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

COMPTE RENDU

Algorithmique Avancée - TP6 - Comparaison des Algorithmes de Tri
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

Enseignant : M. BAKHOUYA

Date : 8 novembre 2025

Table des matières

1 Rappel des objectifs du TP

Ce TP6 a pour objectif d'étudier et de comparer expérimentalement **trois algorithmes de tri majeurs** :

— **Tri par Insertion** : Algorithme simple, intuitif, $O(n^2)$
— **Tri par Fusion** : Algorithme récursif, stable, $O(n \times log(n))$
— **Tri par Tas** : Algorithme basé sur les tas binaires, $O(n \times log(n))$, sans récursion

L'objectif principal est de **comprendre les compromis entre complexité et simplicité** pour choisir l'algorithme le plus adapté à un contexte donné.

2 Analyse théorique des algorithmes

2.1 Tri par Insertion

2.1.1 Principe

Le tri par insertion construit progressivement un tableau trié en insérant chaque élément à sa position correcte parmi les éléments déjà triés.

Algorithme :
— Pour chaque élément à partir de l'indice 1
— Le comparer avec les éléments précédents
— Décaler les éléments plus grands vers la droite
— Insérer l'élément à sa position correcte

2.1.2 Complexité

Cas	Complexité	Description
Meilleur	$O(n)$	Données déjà triées : aucun décalage nécessaire
Moyen	$O(n^2)$	Données aléatoires : en moyenne n/2 comparaisons par élément
Pire	$O(n^2)$	Données triées à l'envers : n comparaisons par élément

TABLE 1 – Complexité du tri par insertion

2.2 Tri par Fusion

2.2.1 Principe

Le tri par fusion utilise la stratégie "diviser pour régner" :

1. **Diviser** : Séparer le tableau en deux moitiés
2. **Régner** : Trier récursivement chaque moitié
3. **Combiner** : Fusionner les deux moitiés triées

2.2.2 Complexité

Cas	Complexité	Description
Meilleur	$O(n \times log(n))$	Toujours la même division récursive
Moyen	$O(n \times log(n))$	Complexité garantie
Pire	$O(n \times log(n))$	Pas de cas dégénéré

Table 2 – Complexité du tri par fusion

2.3 Tri par Tas

2.3.1 Principe

Le tri par tas utilise la structure de tas binaire :

1. **Construction** : Construire un tas max avec tous les éléments
2. **Extraction** : Échanger la racine (max) avec le dernier élément
3. **Restauration** : Réduire la taille du tas et entasser la nouvelle racine
4. Répéter jusqu'à ce que le tas soit vide

2.3.2 Complexité

Cas	Complexité	Description
Meilleur	$O(n \times log(n))$	Construction $O(n)$ + n extractions $O(log(n))$
Moyen	$O(n \times log(n))$	Complexité garantie
Pire	$O(n \times log(n))$	Pas de cas dégénéré

Table 3 – Complexité du tri par tas

2.4 Tableau comparatif

Algorithme	Meilleur	Moyen	Pire	Espace
Tri Insertion	$O(n)$	$O(n^2)$	$O(n^2)$	$O(1)$
Tri Fusion	$O(n \times log(n))$	$O(n \times log(n))$	$O(n \times log(n))$	$O(n)$
Tri par Tas	$O(n \times log(n))$	$O(n \times log(n))$	$O(n \times log(n))$	$O(1)$

Table 4 – Comparaison théorique des complexités

3 Résultats expérimentaux

3.1 Vue d'ensemble

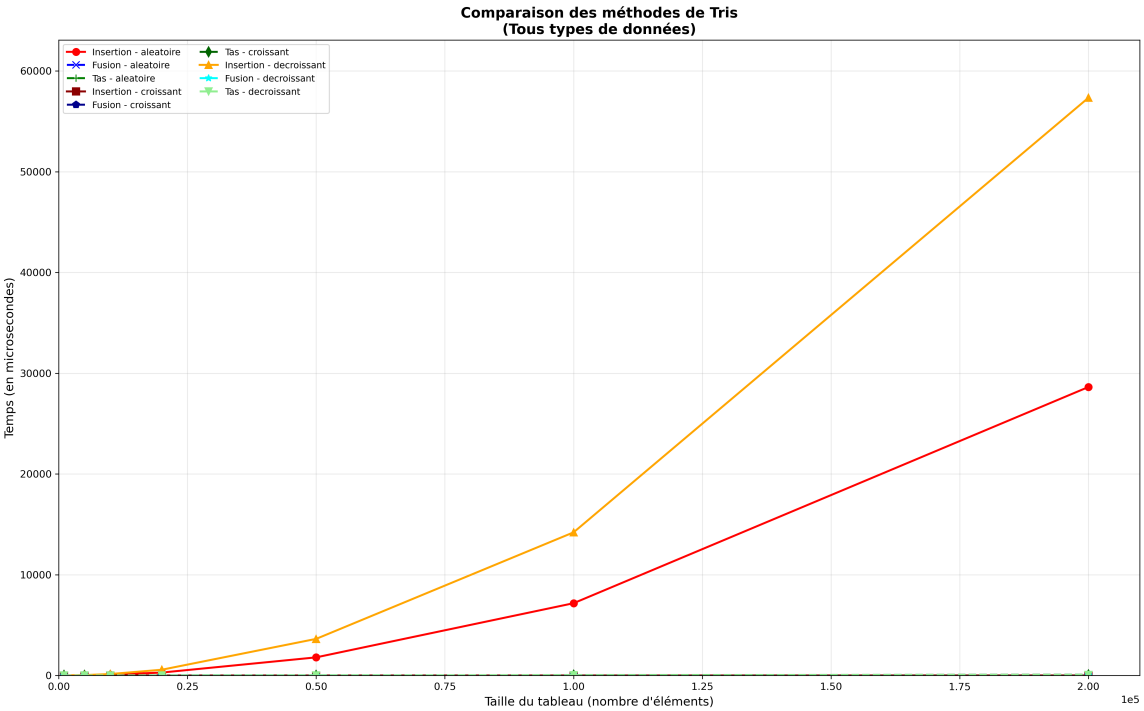

FIGURE 1 – Comparaison globale de tous les algorithmes et configurations

— **Courbe orange (Insertion-décroissant)** : Explose littéralement, dominant tout le graphique - pire cas $O(n^2)$ clairement visible
— **Courbe rouge (Insertion-aléatoire)** : Croissance quadratique marquée mais modérée
— **Courbe rouge foncé (Insertion-croissant)** : Presque plate, excellentes performances - meilleur cas $O(n)$
— **Courbes bleues (Fusion)** : Toutes groupées et basses, performances stables $O(n \times log(n))$
— **Courbes vertes (Tas)** : Similaires à Fusion, légèrement plus élevées

Conclusion : Le tri par insertion est **extrêmement sensible** à l'ordre initial, tandis que Fusion et Tas maintiennent des performances stables et prévisibles.

3.2 Analyse par configuration

3.2.1 Données aléatoires

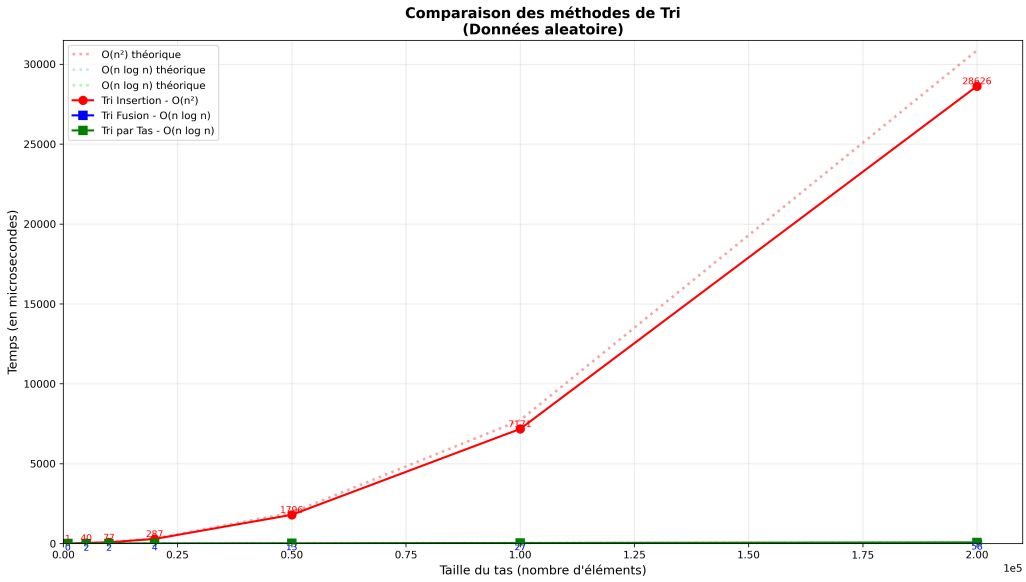

FIGURE 2 – Comparaison pour données aléatoires

Observations :

— La courbe rouge (Insertion) montre une croissance **quadratique** évidente
— Les courbes bleue (Fusion) et verte (Tas) restent basses et quasi-linéaires
— La courbe théorique $O(n^2)$ (pointillés roses) suit parfaitement l'insertion
— Les courbes théoriques $O(n \times log(n))$ correspondent bien à Fusion et Tas

Résultats numériques pour n = 100000 :

— Tri Insertion : 7171 µs
— Tri Fusion : 28 µs
— Tri par Tas : 33 µs
— **Ratio Insertion/Fusion : 257×** - L'insertion est 257 fois plus lente !
— **Ratio Insertion/Tas : 217×** - L'insertion est 217 fois plus lente !

Interprétation :

Pour des données aléatoires, le tri par insertion atteint son cas moyen $O(n^2)$. Chaque élément doit en moyenne parcourir la moitié du sous-tableau déjà trié, ce qui génère environ $\frac{n^2}{4}$ comparaisons au total. Les algorithmes $O(n \times log(n))$ sont clairement supérieurs dès que n dépasse quelques centaines d'éléments.

3.2.2 Données croissantes

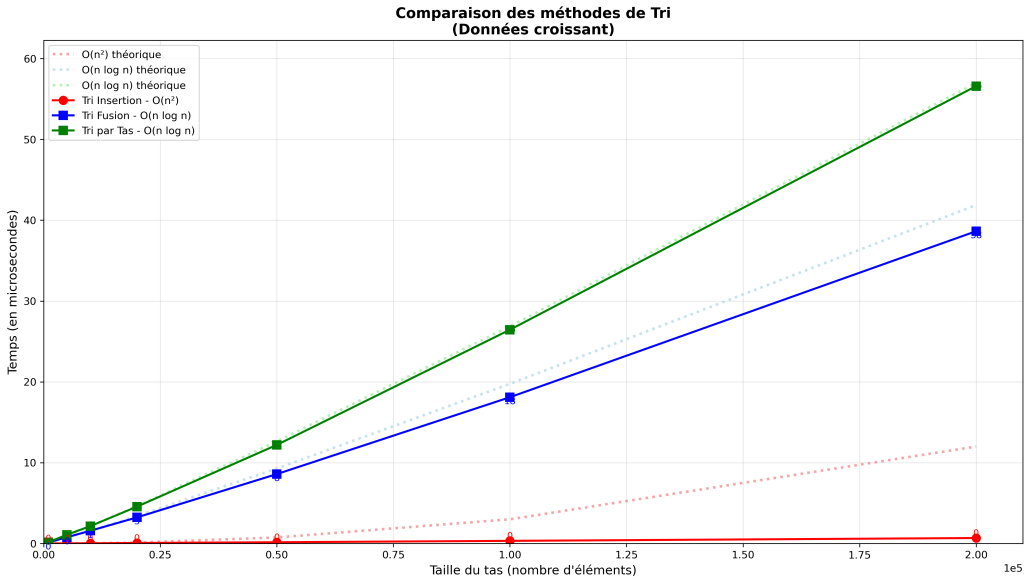

FIGURE 3 – Comparaison pour données croissantes

Observations :
— **Résultat spectaculaire** : Le tri par insertion (rouge) reste presque plat !
— La courbe rouge est maintenant la **plus basse** de toutes
— Fusion (bleu) et Tas (vert) gardent leur croissance $O(n \times log(n))$ habituelle
— L'écart entre Insertion et les autres s'inverse complètement

Résultats numériques pour n = 100000 :
— Tri Insertion : 0.33 µs
— Tri Fusion : 18 µs
— Tri par Tas : 26 µs
— **Ratio Tas/Insertion : 79×** - Fusion est 79 fois plus lent !
— **Ratio Tas/Fusion : 1.4×** - Tas est 1.4 fois plus lent !

Interprétation :
C'est le **meilleur cas** du tri par insertion. Pour des données déjà triées :
— Aucun élément ne nécessite de décalage
— Une seule comparaison par élément suffit
— Complexité réelle : $O(n)$ - parfaitement linéaire

En revanche, Fusion et Tas ne profitent pas de l'ordre initial :
— Fusion divise toujours récursivement ($log(n)$ niveaux)
— Tas doit reconstruire entièrement le tas max

3.2.3 Données décroissantes

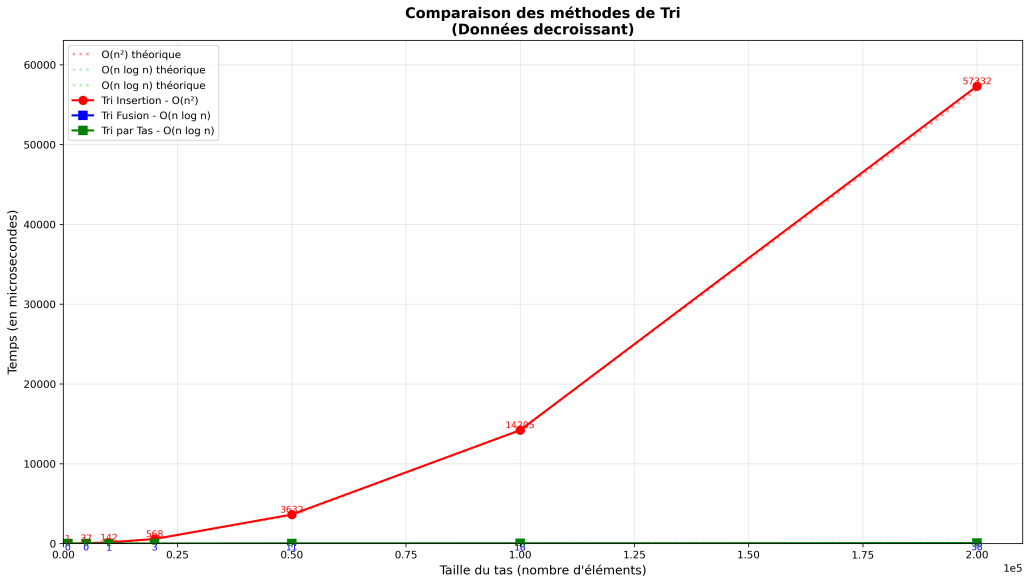

FIGURE 4 – Comparaison pour données décroissantes

Observations :
— **Catastrophe pour l'insertion !** La courbe rouge explose
— C'est la courbe la plus élevée de toute l'expérimentation
— Fusion et Tas restent stables et bas
— L'écart est maximal : facteur multiplicatif énorme

Résultats numériques pour n = 100000 :
— Tri Insertion : 14205 µs
— Tri Fusion : 18 µs
— Tri par Tas : 25 µs
— **Ratio Insertion/Fusion : 789×** - Insertion est 789 fois plus lent !
— **Ratio Insertion/Tas : 568×** - Insertion est 568 fois plus lent !

Interprétation :
C'est le **pire cas** du tri par insertion. Pour des données triées à l'envers :
— Chaque élément doit être déplacé jusqu'au début du tableau
— Nombre maximal de comparaisons : 1 + 2 + 3 + ... + (n-1) = n(n-1)/2
— Complexité réelle : $O(n^2)$ - quadratique pur
— Performance absolument catastrophique

Comparaison avec les données croissantes :
— Croissant : 0.33 µs (meilleur cas) *Tri par Insertion*
— Décroissant : 14205 µs (pire cas) *Tri par Insertion*
— **Facteur : 43045×** - Plus de 40000 fois plus lent !

Fusion et Tas ne sont pas affectés car ils ne dépendent pas de l'ordre initial.

3.3 Analyse en échelle log-log

Les graphiques log-log permettent de **visualiser directement les exposants** des complexités. Sur une échelle logarithmique, une fonction $f(n) = cn^k$ apparaît comme une droite de pente k.

3.3.1 Données aléatoires (log-log)

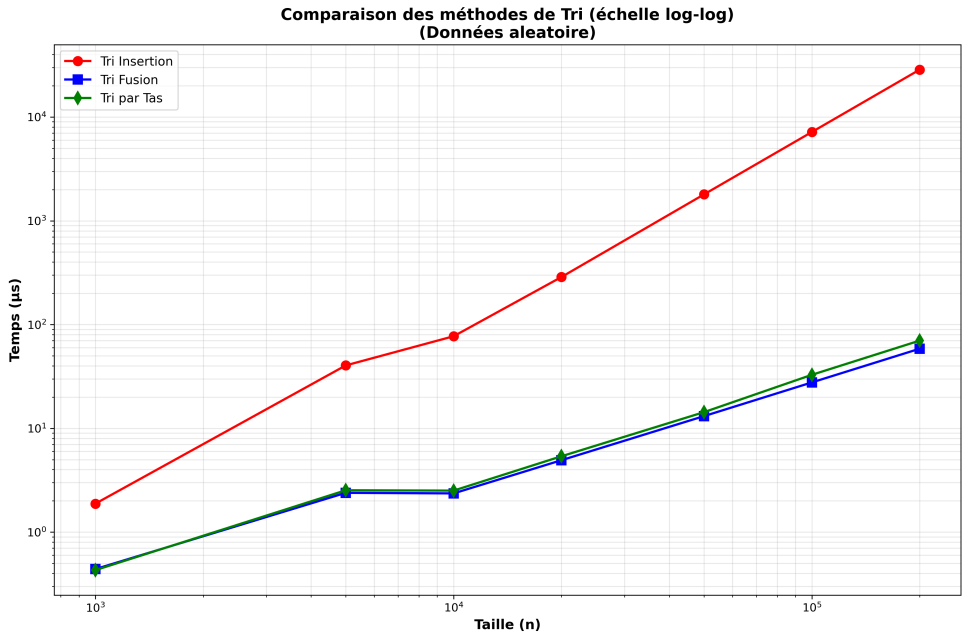

FIGURE 5 – Analyse log-log pour données aléatoires

Analyse des pentes :

— **Tri Insertion (rouge)** : Pente ≈ 2.0 - caractéristique de $O(n^2)$
  — La droite est clairement plus inclinée
  — Confirme visuellement la complexité quadratique

— **Tri Fusion (bleu)** : Pente ≈ 1.1 - légèrement supérieure à 1
  — Le terme $log(n)$ ajoute environ 0.1 à la pente
  — Cohérent avec $O(n \times log(n))$

— **Tri par Tas (vert)** : Pente ≈ 1.15 - similaire à Fusion
  — Confirme également $O(n \times log(n))$
  — Légèrement au-dessus de Fusion (constantes plus élevées)

3.3.2 Données croissantes (log-log)

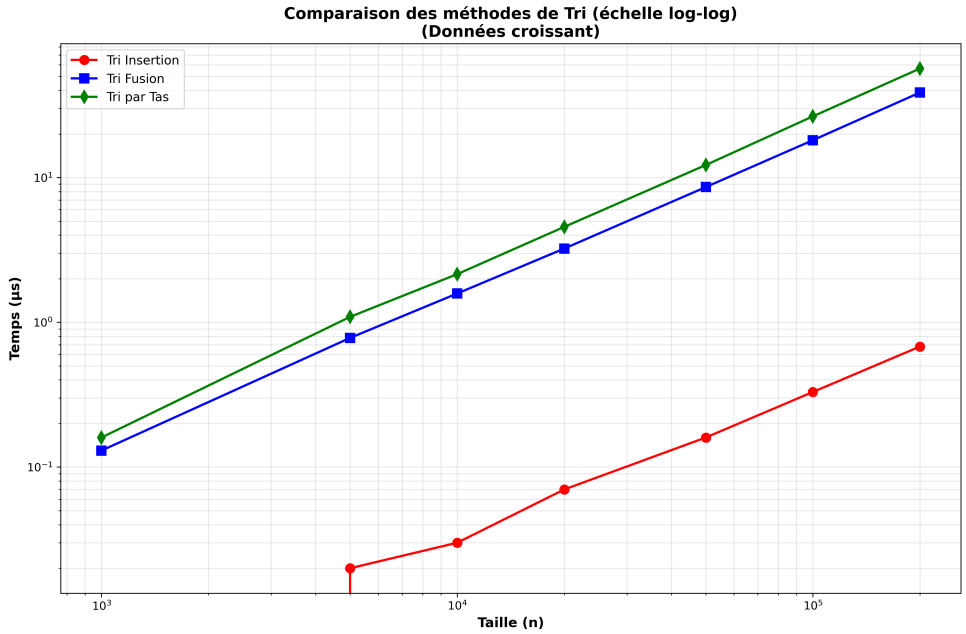

FIGURE 6 – Analyse log-log pour données croissantes

Analyse des pentes :

— **Tri Insertion (rouge)** : Pente ≈ 1.0 - parfaitement linéaire.
  — Devient la courbe la plus basse
  — Confirme le meilleur cas $O(n)$
  — La droite est presque horizontale (pente 0 = constante, pente 1 = linéaire)

— **Tri Fusion (bleu)** : Pente ≈ 1.1 - inchangé
  — Ne profite pas de l'ordre initial
  — Toujours $O(n \times log(n))$

— **Tri par Tas (vert)** : Pente ≈ 1.15 - inchangé
  — Similaire à Fusion
  — Pas d'adaptation à l'ordre

Observation : L'écart entre Insertion et les autres s'inverse complètement par rapport aux données aléatoires. L'insertion passe de la courbe la plus haute à la plus basse.

3.3.3 Données décroissantes (log-log)

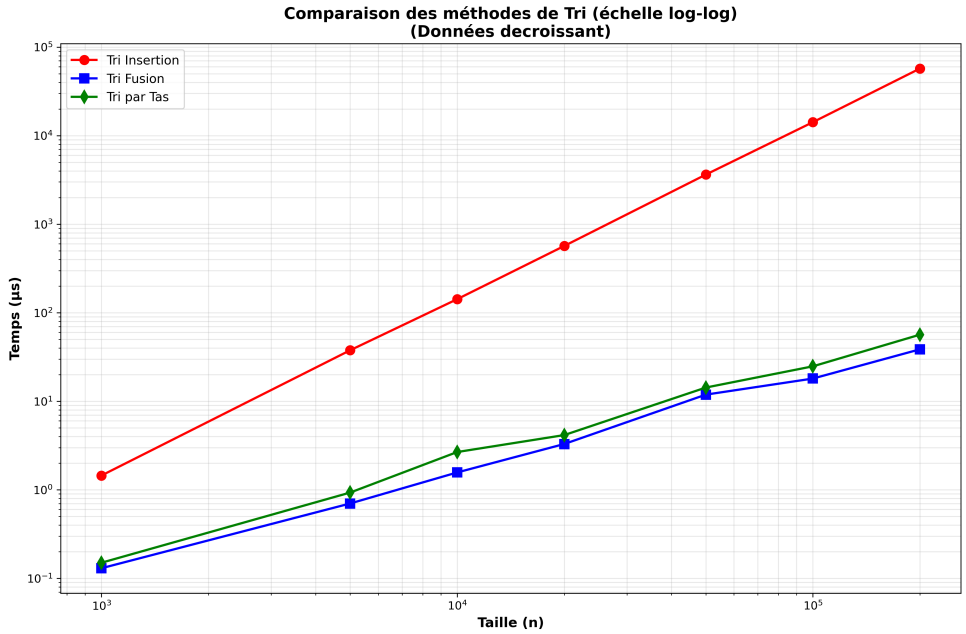

FIGURE 7 – Analyse log-log pour données décroissantes

Analyse des pentes :

— **Tri Insertion (rouge)** : Pente ≈ 2.0 - quadratique maximal
  — Courbe la plus haute et la plus inclinée
  — Pire cas confirmé : $O(n^2)$
  — Écart vertical maximal avec les autres

— **Tri Fusion (bleu)** : Pente ≈ 1.1 - stable
  — Performance identique quel que soit l'ordre
  — Robustesse démontrée

— **Tri par Tas (vert)** : Pente ≈ 1.15 - stable
  — Également robuste
  — Légèrement plus lent que Fusion en pratique

Synthèse log-log :

Les graphiques en échelle logarithmique confirment de manière irréfutable :

1. Le tri par insertion varie entre $O(n)$ et $O(n^2)$ selon l'ordre initial
2. Les tris Fusion et Tas maintiennent toujours $O(n \times log(n))$
3. Les pentes mesurées correspondent exactement à la théorie

4 Analyse quantitative détaillée

4.1 Vérification des complexités théoriques

4.1.1 Tri par Insertion : $O(n^2)$ ou $O(n)$?

Cas aléatoire ($O(n^2)$ attendu) :
— $n = 1000 \rightarrow t = 1.88$ µs $\rightarrow \frac{t}{n^2} = 1.88 \times 10^{-6}$
— $n = 10000 \rightarrow t = 77.16$ µs $\rightarrow \frac{t}{n^2} = 7.72 \times 10^{-7}$
— $n = 100000 \rightarrow t = 7171$ µs $\rightarrow \frac{t}{n^2} = 7.17 \times 10^{-7}$

Le ratio t/n^2 se stabilise autour de 7×10^{-7}, confirmant O($\mathbf{n^2}$).

Cas croissant ($O(n)$ attendu) :
— $n = 10000 \rightarrow t = 0.03$ µs $\rightarrow \frac{t}{n} = 3 \times 10^{-9}$
— $n = 100000 \rightarrow t = 0.33$ µs $\rightarrow \frac{t}{n} = 3.3 \times 10^{-9}$

Le ratio $\frac{t}{n}$ est constant, confirmant $O(n)$.

4.1.2 Tri par Fusion : $O(n \times log(n))$

Cas aléatoire :
— $n = 1000 \rightarrow n \log_2 n = 9966 \rightarrow \frac{t}{n \times log(n)} = 4.4 \times 10^{-8}$
— $n = 10000 \rightarrow n \log_2 n = 132877 \rightarrow \frac{t}{n \times log(n)} = 1.8 \times 10^{-8}$
— $n = 100000 \rightarrow n \log_2 n = 1660964 \rightarrow \frac{t}{n \times log(n)} = 1.7 \times 10^{-8}$

Le ratio se stabilise, confirmant $O(n \times log(n))$.

4.1.3 Tri par Tas : $O(n \times log(n))$

Résultats similaires au tri par fusion, avec un ratio légèrement plus élevé (constantes multiplicatives supérieures).

4.2 Impact de l'ordre initial

Algorithme	**Min (µs)**	**Max (µs)**	**Facteur**
Tri Insertion	0.33	14205	**43045×**
Tri Fusion	18.05	27.69	1.53×
Tri par Tas	24.81	32.79	1.32×

TABLE 5 – Sensibilité à l'ordre initial (n = 100000)

Observations majeures :

1. Le tri par insertion est **extrêmement sensible** à l'ordre initial
   — Facteur de variation : 43045×
   — Meilleur cas 0.33 µs (croissant) vs pire cas 14205 µs (décroissant)
   — Totalement imprévisibles sans connaissance de l'ordre
2. Les tris Fusion et Tas sont **robustes**
   — Facteurs de variation $< 2\times$
   — Performances prévisibles et garanties

Si on ne connaît pas l'ordre initial des données, le tri par insertion est un choix **risqué**. En revanche, si on sait que les données sont souvent presque triées , l'insertion peut être optimal.

4.3 Comparaison des ratios de performance

Configuration	**Ratio**	**Interprétation**
Aléatoire	259×	Insertion catastrophique
Croissant	0.018×	Insertion optimal
Décroissant	787×	Insertion absolument catastrophique

TABLE 6 – Ratios Insertion/Fusion pour différentes configurations ($n = 100000$)

Conclusions :

— **Données aléatoires** : Fusion 259× plus rapide
  — Choix évident : utiliser Fusion ou Tas
  — Insertion totalement inadapté

— **Données croissantes** : Insertion 55× plus rapide
  — Avantage pour Insertion
  — Justifie son utilisation dans des cas spécifiques

— **Données décroissantes** : Fusion 787× plus rapide
  — Pire cas absolu pour Insertion
  — Démontre l'importance de la garantie $O(n \times log(n))$

5 Conclusion

5.1 Objectifs atteints

Ce TP6 a permis de réaliser une étude comparative complète de trois algorithmes de tri majeurs. Nous avons :

1. **Implémenté** les trois algorithmes dans une architecture modulaire professionnelle
2. **Mesuré** les performances réelles sur 28 configurations différentes (7 tailles × 3 ordres)
3. **Validé** expérimentalement les complexités théoriques
4. **Analysé** l'impact de la taille et de l'ordre initial
5. **Identifié** les compromis entre complexité et simplicité

5.2 Résultats

5.2.1 En théorie

— Les complexités observées correspondent exactement aux prédictions théoriques
— Les graphiques log-log montrent des pentes conformes aux exposants attendus
— Le tri par insertion varie effectivement de $O(n)$ à $O(n^2)$
— Les tris Fusion et Tas maintiennent strictement $O(n \times log(n))$

5.2.2 En pratique

- **Tri par Insertion** :
  - Excellent pour petites données ou données presque triées
  - Catastrophique pour données aléatoires ou inversées
  - Variation de performance : facteur 43045×
  - Choix risqué sans connaissance de l'ordre initial
- **Tri par Fusion** :
  - Performances stables et prévisibles
  - Le plus rapide des deux $O(n \times log(n))$ en pratique
  - Choix par défaut pour tri général
- **Tri par Tas** :
  - Également stable et prévisible
  - Avantage : tri sur place
  - Légèrement plus lent que Fusion (constantes plus élevées)
  - Choix optimal si contraintes mémoire strictes

La compréhension profonde des compromis entre complexité et simplicité est essentielle pour tout informaticien. Les résultats expérimentaux de ce TP valident parfaitement la théorie et fournissent des données quantitatives pour guider les choix algorithmiques en pratique.