# ASD1 – LABORATOIRE 3 : TRIS

## INTRODUCTION

Dans ce laboratoire, nous avons mis en pratique plusieurs tris vus au cours, le tri par sélection et le tri rapide, ainsi que découvert le tri casier qui utilise l'arithmétique plutôt que des comparaisons pour effectuer le tri.

Nous avons également comparé les valeurs empiriques avec les valeurs théoriques calculées avec les complexités expérimentales de chaque tri, multiplié avec un facteur temps (détaillé plus bas dans le rapport).

Le rapport est composé de deux grandes parties :

1. Le test 1, qui teste les différents types de tri avec des données de taille différentes mais de même distribution [1-100].
2. Le test 2, qui teste les différents type de tri avec des données de taille fixe (1'000'000) mais de distribution différente [1-x].

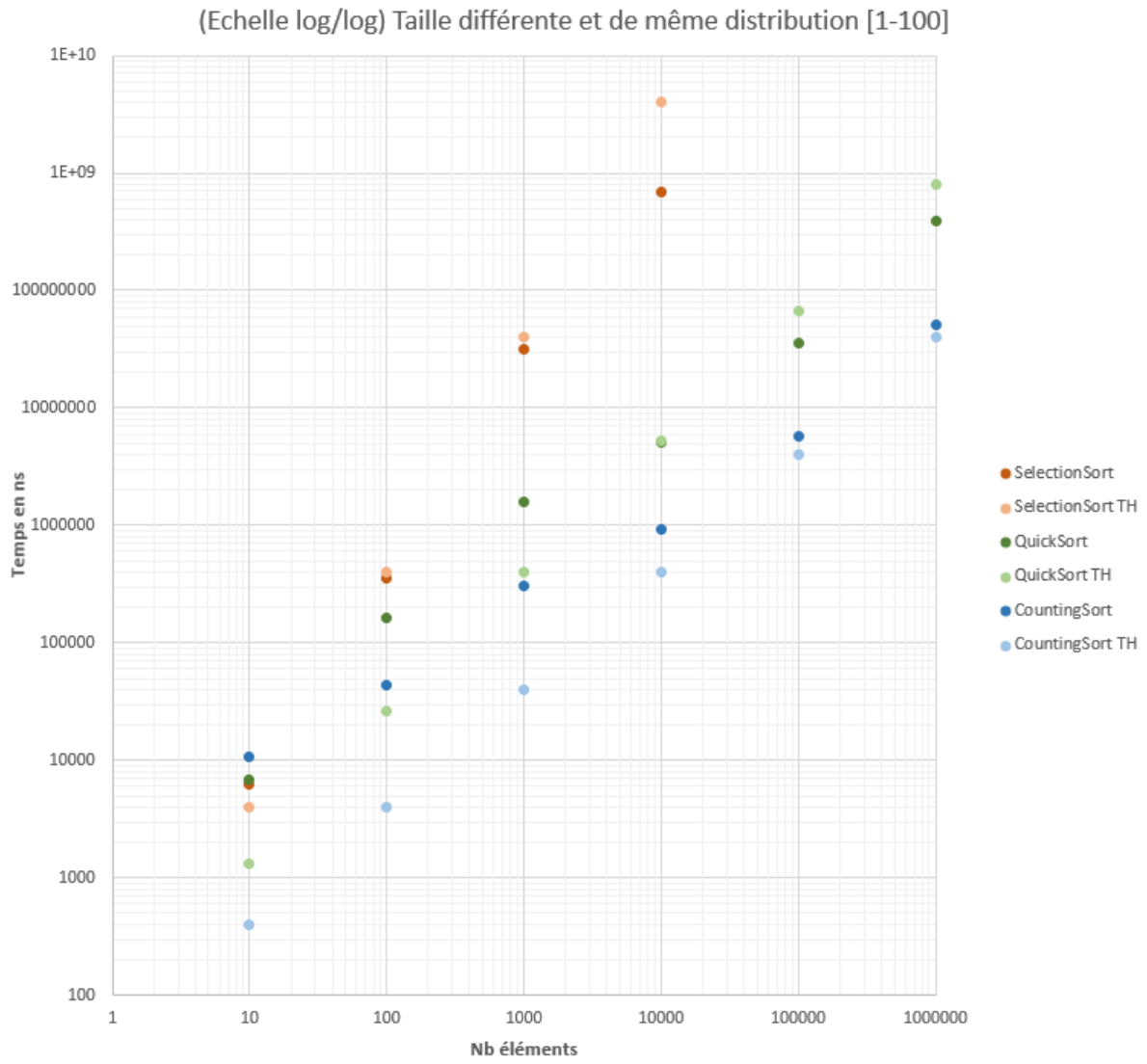
## TEST 1 : TAILLE DIFFÉRENTE ET DE MÊME DISTRIBUTION

Dans cette première série de test, nous évaluons le temps moyen d'exécution sur des vecteurs de taille différente mais ayant la même distribution [1-100], cela répété 15 fois. Nous avons choisi 15 car sinon le temps d'exécution devient beaucoup trop long et répété 15 fois le tri est assez suffisant pour en tirer une bonne moyenne. La taille des différents vecteurs varie de 10 à 1'000'000.

En ce qui concerne le tri par sélection, nous avons choisi de ne pas le tester au-delà de 10'000 éléments car ce type de tri n'est pas optimisé pour un grand nombre d'éléments et c'est le cas ici, le temps devient énormément long si on l'ajoute.

| | **Temps en ns** | | |
|---|---|---|---|
| **Nb d'éléments** | **Selection Sort** | **Quick Sort** | **Counting Sort** |
| 10 | 6189 | 6889 | 10758 |
| 100 | 356174 | 162512 | 44109 |
| 1000 | 31468900 | 1576310 | 305692 |
| 10000 | 679353000 | 5101120 | 920889 |
| 100000 | Trop long | 35877600 | 5660940 |
| 1000000 | | 386968000 | 50832300 |

Pour les valeurs théoriques, elles n'ont pas été ajoutées dans le tableau ci-dessus mais sont dans le graphique ci-dessous. Elles ont été calculées à l'aide de la complexité du tri utilisé multiplié par un facteur temps de 40 nanoseconde (nous estimons qu'une opération se passe en 40 ns). *Par exemple, pour le tri par sélection nous avons une complexité $O(n^2)$ donc pour 100 éléments, nous avons un temps théorique de 100 * 100 * 40 = 400'000 ns.*




*Ce graphique comporte les trois types de tri avec leurs valeurs empiriques et théoriques. Pour plus de visibilité nous avons regroupé les couples (empirique-théorique) par nuance de couleur.*

Nous pouvons voir que le tri par sélection est plus efficace que les autres sur un petit nombre d'éléments, mais il s'accentue de manière exponentielle (dû à sa complexité $O(n^2)$).

Pour les tris rapide et comptage, ils ont une pente similaire (valeurs empiriques) avec seulement un facteur qui change, ce qui est normal car le comptage est en O(n) et le rapide est en O(n * log(n)) nous avons donc un rapport log(n) entre les deux.

En ce qui concerne les couples empirique et théorique, pour chaque tri nous voyons que la théorie est meilleure quand il y a moins d'éléments et devient moins bonne sur le long terme. La raison est sûrement qu'il y a une différence entre la complexité théorique et réelle des algorithmes de tri (par exemple, le tri comptage à une complexité théorique de O(n) mais en pratique nous avons O(2*n + m) m étant le nombre de « boîte ».

Mais de manière générale nous pouvons voir qu'il respecte une droite plus ou moins proche de celle des valeurs empiriques.

## TEST 2 : MÊME TAILLE ET DE DISTRIBUTION DIFFÉRENTE

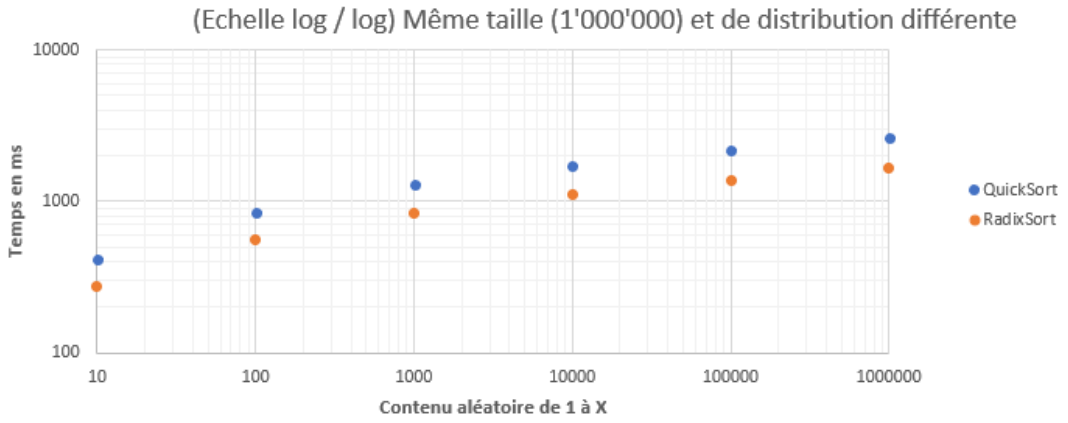
Pour la deuxième batterie de tests, nous avons utilisé un conteneur de 1'000'000 d'éléments avec des valeurs générées aléatoirement variant de 1 à x avec x = {10^m | m ∈ [1,2, ⋯ ,6]} et le tout répété 15 fois également, pour les mêmes raisons que le test1.

Nous avons choisi de ne pas tester le tri par sélection car ce type de tri n'est pas optimisé pour un grand nombre d'éléments et c'est le cas ici, le temps devient énormément long si on l'ajoute.

Selon la donnée, il est suggéré d'utiliser le tri comptage chiffre par chiffre mais nous avons décidé d'utiliser le Radix sort, car il a un comportement similaire, à l'exception qu'il fonctionne octets par octets.

| | | **Borne supérieur => distribution [1-x]** | | | | | |
|---|---|---|---|---|---|---|---|
| | **x** | **10** | **100** | **1'000** | **10'000** | **100'000** | **1'000'000** |
| **Temps en ms** | **QuickSort** | 415 | 854 | 1292 | 1729 | 2189.98 | 2644 |
| | **RadixSort** | 275 | 555 | 840 | 1099 | 1386 | 1655 |

En ce qui concerne les valeurs théoriques des deux algorithmes de tris, nous n'avons décidé de ni les mettre dans le tableau ci-dessus, ni dans le graphique ci-dessous. Ce choix vient du fait que ces valeurs dépendent du nombre d'éléments qui, dans ce cas, est constant (1'000'000 d'éléments) et n'apporte pas grand-chose à la compréhension du graphique.




Sur le graphe ci-dessus, on peut remarquer que les deux fonctions ont exactement la même pente mais qu'il y a un facteur qui fait que le Radix sort est légèrement meilleur que le Quick sort. Ce facteur peut être expliqué lorsque l'on compare leur complexité. Le Quick sort à un complexité O(n*log(n)) et le Radix O(n). Il y a donc un facteur log(n) qui différencie ces deux algorithmes de tri.

## CONCLUSION

Nous avons donc pu observer dans ce laboratoire, que pour peu d'éléments, le tri par sélection est plus efficace que les deux autres. Cependant, cela change très vite, et rapidement nous avons pu constater que le radix sort/counting sort sont bien plus efficace.

De plus, nous avons appris à créer nous-mêmes les algorithmes de tri et donc mettre en pratique ce que nous avons appris en cours.