3I-IN9 : Structures de données
TD - 2

Quelques définitions

— Chaque algorithme de ce TD traite un tableau de n éléments **avec n non nul**.

— Quand l'on parle de meilleur ou pire cas pour un algorithme, l'on cherche la configuration spécifique des données dans la tableau (de n éléments) pour que algorithme dure le moins longtemps ou dure le plus longtemps.

— Quand l'on parle du coût en temps d'un algorithme, l'on affecte à chaque ligne de l'algorithme un temps d'exécution que l'on notera t_i (par exemple t_3 le temps d'exécution de la $3^{\text{ème}}$ ligne de l'algorithme) puis l'on détermine le nombre de fois que chaque ligne est exécuter pour calculer le temps d'exécution total de algorithme.

— Quand l'on parle de complexité majorante asymptotique avec la notation de Landau, l'on cherche à déterminé le terme du coût en temps qui croit le plus vite en fonction de n (par exemple : si l'on à un coût en temps suivant $T(n) = (t_3 + t_4) * n^2 + (t_5 + t_7) * n + (t_1 + t_2)$ le terme qui croit le plus vite est $(t_3 + t_4) * n^2$, la complexité majorante asymptotique est alors $\mathcal{O}(n^2)$.

1 Recherche d'une valeur dans une tableau non-trié

Algorithm 1 Recherche dans un tableau non-trié

```
1: procedure FindValue(T[ ], v)
2:     n ← getNbElement(T)
3:     for i ← 1, n do
4:         if T[i] == v then
5:             return i
6:         end if
7:     end for
8:     return −1
9: end procedure
```

1. Dans l'algorithme 1, quelles est la condition pour que le meilleur cas se produise ?
2. Pour le meilleur cas de cette algorithme,
   2.1. Quel est son coût en temps ?
   2.2. Quelle est sa complexité majorante asymptotique avec la notation de Landau ?
3. Dans l'algorithme 1, quelle est la condition pour que le pire cas se produise ?
4. Pour le pire cas de cette algorithme,
   4.1. Quel est son coût en temps ?
   4.2. Quelle est sa complexité majorante asymptotique avec la notation de Landau ?

2 Recherche d'une valeur dans une tableau trié

Algorithm 2 Recherche par dichotomie

```
procedure FindValue(T[ ], v)
    start ← 1
    end ← getNbElement(T)
    while start ≤ end do
        i ← ⌊(start+end)/2⌋
        if T[i] == v then
            return i
        else if T[i] < v then
            start ← i + 1
        else
            end ← i − 1
        end if
    end while
    return −1
end procedure
```

1. Dans l'algorithme 2, quelles est la condition pour que le meilleur cas se produise ?
2. Pour le meilleur cas de cette algorithme,
   - 2.1. Quel est son coût en temps ?
   - 2.2. Quelle est sa complexité majorante asymptotique avec la notation de Landau ?
3. Dans l'algorithme 2, quelle est la condition pour que le pire cas se produise ?
4. Pour le pire cas de cette algorithme,
   - 4.1. Quel est son coût en temps ?

     Aide

     Posez-vous la questions suivante : Combien de fois peut-on itérativement couper en deux un tableau de n éléments avant de n'avoir plus qu'un seul élément ?
     Simplification : Vous pouvez supposer que n le nombres d'élément dans votre tableau est une puissance de 2.

   - 4.2. Quelle est sa complexité majorante asymptotique avec la notation de Landau ?