ensit **Ecole Nationale Supérieure d'Ingénieurs de Tunis**

Examen **Classe :** 2ème anné **Matière :** Algorithmique Avancée **Enseignant(e) :** **Mme Zoulel KOUKI**	**Date :** 10/01/2023 **Durée :** 02H **N ombre de pages :** 03 **Documents autorisés :** Non ☒ Oui ☐

Exercice 1 : Un Etudiant, faisant sa valise pour un séjour d'études à l'étranger, se trouve hésitant entre n objets qu'il juge à priori utiles. Le i^{eme} de ces objets valant v_i dinars et pesant p_i kilos, v_i et p_i étant des entiers.

L'étudiant veut bien évidemment emporter un butin de plus grande valeur possible mais il ne peut porter que C kilos dans son sac à dos. Il doit se décider: quels objets conviendrait-t-il le mieux, prendre?

1- Proposer un algorithme glouton qui priorise la cueillette des objets ayant un rapport coût-poids plus élevé.

(Possibilité d'appeler des algorithmes sans les développer, dans ce cas, il faut donner la signature de l'algorithme et sa complexité).

2 - Calculer la complexité totale de l'algorithme glouton.

3 - Donnez un exemple d'instance avec quatre objets au plus pour lequel l'algorithme donne une solution optimale.

4 - Montrez au moyen d'un contre-exemple avec quatre objets au plus que l'algorithme glouton n'est pas toujours optimal.

5 - Proposez un algorithme itératif de programmation dynamique.

Indication : construire un tableau T dans lequel les lignes seront indexées par les objets et les colonnes par les valeurs. L'élément T[i][j] représentera la valeur maximale pour un sac à dos de capacité j à l'aide des i premiers objets.

$$T[n, C] = v_n + T[n-1][C - p_n]$$

Si le nième élément n'appartient pas à la solution optimale, cela signifie, que la solution optimale OPT est aussi une solution optimale avec un sac à dos de capacité C et un ensemble O privé du $n^{ème}$ élément, alors $T[n, C] = T[n-1][C]$

Donc par récurence on peut en déduire pour tout $i \in \{1, \dots, n\}$ que

Si $j < v_i$ alors $T[i][j] = 0$ sinon $T[i][j] = max(T[i-1][j], T[i-1][j - p_i] + v_i)$

6 - Donner la trace d'exécution de l'algorithme itératif de programmation dynamique pour les instances des questions 3 et 4.

7 -Calculer la complexité de l'algorithme itératif de programmation dynamique.

Exercice 2

Soit une matrice carrée M d'ordre n ($n= 2^k$) de réels.

Notre objectif est d'écrire un algorithme permettant de déterminer le plus grand élément de la matrice M.

1.Ecrire la fonction Sup (x1, x2) qui retourne le maximum entre deux nombres.

2.Déduire la fonction Maximum qui retourne le maximum entre quatre nombres.

3.Donner la complexité exacte en opérations élémentaires de ces algorithmes .

4.Donner l'algorithme itératif le plus simple permettant de déterminer ce maximum.

5.Déterminer la complexité asymptotique de cet algorithme.