CORRIGÉ DE LA PC 7

GUILLAUME HANROT

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

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

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

Il suffit en fait d'initialiser la valeur $s(0,0,0) = 0$, 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 avec un coût $O(1)$ par case, soit $O(n^3)$ en tout. Cette approche se généralise sans difficulté au cas de k chaînes, et fournit une complexité de $O((2n)^k)$, qui est donc exponentielle en k.

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 toutefois au mieux à un algorithme en $O(n^4)$.

Définissons donc plutôt $t(i,j)$ comme le meilleur alignement d'un suffixe de $u_1 \dots u_i$ avec un suffixe de $v_1 \dots v_j$, En d'autres termes, $t(i,j) = \max_{i' \leq i, j' \leq j} w(i',i,j',j)$.

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

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

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

Date: 29 mars 2004.

dans la table de programmation dynamique ; le premier 0 trouvé marque le point où les chaînes α et β commencent.

2. i. On construit l'automate non-déterministe correspondant au motif, on le rend déterministe et minimal ; cet automate fournit alors un moyen de rechercher le motif en temps linéaire.

ii. On construit en parallèle deux automates identiques reconnaissant le motif cherché, avec des transitions non-déterministes de l'état i du premier vers l'état $(i+1)$ du deuxième pour des caractères différents du i-ème caractère du motif. On rend cet automate déterministe et minimal ; cet automate fournit alors un moyen de rechercher le motif en temps linéaire.

Une autre possibilité consiste à construire tous les motifs différant en une position au plus du motif initial.

3. C'est essentiellement le cours. On pose $S_i = \sum_{k=1}^{i} X_k$. Le problème de la sous-séquence maximale consiste alors à trouver $j < i$ tels que $S_i - S_{j-1}$ soit maximal. La façon la plus simple de le faire consiste à calculer les S_i, tout en stockant le minimum courant des valeurs dans une variable m et le maximum des $S_i - m$. Le pseudo-code suivant trouve i, j et la valeur optimale :

$S \leftarrow 0, m \leftarrow 0, M \leftarrow 0, \mu \leftarrow 0, \mu' \leftarrow 0, \nu \leftarrow 0$
Pour i de 1 à n faire
  $S \leftarrow S + X_i$
  Si $S < m$ faire $m \leftarrow S$, $\mu' \leftarrow i$.
  sinon si $S - m > M$ faire $M \leftarrow S - m$, $\nu \leftarrow i$, $\mu \leftarrow \mu'$ finsi.
  finsi.
finpour.
Renvoyer (S, μ, ν).

ii. On cherche alors à optimiser $\sum_{k=j}^{i} X_k$ avec $j \leq i$ soit $\sum_{k=1}^{i} X_k + \sum_{k=j}^{n} X_k$ avec $j > i$. Si l'on pose $\sigma = \sum_{k=1}^{n} X_i$, la somme dans le second cas vaut exactement $\sigma + S_i - S_{j-1}$. L'optimum cherché est alors

$$\max(\max_{j \leq i} S_i - S_{j-1}, \max_{j > i} S_i - S_{j-1} + \sigma),$$

qui se calcule de façon analogue à ci-dessus. Dans le cas favorable $\sigma = 0$, le résultat vaut $\max S_i - \min S_i$.

iii. On a traité un exercice analogue au TD2 ; il s'agit tout simplement d'une technique de type programmation dynamique / parcours en profondeur.

4. Supposons D ultramétrique, et montrons alors par récurrence que sur tout chemin joignant i à j il existe une arête de poids au moins égal à $D(i, j)$. La récurrence se fait sur la longueur du chemin. Si le chemin est de longueur 1, c'est trivial. Si c'est vrai pour un chemin de longueur l, on considère un chemin de longueur $l+1$, et k le premier point après

i sur ce chemin. Par récurrence il existe entre k et j une arête α de poids au moins égal à $D(k, j)$.

Notons maintenant que

- Soit $D(i, j)$ est égal au maximum des trois distances $D(i, j), D(j, k), D(k, i)$, auquel cas comme une autre distance réalise ce maximum on a $D(i, j) = \max(D(j, k), D(k, i))$.
- Soit $D(i, j)$ est plus petit que le maximum et alors $D(i, j) < \max(D(j, k), D(k, i))$.

Dans tous les cas, $D(i, j) \leq \max(D(j, k), D(k, i))$. Si le premier terme réalise le maximum, l'arête α convient. Sinon, l'arête (i, k) convient.

Montrons maintenant le sens direct de la question. Le chemin construit dans l'arbre couvrant minimum est, par l'algorithme glouton, celui qui minimise $\max D(\alpha)$, α parcourant les arêtes du chemin. On a montré que ce maximum vaut au moins $D(i, j)$, et il vaut au plus $D(i, j)$ grâce au chemin constitué de l'arête (i, j). Cela montre que le chemin dans l'arbre couvrant contient une arête de poids $D(i, j)$.

Réciproquement considérons (i, j, k) trois sommets. Il existe une étape de la construction de l'arbre couvrant dans laquelle deux des sommets (par exemple j et k) sont dans une composante connexe et le troisième dans une autre. Au moment où les deux composantes connexes sont reliées par une arête α, on a donc $D(\alpha) \leq \max(D(i, j), D(j, k))$. En outre, le chemin de i à j et de j à k passe nécessairement par α, et cette arête est celle de poids maximum sur le chemin, donc par hypothèse $D(\alpha) = D(i, j) = D(i, k)$. Mais par hypothèse sur le chemin de j à k, il existe une arête de poids $D(j, k)$ qui a été ajoutée avant α, donc $D(j, k) \leq D(\alpha)$, ce qui conclut la preuve.