Laboratoire 3 : tri

Introduction :

Le but de ce laboratoire est de mettre en œuvre en C++ trois algorithmes de tri et d'en comparer la complexité. Les trois algorithmes sont :

- L'algorithme en $O(n^2)$ sera le tri par sélection (***selection sort***).
- L'algorithme en $O(n * \log_2(n))$ sera le tri rapide (***quick sort***).
- L'algorithme en $O(n)$ sera le tri par comptage, également appelé le tri casier (***counting sort***)

Tests :

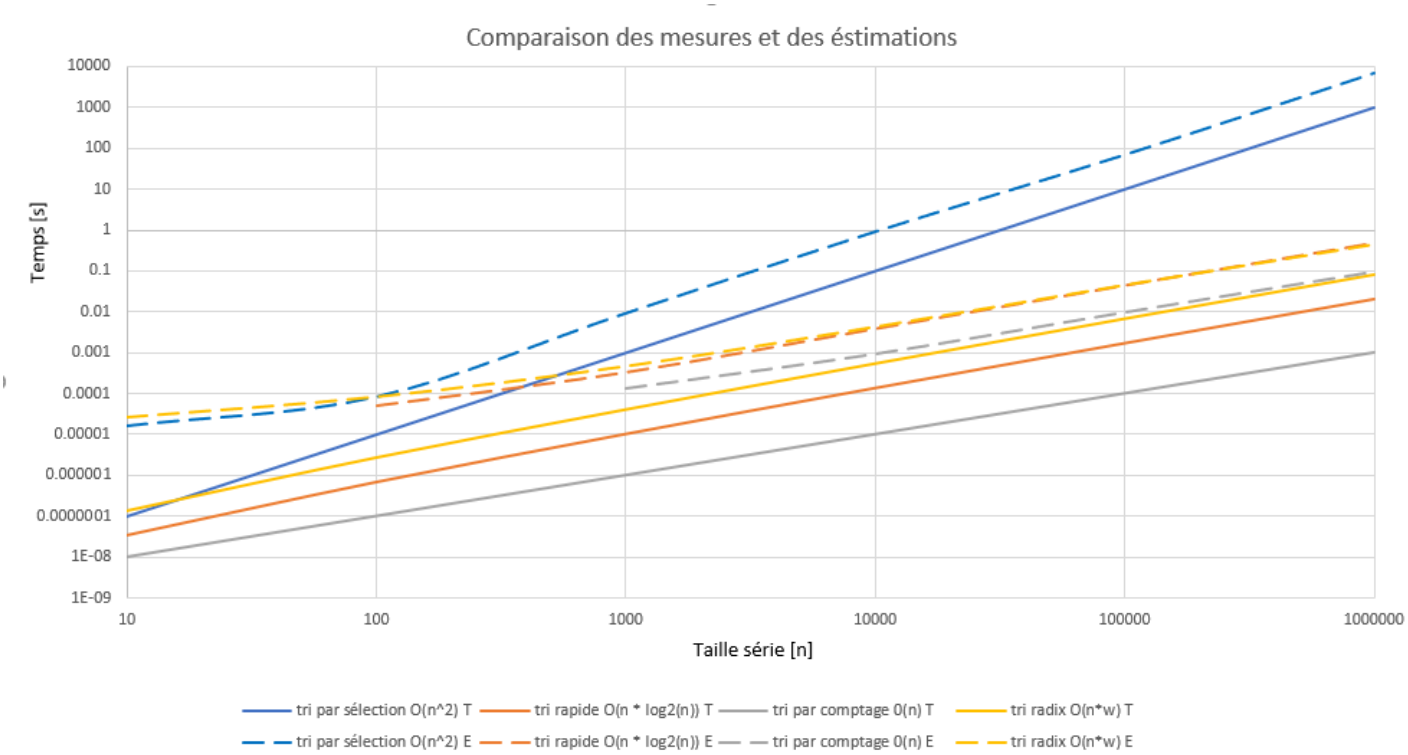
Pour calculer la complexité théorique nous allons partir du fait qu'une **itération ou opération est égale à une nanoseconde** et que les algorithmes de tri sont sur des séries de taille différente mais de même distribution. **Les valeurs sont représentées en nanosecondes.**

Estimation théorique				
Taille tableau [n]	Tri par sélection $O(n^2)$	Tri rapide $O(n * \log_2(n))$	Tri par comptage $O(n)$	Tri radix $O(n*w)$
10	0.0000001	3.32E-08	0.00000001	1.33E-07
100	0.00001	6.64E-07	0.0000001	2.66E-06
1000	0.001	9.97E-06	0.000001	3.99E-05
10000	0.1	0.000132877	0.00001	0.000531508
100000	10	0.001660964	0.0001	0.006643856
1000000	1000	0.019931569	0.001	0.079726274

Pour chaque calcul des mesures nous avons fait 30 simulations. Pour calculer la complexité empirique avec des séries de taille différente mais de même distribution :
Pour la taille selon $n = \{10^n \mid n \in [1,2,3,4,5,6]$. Et des valeurs entre 1-100 [bornes comprises]. Pour le tri par sélection, nous avons fait des calculs avec la taille maximum de 10^4 car les temps de calcul étaient trop long.

Mesures				
Taille tableau [n]	Tri par sélection $O(n^2)$	Tri rapide $O(n * \log_2(n))$	Tri par comptage $O(n)$	Tri radix $O(n*w)$
10	0.0000163	0	0	6.61333E-05
100	0.0000835333	0.0000502	0	8.26333E-05
1000	0.00927537	0.000330633	0.000132267	0.000463
10000	0.92061	0.00386863	0.0009276	0.0042822
100000	-	0.044524	0.00970433	0.0451342
1000000	-	0.48923	0.0954608	0.447777

Nicolas Hungerbühler,
Florian Schaufelberger,
Maurice Lehmann

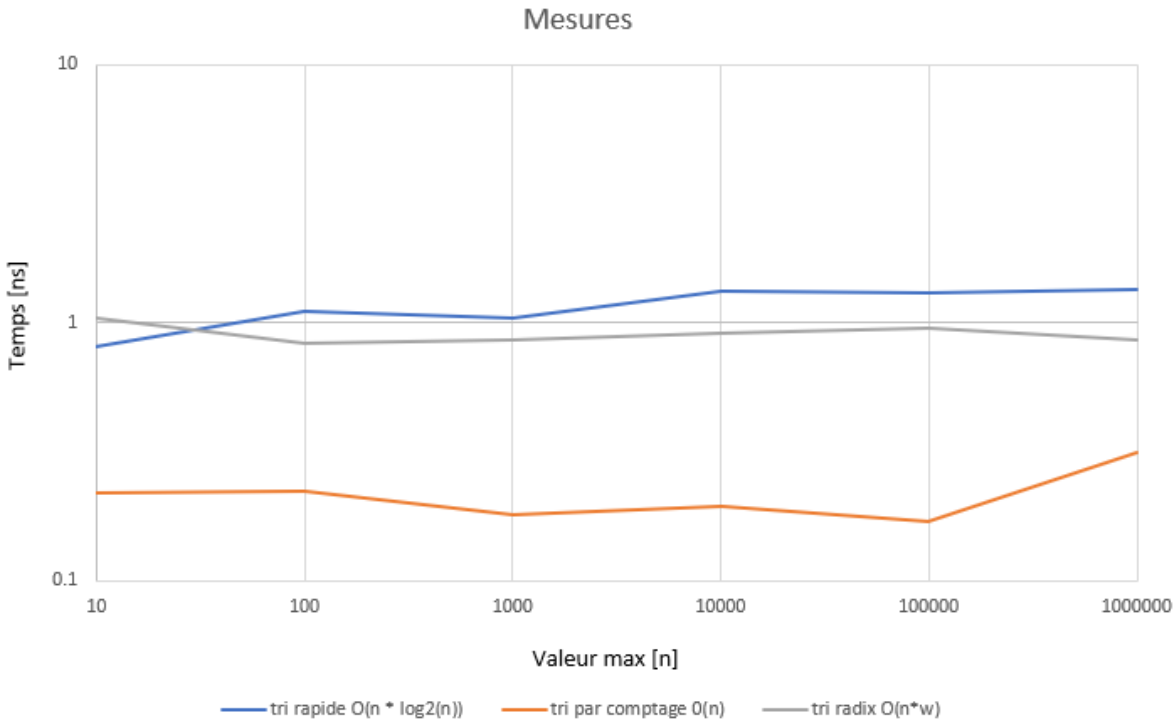


Les courbes ont la même tendance, cependant on constate que les courbes des estimations théoriques sont plus petites que celles des valeurs calculées. Le graphe démontre bien que le tri par comptage est bien le plus rapide. Le tri rapide et le tri par comptage sont très proches, par contre le tri par sélection évolue de manière quadratique ce qui veut dire que ce n'est pas un bon algorithme pour de grandes séries. Les tris de petits tableaux nous donnent un temps de zéro seconde en moyenne, nous ne les avons pas représenté dans le graphique.

Pour calculer les mesures avec des séries de même taille mais de distributions différentes :
On utilise une taille de 10^6. Les valeurs selon $n = \{10^n \mid n \in [1,2,3,4,5,6]$, toujours avec les bornes comprises. Nous avons baissé le nombre de simulations à 20 pour ces tests. Nous n'avons pas testé le tri par sélection car les calculs sont trop longs avec une taille de tableau aussi grande, et on ne voulait pas fausser les résultats avec les autres tris. **Les valeurs sont représentées en nanosecondes.**

Mesures			
Valeur maximale [n]	Tri rapide $O(n * \log_2(n))$	Tri par comptage 0(n)	Tri radix O(n*w)
10	0.805154	0.21927	1.03789
100	1.10199	0.222254	0.830374
1000	1.04482	0.18019	0.863997
10000	1.33691	0.193269	0.907009
100000	1.29938	0.169141	0.958337
1000000	1.348483	0.315384	0.854945

Nicolas Hungerbühler,
Florian Schaufelberger,
Maurice Lehmann



On constate que chaque algorithme est à peu près constant. On en déduit que la distribution d'une série n'influence pas la complexité sur ces algorithmes-là. Le tri radix est, logiquement, à peu près 4x plus long que le tri de comptage. Cependant, on ne calcule pas, mais le tri radix économise bien plus de mémoire que les deux autres tris.

Nicolas Hungerbühler,
Florian Schaufelberger,
Maurice Lehmann