2018 - M1 STL - CPA - Christoph Dürr

Question et réponse type en programmation dynamique

Rendre un mot bien parenthésé

Un mot w sur l'alphabet $\{(,)\}$ est bien parenthésé si on peut associer chaque parenthèse ouvrante x à une parenthèse fermante à droite de x et que les associations soient *sans collision* et *ne se croisent pas*. Voir Figure 1.

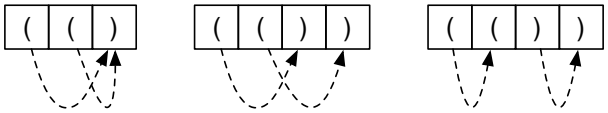

Figure 1: à gauche une association avec collision, au centre une avec croisement, et à droite une sans collision ni croisement.

Formellement on associe à un mot $w \in \{(,)\}^*$ un compteur $p[w]$ défini comme le nombre de parenthèses ouvrantes dans w moins le nombre de parenthèses fermantes dans w. On dit que $w \in \{(,)\}^n$ est bien parenthésé si $p[w] = 0$ et que pour tout $i = 1, \ldots, n-1$, on ait $p[w_{1,\ldots,i}] \geq 0$, où $w_{1,\ldots,i}$ est le mot constitué des i premières lettres de w.

bien parenthésé	mal parenthésé
ε	(
()	)
()()	)(
(())	(()
(())()	())(()

Il est facile de déterminer si un mot est bien parenthésé, en effectuant un unique parcours pendant lequel on maintien un compteur représentant $p[w_{1,\ldots,i}]$. Mais si un mot n'est pas bien parenthésé, on peut toujours le rendre parenthésé en supprimant des parenthèses à différents endroits de la chaîne. Étant donnée un mot $w \in \{(,)\}^n$, déterminer le nombre minimal de parenthèses à supprimer pour rendre w bien parenthésé. Une complexité en $O(n^3)$ est attendue.

Réponse type : Soit $A_{i,j}$ le nombre minimal de parenthèses à supprimer pour rendre le mot $w_{i,\ldots,j}$ bien parenthésé, avec $1 \leq i \leq j \leq n$. Nous étendons la notation à $i = j+1$ et définissons $A_{j+1,j} = 0$, correspondant à la solution optimale pour le mot vide.

La récursion est sur la valeur $j - i$. Le cas de base est $A_{j+1,j} = 0$ pour le mot vide. Pour la récursion avec $j \geq i$, considérons la première lettre w_i du mot $w_{i,\ldots,j}$ et considérons une solution optimale. Soit w_i est supprimée dans la solution optimale, et dans ce cas $A_{i,j} = 1 + A_{i+1,j}$ par composition des solutions. Soit cette lettre est associée à une lettre w_ℓ avec $i+1 \leq \ell \leq j$. Cette situation n'est possible seulement si $w_i = ($ et $w_\ell =)$. Les

chaînes $w_{i+1,\ldots,\ell-1}$ et $w_{\ell+1,\ldots,j}$ forment des sous-problèmes indépendants car les associations de parenthèses ne peuvent pas se croiser. Dans ce cas nous avons alors la récursion $A_{i,j} = A_{i+1,\ell-1} + A_{\ell+1,j}$.

En résumé le programme dynamique consiste en $O(n^2)$ variables $A_{i,j}$, dont le cas de base est $A_{i,j} = 0$ si $i = j + 1$ et sinon

$$A_{ij} = \min \left\{ 1 + A_{i+1,j}, \min_{\ell} A_{i+1,\ell-1} + A_{\ell+1,j} \right\},$$

où le minimum intérieur est pris sur les valeurs $i + 1 \leq \ell \leq j$ avec $w_i = ($ et $w_\ell =)$. Par convention le minimum sur l'ensemble vide est défini comme valant $+\infty$. La réponse au problème est $A_{1,n}$. Ces variables doivent être calculées dans l'ordre suivant. D'abord pour tout $j = \{1, \ldots, n\}$ $A_{j+1,j} = 0$ est posé. Puis pour tout $k = 0, \ldots, n-1$ (et dans cet ordre) et pour tout $j \in \{1 + k, \ldots, n\}$, $A_{j-k,j}$ est calculée en utilisant la récursion ci-haute.

Comme il y a $O(n^2)$ variables et que chacune est calculée par minimisation sur $O(n)$ alternatives, le programme dynamique a une complexité en temps de $O(n^3)$. Il existe de meilleurs algorithmes pour ce problème, mais ce n'est pas le propos pour l'instant.