Examen TPGO 2020/2021 // Doc non autorisés // Durée : 2H

Exercice 1

Un algorithme récursif relatif à la technique de conception "diviser pour résoudre" a comme équation de recurrence T(n) = aT(n/b) + f(n) et T(1) = c (a, b, et c sont des constantes)
où n est la taille des entrées, a est le nombre de sous-problèmes, n/b est la taille des sous-problèmes, et f(n) est le temps additionnel exigé par la décomposition.
Quelle est sa complexité ? L'appliquer pour T(n) = 3T(n/2) + n

Exercice 2

Considérons le problème du k-centre suivant : Étant donné n villes et les distances entre chaque paire de villes, sélectionnez k villes (pour placer par exemple des guichets automatiques) de telle sorte que la distance maximale d'une ville à un guichet soit minimale.
- Considérer n=4 villes V1, V2, V3 et V4 avec les distances suivantes entre elles : (V1, V2) = 10; (V1, V3) = 7; (V1, V4) = 5; (V2, V3) = 6; (V2, V4) = 8; (V3, V4) = 12.
Dresser une trace pour k=2. - Est-il un problème de décision ? - Si oui donner sa classe. Justifier, - Donner une solution gloutonne à ce problème et la dérouler sur l'exemple considéré.
N. B : On démontre que le problème du vertex cover (ensemble dominant) est réductible polynomialement vers le problème du K-centre (sachant que vertex cover est défini comme suit:Existe-t-il un sous ensemble W de V ayant une taille k et tel que pour chaque arc (u,v) dans E, au moins une extrémité (u ou v) appartient à W ?)

Exercice 3

Considérer le programme suivant : r := 1; i := 0; TQ i < m do r := r ∗ n; i := i + 1 FTQ ayant pour postcondition r = n^m.
- Trouvez sa précondition, - Dresser l'arbre de preuve avec un invariant X, - Proposer X et compléter la preuve.
Rappel règle de la boucle : Démontrer E {Tq B: P Ftq } E & Non B revient à montrer E & B {P} E. E étant l'invariant de boucle.

Exercice 4

Écrire le prédicat Prolog Partie(L, K) qui est vrai si L est une partie (sous ensemble) de K. Donner les arbres de preuve pour Partie ([4,3], [2, 3,5,4]) et pour Partie(X, [2, 3, 5, 4]).

Exercice 5

- Écrire la fonction Lisp Concat (L1, L2) qui concatène deux listes L1 et L2.
- Évaluer (AND FALSE TRUE) avec TRUE = λx. λy. x ; FALSE = λx. λy. y et AND = λb1. λb2. b1 b2 FALSE