Cours *conception et analyse d'algorithmes*
Correction du TD 9

1. Arbre d'évolution optimal pour la distance de Hamming.

i. On considère le graphe complet à m sommets où chacun des sommets représente un des f_i et où l'arête entre f_i et f_j a un poids égal à la distance de Hamming entre ces deux mots.

On peut ensuite appliquer l'algorithme de Kruskal sur ce graphe pour obtenir un arbre d'evolution de coût minimal.

ii. On utilise la construction donnée par l'énoncé. Étant donné un arbre T d'évolution de coût k, on construit un ensemble couvrant de sommets. Remarquons que toutes les arêtes sont de poids au plus 2, en effet $w_\emptyset$ appartient à l'arbre et est à distance au plus 2 de tous les éléments. S'il existe une arête entre deux mots qui sont à distance égale à deux alors on peut diviser cette arête en une chaîne de 2 arêtes ayant chacune poids 1 et telle que tous les sommets sur cette chaîne appartiennent à F ou à G.

Soit T' l'arbre ainsi obtenu (dont le coût est le même que T). On peut à nouveau modifier T' sans changer son coût pour qu'il soit enraciné en $w_\emptyset$ et soit de hauteur 2. Dans T', chaque sommet f_p de $F\backslash\{w_\emptyset\}$ a un voisin dans G qui est par construction une extrémité de f_p dans le graphe Γ. En sélectionnant pour chaque sommet f_p de F un de ses voisins, on obtient un ensemble de sommets de G couvrant toutes les arêtes et de cardinal inférieur à $k-m$.

Réciproquement, étant donné un ensemble couvrant V de sommets de cardinal inférieur ou égal à k. On construit un graphe d'évolution enraciné en $w_\emptyset$ et dont les fils sont l'ensemble V. On ajoute enfin les m feuilles de l'arbre, ce qui donne un arbre d'évolution de poids égal à $k+m$.

2. Programmation dynamique : Alignement de séquences

i. Définissons $s(i,j,k)$ comme étant le meilleur alignement entre $u_1,\dots,u_i$, $v_1,\dots,v_j$ et $w_1,\dots,w_k$. On établit alors facilement la récurrence, de façon analogue à ce qui se passe dans le cours pour le cas de deux chaînes :

$$s(i,j,k) = \min \begin{cases} s(i-1,j-1,k-1) + \Delta(u_i, v_j, w_k) \\ s(i-1,j-1,k) + \Delta(u_i, v_j, -) \\ s(i-1,j,k-1) + \Delta(u_i, -, w_k) \\ s(i,j-1,k-1) + \Delta(-, v_j, w_k) \\ s(i-1,j,k) + \Delta(u_i, -, -) \\ s(i,j-1,k) + \Delta(-, v_j, -) \\ s(i,j,k-1) + \Delta(-, -, w_k). \end{cases}$$

On a posé $\Delta(\alpha,\beta,\gamma) = \delta(\alpha,\beta) + \delta(\alpha,\gamma) + \delta(\beta,\gamma)$.

L'initialisation se fait par $s(0,0,0) = 0$ et $s(i,j,k) = +\infty$ si i, j ou k est strictement négatif.

La complexité est le coût du remplissage de la table, soit $O(n^3)$ cases, le calcul coutant $O(1)$ par case, soit $O(n^3)$ en tout.

ii. L'intuition naturelle consiste à poser $w(i',i,j',j)$ comme le meilleur alignement de $u_{i'}\dots u_i$ avec $v_{j'}\dots v_j$, puis à chercher à calculer $w(i',i,j,j')$; cela conduit cependant au mieux à un algorithme en $O(n^4)$.

On définit alors $t(i,j)$ comme le meilleur alignement d'un suffixe de $u_1 \dots u_i$ avec un suffixe de $v_1 \dots v_j$, ie : $t(i,j) = \max_{i' \leq i, j' \leq j} w(i', i, j, j')$.

La relation de récurrence pour $t(i,j)$ est similaire à celle obtenue dans le cas d'alignement de deux chaînes ; la seule différence est le fait que le meilleur alignement d'un suffixe petu correspondre au cas où l'un des deux suffixes est trivial (et dans ce cas l'autre l'est nécessairement). Cela conduit à :

$$t(i,j) - \max \begin{cases} t(i-1, j-1) + \delta(u_i, v_j) \\ t(i-1, j) + \delta(u_i, -) \\ t(i, j-1) + \delta(-, v_j) \\ 0. \end{cases}$$

Une fois la table construite, le maximum permet d'obtenir les positions i et j où se terminent α et β, ainsi que la valeur de l'alignement optimal. Pour déterminer l'endroit où commencent α et β, il suffit de chercher à construire l'alignement optimum en remontant dans la table de programmation dynamique. Le premier 0 trouvé marque le point où les chaînes α et β commencent.

3. Placement optimal de ressources

i. La version de décision du problème consiste à se fixer un entier k et à répondre à la question : existe-t-il un ensemble dominant de taille inférieure ou égale à k ?

ii. Soit $(E, (S_i)_{i \in I})$ une instance de recouvrement par ensemble où les S_i sont des sous-ensembles de E. On construit le graphe dont l'ensemble des sommets est $V = E \cup I$ tel que sa restriction à I soit le graphe complet et il existe une arête $(e, i) \in E \times I$ si et seulement si $e \in S_i$.

Soit X un ensemble dominant du graphe de taille k. On peut supposer que tous les sommets de X appartiennent à I, en effet si $x \in X$ appartient à E alors peut-être remplacé par n'importe lequel de ses voisins.

Cela nous donne donc une solution au problème de recouvrement d'ensembles de taille inférieure ou égale à k.

iii. Soit S la solution au problème k-CENTRES. Si $\phi(S) = 1$, alors S est un ensemble dominant du graphe G, puisque pour tout $y \notin S$, il existe $x \in S$ tel que $d(x, y) = 1$ ce qui signifie que (x, y) est une arête de G.

iv. S'il existe un algorithme polynomial qui renvoie un ensemble S de taille k tel que $\phi(S) < 2\phi^\star$, alors dans la construction précédente, on obtiendrait une solution au problème. Il est en effet évident que $\Phi(S) = 1$ ou 2. On aurait donc un algorithme polynomial pour résoudre ensemble dominant.