Université d'Ottawa
Faculté de génie

École d'ingénierie et de
technologie de l'information

uOttawa
L'Université canadienne
Canada's university

University of Ottawa
Faculty of Engineering

School of Information
Technology and Engineering

Devoir 1 (6% - 24 points)

CSI2510

A remettre sur le Campus Virtuel avant 9h00am le 8 octobre 2019

Question 1. **[6 points]**

Démontrer si les énoncés ci-dessous sont vrais ou faux. Lorsqu'elles existent identifier des valeurs pour les constantes `c` et n_0 telles que définies dans les définitions de Grand O et Grand Θ.

a) 2^{n+a} est $O(n^2)$, pour `a` étant une constante positive
b) 2^{2n+a} est $O(n^2)$, pour `a` étant une constante positive
c) $\sum_{k=1}^{n} k^3$ est $O(n^4)$
d) $\sum_{k=1}^{n} k^3$ est $\Theta(n^3)$
e) $(2n+8)\log(n^{10})$ est $O(n\log(n))$
f) $(2n+8)\log(n^{10})$ est $\Theta(n\log(n))$

Question 2. **[6 points]**

La recherche dans un tableau trié se fait normalement par division successive du tableau en 2 parties égales (recherche binaire ou recherche dichotomique). On vous propose maintenant d'effectuer cette recherche en divisant l'intervalle en 3. L'algorithme procéderait donc ainsi afin d'effectuer une recherche dans l'intervalle `[low, high]`:

- Diviser l'intervalle en trois aux index `i1=low+(high-low)/3` et `i2=low+2(high-low)/3`;
- Si l'élément recherché se trouve à `i1`, la recherché s'arrête
- Si l'élément recherché est inférieur à l'élément à l'index `i1`, alors la recherche se poursuit dans le premier intervalle;
- Si l'élément recherché se trouve à `i2`, la recherché s'arrête
- Si l'élément recherché est inférieur à l'élément à l'index `i2`, alors la recherche se poursuit dans le second intervalle;
- Sinon la recherche se poursuit dans le dernier intervalle.

Question 2. **[**suite**]**

a) Quel sera donc, au pire cas, le nombre de comparaisons nécessaire afin de trouver un élément dans un tableau de `N` éléments en utilisant cette stratégie? Bien détailler votre réponse
b) La complexité de cette recherche est-elle encore $O(\log_2(n))$? Justifier votre réponse

Question 3. **[6 points]**

Voici un algorithme s'appliquant sur un tableau `a` contenant `n` éléments.

```
for (i = 0, length = 1; i < n-1; i++) {
   for (i1 = i2 = k = i; k < n-1 && a[k] < a[k+1]; k++, i2++);
   if (length < i2 - i1 + 1)
      length = i2 - i1 + 1;
}
return length;
```

a) Que fait cet algorithme ? Quel résultat retourne-t-il ?

b) Donner l'ordre de grandeur de la complexité de cet algorithme en termes de Grand O. Effectuer cet analyse pour le meilleur cas et le pire cas. Spécifier un ordre de grandeur aussi simple que possible.

c) Donner un exemple d'un tableau produisant le meilleur et le pire cas.

Question 4. **[6 points]**

On vous demande de concevoir un algorithme permettant de savoir si un tableau non-ordonné contient plus d'une fois la même valeur. L'algorithme doit simple retourner vrai dès que deux éléments de même valeur sont trouvés. Vous ne devez pas changer l'ordre des éléments dans le tableau (donc vous ne pouvez pas trier le tableau) et vous ne devez pas utiliser un tableau auxiliaire pour effectuer cette tâche. Donner le pseudo-code de cet algorithme et donner sa complexité Grand Ω dans le pire cas.