Université de Nice - Sophia Antipolis

Polytech Nice Sophia

Sciences Informatiques - 5ème année

Programmation Fine et Complexité Empirique

Etude d'algorithmes en C++

Alexis Gardin et Jérémi Ferré

Enseignant : Christophe Papazian

1 Contexte

Pour cette étude, nous devions choisir un langage de programmation à étudier en terme de performance. Nous avons décider de nous pencher sur le **C++**, un langage multi-paradigmes utilisé depuis déjà quelques années dans l'industrie, et célèbre entre autres pour sa flexibilité, ses possibilités de robustesse et sa difficulté de lisibilité.

L'étude de la performance est évaluée par le temps d'exécution de **différents algorithmes de tri**, tels que le tri par insertion, le tri par sélection, le tri fusion, le tri par tas, le tri rapide, le tri comptage, et enfin le tri natif du C++.

2 Fonctionnement du code

*Notre code source est disponible **ici**.*

Le projet est divisé en 3 fichiers principaux :

— **Generator.h**, une classe utilitaire permettant de générer des listes de test, contenant des valeurs alétoires ou ordonnées.

— **Tris.h**, notre classe contenant les différents algorithmes de tris précédemment cités, toujours implémentés de telle sorte que l'on puisse passer en paramètre un tableau à trier **tab** et sa taille **N**.

— **main.cpp**, le point d'entrée de notre programme, lançant un benchmark sur tous les algorithmes de tri de la classe *Tris.h*, en effectuant toujours **3 itérations** sur un nombre d'entrées n suivant une évolution en 2^n, et comparé entre des tableaux **aléatoires**, des tableaux triés dans l'**ordre croissant**, et des tableaux triés dans l'**ordre décroissant**.

2.1 Exécuter le projet

— *>$ mkdir build && cd build*
— *>$ cmake .. && make*
— *>$ ./ProgFine*

2.2 Génération des résultats

L'exécution du programme produit un fichier *benchmarkresult.csv* contenant les résultats comparatifs des différents algorithmes de tri avec les différentes tailles et ordres de tableaux.

Nous avons utilisé ce fichier pour produire des statistiques et des graphiques en utilisant Python (plus précisémment Jupyter Notebook).

3 Comparaison des algorithmes en C++

Après avoir mis nos résultats avec une échelle logarithmique, nous avons appliqué une régression linéaire pour une meilleure représentation des différences de performances des différents tris. Les 3 graphiques ont pour l'axe des abscisses une représentation du temps et en ce qui concerne l'axe des ordonnées il s'agit de la taille de cette liste. Pour pouvoir mieux comparer les différents algorithmes nous les avons testé avec des liste

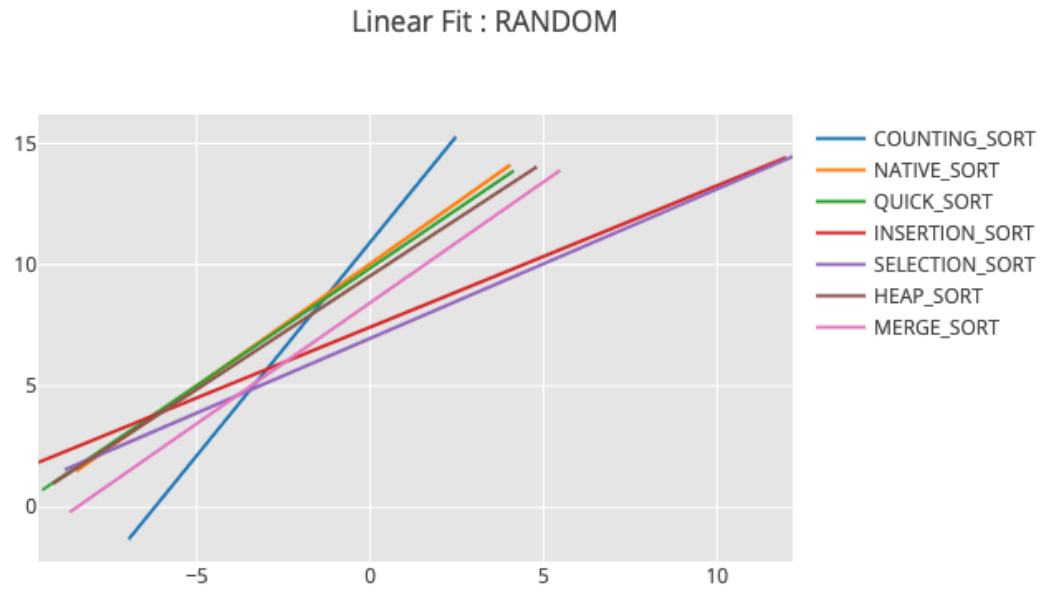

FIGURE 1 – Comparatif des tris avec des tableaux aléatoires

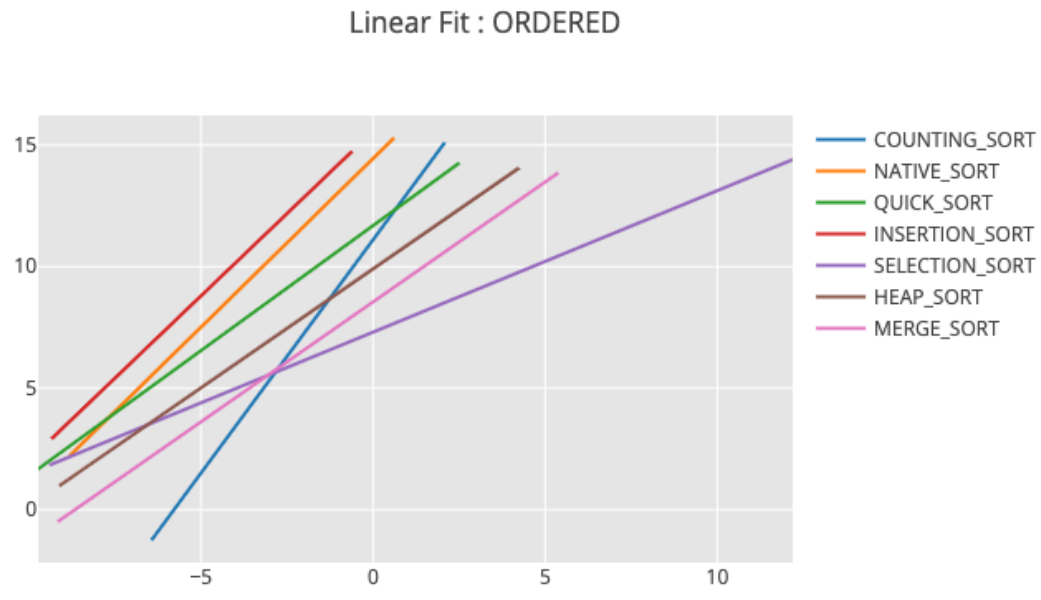

FIGURE 2 – Comparatif des tris avec des tableaux déjà triés

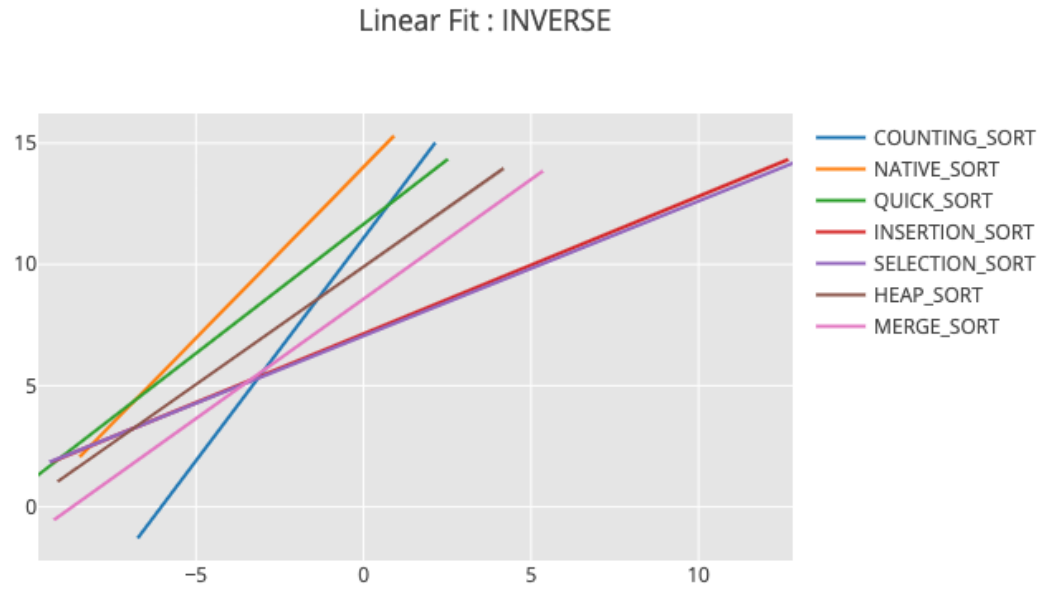

FIGURE 3 – Comparatif des tris avec des tableaux triés en sens inverse

4 Difficultés rencontrées

Au départ, nous avions choisi d'utiliser la librairie de micro-benchmarking **Google Benchmark** pour calculer le temps d'exécution des algorithmes. Néanmoins, des résultats nous ont paru anormaux sur certaines valeurs et certains tris, nous avons donc basculé sur notre propre benchmark manuel.