FACULTÉ DES SCIENCES ET TECHNIQUES
DE MOHAMMEDIA
UNIVERSITÉ HASSAN II DE CASABLANCA

Prog Orientée Objet Avancée et Design Patterns.

Compte Rendu : RED-BLACK-TREE

Filière d'ingénieur :

Ingénierie Logicielle et Intégration des Systèmes Informatiques

Encadré par :

- Pr A.EL BOUZIRI

Réalisé par :

- MARZAQ Mouad

Année universitaire : 2025-2026

Table des matières

1 Introduction

Ce rapport présente une analyse comparative des performances de différentes structures de données en C++. Les tests ont été effectués sur un ensemble de **1 000 000 éléments** avec des opérations d'insertion, de recherche et de suppression.

1.1 Accès au Code Source

Le code source complet de ce projet est disponible sur GitHub :

Dépôt GitHub :

https://github.com/Md-marzak/Benchmarking-RougeNoirArbre

2 Structures de Données Testées

Les cinq structures suivantes ont été évaluées :

1. `RougeNoirArbre<int>`
2. `std::set<int>`
3. `std::unordered_set<int>`
4. `std::map<int,string>`
5. `Arbre<string>`

3 Méthodologie

3.1 Configuration des Tests

— **Nombre d'éléments** : 1 000 000
— **Type de clés** : Entiers aléatoires dans [0, 1 000 000]
— **Opérations testées** : insertion, recherche, suppression

4 Résultats des Tests

4.1 Tableau Comparatif Global

TABLE 1 – Comparaison des performances (en millisecondes)

Structure de Données	Insertion	Recherche	Suppression
RougeNoirArbre<int>	1025.800	0.0011	0.0026
std : :set<int>	372.015	0.0047	0.1233
std : :unordered_set<int>	**112.046**	**0.0006**	**0.0019**
std : :map<int,string>	383.426	0.0087	0.0092
Arbre<string>	257.899	0.0319	0.0101

4.2 Visualisation Graphique des Performances

4.2.1 Comparaison des Temps d'Insertion

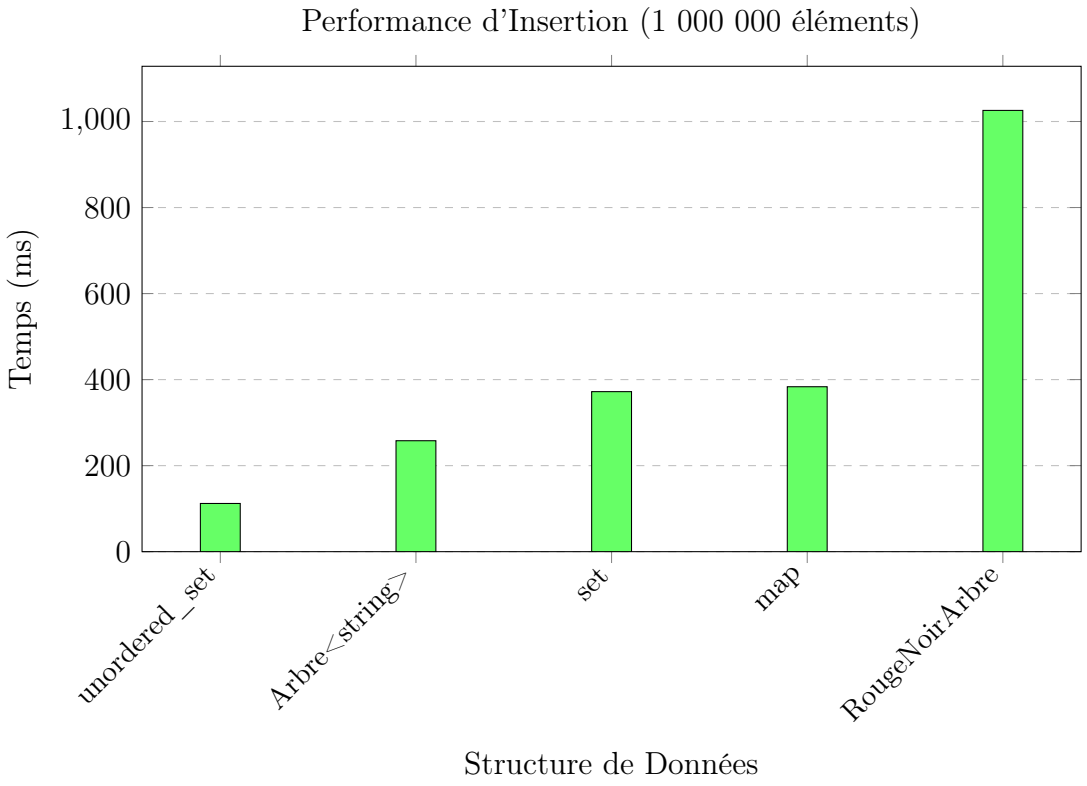

FIGURE 1 – Comparaison des temps d'insertion

4.2.2 Comparaison des Temps de Recherche

4.2.3 Comparaison des Temps de Suppression

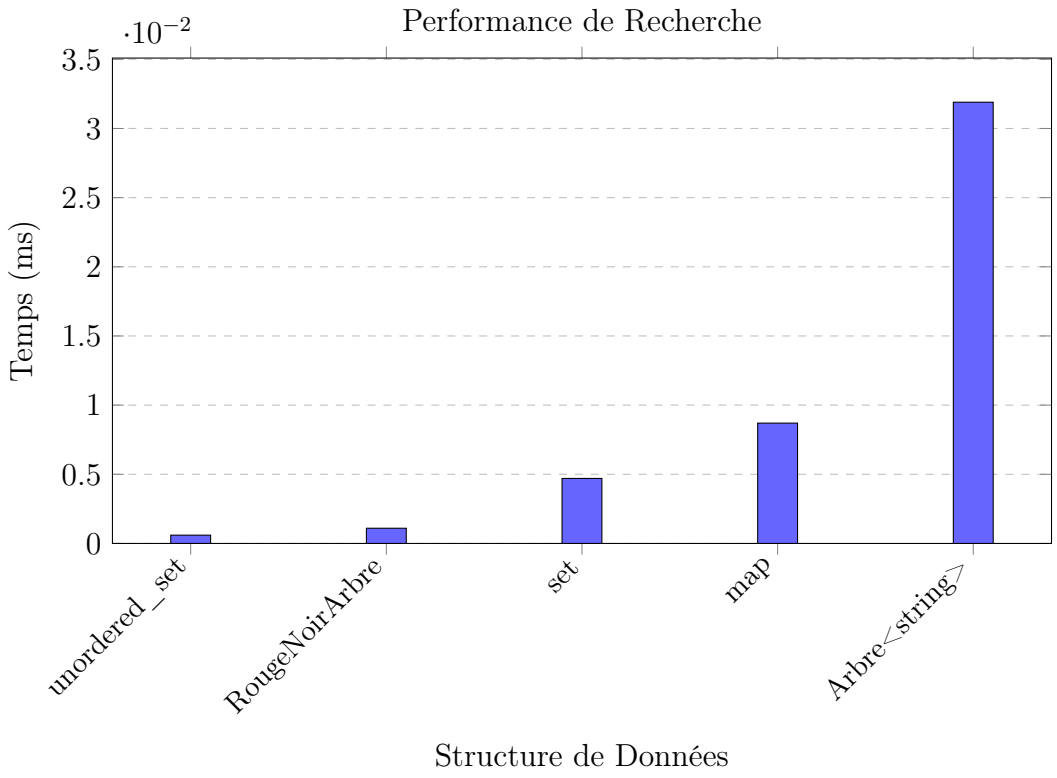

FIGURE 2 – Comparaison des temps de recherche

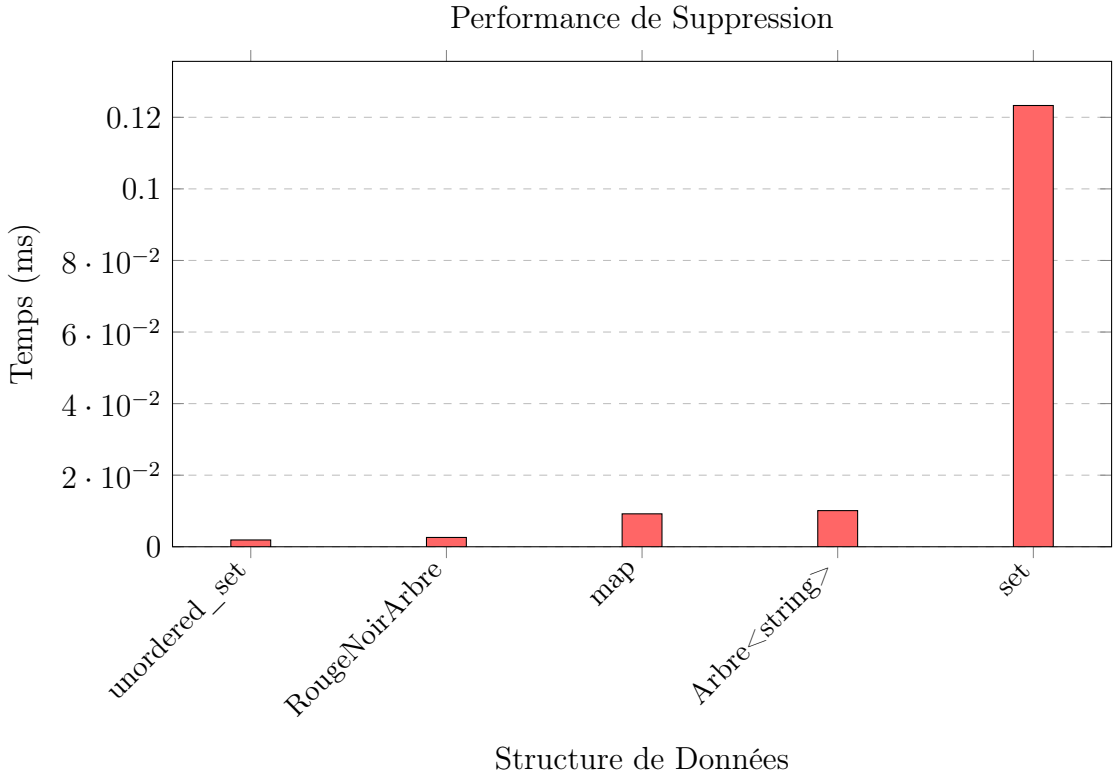

FIGURE 3 – Comparaison des temps de suppression

4.2.4 Vue d'Ensemble des Trois Opérations

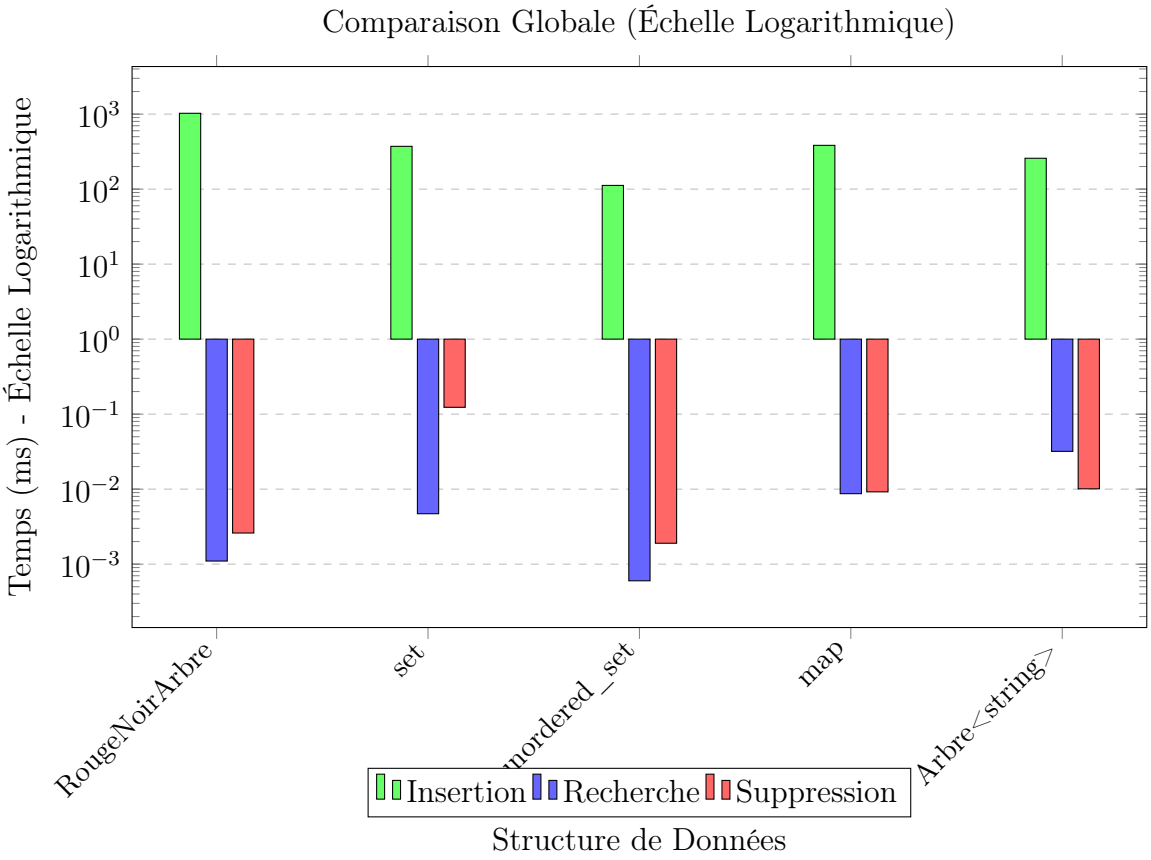

FIGURE 4 – Comparaison globale des trois opérations

5 Analyse par Opération

5.1 Performance d'Insertion

TABLE 2 – Classement pour l'insertion

Rang	Structure	Temps (ms)
1	std : :unordered_set<int>	112.046
2	Arbre<string>	257.899
3	std : :set<int>	372.015
4	std : :map<int,string>	383.426
5	RougeNoirArbre<int>	1025.800

5.2 Performance de Recherche

TABLE 3 – Classement pour la recherche

Rang	Structure	Temps (ms)
1	std : :unordered_set<int>	0.0006
2	RougeNoirArbre<int>	0.0011
3	std : :set<int>	0.0047
4	std : :map<int,string>	0.0087
5	Arbre<string>	0.0319

5.3 Performance de Suppression

TABLE 4 – Classement pour la suppression

Rang	Structure	Temps (ms)
1	std : :unordered_set<int>	0.0019
2	RougeNoirArbre<int>	0.0026
3	std : :map<int,string>	0.0092
4	Arbre<string>	0.0101
5	std : :set<int>	0.1233

6 Conclusion

À travers cette étude comparative, nous pouvons tirer les conclusions suivantes :

— **std : :unordered_set** est la structure la plus performante pour toutes les opérations testées, grâce à son accès en temps quasi constant.

— **RougeNoirArbre** assure un temps de recherche et de suppression faible, mais l'insertion reste relativement coûteuse par rapport aux autres structures.

— **std : :set** et **std : :map** présentent des performances correctes mais sont moins efficaces que `unordered_set` et `Arbre` pour certaines opérations.

— **Arbre<string>** montre de bonnes performances pour l'insertion mais est moins performant pour la recherche, surtout avec des chaînes de caractères.

En résumé, le choix de la structure de données dépend fortement de l'usage prévu : pour un accès rapide et intensif, `std::unordered_set` est recommandé, tandis que pour une structure ordonnée avec recherche rapide et équilibrée, `RougeNoirArbre` reste un choix intéressant.