Ecole Polytechnique de Sousse Année Universitaire : 2023 -2024

Classe : 3ème Génie Informatique	DS S2
Matière : Algorithmique Avancée et Langage C	Durée : 1h30
Enseignant : Mme. Feyrouz Hamdaoui	Date : 01 - 03 - 2024
Documents autorisés : Non	Nombre de Pages : 02

Exercice 1 (10 points)

Q1. Quelle est la différence entre $O(n^3)$ et $O(2n^2+ n^3)$? Justifiez votre réponse.

Q2. Soit la complexité temporelle d'une fonction : $T(n)= T(n/2) + T(n/2) + O(n)+O(1)$.

Proposer un squelette de la fonction correspondante.

Q3. Donner l'ordre de grandeur de la procédure ci-dessous au pire, moyenne et meilleure des cas. Justifiez votre réponse.

```
void exp1() {
 int i ;
 for (int i = 1 ; i <= 100 ; i++)
    printf (i*2) ;
}
```

Q4. Donner l'ordre de grandeur de chacun des deux algorithmes ces dessous au pire et au meilleure des cas.

```
Algorithm whoAmI(n)
  t=0
  i=1
  while  log2(i) ) < n do {
     t = t + 1
     i = i + 1
  return t
```

```
Algorithm GoodLuck(n)
  i = 1; j = 1;
  while i <= n do {
     j = j + 3;
     i = i * 2;
  }
```

Exercice 2 (10 points)

Etant donnée un ensemble A de n éléments, on dit qu'un élément x est majoritaire dans A si le nombre d'occurrences de x (x non nul) est strictement supérieur à n/2.

Q1. Ecrivez une fonction itérative **Occ_it(x, A, i, j)** qui calcule le nombre d'occurrences d'une valeur x présente entre les indices i et j d'un tableau A.

Indication : l'appel dans le programme principal se fait avec une initialisation de i à 1, de j à n.

Q2. Analyser sa complexité temporelle au pire des cas.

Q3. Ecrivez une procédure récursive **Occ_rec (i, A, n, x, p)** qui calcule le nombre d'occurrences p d'une valeur x présente dans un tableau A.

Indication : l'appel dans le programme principal se fait avec une initialisation de p à 0 et de i à 1.

Q4. Analyser sa complexité temporelle au pire des cas en utilisant la méthode de substitution.

Q5. ***Sans faire appel aux fonctions Occ***, proposer une fonction itérative **Majo_it(A : tab)** de complexité **quadratique** qui vérifie si un tableau A contient un élément majoritaire quelconque ou non.

Q6. On considère maintenant l'approche suivante : on divise en deux le tableau A. Il renverra vrai si le tableau contient un élément majoritaire quelconque et renverra faux si le tableau ne contient pas d'élément majoritaire.

Ecrivez une fonction récursive **Majo_rec(A, i, j, K)** qui correspondant à cette idée.

Indication :

- L'appel dans le programme principal se fait avec une initialisation de i à 1, de j à n et de K à 0.
- Cette fonction renvoie également l'élément majoritaire trouvé (sinon 0), en paramètre d'entrée K passé par variable.
- Vous pouvez appeler la fonction Occ_it de la question 1.

Q7. Analyser sa complexité temporelle au pire des cas (vous pouvez utiliser le théorème en annexe).

Annexe : Théorème

1. Soit T(n) une fonction définie par l'équation de récurrence suivante, où k ≥ 0, b > 0, et c ≥ 1 :

$$T(n) = c * T(n-1) + b * n^k \ \ Alors \ :$$
$$- Si\ c = 1\ alors\ T(n) = O(n^{k+1})$$
$$- Si\ c > 1\ alors\ T(n) = O(c^n)$$

2. Soit T(n) une fonction définie par l'équation de récurrence suivante, où d≥2, k ≥0, c>0, b > 0 :

$$T(n) = c * \ T(n/d) \ + \ b * n^k$$

La relation entre c, d et k détermine la fonction T(n) comme suit :

$$- Si\ c > \ d^k, alors\ T(n) = \ O(n^{log_d(c)})$$
$$- Si\ c = \ d^k, alors\ T(n) = \ O(n^k * \ log(n))$$
$$- Si\ c < \ d^k, alors\ T(n) = \ O(n^k)$$

Bon travail