L3 Informatique **année 2012-2013**

Méthodes Algorithmiques

Examen du 15 Avril 2013
Responsable : Sophie Pinchinat

Seule une feuille manuscrite en format A4 recto-verso est autorisée.
Le barème est donné à titre indicatif.

Exercice 1 : Diviser pour Régner (5 points) Vous disposez de trois algorithmes pour résoudre un problème donné.

- L'Algorithme A résoud le problème en le divisant en 5 sous-problèmes de taille moitié moindre, en résolvant ces sous-problèmes récursivement, et en recombinant les solutions en temps linéaire.
- L'Algorithme B résoud le problème (de taille n) en résolvant récursivement 2 sous-problèmes de taille $n-1$ et en recombinant les solutions en temps constant.
- L'Algorithme C résoud le problème (de taille n) en le divisant en 9 sous-problèmes de taille $n/3$, en résolvant récursivement chaque sous-problème, et en recombinant les solutions en temps $O(n^2)$.

1. Si on note $T_X(n)$ le temps de calcul de l'Algorithme X sur une entrée de taille n, donnez les équations qui décrivent $T_X(n)$ pour chaque $X \in \{A, B, C\}$.
2. Quel est asymptotiquement le temps d'exécution de chaque algorithme ?
3. Lequel choisiriez-vous ?

Exercice 2 : Identification (8 points) Soit un mélange de n peptides. On connaît la masse moléculaire de chacun des peptides présents $w_1, \ldots, w_n$ (toutes distinctes). On sait séparer les peptides mais ensuite il faut les identifier, c-à-d. associer à chaque peptide séparé j une masse moléculaire w_i d'un des n peptides. Pour cela, on mesure la masse moléculaire de chaque peptide à l'aide d'un spectromètre de masse, obtenant ainsi une masse x_j pour le j-ème peptide.

Malheureusement le spectromètre n'est pas exact et on cherche donc à associer à chaque mesure x_j une masse moléculaire w_{i_j} correspondante de sorte que :

1. l'association des x_j vers les w_i est injective : deux peptides de mesures différents $x_j \neq x_\ell$ sont associés à des masses moléculaires différentes $w_{i_j} \neq w_{i_\ell}$;
2. l'association minimise la marge d'erreur, décrite par la valeur $\sum_{j=1}^{n} |x_j - w_{i_j}|$.

On peut formaliser le problème de la manière suivante. Étant données deux ensembles d'entiers $X = \{x_1, \ldots, x_n\}$ et $W = \{w_1, \ldots, w_n\}$, une *association* entre X et W est une fonction bijective $A : X \to W$, et sa *valeur* est $v(A) = \sum_{j=1}^{n} |x_j - A(x_j)|$. Une *inversion* de l'association A est un couple (j, k), où $1 \leq j, k \leq n$, tel que $x_j > x_k$ mais $A(x_j) < A(x_k)$.

Le reste de l'exercice consiste à résoudre :

Problème IDENTIFICATION

Entrée : Deux ensembles d'entiers $X = \{x_1, \ldots, x_n\}$ et $W = \{w_1, \ldots, w_n\}$
Sortie : Une association entre X et W de valeur minimum.

1. Donnez un exemple non trivial avec $n = 3$ d'ensembles X et W et d'une association optimale dont vous donnerez la valeur.

2. Décrivez un algorithme efficace qui étant donnés deux ensembles X et W calcule une association entre X et W de valeur la meilleure possible parmi l'ensemble de toutes les associations.

   Quelle est la complexité de votre solution ?

3. Votre algorithme est-il optimal ? Justifiez votre réponse.

Exercice 3 : Ordonnancement sur un ensemble (7 points) Vous disposez de n objets que vous désirez ordonner avec les deux relations "$<$" et "$=$". Par exemple, il y 13 ordonnancements entre 3 objets :

$a = b = c$	$a = b < c$	$a < b = c$	$a < c < b$
$a < b < c$	$a = c < b$	$b < a = c$	$b < a < c$
$b = c < a$	$b < c < a$	$c < a = b$	$c < a < b$
$c < b < a$			

On cherche à développer un algorithme de programmation dynamique capable de déterminer en fonction de n, le nombre d'ordonnancements différents.

On remarque qu'un ordonnancement de n éléments se décompose en un "bloc" (maximal) de taille $1 \leq k \leq n$ de premiers éléments tous égaux : par exemple pour l'ordonnancement $\boxed{a = b = c}$ le bloc est de taille 3, pour l'ordonnancement $\boxed{\text{c}} < b < a$ le bloc est de taille 1, et pour l'ordonnancement $\boxed{\text{b=c}} < a$ le bloc est de taille 2. Ce premier critère permet de distinguer quelques ordonnancements, mais pas tous. En effet, il existe plusieurs ordonnancements correspondant à une même taille de bloc : les ordonnancements $\boxed{\text{b=c}} < a$ et $\boxed{\text{a=c}} < b$ ont tous deux un bloc de taille 2, mais leur différence porte plutôt sur le choix des 2 éléments formant le bloc ($\{b, c\}$ pour l'ordonnancement $b = c < a$ et $\{a, c\}$ pour l'ordonnancement $a = c < b$).

Ce point de vue permet de réorganiser les 13 ordonnancements par taille croissante de bloc comme suit : $\{a < b < c, a < c < b, b < a < c, b < c < a, c < a < b, c < b < a, a < b = c, b < a = c, c < a = b\} \cup \{a = b < c, a = c < b, b = c < a\} \cup \{a = b = c\}$.

Pour une taille de bloc k fixée, on notera $\binom{i}{k}$ le nombre de manières de choisir k éléments parmi i (ceux qui peupleront le bloc de taille k).

1. Proposez une équation récursive pour calculer le nombre $Ord(i)$ d'ordonnancements différents pour i éléments. Cette équation combine les différents choix pour les éléments d'un bloc et les différents ordonnancements sur les éléments restants.

2. Démontrez que $\binom{i}{k} = \binom{i-1}{k-1} + \binom{i}{k-1}$.

3. Utilisez les deux questions précédentes pour décrire un algorithme de programmation dynamique qui calcule $Ord(n)$.

4. Quel est le temps d'exécution de votre algorithme ?

5. Quelle est son occupation en espace ?