléments sur mon implémentation de l'algorithme de Needle-Wunsch.
On rappelle que cet algorithme permet de trouver les meilleurs alignements de chaînes de charactères symboliques (?). L'algorithme s'appuie sur la construction d'une matrice de similarité entre les deux éléments que l'on veut aligner. J'ai opté pour une approche orientée objet.

1.1 Objet : *nw.slot*

Chaque emplacement de la matrice contient un objet *slot*. Ses variables de champs sont :

— Un couple de symboles il s'agit des deux caractères que l'on aligne à cet endroit particulier de la matrice.

— Une pénalité Score à retrancher aux cases adjacentes en fonction d'une correspondance ou non-correspondance des deux caractères (*match* ou *mismatch*)

— Un score Calculé en fonction des case : en haut, en haut à gauche, à gauche. à chacune des cases adjacentes listées est ajoutée une pénalité en fonction de. Le score de la case est le maximum de ces trois sous-scores.

— Une direction L'algorithme mémorise l'origine de ce score.

1.2 Objet : *nw.matrix*

La matrice elle même est un objet. Les variables de champs sont :

— matrix la matrice de similarité contenant les *slot*

— N et M caractérisent la taille de la matrice

— path Liste tous les alignements possibles

— strA strB Les chaînes de symboles que l'on compare

2 Opérations sur la base de donnée

2.1 Tri de la base de donnée par ordre décroissant du nombre de tracks

Dans cette partie, j'ai implémentée l'algorithme de *quicksort* de Lomuto, en inversant l'ordre du tri. Le pseudo code de l'algorithme original est disponible sur la page wikipédia (https ://en.wikipedia.org/wiki/Quicksort), listé ci-dessous :

```
algorithm quicksort(A, lo, hi) is
    if lo < hi then
        p := partition(A, lo, hi)
        quicksort(A, lo, p-1)
        quicksort(A, p + 1, hi)

algorithm partition(A, lo, hi) is
    pivot := A[hi]
    i := lo - 1
    for j := lo to hi - 1 do
        if A[j] < pivot then
            i := i + 1
            swap A[i] with A[j]
    if A[hi] < A[i + 1] then
        swap A[i + 1] with A[hi]
    return i + 1
```

Listing 1 – Algorithme quicksort de Lomuto

Références