Formules Algo

Algorithmique Avancée - Préparation Partiels mi-semestre
3e année Cybersécurité - École Supérieure d'Informatique et du Numérique (ESIN)
Collège d'Ingénierie & d'Architecture (CIA)

Étudiant :	HATHOUTI Mohammed taha
Filière :	Cybersécurité
Année :	2025/2026
Enseignants :	M.BAKHOUYA
Date :	October 26, 2025

1 Master Theorem - FORMULE CLÉ

Forme générale

$$T(n) = a \cdot T(n/b) + f(n)$$

Étape préliminaire : Calculer $n^{\log_b a}$

CAS 1 : Les feuilles dominent

$$f(n) = O(n^{\log_b a - \varepsilon}) \text{ pour un } \varepsilon > 0$$

$$\Rightarrow \boxed{T(n) = \Theta(n^{\log_b a})}$$

CAS 2 : Équilibre

$$f(n) = \Theta(n^{\log_b a})$$

$$\Rightarrow \boxed{T(n) = \Theta(n^{\log_b a} \log n)}$$

CAS 3 : La racine domine

$$f(n) = \Omega(n^{\log_b a + \varepsilon}) \text{ pour un } \varepsilon > 0$$

$$+ \text{ régularité : } a \cdot f(n/b) \leq c \cdot f(n), \text{ avec } c < 1$$

$$\Rightarrow \boxed{T(n) = \Theta(f(n))}$$

1.1 Exemples d'application

Exemple 1 : (Tri Fusion)

$$T(n) = 2T(n/2) + n$$

- $a = 2,\ b = 2,\ f(n) = n$
- $n^{\log_2 2} = n^1 = n$
- $f(n) = n = \Theta(n) \rightarrow$ **CAS 2**
- **Résultat :** $\boxed{T(n) = \Theta(n \log n)}$

Exemple 2 :

$$T(n) = 9T(n/3) + n$$

- $a = 9,\ b = 3,\ f(n) = n$
- $n^{\log_3 9} = n^2$
- $f(n) = n = O(n^{2-1}) \rightarrow$ **CAS 1**

- **Résultat :** $\boxed{T(n) = \Theta(n^2)}$

Exemple 3 : (Recherche)

$$T(n) = T(2n/3) + 1$$

- $a = 1,\ b = 3/2,\ f(n) = 1$
- $n^{\log_{3/2} 1} = n^0 = 1$
- $f(n) = 1 = \Theta(1) \rightarrow$ **CAS 2**
- **Résultat :** $\boxed{T(n) = \Theta(\log n)}$

Exemple 4 :

$$T(n) = 3T(n/4) + n \log n$$

- $a = 3,\ b = 4,\ f(n) = n \log n$
- $n^{\log_4 3} \approx n^{0.79}$
- $n \log n > n^{0.79} \rightarrow$ **CAS 3** (vérifier régularité)
- **Résultat :** $\boxed{T(n) = \Theta(n \log n)}$

1.2 Rappel : Calcul de $\log_b a$

Formule

$$\boxed{\log_b a = \frac{\log a}{\log b}}$$

Exemples :

- $\log_2 4 = \frac{\log 4}{\log 2} = 2$
- $\log_3 9 = \frac{\log 9}{\log 3} = 2$
- $\log_4 3 = \frac{\log 3}{\log 4} \approx 0.79$
- $\log_2 8 = \frac{\log 8}{\log 2} = 3$

2 Notations Asymptotiques

Big-O (Borne Supérieure)

$$f(n) = O(g(n)) \Leftrightarrow \exists c > 0, \exists n_0 \text{ tels que :}$$
$$0 \leq f(n) \leq c \cdot g(n) \quad \forall n \geq n_0$$

Signification : f croît au plus aussi vite que g

Omega (Borne Inférieure)

$$f(n) = \Omega(g(n)) \Leftrightarrow \exists c > 0, \exists n_0 \text{ tels que :}$$
$$0 \leq c \cdot g(n) \leq f(n) \quad \forall n \geq n_0$$

Signification : f croît au moins aussi vite que g

Theta (Borne Exacte)

$$f(n) = \Theta(g(n)) \Leftrightarrow \exists c_1, c_2 > 0, \exists n_0 \text{ tels que :}$$
$$0 \leq c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n) \quad \forall n \geq n_0$$

Équivalence :

$$\boxed{f(n) = \Theta(g(n)) \Leftrightarrow f(n) = O(g(n)) \text{ ET } f(n) = \Omega(g(n))}$$

3 Échelle de Complexité

Hiérarchie à connaître PAR CŒUR

$$\boxed{O(1) < O(\log n) < O(\sqrt{n}) < O(n) < O(n \log n) < O(n^2) < O(n^3) < O(2^n) < O(n!)}$$

Règles importantes

1. **Les constantes disparaissent :**

$$O(5n^2) = O(n^2) \quad \text{et} \quad O(100n) = O(n)$$

2. **Seul le terme dominant compte :**

$$O(n^2 + n + 100) = O(n^2)$$
$$O(3n^3 + 2n^2 + n) = O(n^3)$$

3. **Pour polynômes :** Garder le terme de plus haut degré sans coefficient

4 Formules de Sommes Importantes

À mémoriser

$$\boxed{\sum_{i=1}^{n} i = \frac{n(n+1)}{2} = \Theta(n^2)}$$

$$\boxed{\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6} = \Theta(n^3)}$$

$$\boxed{1 + 2 + 4 + 8 + \cdots + 2^n = 2^{n+1} - 1 = \Theta(2^n)}$$

$$\boxed{\log(n!) = \Theta(n \log n)}$$

$$\boxed{\sum_{i=0}^{n} x^i = \frac{x^{n+1} - 1}{x - 1} \text{ (série géométrique)}}$$

5 Complexité de Boucles

Boucle simple

```
pour i de 1 à n faire
    instruction O(1)
fin pour
```

Complexité : $O(n)$

Deux boucles imbriquées (indépendantes)

```
pour i de 1 à n faire
    pour j de 1 à n faire
        instruction O(1)
    fin pour
fin pour
```

Complexité : $n \times n = O(n^2)$

Boucles dépendantes

```
pour i de 1 à n faire
    pour j de 1 à i faire
        instruction O(1)
    fin pour
fin pour
```

Analyse : $1 + 2 + 3 + \cdots + n = \frac{n(n+1)}{2}$
Complexité : $O(n^2)$

Boucle logarithmique (doublement)

```
i = 1
tant que i < n faire
    instruction O(1)
    i = i * 2
fin tant que
```

Valeurs de i : $1, 2, 4, 8, 16, \ldots, 2^k < n$
Nombre d'itérations : $k = \lfloor \log_2 n \rfloor$
Complexité : $O(\log n)$

Boucles successives (non imbriquées)

```
pour i de 1 à n faire
    instruction O(1)
fin pour

pour j de 1 à n faire
    instruction O(1)
fin pour
```

Complexité : $O(n) + O(n) = O(n)$
Règle : Boucles successives $\rightarrow$ ADDITIONNER puis garder le terme dominant

6 Algorithmes Classiques

Recherche Dichotomique

Récurrence :

$$T(n) = T(n/2) + O(1)$$

Application Master Theorem :

- $a = 1,\ b = 2,\ f(n) = 1$
- $n^{\log_2 1} = 1 \rightarrow$ CAS 2

Complexité : $\boxed{O(\log n)}$

Tri Fusion

Récurrence :

$$T(n) = 2T(n/2) + \Theta(n)$$

Application Master Theorem :

- $a = 2$, $b = 2$, $f(n) = n$
- $n^{\log_2 2} = n \rightarrow$ CAS 2

Complexité : $\boxed{\Theta(n \log n)}$ (dans tous les cas)

Tri par Insertion

Meilleur cas (tableau trié) : $O(n)$
Pire cas (tableau inversé) : $\boxed{O(n^2)}$
Cas moyen : $O(n^2)$

7 Pièges à Éviter

✗ ERREURS FRÉQUENTES

Erreur 1 : Oublier d'enlever les constantes

- ✗ Faux : $5n^2 = O(5n^2)$
- ✓ Correct : $5n^2 = O(n^2)$

Erreur 2 : Additionner au lieu de multiplier (boucles imbriquées)

- ✗ Faux : 2 boucles imbriquées de $n \to O(n + n) = O(n)$
- ✓ Correct : 2 boucles imbriquées de $n \to O(n \times n) = O(n^2)$

Erreur 3 : Confondre les cas du Master Theorem

- Si $f(n) = n^{\log_b a} \to$ CAS 2 (pas CAS 3 !)
- Ex : $T(n) = 2T(n/2) + n \to$ CAS 2, pas CAS 1 ni 3

Erreur 4 : Oublier la condition de régularité (CAS 3)

- Le CAS 3 nécessite DEUX conditions
- Ne pas oublier de vérifier : $a \cdot f(n/b) \leq c \cdot f(n)$

Erreur 5 : Ne pas reconnaître les algorithmes classiques

- Recherche dichotomique $\to$ TOUJOURS $O(\log n)$
- Tri fusion $\to$ TOUJOURS $O(n \log n)$
- 2 boucles for(1 à n) imbriquées $\to$ TOUJOURS $O(n^2)$

8 Checklist Avant le CC1

✓ VÉRIFICATION FINALE

Je connais :

- ☐ Les 3 cas du Master Theorem
- ☐ Comment calculer $n^{\log_b a}$
- ☐ L'échelle de complexité complète
- ☐ Dichotomie $= O(\log n)$
- ☐ Tri fusion $= O(n \log n)$
- ☐ 2 boucles imbriquées $= O(n^2)$
- ☐ Formule $\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$

Je sais faire :

- ☐ Appliquer le Master Theorem (identifier a, b, $f(n)$)
- ☐ Calculer la complexité de boucles
- ☐ Prouver qu'une fonction est $O(g(n))$
- ☐ Analyser un algorithme récursif
- ☐ Enlever les constantes multiplicatives
- ☐ Garder uniquement le terme dominant

J'évite :

- ☐ De garder les constantes dans Big-O
- ☐ D'additionner les boucles imbriquées
- ☐ De confondre les 3 cas du Master Theorem
- ☐ D'oublier la régularité dans le CAS 3
- ☐ De ne pas reconnaître les algos classiques

BON COURAGE POUR VOTRE Partiel !

Références

Sources du cours

- **Prof. M. BAKHOUYA**, *Algorithmique et Structures de Données Avancées*, Chapitre 1, UIR 2025-2026
- **T. CORMEN et al.**, *Introduction à l'algorithmique*, DUNOD, 2002
- **Assistant IA :** Claude (Sonnet 4.5), Anthropic, https://claude.ai

Remerciements

Ce document de révision a été créé pour faciliter la préparation au CC1 d'Algorithmique Avancée. Il synthétise le Chapitre 1 du cours du Prof. BAKHOUYA et inclut des exemples, exercices corrigés et conseils pratiques.

Document créé le October 26, 2025
Documents autorisés - Imprimez et annotez !