TD2 – Création et analyse d'algorithmes

Killian Reine

Rappels de cours.

Introduction au TD

L'objectif de ce TD est de mettre en pratique la création d'algorithmes et d'apprendre à les analyser de manière simple, en s'appuyant sur les notions de TD1 : entrées/sorties, boucles, opérations, meilleur/pire cas.
On ne parle pas encore de récursivité avancée ou de complexité asymptotique formelle, mais plutôt de raisonnement étape par étape et logique de programme « propre ».

Exercice 1

Description, construction et analyse

On veut créer un algorithme qui compte combien de fois un nombre apparaît dans un tableau.

1 **Identifier l'entrée et la sortie de l'algorithme.**

- **Entrées :**
  - un tableau T de taille $n \geq 1$, contenant des éléments comparables ;
  - un élément x à rechercher, appartenant au même ensemble que les éléments de T.
- **Sortie :** un entier $c \in \mathbb{N}$ représentant le nombre d'occurrences de x dans le tableau T.

2 **Écrire une description pas à pas en français.**

(a) Initialiser un compteur à zéro.
(b) Pour chaque élément du tableau, comparer cet élément avec le nombre cherché.
(c) Si l'élément courant est égal au nombre cherché, incrémenter le compteur de 1.
(d) Une fois tous les éléments parcourus, retourner la valeur du compteur.

3 **Construire l'algorithme.**

On suppose que le tableau T contient $n \geq 1$ éléments et que x est l'élément à compter.

```
compteur ← 0
pour i de 1 à n faire
    si T[i] = x alors
        compteur ← compteur + 1
    fin si
fin pour
retourner compteur
```

4 **Application avec le tableau $T = \{3, 5, 3, 2, 1, 3\}$ pour compter le nombre d'occurrences du nombre 3.**

Déroulons l'algorithme étape par étape.

5 **Faire un tableau étape par étape pour chaque itération.**

Itération	Valeur courante	Comparaison avec 3	Valeur du compteur
Initialisation	-	-	0
$i = 1$	$T[1] = 3$	$3 = 3$ (Vrai)	1
$i = 2$	$T[2] = 5$	$5 = 3$ (Faux)	1
$i = 3$	$T[3] = 3$	$3 = 3$ (Vrai)	2
$i = 4$	$T[4] = 2$	$2 = 3$ (Faux)	2
$i = 5$	$T[5] = 1$	$1 = 3$ (Faux)	2
$i = 6$	$T[6] = 3$	$3 = 3$ (Vrai)	3

6 **Quel est le résultat final ?**

Le résultat final est **3**. Le nombre 3 apparaît exactement 3 fois dans le tableau $T = \{3, 5, 3, 2, 1, 3\}$.

Exercice 2

Étude d'un algorithme

Pour l'algorithme du maximum :

1 **Donner l'algorithme.**

On suppose que le tableau T contient $n \geq 1$ éléments comparables.

```
max ← T[1]
pour i de 2 à n faire
    si T[i] > max alors
        max ← T[i]
    fin si
fin pour
retourner max
```

2 **Identifier :**

a **Nombre d'affectations dans le meilleur cas.**

Le meilleur cas se produit lorsque le maximum se trouve en première position du tableau (ou que le tableau est trié dans l'ordre décroissant).

Dans ce cas, l'affectation `max ← T[i]` n'est jamais exécutée dans la boucle.

Nombre total d'affectations dans le meilleur cas : $\boxed{1}$ (uniquement l'initialisation `max ← T[1]`).

b **Nombre d'affectations dans le pire cas.**

Le pire cas se produit lorsque le tableau est strictement croissant. À chaque itération, l'élément courant est strictement plus grand que `max`, donc une affectation est effectuée.

- 1 affectation lors de l'initialisation : `max ← T[1]` ;
- $n - 1$ affectations supplémentaires dans la boucle (une par itération).

Nombre total d'affectations dans le pire cas : $\boxed{n}$.

Exemple : Pour $T = \{1, 2, 3, 4, 5\}$ avec $n = 5$, il y a 5 affectations.

3 **Nombre de contrôles (tests de boucle) pour un tableau de taille N.**

Un contrôle (test de boucle) correspond à la vérification de la condition de continuation de la boucle, c'est-à-dire le test $i \leq n$.

- La condition est testée avant la première itération.
- Elle est testée après chaque itération.

La boucle commence à $i = 2$ et se termine lorsque $i > n$.

Les valeurs de i testées sont : $2, 3, 4, \ldots, n, n + 1$.

Nombre de contrôles : $\boxed{n}$ (tests pour $i = 2$ jusqu'à $i = n + 1$).

Remarque : On compte n tests au total, car :

- $n - 1$ tests pour les itérations $i = 2$ à $i = n$ (la boucle s'exécute $n - 1$ fois) ;
- 1 test supplémentaire pour $i = n + 1$ qui provoque la sortie de la boucle.

4 **Vérifier que l'algorithme renvoie bien un résultat pour toutes les entrées.**

Pour que l'algorithme renvoie toujours un résultat valide, il faut vérifier :

(a) **Terminaison :** La boucle parcourt les éléments de $i = 2$ à $i = n$, ce qui constitue un nombre fini d'itérations. L'algorithme termine donc toujours.

(b) **Correction :**

- À l'initialisation, `max` contient le premier élément du tableau.
- À chaque itération, si un élément plus grand est trouvé, `max` est mis à jour.
- À la fin du parcours, `max` contient nécessairement le plus grand élément du tableau.

(c) **Précondition :** L'algorithme suppose que $n \geq 1$ (le tableau n'est pas vide). Si cette condition est respectée, l'algorithme renvoie toujours un résultat valide.

Conclusion : L'algorithme renvoie bien un résultat pour toutes les entrées valides (tableaux non vides de taille $n \geq 1$).