

Master Biologie Structurale et Bioinformatique

Examen écrit – 9h - 11h, mardi 15 décembre 2015

Formuler des réponses concises, mais précises. Documents autorisés.
Moyens de communication (téléphone, ordinateurs, etc.) interdits.
Les trois exercices sont indépendants les uns des autres.

Exercice 1 : Recherche d'une star

Dans un groupe de n personnes (numérotées de 1 à n pour les distinguer), on définit **une star** comme quelqu'un qui ne connaît personne mais que tous les autres connaissent. Pour démasquer une star, s'il en existe une, vous avez juste le droit de poser des questions, à n'importe quel individu i du groupe, du type "est-ce que vous connaissez j ?" (la question est appelée par une fonction notée *connais*(i,j) $\rightarrow$ booléen). On suppose que les individus répondent la vérité. On veut un algorithme qui trouve une star, s'il en existe, ou sinon qui garantit qu'il n'y a pas de star dans le groupe, en posant le moins de questions possibles.

- 1.1 Combien peut-il y avoir de star dans un groupe de n personnes ?
- 1.2 Ecrire un algorithme appelant un minimum de fois la fonction *connais*() et donner sa complexité.

Exercice 2 : Le problème du sac à dos

On se donne n objets ayant pour valeurs $c_1, \dots, c_n$ et pour poids $w_1, \dots, w_n$. Le but est de remplir le sac à dos en maximisant la valeur somme sous la contrainte que le poids global reste inférieur ou égal à W , W étant la contenance maximale du sac.

2.1 Ecrire un algorithme naïf qui remplit le sac en tenant en compte le rapport valeur sur poids. Montrer par un contre exemple que cet algorithme n'est pas optimal.

2.2 Solution optimale par récurrence

Posons $C(v,i)$ comme l'expression du meilleur coût pour remplir un sac de poids v avec les i premiers objets. Résoudre le problème de remplissage du sac de taille W avec les n objets revient alors à calculer et déterminer $C(W,n)$.

Exprimer le calcul optimal de $C(v,i)$ par une relation de récurrence, selon laquelle on choisit d'ajouter le dernier élément i de valeur c_i et poids w_i , soit de ne pas le prendre. En déduire un algorithme récursif de calcul.

Exercice 3 : Sous-séquences croissantes

Ecrire une procédure *determinerSequences()*, qui à partir d'un tableau d'entiers t de n éléments, fournit le nombre de sous-séquences strictement croissantes de ce tableau, ainsi que les indices de début et de fin de la plus grande sous-séquence.

Exemple : t un tableau de 15 éléments : 1,2,5,3,12,25, 13,8,4,7,24,28,32,11,14

Les séquences strictement croissantes sont : $\langle 1,2,5 \rangle$, $\langle 3,12,25 \rangle$, $\langle 13 \rangle$, $\langle 8 \rangle$, $\langle 4,7,24,28,32 \rangle$, $\langle 11,14 \rangle$. Le nombre de sous-séquences est : 6 et la plus grande sous-séquence est : $\langle 4,7,24,28,32 \rangle$.

Quelle est la complexité de votre algorithme.