TD1: complexité d'algorithme et tableaux

2025-2026

Exercice 1. Calcul du coût d'un algorithme

a. Déterminer le nombre d'affectations, en fonction de n et m, dans les algorithmes suivants.

Algorithme : A(m, n : entiers) : entier

```
1 i ← 1;
2 tant que i ≤ m et i ≤ n
3 |  i ← i + 1;
```

Algorithme : B(m, n : entiers) : entier

```
1 i ← 1;
2 tant que i ≤ m ou i ≤ n
3 |  i ← i + 1;
```

Algorithme : C(m, n : entiers) : entier

```
1 i ← 1;
2 j ← 1;
3 tant que j ≤ n
4 |  si i ≤ m
5 |  |  i ← i + 1;
6 |  sinon
7 |  |  j ← j + 1;
```

Algorithme : D(m, n : entiers) : entier

```
1 i ← 1;
2 j ← 1;
3 tant que j ≤ n
4 |  si i ≤ m
5 |  |  i ← i + 1;
6 |  sinon
7 |  |  j ← j + 1;
8 |  |  i ← 1;
```

Exercice 2. Algorithme mystère

a. Que calcule l'algorithme suivant ? Quelle est sa complexité ?

Algorithme : f(n : entier) : entier

```
1 r ← 0;
2 pour i de 1 à n
3 |  r ← 2r + 1
4 retourner r
```

b. Que calcule l'algorithme suivant et quelle est sa complexité ? Environ quel temps faut-il pour calculer $g(100)$ en supposant que l'on fait 10^{10} additions par secondes.

Algorithme : $g(n$: entier) : entier

```
1 si n = 0
2 |  retourner 0
3 sinon
4 |  retourner g(n-1)+g(n-1)+1
```

Exercice 3. Taille d'un entier

a. Écrire un algorithme itératif et un algorithme récursif qui prend en entrée un entier n et renvoie sa taille en base 10. La taille d'un entier est le nombre de chiffres qu'il faut pour l'écrire dans sa représentation en base 10.

b. La complexité est-elle différente si on change la base (base 2 par exemple) ?

Exercice 4. Diviser pour régner

Le problème est de définir à partir de quel étage d'un immeuble une bille en verre, lâchée depuis cet étage, se casse. On numérote les étages de 1 à n. Il y a k billes disponibles toutes identiques. Pour tester si la hauteur d'un étage est fatale pour la bille, on peut en lâcher une encore disponible. Si elle ne casse pas, on peut la réutiliser. Vous devez proposer un algorithme pour trouver l'étage limite.

a. Si vous avez uniquement une seule bille, quel sera votre algorithme et de combien de tests aurez-vous besoin dans le pire cas ?

b. Si $k \geq \log n$ donner un algorithme en $O(\log n)$ tests.

c. Si $k = 2$ proposer un algorithme en $O(\sqrt{n})$ tests.

d. Si $k < \log n$ donner le meilleur algorithme possible.

Exercice 5. Minimum d'un tableau

Pour un tableau t, on note $t.n$ sa taille et $t[i]$ l'élément d'indice i qui doit être compris entre 0 et $t.n - 1$.

a. Écrire un algorithme qui retourne la valeur minimale d'un tableau.

b. Combien de comparaisons effectue cet algorithme (on ne compte que les comparaisons avec des valeurs du tableau) en fonction de $t.n$?

c. Écrire alors un algorithme qui retourne les valeurs à la fois du minimum et du maximum dans un tableau. Combien de comparaisons fait-il ?

d. Si ce n'est pas déjà le cas, trouver un algorithme qui n'effectue que $\frac{3n}{2}$ comparaisons.

Exercice 6. Doublons

a. Dans un tableau de n entiers, proposer un algorithme qui retourne un tableau dans lequel les doublons ont été supprimés.

b. Même question si le tableau est trié.

c. Pour la première question, est-il intéressant de le trier et d'appliquer ensuite l'algorithme sur les tableaux triés (la complexité du tri est $O(n \log n)$) ?

Exercice 7. Fusion de tableaux triés

La fusion de deux tableaux triés est le tableau qui contient tous les éléments des 2 tableaux, éventuellement répétés, et qui est lui-même trié. Par exemple, la fusion du tableau $[1, 5, 6, 7, 7, 8]$ avec le tableau $[2, 5, 7, 13]$ donne le tableau $[1, 2, 5, 5, 6, 7, 7, 7, 8, 13]$.

a. Écrire un algorithme de complexité linéaire en la somme de la taille des tableaux.

b. Supposons que l'on ait qu'un seul tableau de taille $n = 2^k$, et que chaque moitié du tableau soit triée. Comment réaliser la fusion des deux moitiés de tableau en place (sans utiliser de tableau auxiliaire) ? Quelle est alors la complexité, et celle du tri fusion qui utiliserait cet algorithme ? (*indice : montrer que 3 échanges de valeurs sont suffisants pour un tableau de taille 4, puis généraliser en utilisant la récursivité.*)

Exercice 8. Calcul de la médiane d'un tableau

La valeur *médiane* d'un tableau de taille n est la valeur qui se trouve à l'indice $\lfloor n/2 \rfloor$ une fois que le tableau est trié.

a. Écrire un algorithme qui calcule la médiane quand le tableau est trié. Quelle est sa complexité ?

b. Même question si le tableau n'est pas trié.

c. On suppose qu'on a deux tableaux t_1 et t_2 de taille n qui sont triés. Écrire un algorithme qui calcule la médiane de la fusion de ces deux tableaux. La complexité doit être $O(\log n)$.

d. Adapter le principe du tri rapide afin de trouver la médiane d'un tableau quelconque. En particulier vous pouvez utiliser l'algorithme de partitionnement du tri rapide.

e. Quelle est la complexité de votre algorithme dans le pire cas et en moyenne ?

f. Pour améliorer la complexité dans le pire cas, on peut utiliser la méthode suivante : on groupe les éléments par 5, on cherche l'élément médian du bloc, puis on calcule le médian de ces médians afin de partitionner le tableau. Adapter votre algorithme pour tenir compte de cette amélioration, et montrer que la complexité est linéaire.