Epreuve finale : Algorithmique avancée et Complexité

Exercice 1 : (9 pts)

A tout graphe orienté G=(X, U) d'ordre n on peut associer une matrice d'Adjacence M d'ordre n, dont les éléments notés M_{ij}, représente l'absence ou la présence d'un arc ayant le sommet i comme extrémité initiale et le sommet j comme extrémité terminale. Si on pose |U| = m, on peut représenter le graphe par deux vecteurs PS et LS. PS est un tableau de positions à n+1 éléments, et LS est un tableau à m éléments où :

- PS[i] : indique la case contenant le premier successeur du sommet i dans LS
- PS[1] = 1
- Pour tout $i \geq 1 \; et \; i \leq n-1, PS[i+1] = PS[i] + k$, k étant le nombre de successeur de i
- PS[n+1] = m+1
- Si un sommet i n'a pas de successeur, on aura PS[i] = PS[i+1]
- Les successeurs d'un sommet i se trouvent entre la case d'indice PS[i] et la case (PS[i+1] – 1) du tableau LS.

1. Ecrire un algorithme permettant de construire à partir de la matrice M les deux vecteurs PS et LS. Calculer la complexité temporelle et spatiale **(4 pts)**
2. Ecrire un Algorithme permettant de reconstituer la matrice M à partir des deux vecteurs PS et LS. Calculer la complexité temporelle et spatiale **(4 pts)**
3. Selon vous quelle serait la structure la plus optimale pour les opérations élémentaires dans un graphe ? Justifier votre réponse. **(1 pt)**

Solution :

1. De la matrice M vers les deux vecteurs PS et LS (3 pts algo, 1 pt complexité)

```
Début
K :=1 ;
Pour i :=1 à N
Faire   PS[i] :=k ;
        Pour j :=1 à N faire si (M[i,j] = 1) alors LS[k] := j ; k :=k+1 ;
                                Fsi ;
        Fait ;
Fait ;
PS[n+1] := m+1 ;
Fin.
```

Complexité temporelle = $O(n^2)$, Complexité spatiale = $O(n^2 + n + m) = O(n^2)$

Bon Courage !

2. Reconstitution de la matrice à partir des deux vecteurs PS et LS (3 pts algo, 1 pt complexité)

Début

K :=1 ;

Pour i :=1 à N faire Pour j :=1 à N faire M [i, j] := 0 ;

Pour i := 1 à N

Faire k := PS[i] ;

Tant que k < PS [i+1] faire M[i, LS[k]] := 1 ; k := k+1 ;

Fait ;

Fait ;

Fin.

Complexité temporelle = $O(n^2)$; Complexité spatiale = $O(n^2 + n + m) = O(n^2)$

La structure la plus optimale est les tableaux si le graphe n'est pas complet. Autrement les deux structures deviennent équivalente

Exercice 2 : (11 pts)

Considérer le problème de gestion des ascenseurs l'objectif étant de réduire au mieux le nombre d'aller-retour effectué. Le problème se définit comme suit : à l'arrivé d'un certain nombre de personne un concierge se charge de déterminer le nombre maximal des personnes pouvant prendre l'ascenseur en tenant compte de leur poids et de la limite de surcharge pondérale défini par l'ascenseur ; autrement dit, il doit déterminer le plus grand sous ensemble de personnes pour lesquelles la somme des poids ne dépasse pas la limite de surcharge pondérale. La définition formelle du problème est comme suit :

Instance : Étant donné un ensemble *E* de *n* entiers positifs, et une somme cible S

Question : Existe-t-il le plus grand sous-ensemble de *E* dont la somme des éléments est S ?

Exemple : Soit E = {81, 72, 17, 100, 90, 15, 21, 19, 45, 70, 120, 60, 35, 110, 85}, et S = 300

1. Illustrer sur l'exemple les étapes de construction d'une solution en spécifiant la modélisation la plus adéquate **(3 pts)**
2. Estimer approximativement la taille de l'arbre de résolution et en déduire l'ordre de complexité de l'algorithme de résolution **(2 pts)**
3. Quelles sont les critères que doit satisfaire une solution donnée S' pour être valide **(1 pt)**
4. Proposer un algorithme de validation d'une solution donnée S'et calculer sa complexité. **(4 pts)**
5. En déduire la classification associée au problème étudié. Justifier votre réponse **(1 pt)**

Solution :

Bon Courage !

1. Illustration et modélisation (1 pt modélisation, 2 pts illustration)

Selon la modélisation il est possible d'avoir différentes représentation de l'arbre de résolution

   a. Modélisation la plus adéquate vecteur binaire. Une solution est représenté sous forme d'un vecteur binaire noté A ; de dimension N (N étant la taille de l'ensemble E en entrée). A[i] représente l'absence ou la présence de l'individu i associé dans l'ensemble E. Initialement A[i] := 0, pour tout i. Au fil de la résolution, les individus seront rajoutés progressivement donc la case associé dans la solution sera mise à 1 jusqu'à atteindre une solution binaire pour laquelle la somme des poids est égale à 300. Le cas inférieur ou égale peut aussi être toléré à condition que la solution retenue soit la plus proche possible 300. i.e., face à plusieurs solutions qui vérifient la propriété <=, la plus grande sera retenue
   b. Modélisation en sous ensemble. Une solution correspond à un vecteur d'entier noté A de dimension p. chaque A[i] représente un individu de l'ensemble de départ E.

Illustration 1ère modélisation (Schématisé sous forme d'un arbre binaire)

Un arbre binaire à N niveau, N étant le nombre d'individus dans l'ensemble E. A chaque niveau une personne peut être prise ou non. A chaque niveau la somme actuelle est calculé pour décider si la branche mérite d'être étendu (rajouter d'autre individus) ou pas.

Illustration 2ème modélisation (Schématisé sous forme d'un arbre n-aire)

Un arbre n-aire, initialement le sous ensemble est vide, donc la racine correspond {⦸}, au premier niveau chaque valeur de l'ensemble peut être considéré. Ainsi la racine devra avoir n fils. Puisque chaque valeur apparait une seule fois, au niveau suivant il restera (n-1) possibilité, et de proche en proche jusqu'à ce qu'il n'en reste aucune. A chaque niveau les branches seront évaluées (calcul de la somme) afin d'exclure les combinaisons qui dépassent la somme cible S.

2. Estimation de la taille de l'arbre (1 pt pour l'estimation, 1 pt pour la complexité)
   a. Cas binaire : 2^N, donc la complexité = $O(2^n)$
   b. Cas sous ensemble : $A_p^N = (n ! * p !)/ (n – p) ! <= p^n$, P étant la taille maximale du sous ensemble, donc la complexité = $O(p^n)$.
3. Une solution valide est une solution qui vérifie les contraintes suivantes :
   a. Solution non vide
   b. Contenant les individus de l'ensemble en entrée (Par définition dans la première modélisation mais elle doit être vérifiée dans la seconde)
   c. Chaque individu apparait une seule fois dans la solution (en supposant que deux individus différents ne peuvent pas avoir le même poids. Doit être vérifier dans la seconde modélisation)
   d. La somme des individus présent dans la solution est égale à S
   e. Pour le cas <=, la solution doit être la plus proche possible de 300. Ce qui signifie, soit trouver toute les solutions qui sont <= 300 et ensuite prendre la plus grande. Soit déterminer un seuil à partir duquel une solution peut être acceptée. Le seuil peut tenir

Bon Courage !

compte soit de la taille de la solution (donc P) soit de l'écart possible entre la solution trouvée et S.

4. Algorithme de validation et complexité (3 pts algorithme, 1 pt complexité)

```
Algo_1 (S' : solution potentiel, S : somme ciblé) : booléen ; /* cas modélisation binaire */
Début
Booléen Valide = vrai ;
Entier i :=1, som := 0 ;
Tant que ( i<= N) et (valide = vrai) et (som < S)
Faire si (S'[i] <> 0) alors      som := som + E [i] ;
                                 Si som > S alors valide = faux ;
                                 Fsi ;
        Fsi ;
        I :=i + 1 ;
Fait ;
Si som <> S'  alors valide = faux ; /* si le cas <= est pris en compte, il doit être traité à partir d'ici
                                    Selon la contrainte supplémentaire choisie */
Fsi ;
Retourner (valide) ;
Fin.
```

Complexité = O(n)

```
Algo_2(S' : solution potentiel, S : Somme ciblé) : Booléen ; /* cas sous-ensemble */
Début
Booléen valide = vrai ;
Entier i=1, j=1, Som =0 ;
Tant que i<= P et valide = vrai
Faire j = 1 ;
Tant que j<= N et trouve = faux faire si S'[i] = E[j] alors trouve = vrai ; // appartenance à E
                                    Sinon j :=j+1 ;
```

Bon Courage !

Fsi ;

Fait ;

K = i +1 ;

// l'unicité de la valeur dans la solution S'

Tant que k <= P et trouve = faux faire si S'[i] = S'[k] alors trouve = vrai ;

Sinon k := k + 1 ;

Fsi ;

Fait ;

Si trouve :=vrai alors valide := faux ;

Sinon som := som + S'[i] ;

Si som > S alors valide = : faux;

Sinon i := i+1 ;

Fsi ;

Fsi ;

Fait ;

Si som <> S' alors valide = faux ; /* si le cas <= est pris en compte, il doit être traité à partir d'ici

Selon la contrainte supplémentaire choisie */

Fsi ;

Retourner (valide) ;

Fin.

Complexité $O(n^2)$.

5. Le problème appartient à la classe NP car l'algorithme de validation associé est polynomial. Le problème n'appartient pas à la classe P puisque l'algorithme de résolution ne peut être polynomial.

Bon Courage !