EXAMEN

Semestre : 1 ■ 2 □

Session : Principale ■ Rattrapage □

Unité d'enseignement : Recherche opérationnelle
Module : Complexité appliquée à la RO
Classes : 4SAE, 4ERPBI, 4DS, 4TWIN, 4INFINI et 4NIDS

Nombre d'exercices : 3 Nombre de pages : 2

Date : 13/01/2023 Heure : 11h00 Durée : 1h30

Exercice 1 (6 points):

Pour chaque algorithme proposé ci-dessous, calculer l'ordre de complexité.

Algo 1

```
Pour i =1 à n faire
   Pour j=1 à n faire
      s=j+i
  Fin Pour
Fin Pour
```

Algo 2

```
Pour i=1 à n-1 faire
   Pour j=1 à i faire
      k=1
      Tant que k<m faire
          S=i+j
          k=k+1
      Fin Tant que
   Fin Pour
Fin Pour
```

Algo 3

```
Pour i=5 à n-5 faire
    Pour j=i-5 à i+5 faire
      S=S+i
    Fin Pour
Fin Pour
```

Algo 4

```
i=n
Tant que i > 1 faire
    Ecrire(« i= »,i)
    i=i/2
Fin Tant que
```

Algo 5

```
Pour i =1 à n faire
    Pour j =1 à  i faire
      y(i) = y(i) + A(i,j)*x(j)
    Fin Pour
FinPour
```

Algo 6

```
Pour i =1 à n faire
    Pour j =I à min(n,i+1) faire
      y(i) = y(i) + A(i,j)*x(j)
    Fin Pour
Fin Pour
```

Exercice 2 : (7 points)

Soit la fonction récursive « **FCT** »

```
Fonction FCT (T: tab, n, m: entier) : booléen
Début
  Si n > 0 Alors
     Si (m = T[n]) alors
         FCT ← vrai
     FinSi
     FCT ← FCT(T, n-1, m)
  Sinon
     FCT ← faux
  FinSi
Fin
```

1. Que fait la fonction **FCT** ? sachant que **T** est un tableau de taille **n** et que la première case dans le tableau est d'indice 1 et la dernière case est d'indice n.
2. Quel est le type de récursivité de la fonction **FCT** ?
3. On cherche à calculer la complexité en nombre de comparaisons. Donner l'équation récurrente de la fonction **FCT** ?
4. Détailler le calcul de complexité en résolvant l'équation de récurrence obtenue dans la question 3.
5. Déduire la classe de complexité de la fonction **FCT.**

Exercice 3: (7 points)

On considère A[] un tableau d'entiers, on veut chercher la différence A[j] - A[i] maximale entre deux éléments A[i] et A[j] tel que A[i] < A[j] et i < j.

Exemples:
Input: A[] = [1, 4, 9, 5, 3, 7], Output: 8
Input: A[] = [9, 8, 1, 6, 3, 2], Output: 5
Input: A[] = [9, 8, 6, 3, 2], Output: -1 (tableau décroissant)

Pour résoudre ce problème, on considère l'algorithme récursif « **maxDifference** » qui suit le principe diviser pour régner, son code est le suivant :

```
int maxDifference(int A[], int l, int r)
{
   if (l >= r)
      return -1;
   int maxDiff = -1;
   int mid = l + (r - l) / 2;
   int leftMaxDiff = maxDifference(A, l, mid);
   int rightMaxDiff = maxDifference(A, mid + 1, r);
   int minLeft = findMin(A, l, mid);
   int maxRight = findMax(A, mid + 1, r);
   maxDiff = max(max(leftMaxDiff, rightMaxDiff), maxRight - minLeft);
   return maxDiff;
}
```

1. On suppose que les fonctions « findmin » et « findmax » sont de complexité au pire O(n), donner alors l'équation récurrente de la fonction « **maxDifference** ».
2. Montrer que la complexité de cet algorithme est $\Theta(nLog(n))$.
3. Cet Algorithme est-t-il considéré comme rapide ? Justifier votre réponse.
4. Soient « **maxDifference2** » et « **maxDifference3** » deux autres algorithmes qui permettent de résoudre le même problème que « **maxDifference** » et ils ont comme complexité respectivement $O(n^2)$ et $O(n^3)$. Comparer les trois algorithmes en termes de performance.