2I003 – QCM sur le projet
durée : 15min

22 novembre 2016

Barème : chaque réponse correcte rapporte 0,5 point, chaque mauvaise réponse fait perdre 0,5 point. L'absence de réponse est notée 0.

On rappelle les définitions données dans le projet.

Soit $\mathcal{A}$ un alphabet et deux mots $u = u_0 \cdots u_{n-1}$ et $v = v_0 \cdots v_{m-1}$ composés respectivement de n et m lettres de $\mathcal{A}$. Un sous-mot commun de taille p est défini par deux suites d'indices croissantes $0 \leq i_1 < i_2 < \cdots < i_p \leq n-1$ et $0 \leq j_1 < j_2 < \cdots < j_p \leq m-1$ telles que pour tout $k \in \{1, \ldots, p\}, u_{i_k} = v_{j_k}$. Le mot vide correspond à $n = 0$.

Par exemple, si on considère l'alphabet latin et les mots $u = abracadabra$ et $v = caramba$, on observe que $araba$ est un sous-mot commun. Les suites associées sont $(i_1, i_2, i_3, i_4, i_5) = (0, 2, 3, 8, 10)$ et $(j_1, j_2, j_3, j_4, j_5) = (1, 2, 3, 5, 6)$.

Pour tout couple $(i, j) \in \{0, \cdots, n\} \times \{0, \ldots, m\}$, on note $L(i, j)$ la taillle maximale d'un sous-mot commun pour les préfixes $u_0 \cdots u_{i-1}$ et $v_0 \cdots v_{j-1}$ de u et v.

On rappelle aussi qu'au cours du projet, vous deviez programmer :

- une fonction récursive `tailleMaxRec` qui calcule $L(i, j)$ grâce à des relations de récurrence sur L ;
- une fonction itérative `CalculMaxIt` qui calcule itérativement la matrice de tous les $L(i, j)$;
- une fonction récursive `PlusLongMot` qui calcule récursivement un plus long sous-mot commun de $u_0 \cdots u_{i-1}$ et $v_0 \cdots v_{j-1}$.

Questions.

Énoncé	**Vrai**	**Faux**
1) Si $u = abbab$ et $v = aaba$, alors $L(3, 3) = 1$.	☐	☐
2) $L(i, 0) = 0$ pour tout $i \in \{0, \ldots, n\}$.	☐	☐
3) À j fixé, la suite $\big(L(i, j)\big)_{0 \leq i \leq n}$ est croissante.	☐	☐
4) À j fixé, la suite $\big(L(i, j)\big)_{0 \leq i \leq n}$ est strictement croissante.	☐	☐
5) Si $u_{i-1} = v_{j-1}$, alors on a $L(i, j) = 1 + L(i-1, j-1)$.	☐	☐
6) La fonction récursive `tailleMaxRec` de paramètres i et j a une complexité exponentielle en $i + j$.	☐	☐
7) La fonction itérative `CalculMaxIt` a une complexité exponentielle en $n+m$.	☐	☐
8) Pour vérifier expérimentalement qu'une fonction $f(n)$ a une croissance exponentielle, on trace simplement le graphe de $f(n)$ en fonction de n.	☐	☐
9) Pour vérifier expérimentalement qu'une fonction $f(n)$ a une croissance polynomiale de la forme $\mathcal{O}(n^\alpha)$, on trace $f(n)/n^\alpha$ en fonction de n.	☐	☐
10) La complexité en pire cas de la fonction `PlusLongMot` vient lorsque les deux mots sont égaux.	☐	☐