# TD $12^4$

**Exercice 1.** *Bienvenue...*

Un problème récurrent en bioinformatique est l'alignement de séquences. Si l'on a deux mots $u$ et $v$ formés à partir d'un alphabet $\mathcal{A}$ (par exemple les nucléotides : $\mathcal{A} = \{G, T, A, C\}$), on appelle alignement de $u$ et $v$ un couple de mots $u'$, $v'$ sur l'alphabet $\mathcal{A} \cup \{e\}$ (où $e \notin \mathcal{A}$ est le caractère d'espacement). $u'$ et $v'$ satisfont les propriétés suivantes :
– $u'$ et $v'$ ont la même longueur ($|u'| = |v'| = n$)
– si on supprime tous les $e$ dans $u'$ et $v'$ on obtient respectivement $u$ et $v$
– les lettres $e$ ne sont pas à la même position dans $u'$ et $v'$
Par exemple un alignement de $u = CGATTAG$ et $v = GATCGA$ est

$$\begin{aligned} u' &= CGATTeAG \\ v' &= eGATCGAe \end{aligned}$$

Afin de déterminer le meilleur alignement, il nous faut une métrique. Soit $p$ une fonction de similarité entre un couple de lettres. La valeur de $p(a, b)$ est un réel d'autant plus grand que les lettres sont considérées comme similaires d'un point de vue biologique. On prendra donc une valeur négative lorsque les lettres sont différentes. On a donc $b \neq a, p(a, a) > 0 > p(a, b)$, la fonction de similarité est également symétrique : $p(a, b) = p(b, a)$. De plus, on se donne un réel $q < 0$ qui exprime la similarité d'une lettre avec un espacement, elle est indépendante de la lettre, on a donc $\forall a \in \mathcal{A}, p(a, e) = p(e, a) = q$. Le score d'un alignement est simplement la somme des similarités entre les lettres de $u'$ et $v'$ :

$$\sum_{i=1}^{n} p(u'_i, v'_i)$$

1. On pose $p(a, b) = -1$ pour $a \neq b$, $p(a, a) = 5$ et $q = -2$. Quel est le score de l'alignement donné en exemple ? Donnez le meilleur alignement et le score correspondant pour les chaînes suivantes : $GATTACA$ et $ATACGTA$.

2. Donnez un algorithme permettant de trouver le meilleur alignement (celui ayant le score le plus élevé). Expliquez comment vous reconstruisez la solution.

3. Quelle est la complexité en temps et en espace de votre algorithme ?

4. Vous disposez maintenant d'une famille de $k$ chaînes de caractères $S_1, S_2, \ldots, S_k$. On souhaite obtenir un alignement des $k$ séquences, c'est à dire $k$ nouvelles chaînes de caractères $S'_1, S'_2, \ldots, S'_k$ toutes de même longueur $n$. On veut en fait minimiser la somme des distances entre toutes les paires de séquences, en reprenant le principe d'alignement de séquences précédent : on fait correspondre au mieux les séquences en ajoutant au besoin des espacements. La distance entre deux séquences étant la somme des distances caractère par caractère :

   $$\sum_{1 \leq i < j \leq k} \sum_{1 \leq l \leq n} \delta(S'_i[l], S'_j[l])$$

   avec $\delta(S_i, S_i) = 0$, $\delta \geq 0$ et $\delta$ est symétrique et respecte l'inégalité triangulaire.
   Pour $k = 2$, cela revient au problème précédent. Est-il possible d'étendre votre algorithme pour traiter le problème pour $k$ quelconque ? Quelle serait alors la complexité temporelle de l'algorithme ? Est-ce polynomial en la taille des données (en le nombre de séquences à traiter, $k$ ; et en la taille des chaînes, $n$) ?

5. Après l'avoir proprement défini, montrer que le problème de décision associé à l'alignement de $k$ séquences est NP-complet.

**Exercice 2.** *Face*

On considère une pile munie des opérations suivantes :
- $PUSH(S, x)$ : empile un objet $x$ sur la pile $S$
- $POP(S)$ : dépile le sommet de la pile $S$ et retourne l'objet dépilé
- $MULTIPOP(S, k)$ : dépile au plus $k$ objets de la pile $S$

**Algorithm 1:** $MULTIPOP(S, k)$

```
début
    tant que S ≠ ∅ et k ≠ 0 faire
        POP(S);
        k ← k − 1;
```

1. Quelle est la complexité de chacune des 3 opérations ? En déduire avec la méthode globale (méthode de l'agrégat) le coût amorti pour une suite de $n$ opérations $PUSH$, $POP$ et $MULTIPOP$ sur une pile initialement vide.
2. Même question avec la méthode des acomptes.
3. Même question avec la méthode des potentiels.
4. On souhaite implémenter une file à l'aide de deux piles, de telle façon que le coût amorti des opérations $ENQUEUE$ et $DEQUEUE$ soit $O(1)$. Comment peut-on faire ?

**Exercice 3.** *Fusion*

On s'intéresse à la fusion de deux ensembles triés, l'un de taille $m$ et l'autre de taille $n$. Les $m + n$ éléments à fusionner sont tous distincts et notés :

$$A_1 < A_2 < ... < A_m \quad \text{et} \quad B_1 < B_2 < ... < B_n$$

1. Montrer qu'il faut au moins $\lceil \log C_{m+n}^n \rceil$ comparaisons pour effectuer la fusion.
2. En déduire que pour $n = m$, il faut au moins $2n - \frac{1}{2}\log n + O(1)$ comparaisons.
3. Rappeler brièvement l'algorithme usuel de fusion et donner sa complexité.
4. Démontrer que pour $n = m$, on ne peut pas faire mieux que l'algorithme usuel. La borne $2n - \frac{1}{2}\log n + O(1)$ ne peut donc pas être atteinte.