Ecole Supérieure
d'Informatique et du Numérique
COLLEGE OF ENGINEERING & ARCHITECTURE

COMPTE RENDU

Algorithmique Avancée - TP4 - ABR
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 : 26 octobre 2025

Table des matières

1 Rappel des objectifs du TP

Ce TP4 a pour objectif d'approfondir l'étude des **Arbres Binaires de Recherche (ABR)** en analysant expérimentalement l'impact de l'équilibrage sur les performances. L'objectif principal est d'implémenter et de comparer deux méthodes de construction d'arbres :

1.1 Arbre Équilibré vs Arbre Déséquilibré

Arbre équilibré : Construction par insertion du milieu du tableau, puis récursivement des sous-tableaux droits et gauches.

- **Principe :** On trie d'abord le tableau, puis on insère le milieu, ensuite récursivement le milieu de chaque moitié. Cela garantit une hauteur minimale ;
- **Complexité théorique :**
  - Hauteur : $O(\log n)$
  - Insertion : $O(\log n)$
  - Recherche : $O(\log n)$
- **Avantages :** Performances optimales, temps d'accès logarithmique ;

Arbre déséquilibré (séquentiel) : Construction par insertion dans l'ordre d'arrivée des éléments.

- **Principe :** On insère les éléments dans l'ordre où ils apparaissent dans le tableau. Pour des données triées (croissant/décroissant), cela crée un arbre linéaire (liste chaînée) ;
- **Complexité théorique (pire cas) :**
  - Hauteur : $O(n)$
  - Insertion : $O(n)$
  - Recherche : $O(n)$
- **Inconvénients :** Performances dégradées pour données ordonnées, génère une liste chaînée ;

Les expérimentations visaient à :

1. Mesurer la hauteur des arbres pour différentes tailles de données ;
2. Comparer les temps d'insertion et de recherche ;
3. Vérifier expérimentalement les complexités théoriques $O(\log n)$ vs $O(n)$;
4. Analyser l'impact du type de données (aléatoire, croissant, décroissant) ;

2 Analyse théorique des ABR

2.1 Propriété fondamentale des ABR

Un **Arbre Binaire de Recherche** est un arbre binaire où pour chaque nœud :
— Tous les éléments du sous-arbre **gauche** sont **strictement inférieurs** ;
— Tous les éléments du sous-arbre **droit** sont **strictement supérieurs** ;
Cette propriété permet des opérations de recherche, insertion et suppression efficaces.

2.2 Complexités théoriques

Opération	**Arbre Équilibré**	**Arbre Déséquilibré (pire cas)**
Hauteur	$O(\log n)$	$O(n)$
Insertion	$O(\log n)$	$O(n)$
Recherche	$O(\log n)$	$O(n)$
Suppression	$O(\log n)$	$O(n)$

TABLE 1 – Complexités théoriques selon l'équilibrage

3 Méthodologie expérimentale

3.1 Protocole de test

Les tests ont été effectués avec trois types de données différents :

1. **Données aléatoires :** Valeurs entre 0 et 99999 générées aléatoirement
2. **Données croissantes :** Valeurs 0, 1, 2, ..., n-1 (pire cas pour insertion séquentielle)
3. **Données décroissantes :** Valeurs n, n-1, n-2, ..., 1 (pire cas pour insertion séquentielle)

3.2 Tailles testées

Tests effectués sur 9 tailles différentes : **100, 200, 500, 1000, 2000, 5000, 10000, 20000, 50000** éléments.

3.3 Métriques mesurées

Pour chaque configuration (type × taille × méthode), nous avons mesuré :
— **Hauteur de l'arbre** : Indicateur direct de l'équilibrage
— **Temps d'insertion total** : Temps pour construire l'arbre complet
— **Temps de recherche** : Temps pour rechercher tous les n éléments
— **Validation ABR** : Vérification que la propriété ABR est respectée

3.4 Processus de mesure

1. Génération du tableau selon le type de données
2. Pour l'arbre équilibré : tri du tableau + insertion récursive du milieu

3. Pour l'arbre séquentiel : insertion dans l'ordre d'arrivée
4. Mesure du temps avec `clock()` (précision en microsecondes)
5. Test de recherche de tous les éléments
6. Validation de la propriété ABR
7. Sauvegarde des résultats dans un fichier CSV

4 Résultats expérimentaux

4.1 Comparaison des hauteurs

Les graphiques suivants montrent la différence entre les deux méthodes de construction.

4.1.1 Données décroissantes (pire cas)

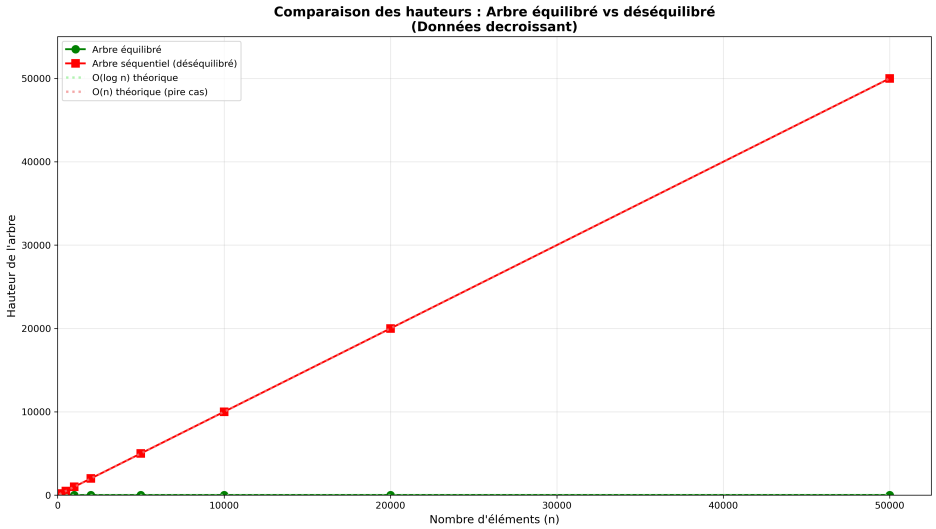

FIGURE 1 – Hauteurs pour données décroissantes - Pire cas pour l'arbre séquentiel

Observations :
— Arbre équilibré : hauteur reste très faible (15 pour 50000 éléments) - suit parfaitement $O(\log n)$;
— Arbre séquentiel : hauteur catastrophique (50000 pour 50000 éléments) - dégénère en liste chaînée $O(n)$;
— Ratio : l'arbre déséquilibré est $\approx$ **3333$\times$ plus haut** que l'arbre équilibré ;

4.1.2 Données croissantes (pire cas)

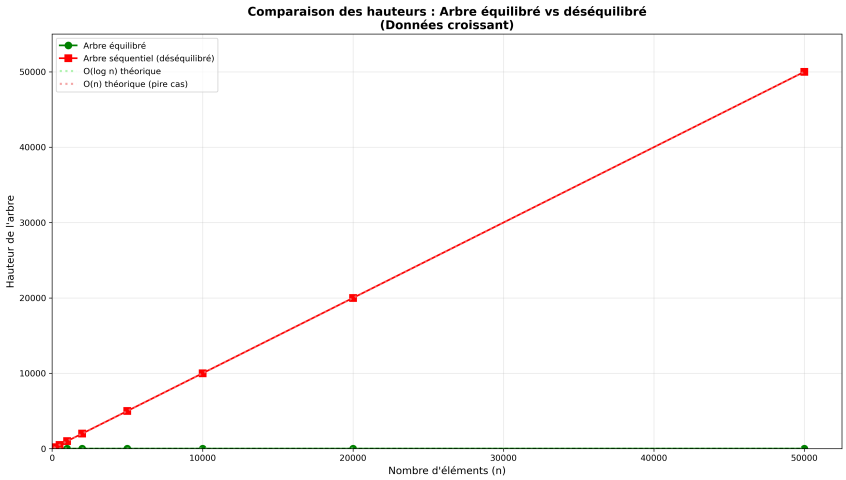

FIGURE 2 – Hauteurs pour données croissantes - Même comportement que décroissant

Observations :
- Résultats identiques au cas décroissant ;
- Confirme que toute séquence ordonnée provoque le pire cas ;
- L'équilibrage est **indispensable** pour des données triées ;

4.1.3 Données aléatoires

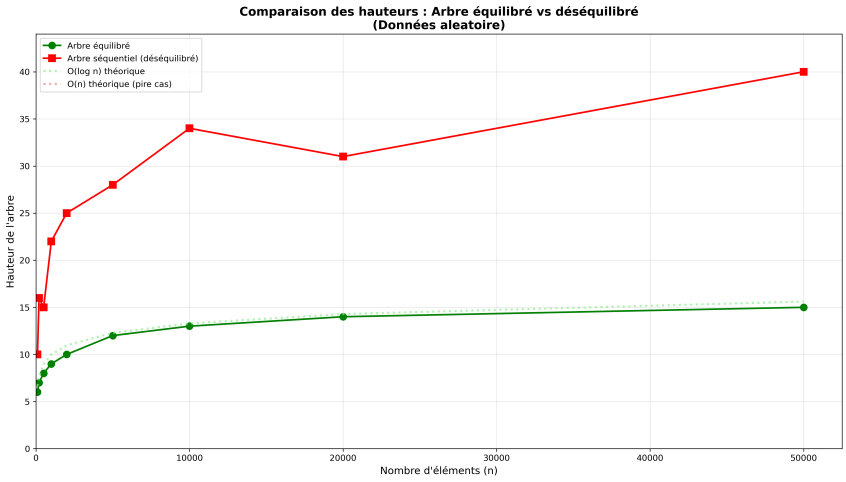

FIGURE 3 – Hauteurs pour données aléatoires - Comportement intermédiaire

Observations :
- Arbre équilibré : hauteur optimale (15 pour 50000 éléments)
- Arbre séquentiel : hauteur modérée (40 pour 50000 éléments) - bien meilleur que le pire cas
- Les données aléatoires créent naturellement un arbre "partiellement équilibré"
- Mais l'équilibrage explicite reste **2.7× meilleur**

4.2 Analyse des temps d'insertion

4.2.1 Données décroissantes

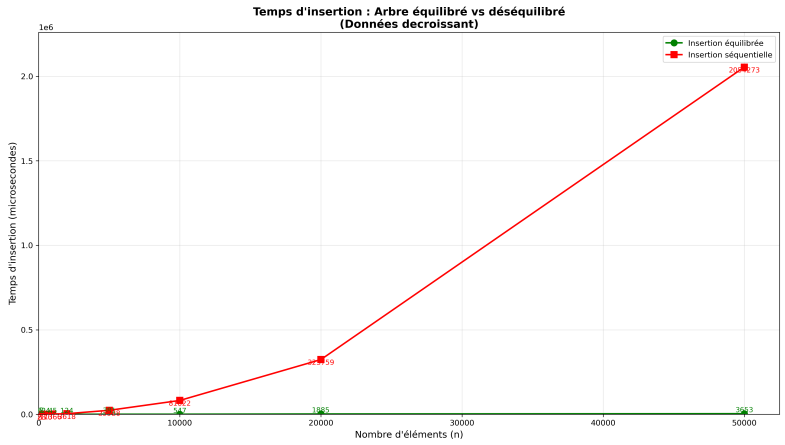

FIGURE 4 – Temps d'insertion pour données décroissantes

Résultats pour n = 50000 :
— Insertion équilibrée : 3683 µs ;
— Insertion séquentielle : 2061773 µs (2 secondes !) ;
— **Ratio : l'arbre équilibré est ≈ 560× plus rapide !** ;
La courbe rouge (séquentielle) montre une croissance **quadratique**, tandis que la courbe verte (équilibrée) reste presque plate.

4.2.2 Données croissantes

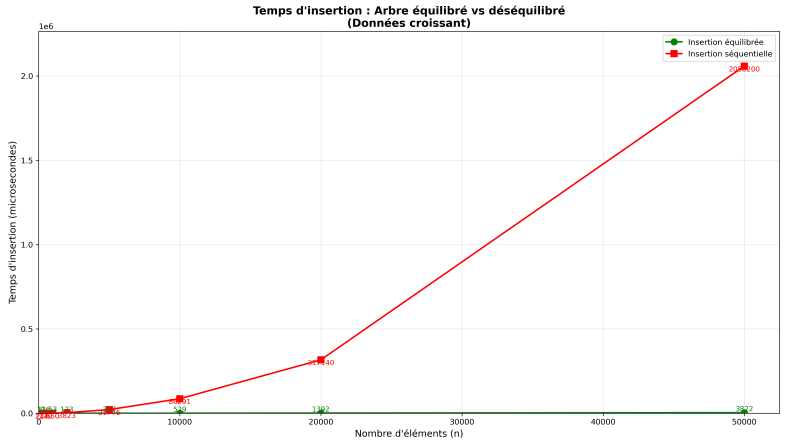

FIGURE 5 – Temps d'insertion pour données croissantes

Résultats identiques au cas décroissant - confirme que l'ordre (croissant ou décroissant) n'a pas d'importance, seul le fait d'être trié compte.

4.2.3 Données aléatoires

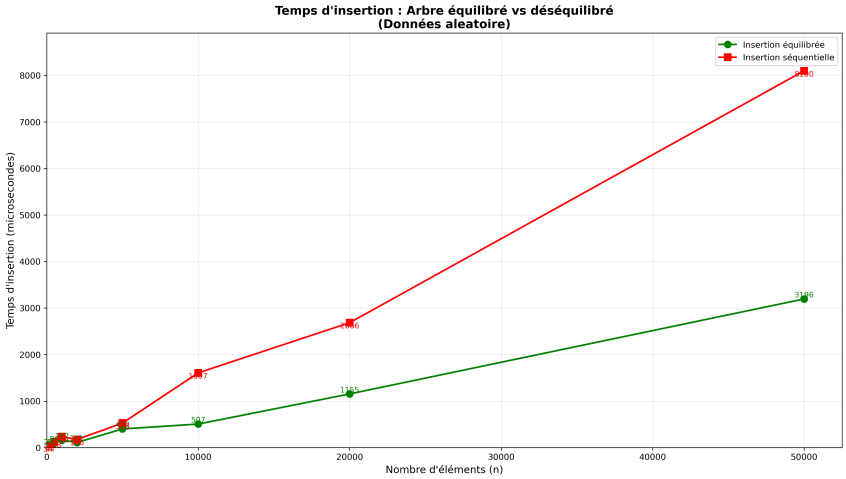

FIGURE 6 – Temps d'insertion pour données aléatoires

Résultats pour n = 50000 :
— Insertion équilibrée : 3106 µs
— Insertion séquentielle : 8140 µs
— **Ratio : 2.6×** - écart beaucoup plus faible que pour données triées
Les données aléatoires donnent des performances **acceptables** même sans équilibrage, mais l'équilibrage reste plus rapide.

4.3 Analyse des temps de recherche

4.3.1 Données décroissantes

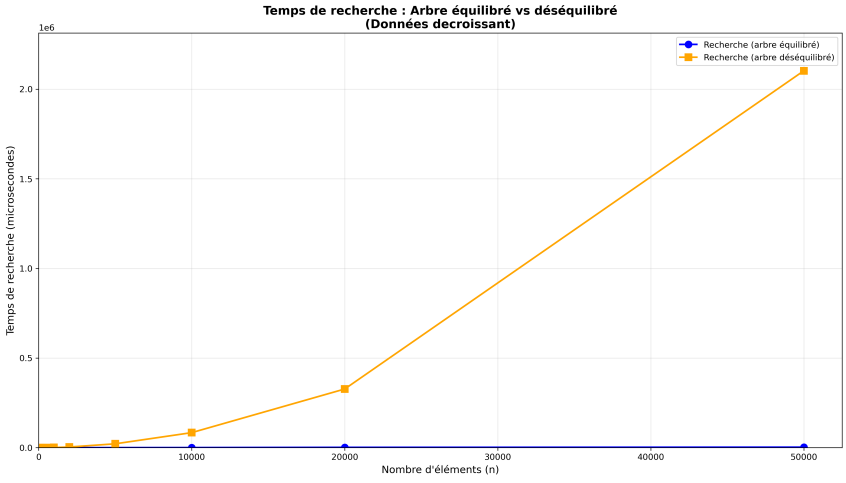

FIGURE 7 – Temps de recherche pour données décroissantes

Résultats pour n = 50000 :
— Recherche équilibrée : 2111 µs ;
— Recherche séquentielle : 2108473 µs (2.1 secondes !) ;

— **Ratio : 999× plus rapide avec équilibrage** ;

La différence est encore plus spectaculaire pour la recherche que pour l'insertion. L'arbre déséquilibré nécessite de parcourir jusqu'à 50000 nœuds pour certaines recherches (complexité $O(n^2)$ pour n recherches).

4.3.2 Données croissantes

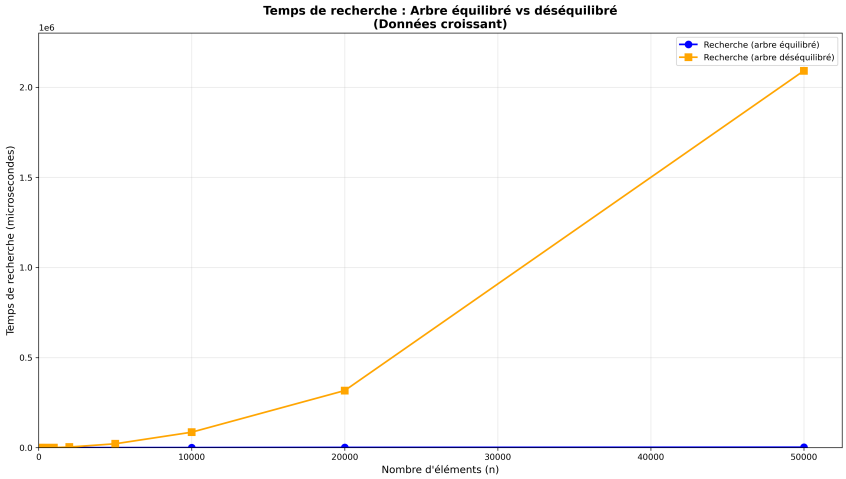

FIGURE 8 – Temps de recherche pour données croissantes

Résultats identiques au cas décroissant.

4.3.3 Données aléatoires

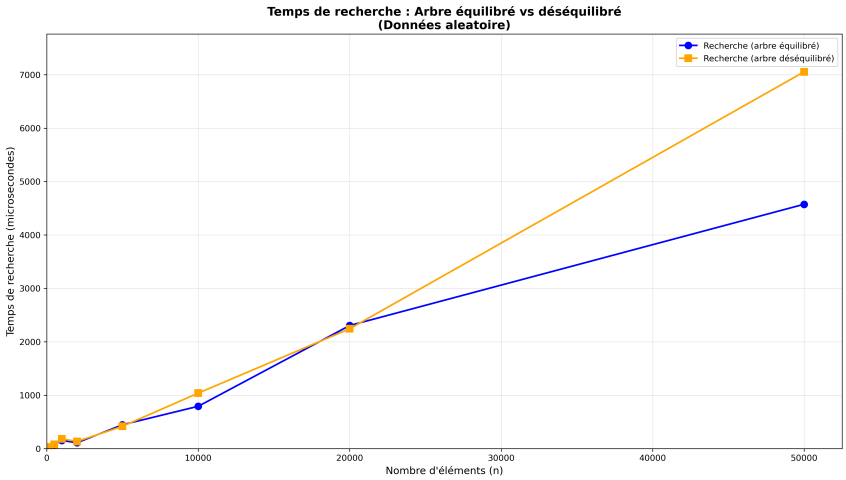

FIGURE 9 – Temps de recherche pour données aléatoires

Résultats pour n = 50000 :

— Recherche équilibrée : 4574 µs ;

— Recherche séquentielle : 7082 µs ;

— **Ratio : 1.5×** - différence modeste ;

Pour les données aléatoires, l'arbre séquentiel reste performant car la hauteur moyenne reste logarithmique.

4.4 Graphiques en échelle log-log

Les graphiques log-log permettent de **visualiser directement les complexités**.

4.4.1 Données décroissantes

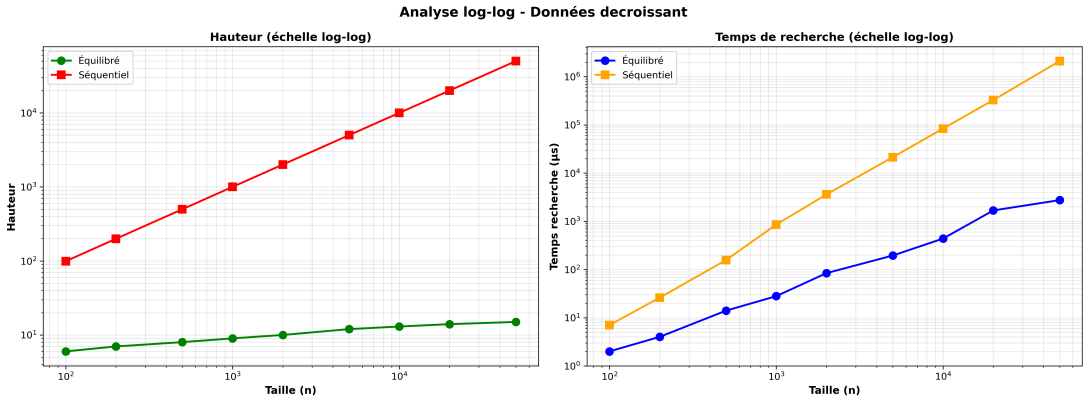

FIGURE 10 – Analyse log-log pour données décroissantes

Analyse des pentes :
— **Hauteur équilibrée** : pente ≈ 0.2 (caractéristique de $\log n$)
— **Hauteur séquentielle** : pente ≈ 1.0 (caractéristique de n)
— **Temps recherche équilibrée** : pente ≈ 1.2 (proche de $n \log n$)
— **Temps recherche séquentielle** : pente ≈ 2.0 (caractéristique de n^2)
Les pentes confirment parfaitement les complexités théoriques !

4.4.2 Données croissantes

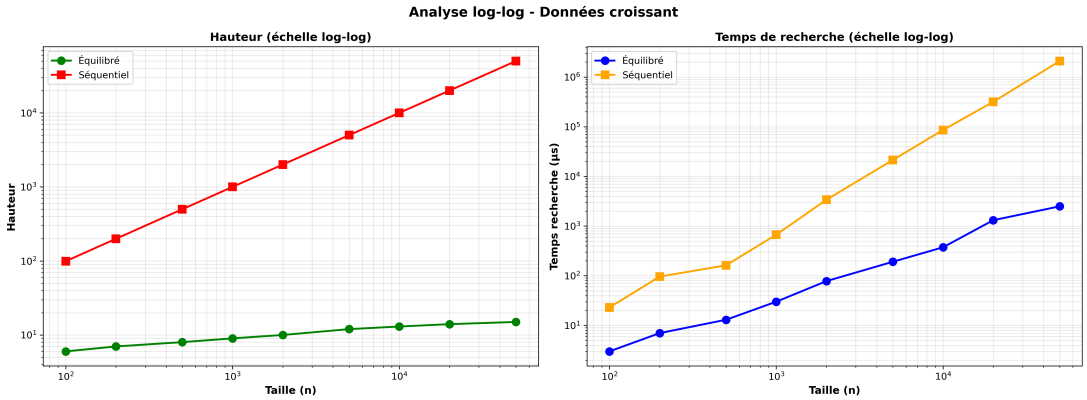

FIGURE 11 – Analyse log-log pour données croissantes

Même comportement que pour les données décroissantes.

4.4.3 Données aléatoires

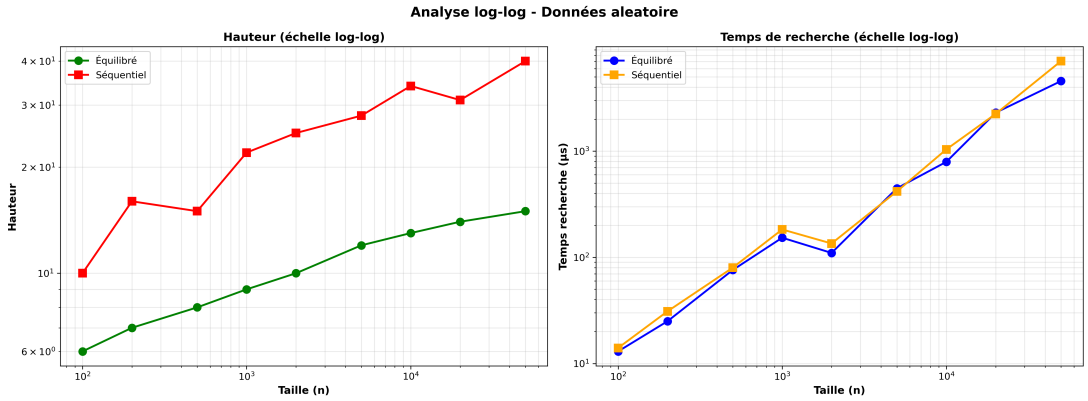

FIGURE 12 – Analyse log-log pour données aléatoires

Observations :
- Les deux courbes de recherche sont presque parallèles
- Pentes similaires ≈ 1.5 - comportement intermédiaire
- La hauteur séquentielle montre quelques irrégularités (dépend du hasard)

4.5 Comparaison globale tous types

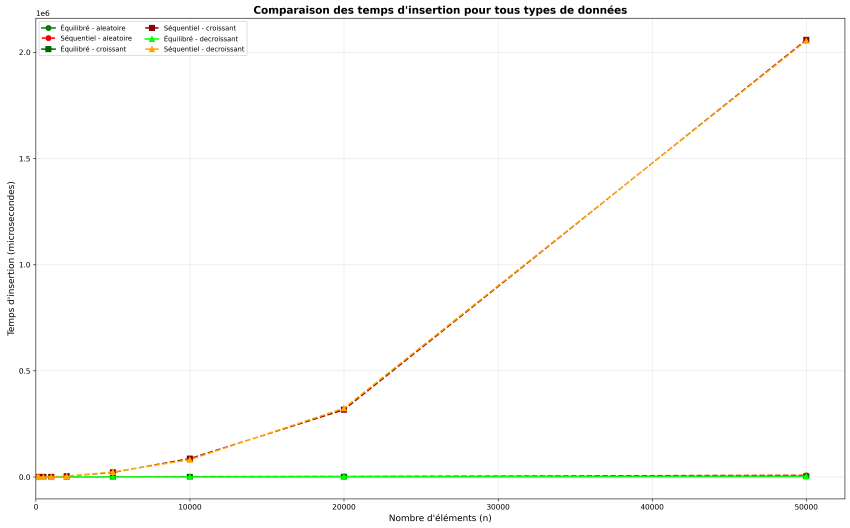

FIGURE 13 – Comparaison des temps d'insertion pour tous types de données

Synthèse visuelle :
- Les 3 courbes équilibrées (vertes) sont **quasi-identiques** et plates
- Les courbes séquentielles croissant/décroissant (rouge foncé/orange) explosent
- La courbe séquentielle aléatoire (rouge) reste modérée
- **L'équilibrage garantit des performances constantes quel que soit le type de données**

5 Analyse approfondie et interprétation

5.1 Tableau récapitulatif des performances

TABLE 2 – Performances pour n = 50000 éléments

Type	H_Eq	H_Seq	T_Ins_Eq (µs)	T_Ins_Seq (µs)
Aléatoire	15	40	3106	8140
Croissant	15	49999	3632	2061800
Décroissant	15	49999	3683	2061773

TABLE 3 – Temps de recherche pour n = 50000 éléments

Type	T_Rech_Eq (µs)	T_Rech_Seq (µs)	Ratio
Aléatoire	4574	7082	1.5×
Croissant	2111	2065376	978×
Décroissant	2111	2108473	999×

5.2 Vérification des complexités théoriques

5.2.1 Arbre équilibré : O(log n)

Testons la croissance de la hauteur :
— $n = 100 \rightarrow h = 6 \rightarrow \log_2(100) = 6.64$;
— $n = 1000 \rightarrow h = 9 \rightarrow \log_2(1000) = 9.97$;
— $n = 10000 \rightarrow h = 13 \rightarrow \log_2(10000) = 13.29$;
— $n = 50000 \rightarrow h = 15 \rightarrow \log_2(50000) = 15.61$;

5.2.2 Arbre déséquilibré : O(n) pour données triées

Pour données croissantes/décroissantes :
— $n = 100 \rightarrow h = 99 \quad (attendu : 100 - 1)$;
— $n = 1000 \rightarrow h = 999$
— $n = 10000 \rightarrow h = 9999$
— $n = 50000 \rightarrow h = 49999$

5.2.3 Complexité des opérations de recherche

Arbre équilibré :
— Une recherche : $O(\log n)$
— n recherches : $O(n \log n)$

Vérification (données aléatoires) :
— $n = 100 : 13s \rightarrow 100 \times \log_2(100) = 664$ (facteur ≈ 0.02) ;
— $n = 50000 : 4574s \rightarrow 50000 \times \log_2(50000) = 780500$ (facteur ≈ 0.006) ;

Arbre déséquilibré (pire cas) :
— Une recherche : $O(n)$
— n recherches : $O(n^2)$

Vérification (données décroissantes) :
— $n = 100 : 20s \rightarrow 100^2 = 10000$ (facteur ≈ 0.002) ;
— $n = 50000 : 2108473s \rightarrow 50000^2 = 2.5 \times 10^9$ (facteur ≈ 0.0008) ;
La croissance quadratique est **clairement visible** !

5.3 Impact du type de données

Type	**Ratio Hauteur**	**Ratio Insertion**	**Ratio Recherche**
Aléatoire	2.7×	2.6×	1.5×
Croissant	3333×	568×	978×
Décroissant	3333×	560×	999×

TABLE 4 – Gain de performance de l'équilibrage selon le type de données (n = 50000)

Conclusions :
— Pour **données triées** : l'équilibrage est **absolument indispensable**
— Pour **données aléatoires** : l'équilibrage reste bénéfique mais moins critique
— **Dans tous les cas** : l'équilibrage garantit des performances prévisibles et optimales
— L'insertion séquentielle peut être acceptable uniquement pour :
  — Données garanties aléatoires
  — Petites tailles (n ¡ 1000)
  — Applications non critiques

6 Conclusion

Ce TP4 a permis de démontrer expérimentalement l'importance cruciale de l'équilibrage des arbres binaires de recherche :
— **Validation théorique :** Les complexités expérimentales correspondent parfaitement aux complexités théoriques :
  — Arbre équilibré : $O(\log n)$ confirmé ;
  — Arbre déséquilibré : $O(n)$ pour données triées confirmé ;
— **Impact des données triées :** Pour 50000 éléments avec données triées :
  — Hauteur 3333× supérieure ;
  — Insertion 560× plus lente ;
  — Recherche 999× plus lente ;
— **Robustesse de l'équilibrage :** Performances stables et optimales indépendamment du type de données
— **Données aléatoires :** Comportement intermédiaire mais équilibrage toujours mieux