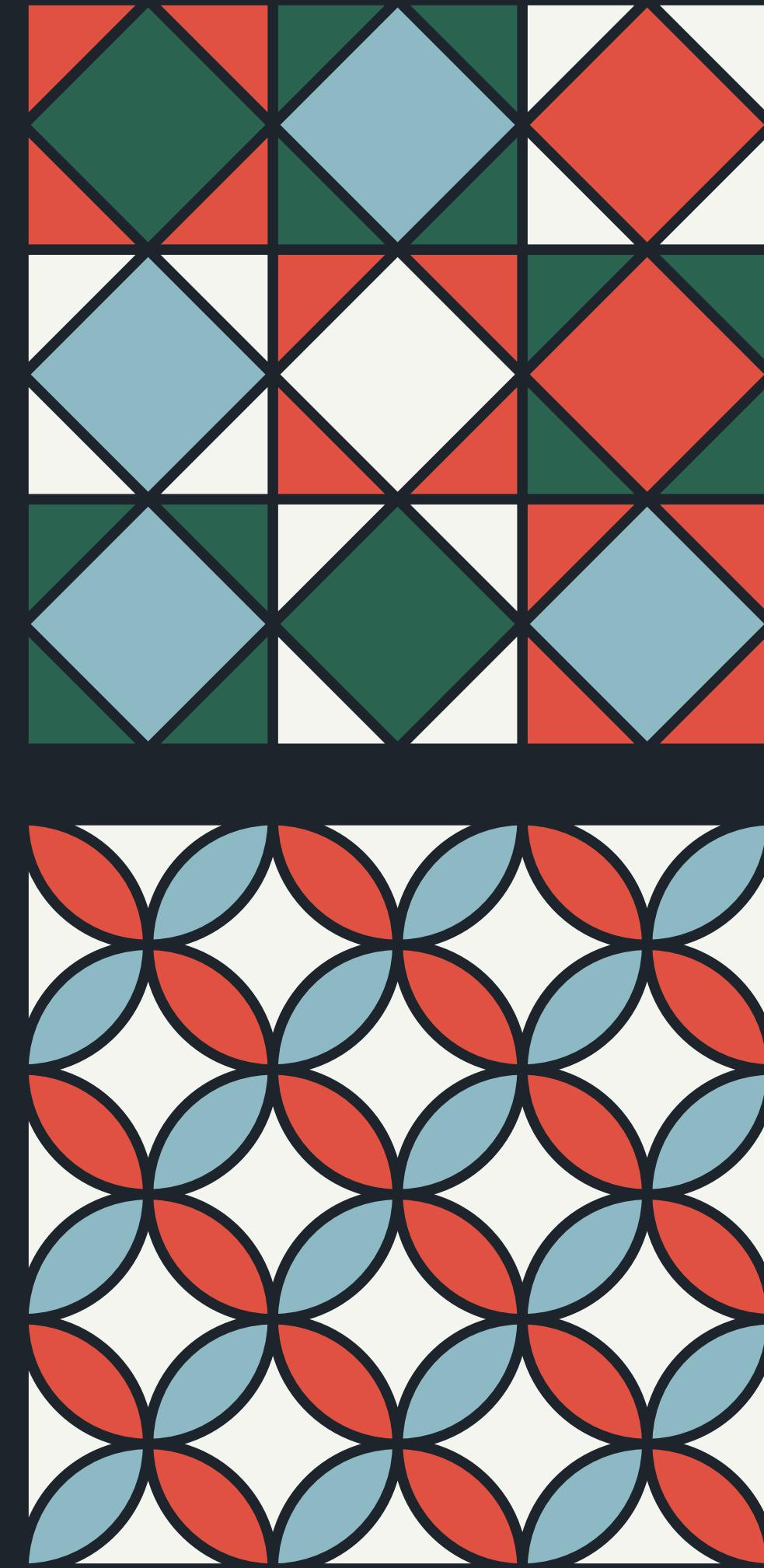
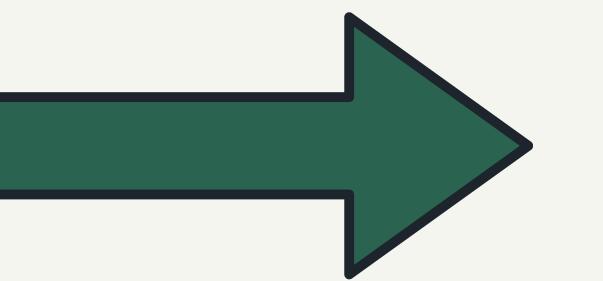
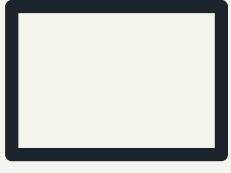


ALGORITMI DE SORTARE



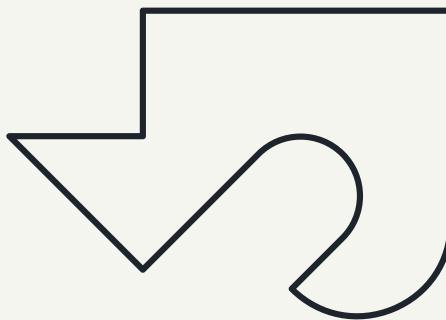
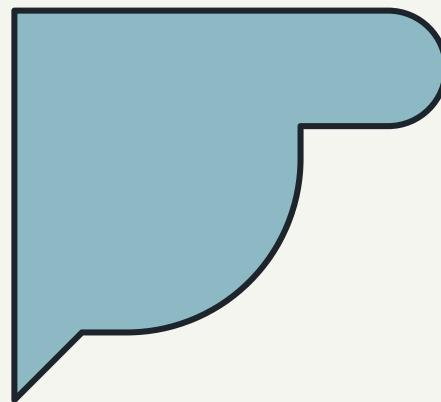


ALGORITMI DE SORTARE



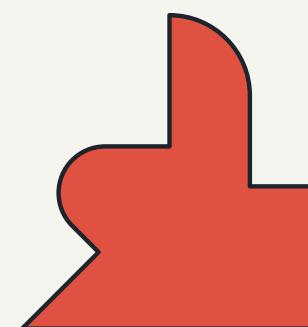
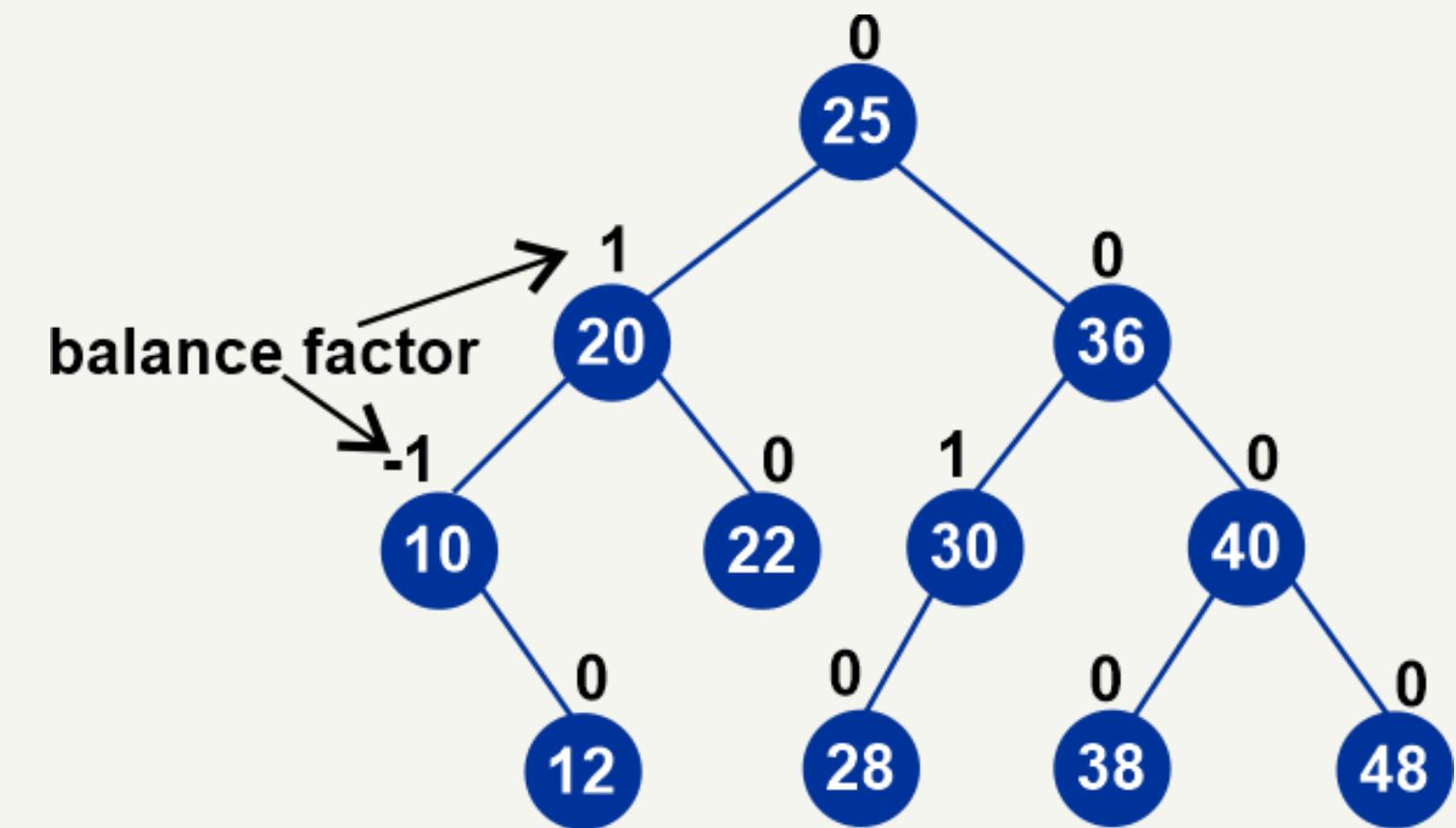
ALGORITMI ALESI:

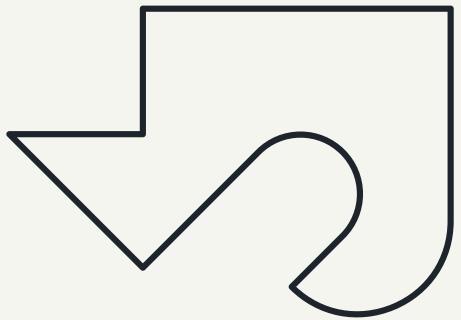
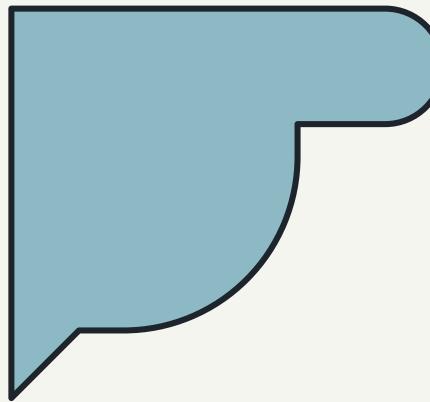
1. AVL Sort
2. Counting Sort
3. Merge Sort
4. Quick Sort
5. Radix Sort
6. Shell sort
7. STL sort



AVL SORT

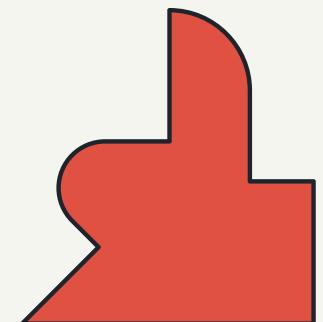
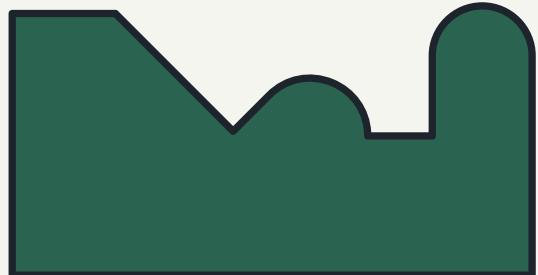
Sortarea AVL este un algoritm de sortare care folosește structura de date a arborelui AVL. Acesta construiește un arbore AVL din elementele de intrare și apoi efectuează o traversare în ordine a arborelui rezultat pentru a obține elementele sortate. În implementarea noastră retinem și frecvența fiecarui număr.





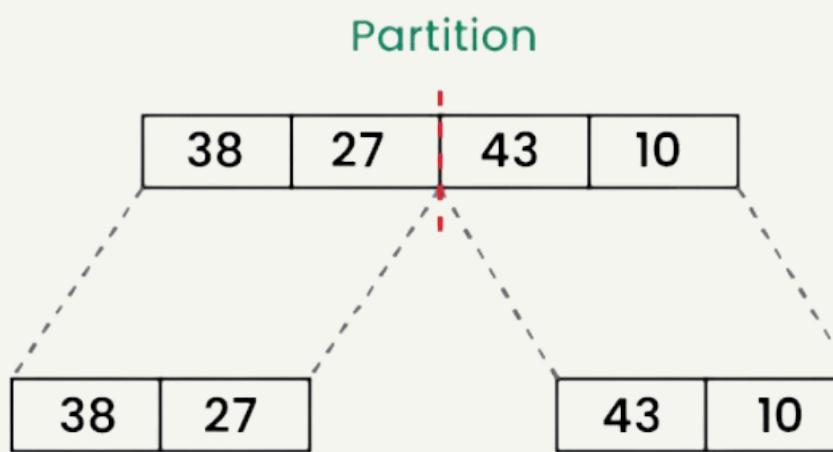
COUNTING SORT

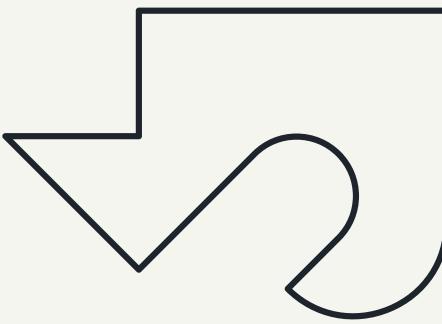
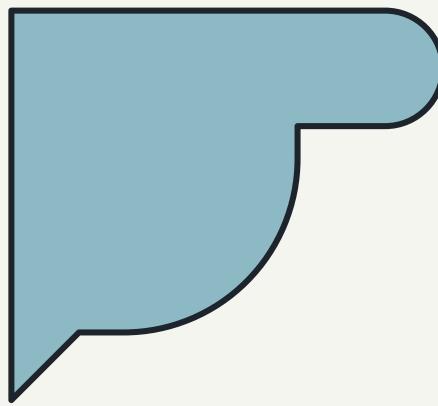
Sortarea prin numărare (Counting Sort) este un algoritm eficient de sortare care sortează elementele în funcție de numărul lor de apariții într-o secvență. Algoritmul presupune că elementele sunt întregi și apar într-un anumit interval finit.



MERGE SORT

Merge sort este un algoritm recursiv care împarte continuu matricea în jumătate până când nu poate fi împărțită încă odată, adică vectorul mai are un singur element (un vector cu un element este întotdeauna sortat). Apoi sub-vectorii sortați sunt îmbinati într-o singură matrice sortată.

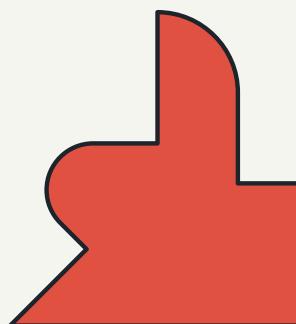
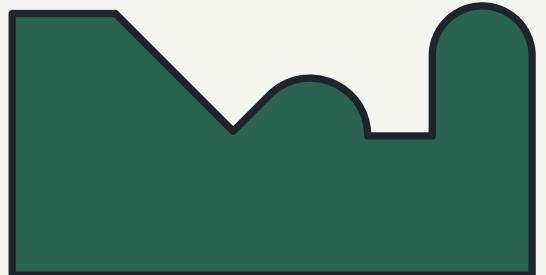


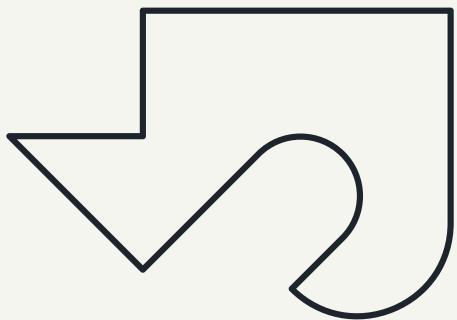
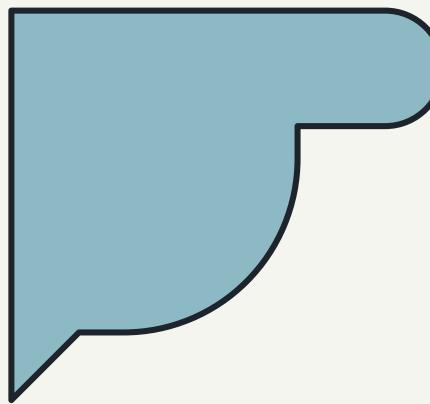


QUICK SORT

Quick sort este un algoritm de sortare "împarte și cucerește". Alegând un **pivot** și rearanjând lista astfel încât elementele mai mici să fie înaintea pivotului, iar cele mai mari după acesta, algoritmul sortează recursiv subliste până când lista este complet sortată.

O sa facem mai multe variatii ale acestui algoritm si o sa le comparam!

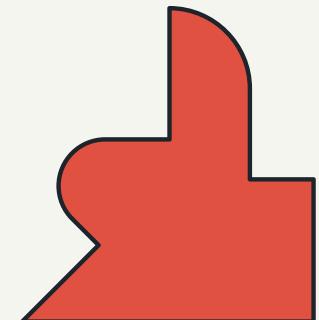


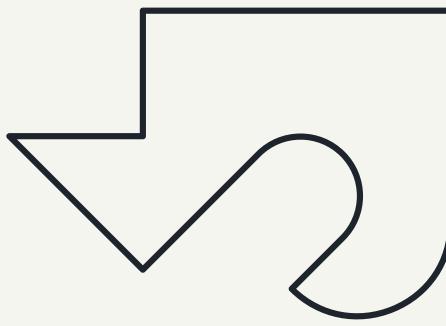
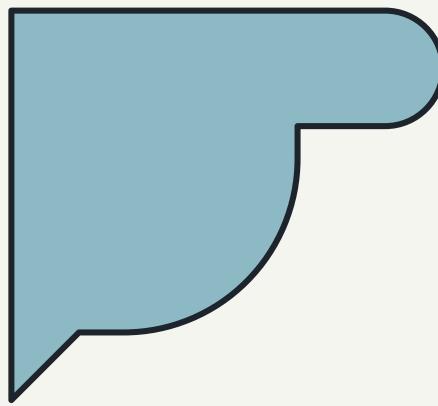


RADIX SORT

Sortarea radix (Radix sort) este un algoritm de sortare care sortează elementele pe baza cifrelor acestora. Algoritmul funcționează prin efectuarea unui număr fix de treceri prin lista de elemente, împărțind elementele în grupuri pe baza cifrelor lor de la cifra cea mai puțin semnificativă la cea mai semnificativă. La fiecare trecere, elementele sunt rearanjate în funcție de cifra curentă, astfel încât la final lista este sortată în întregime.

În implementarea noastră acesta folosește Counting Sort!

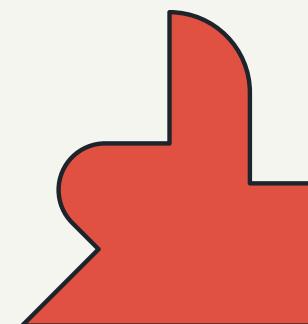
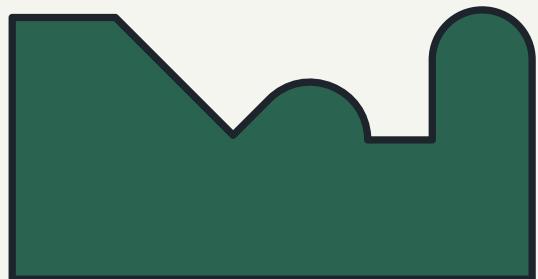




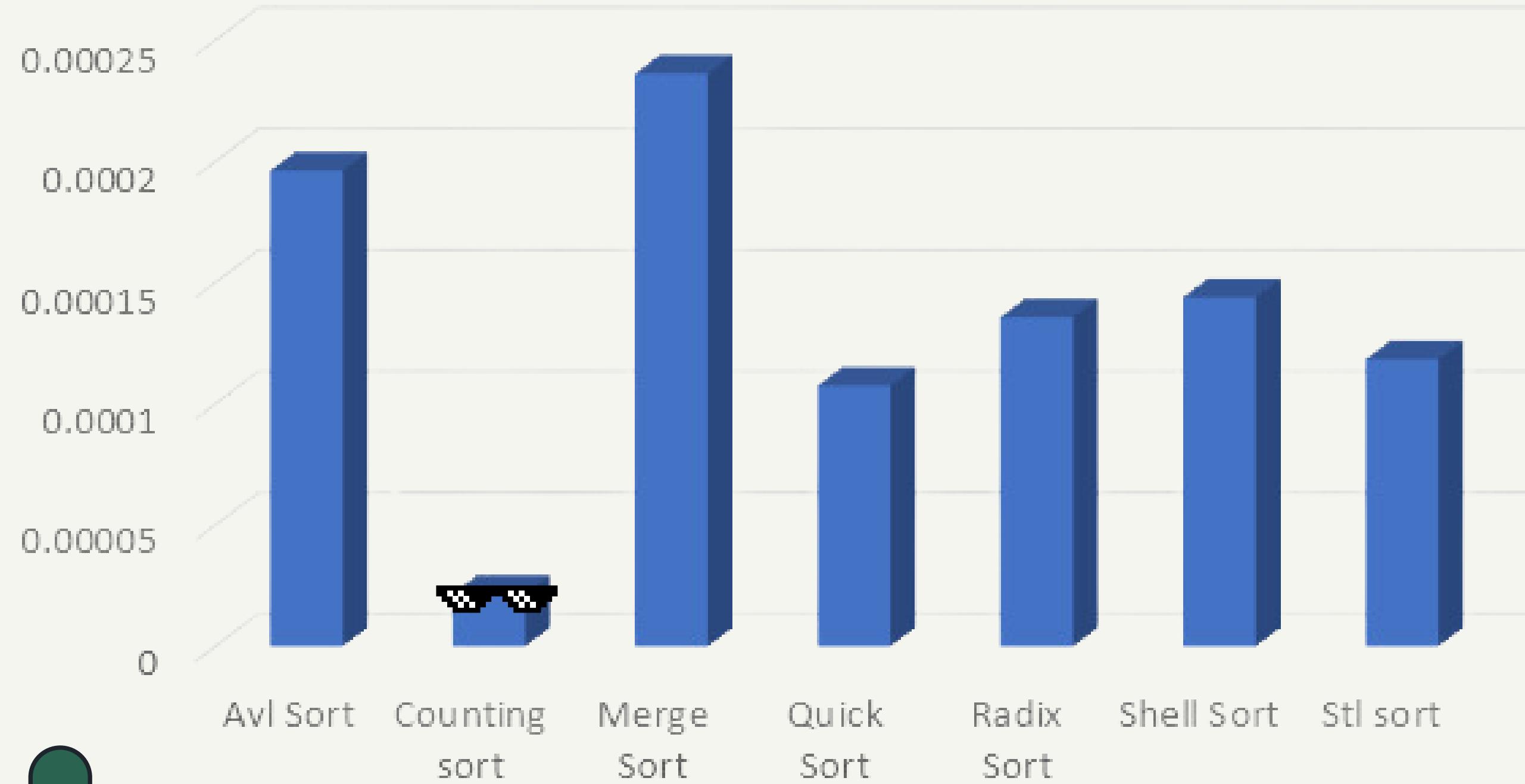
SHELL SORT

Shell sort este un algoritm de sortare care extinde metoda de **sortare prin inserție**. Acesta funcționează prin împărțirea listei în subliste mai mici, care sunt sortate folosind sortarea prin inserție. Aceste subliste sunt apoi "împinse" împreună printr-o serie de pasări, folosind un pas mic de comparare, până când întreaga listă devine aproape sortată. În cele din urmă, sortarea prin inserție este aplicată din nou pentru a finaliza sortarea listei.

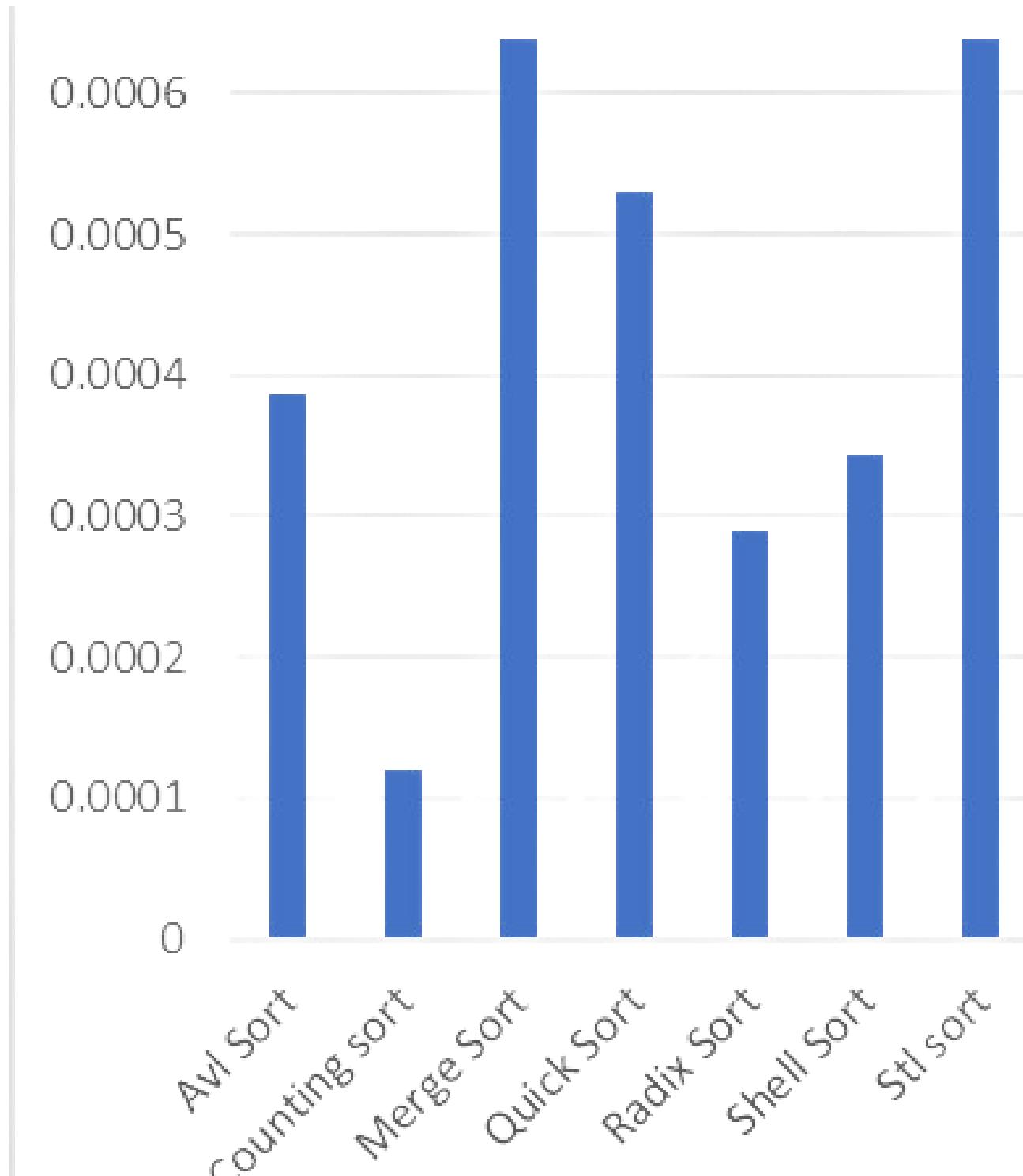
În implementare gapul porneste de la $n/2$ și se injumatatește succesiv!



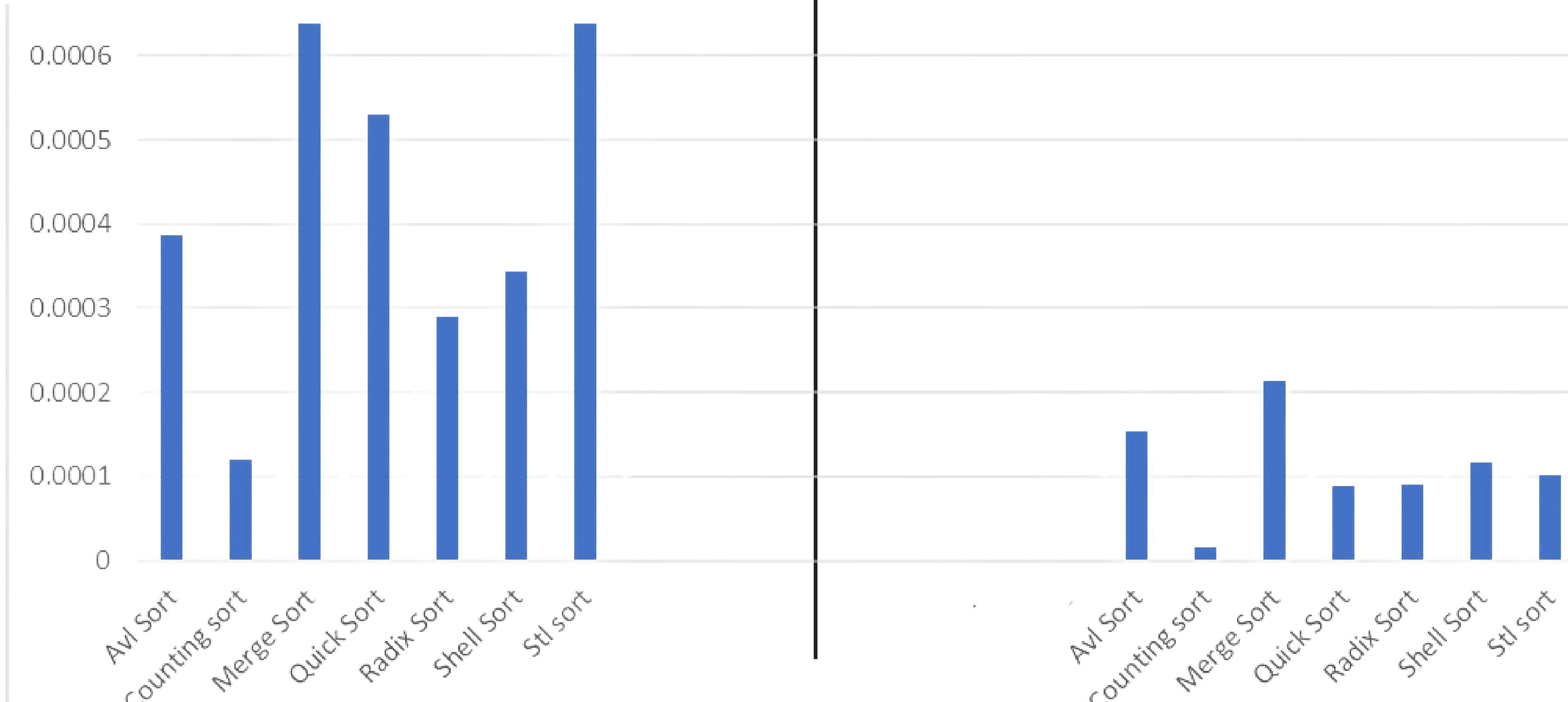
Teste $n = 10^3$, Max = 10^3



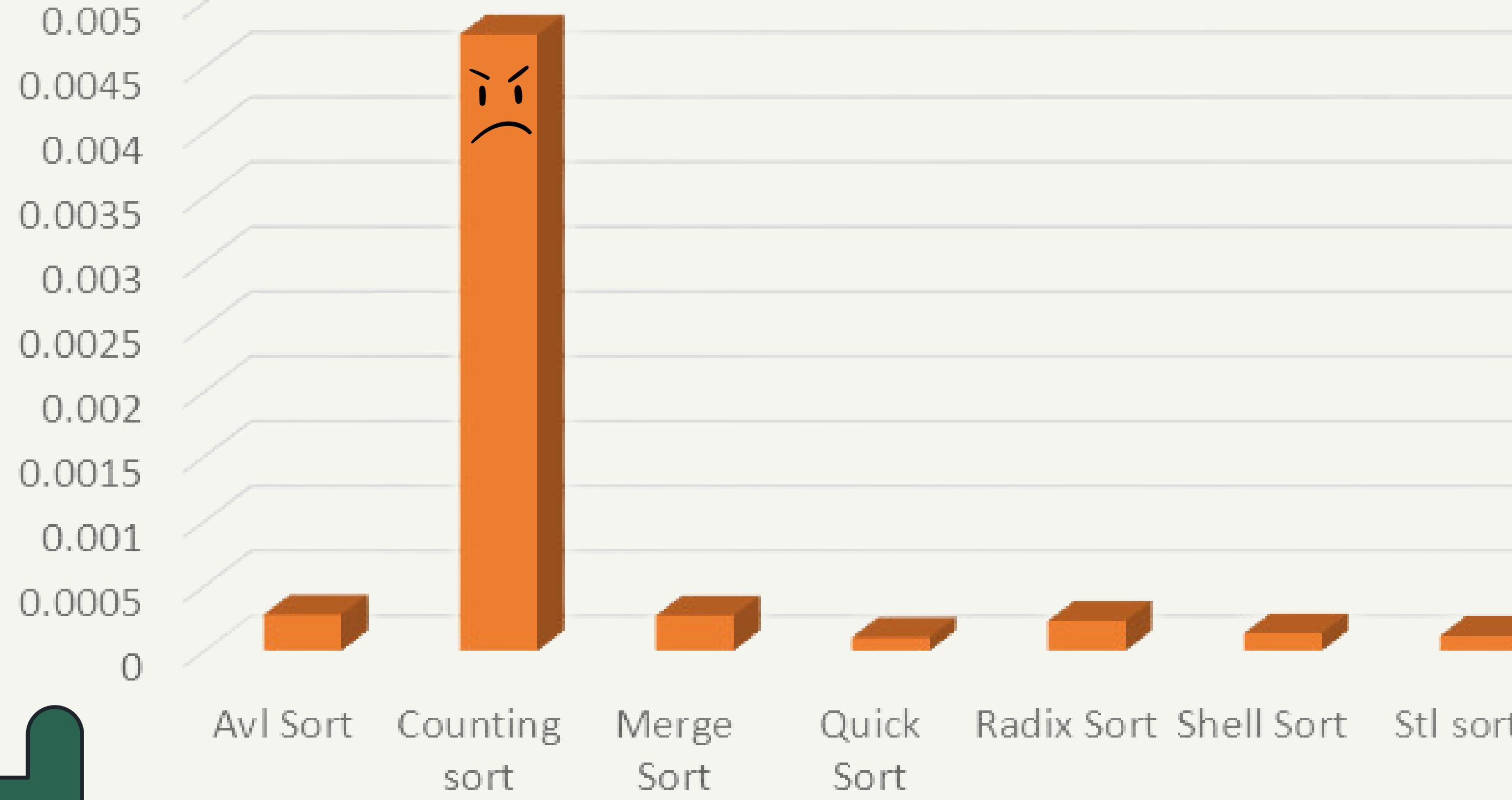
Worst case



Best Case

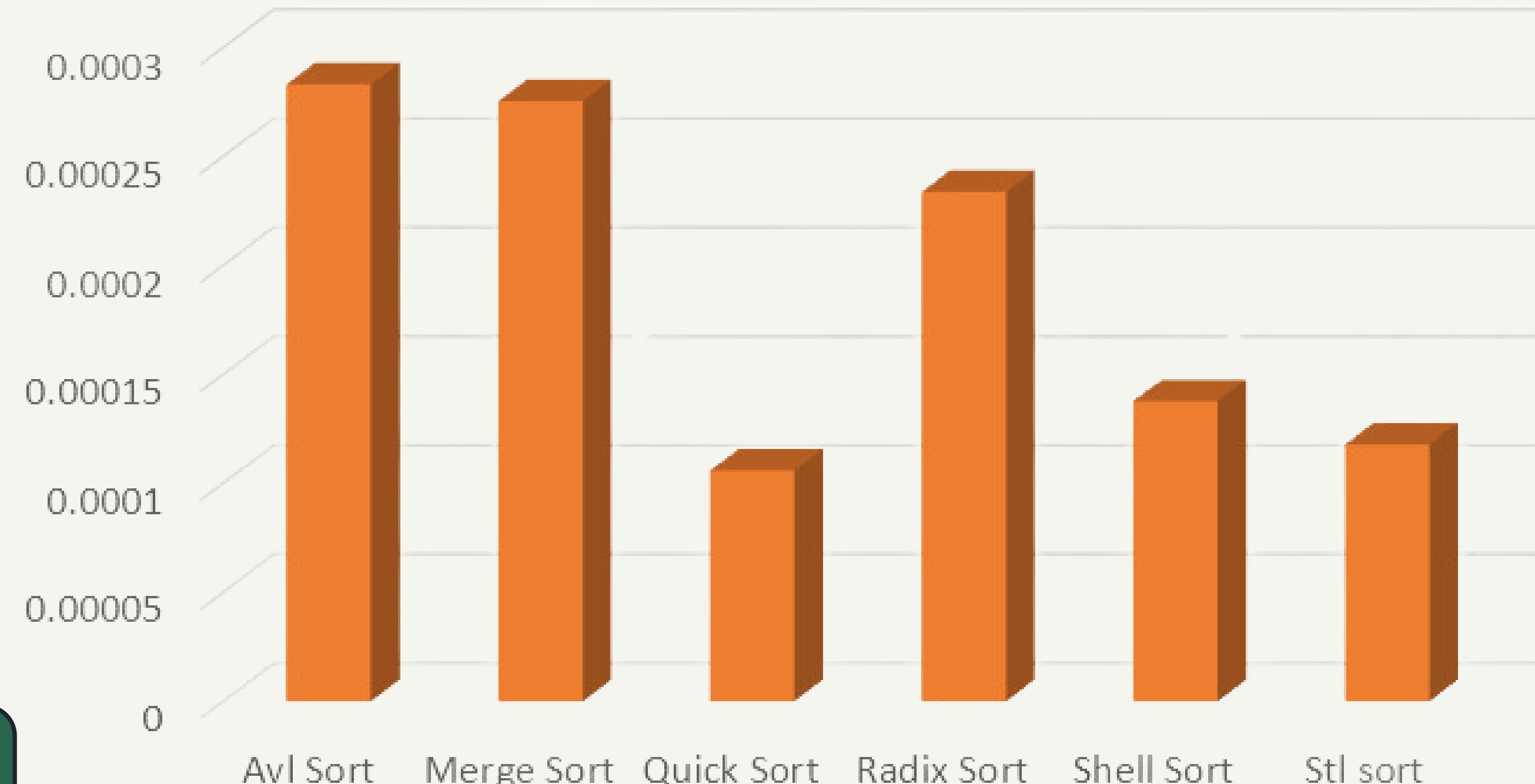


Teste $n = 10^3$, Max = 10^6

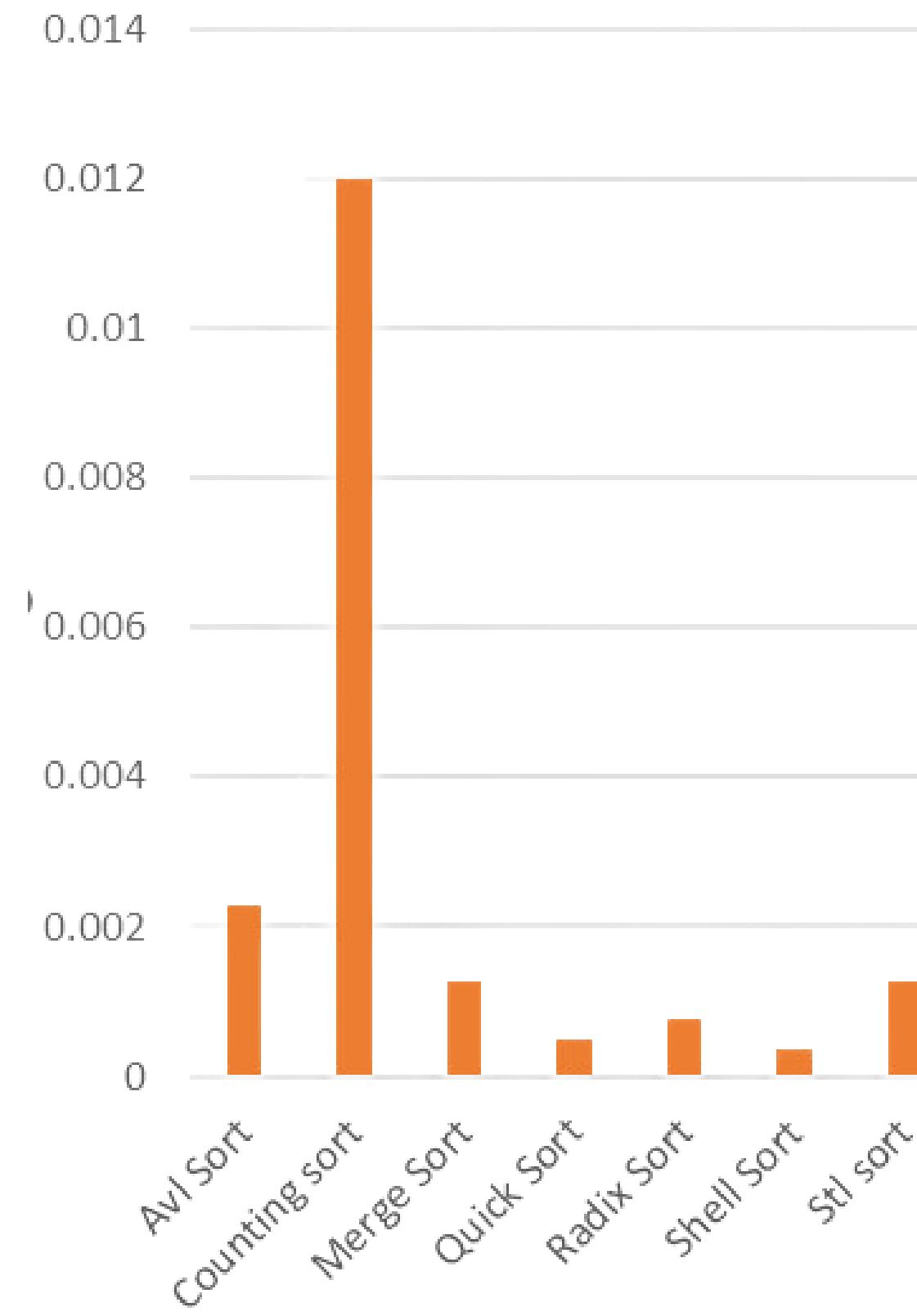


Teste $n = 10^3$, Max = 10^6

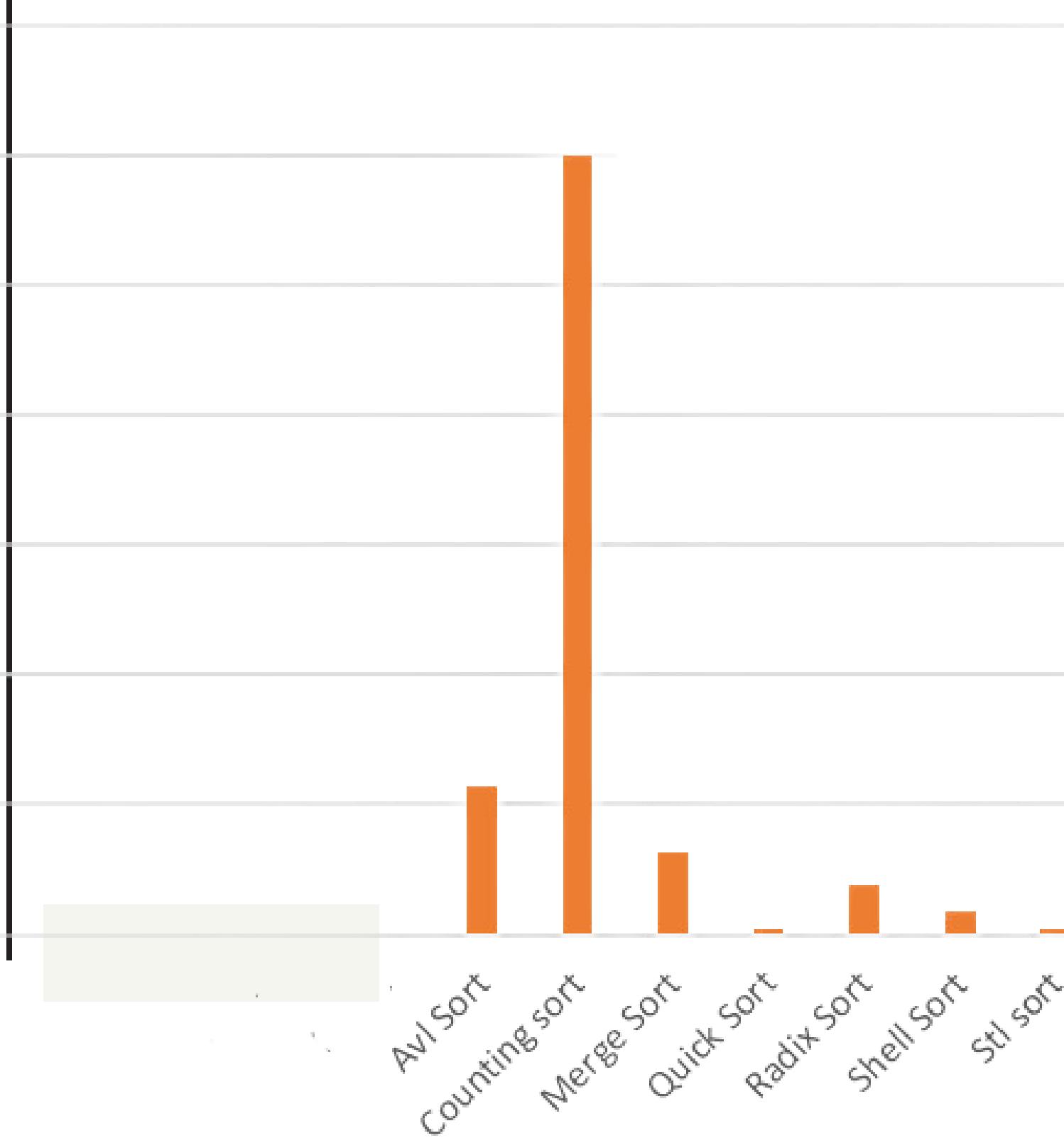
fara Counting Sort



Worst case



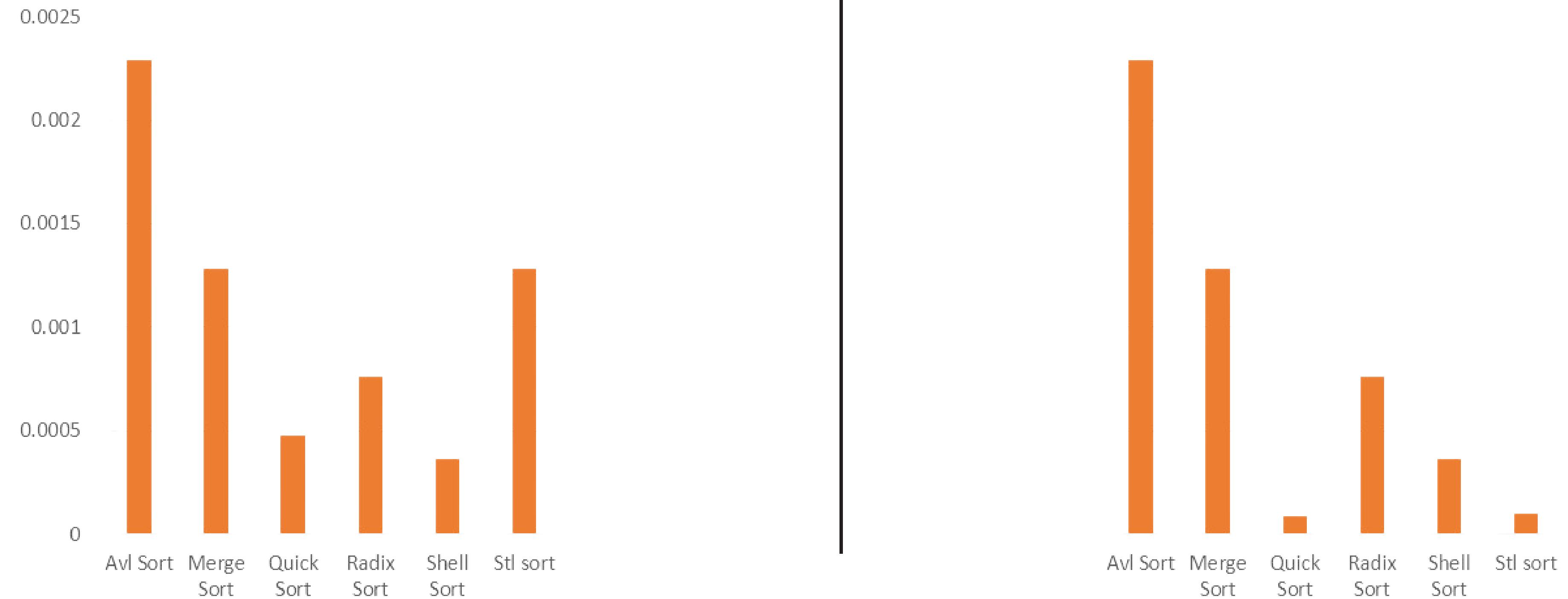
Best Case



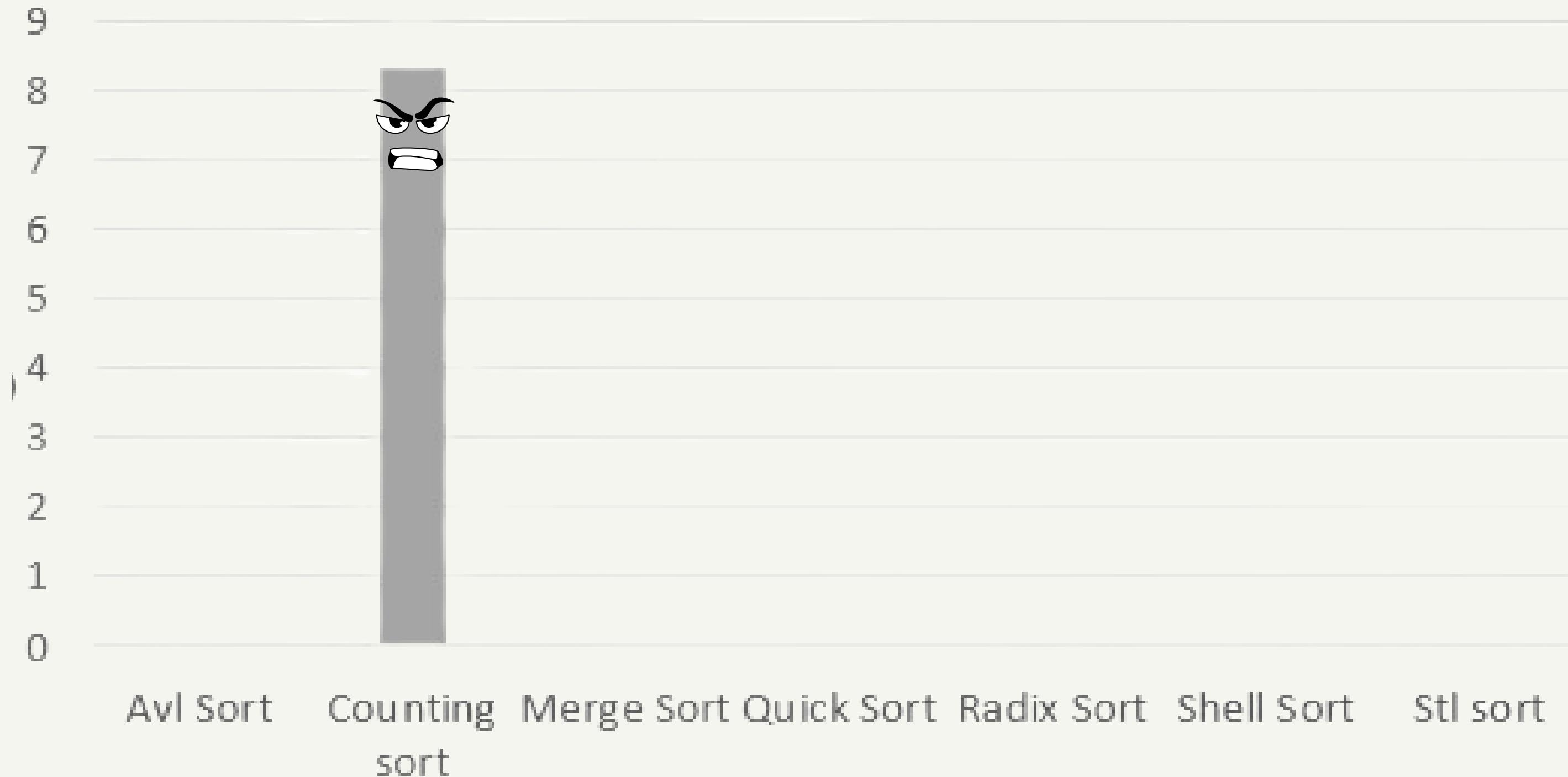
para Counting Sort

Worst case

Best Case



Teste $n = 10^3$, Max = 10^9



Teste $n = 10^3$, Max = 10^9

0.0012

0.001

0.0008

0.0006

0.0004

0.0002

0

Avl Sort

Merge Sort

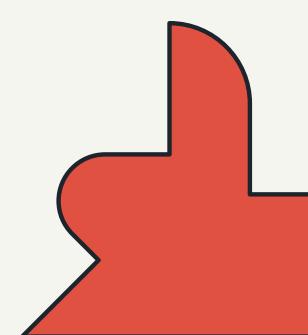
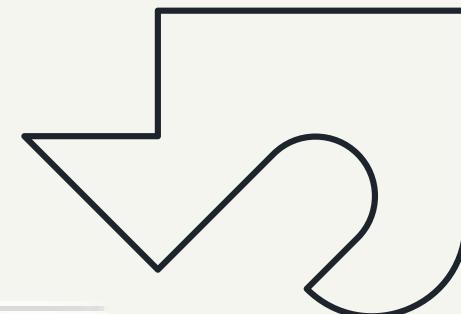
Quick Sort

Radix Sort

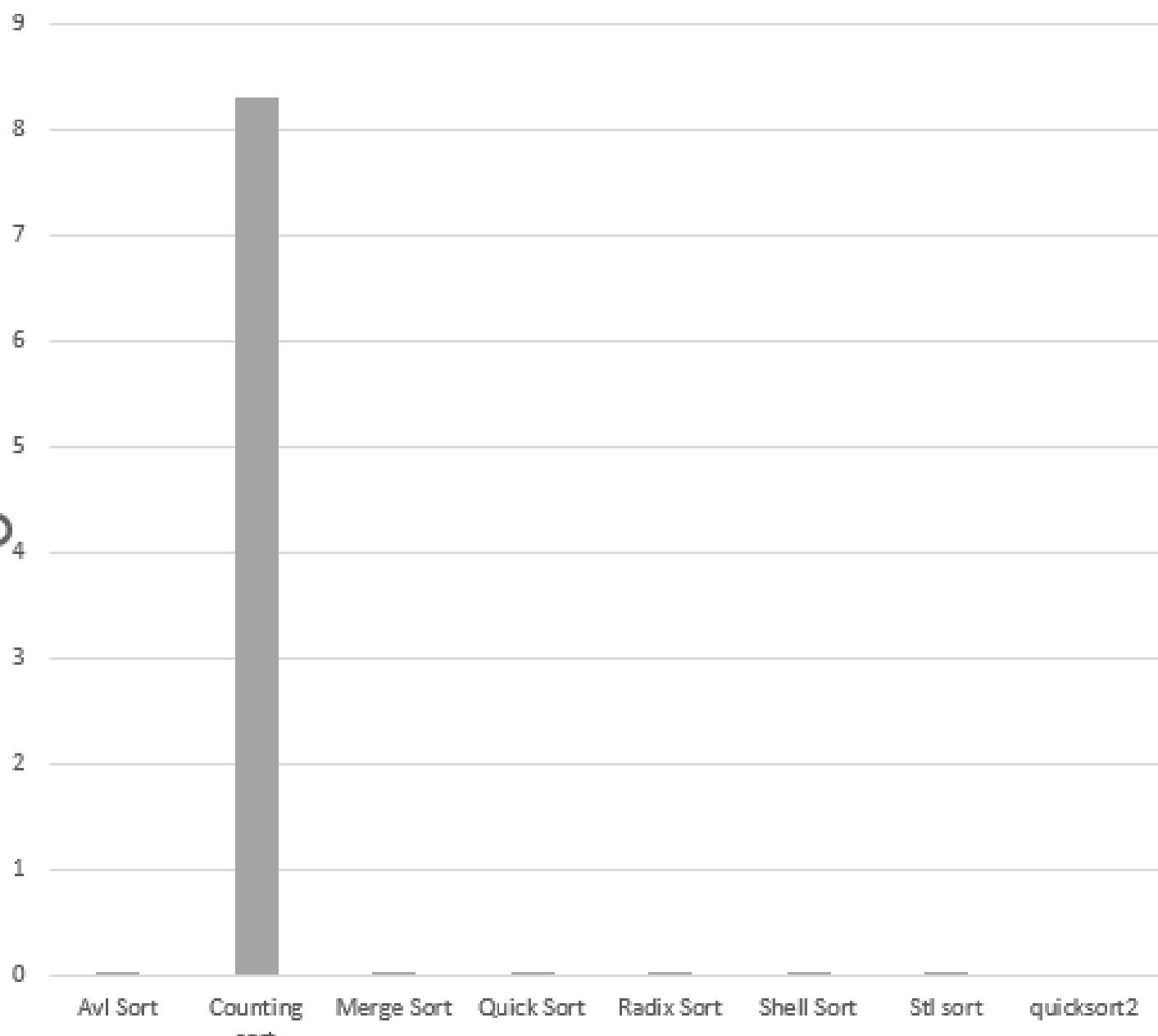
Shell Sort

Stl sort

quicksort2



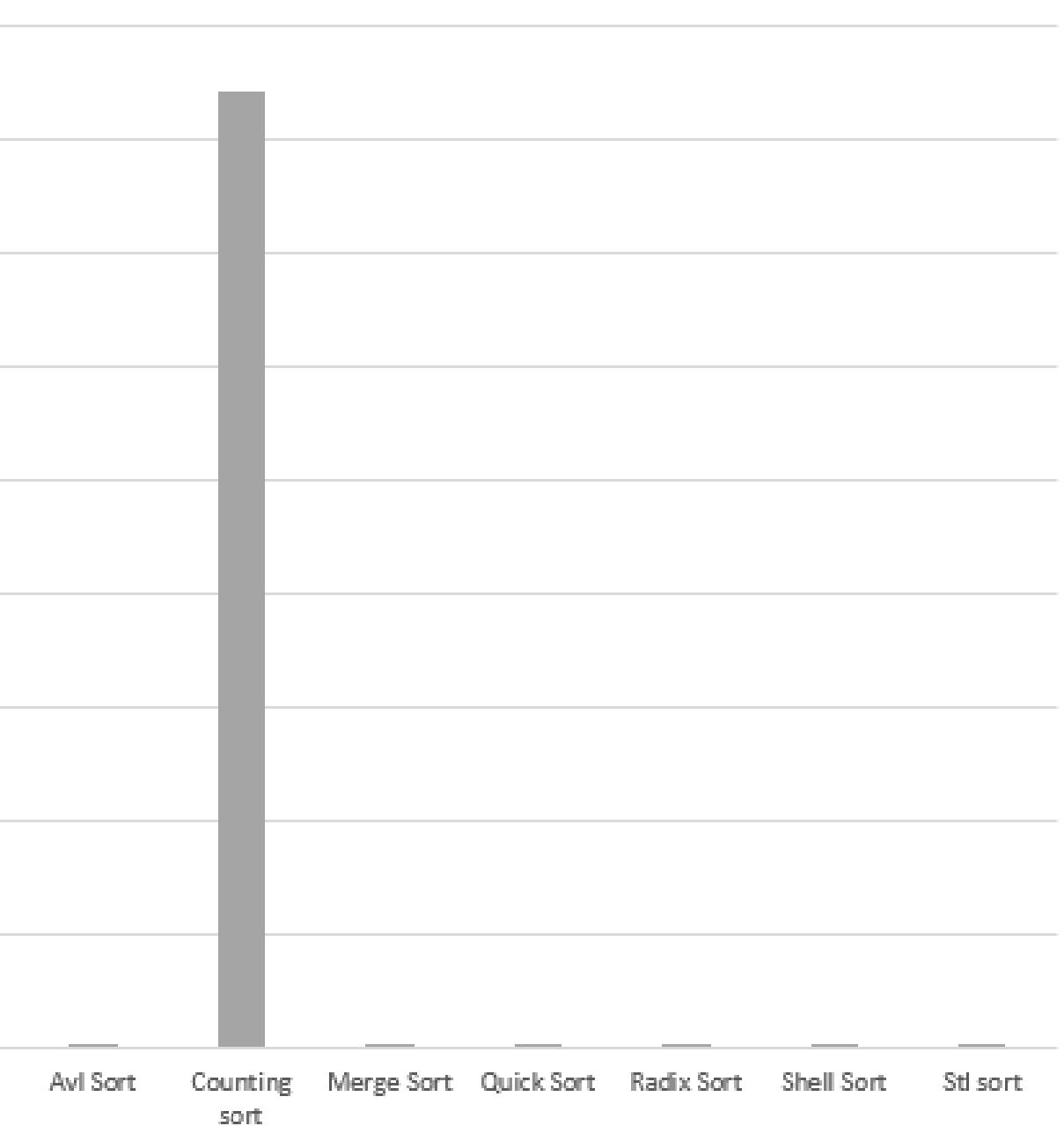
Worst case



■ Teste $n = 10^3$, Max = 10^9

Teste $n = 10^3$, Max = 10^9

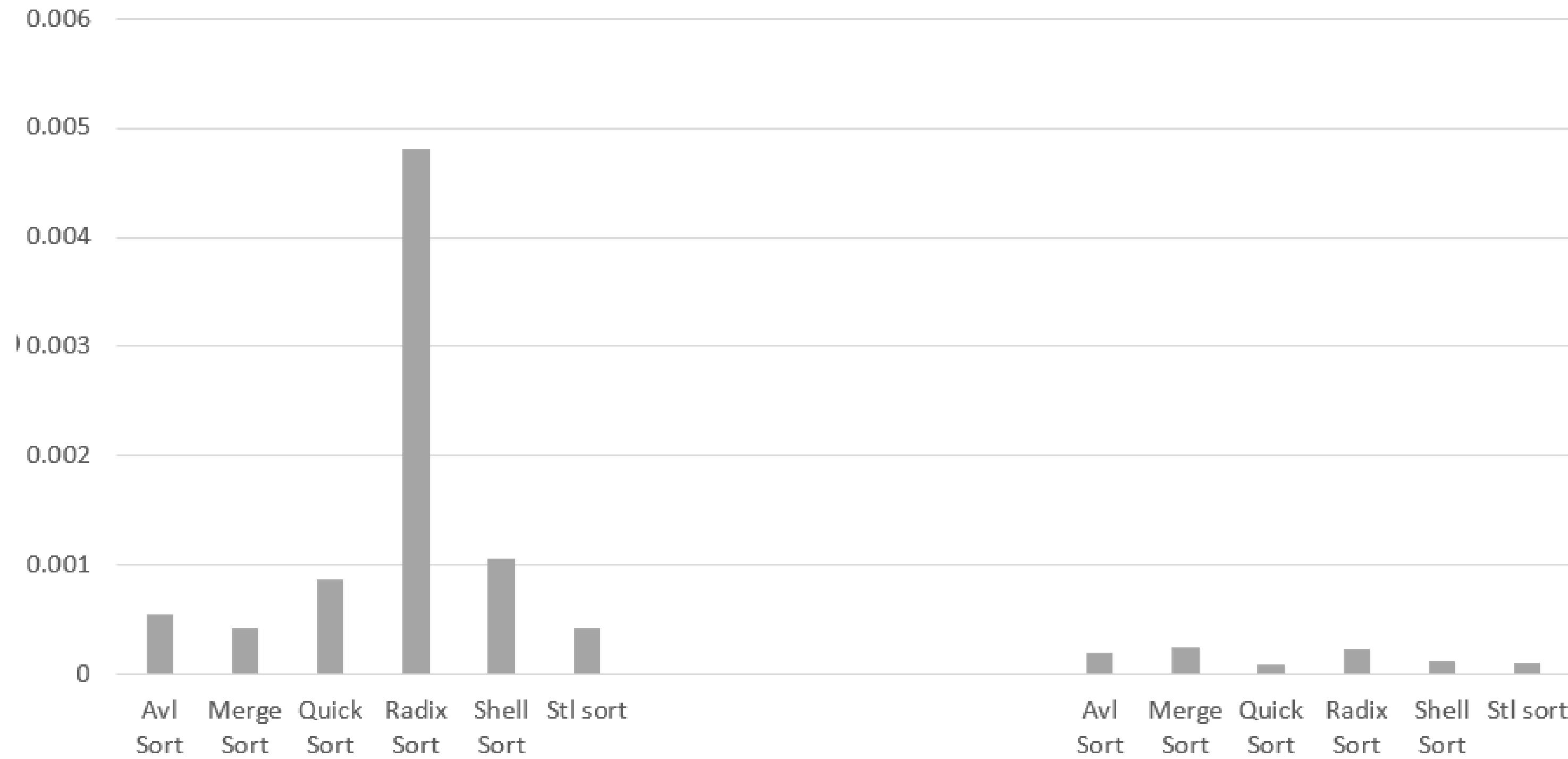
Best Case

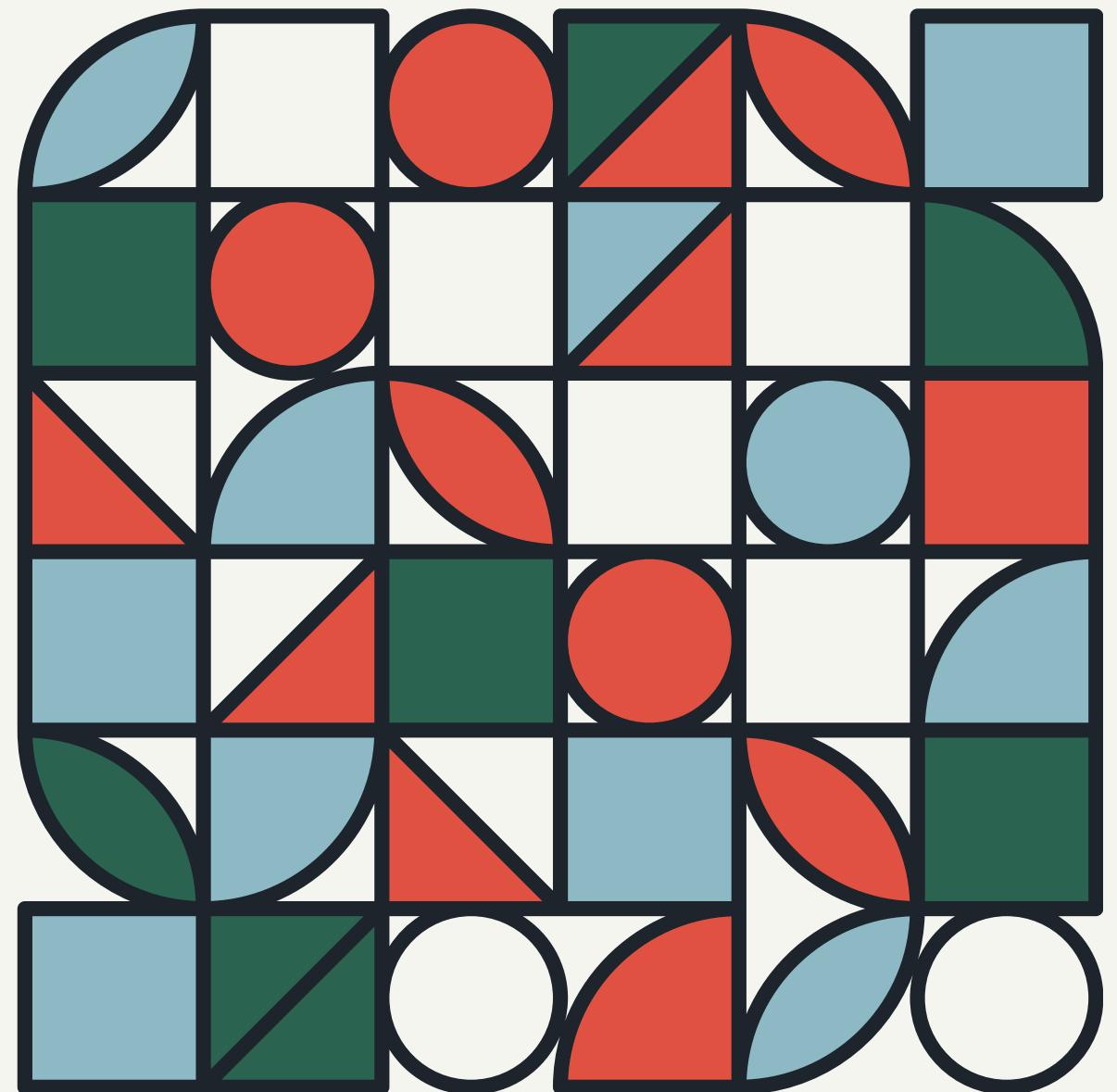


Worst case

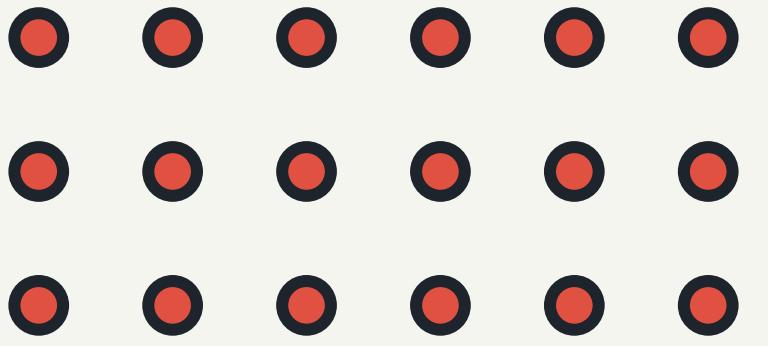
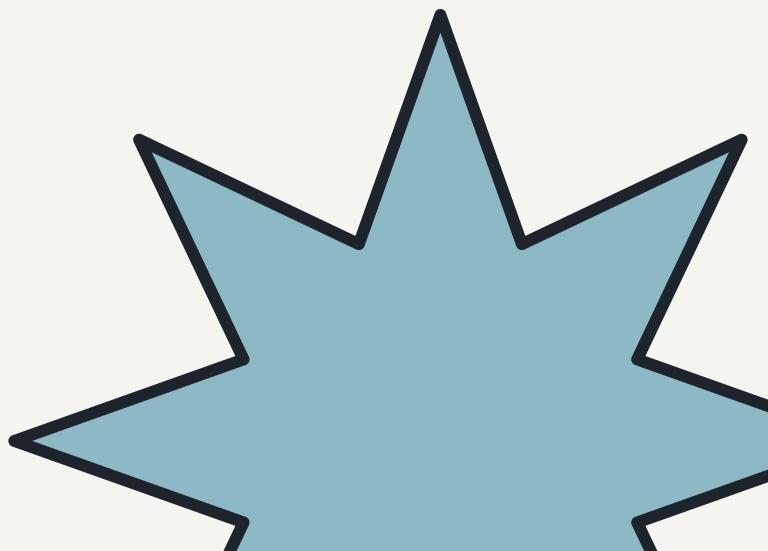
Best Case

Teste n = 10^3 , Max = 10^9

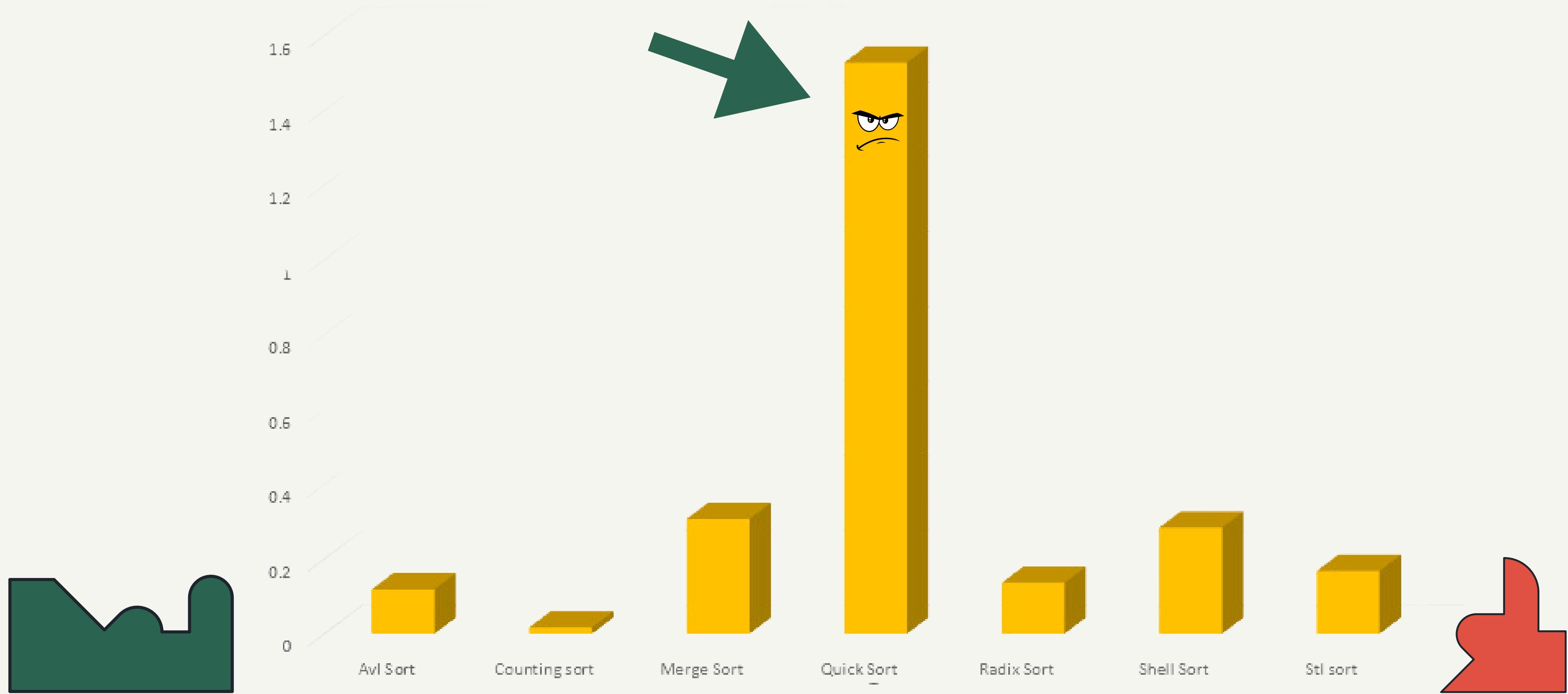
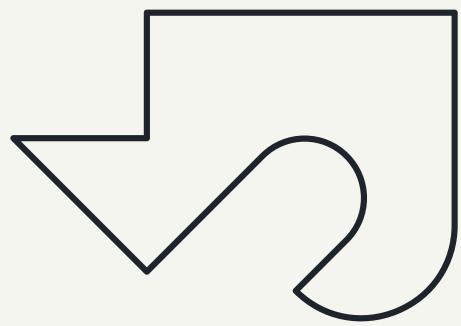
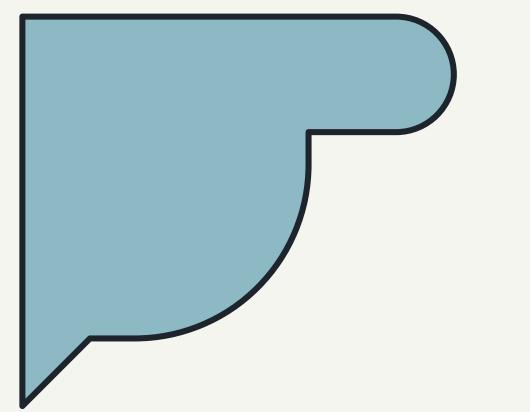




MAI
MULTE
NUMERE?

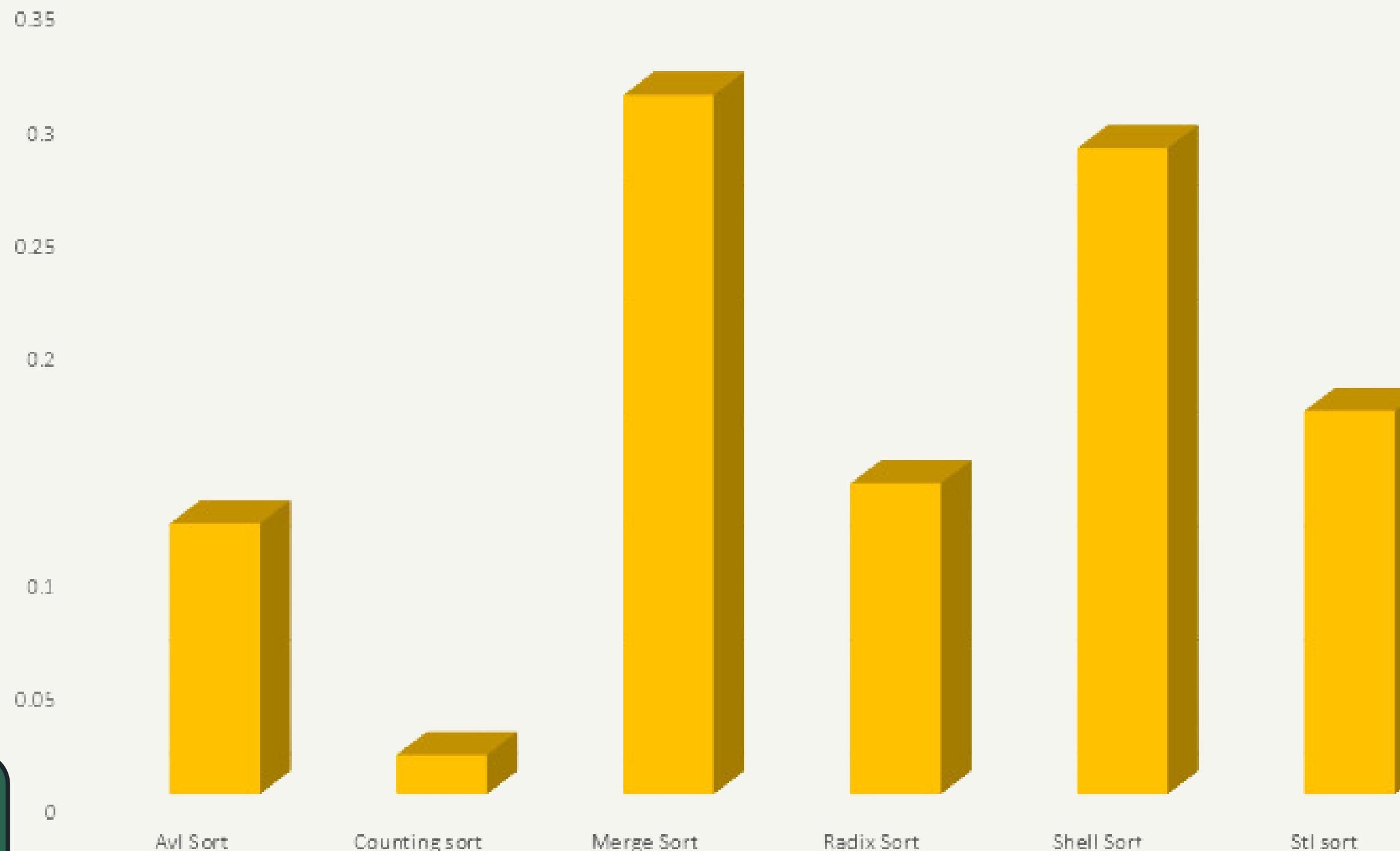


Teste $n = 10^6$, Max = 10^3

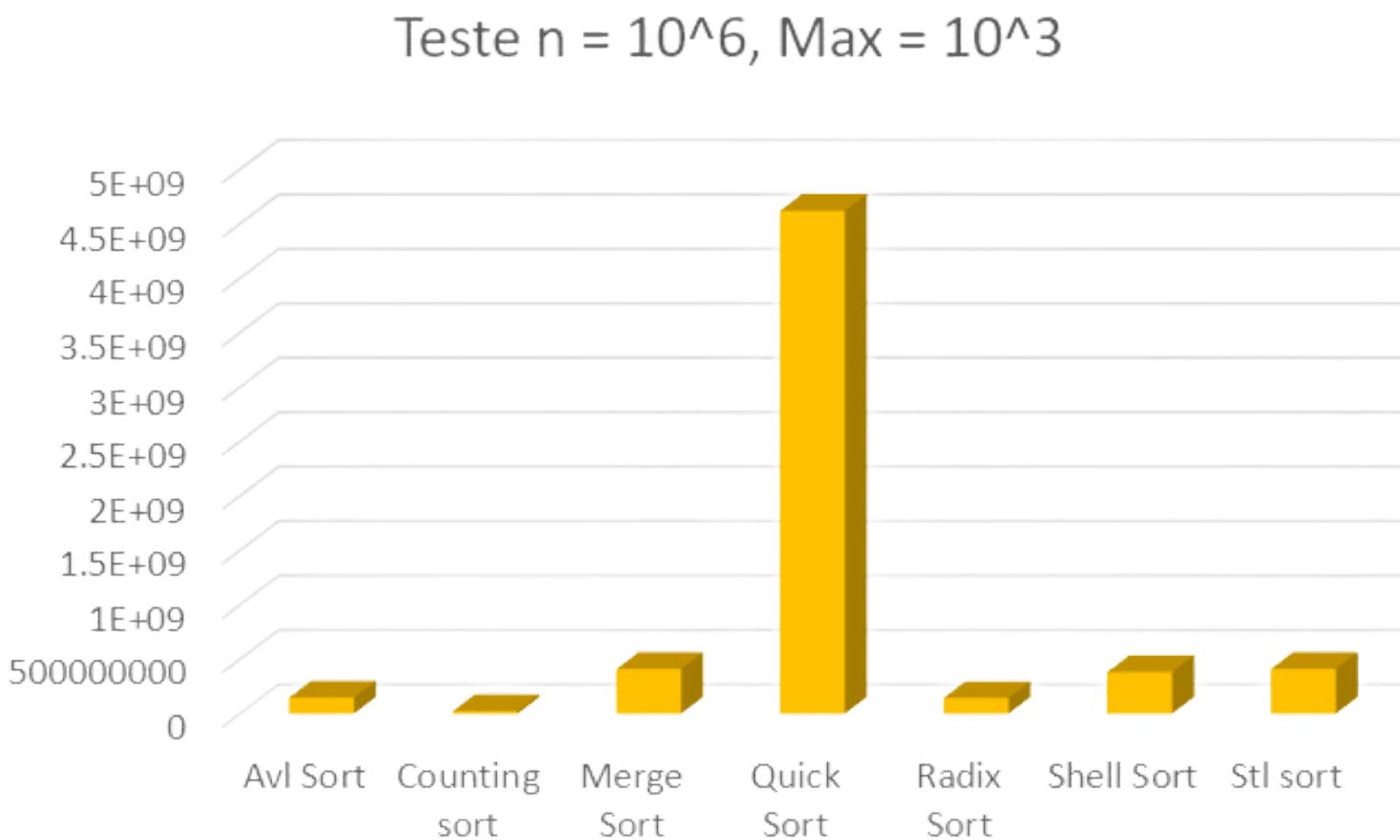


Teste $n = 10^6$, Max = 10^3

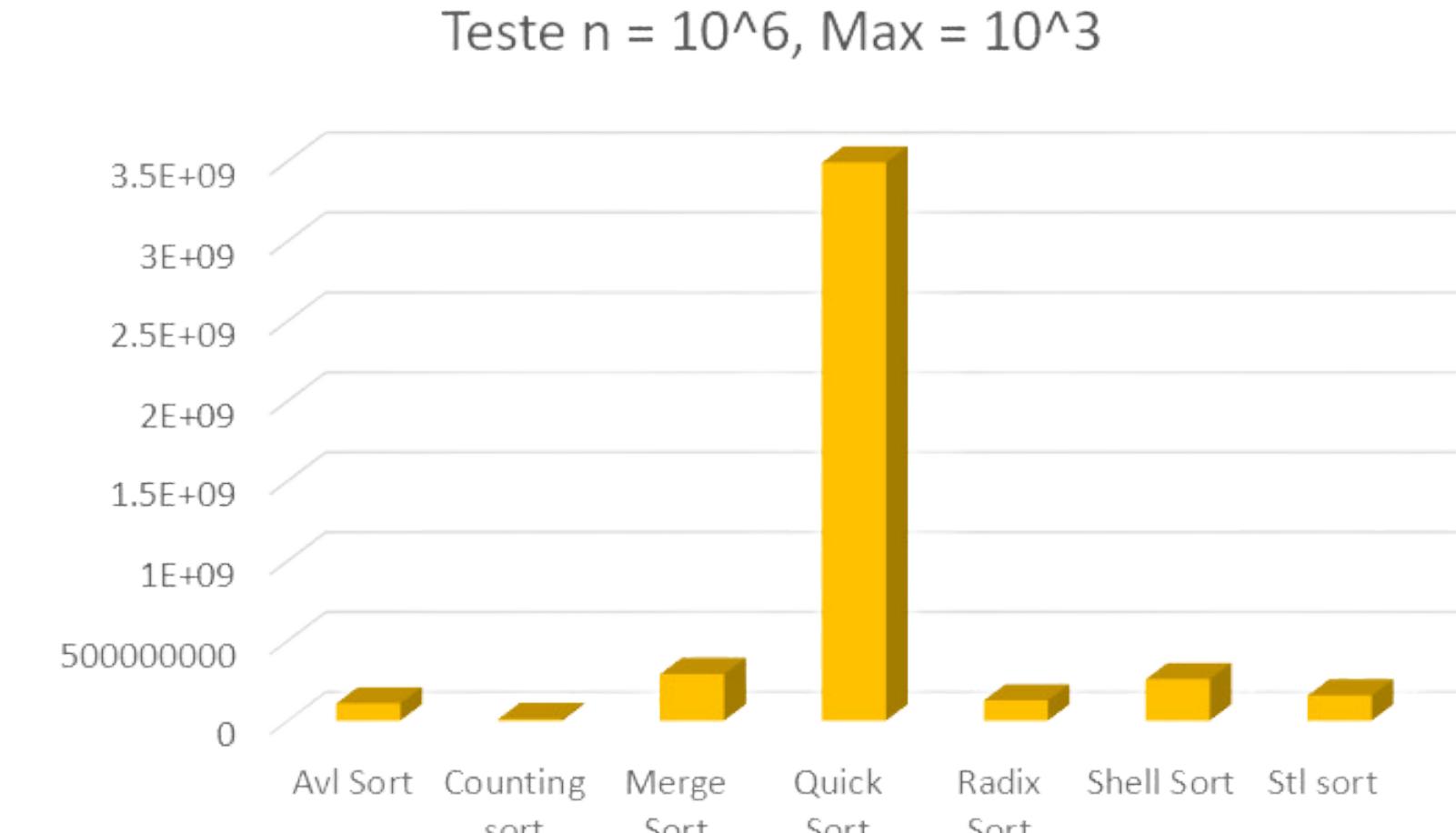
fără Quick Sort



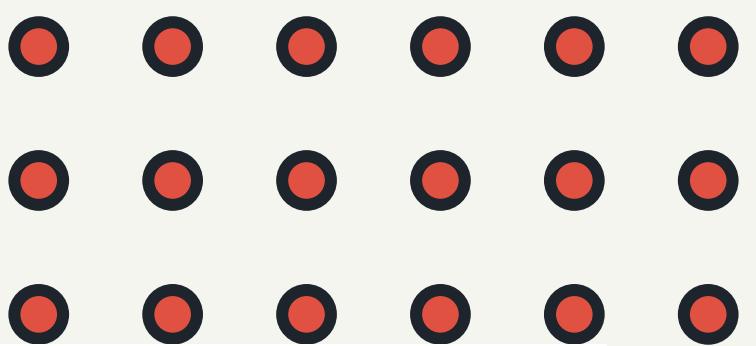
Worst case



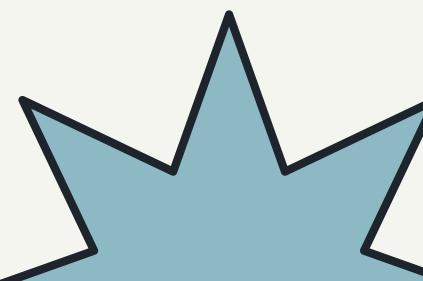
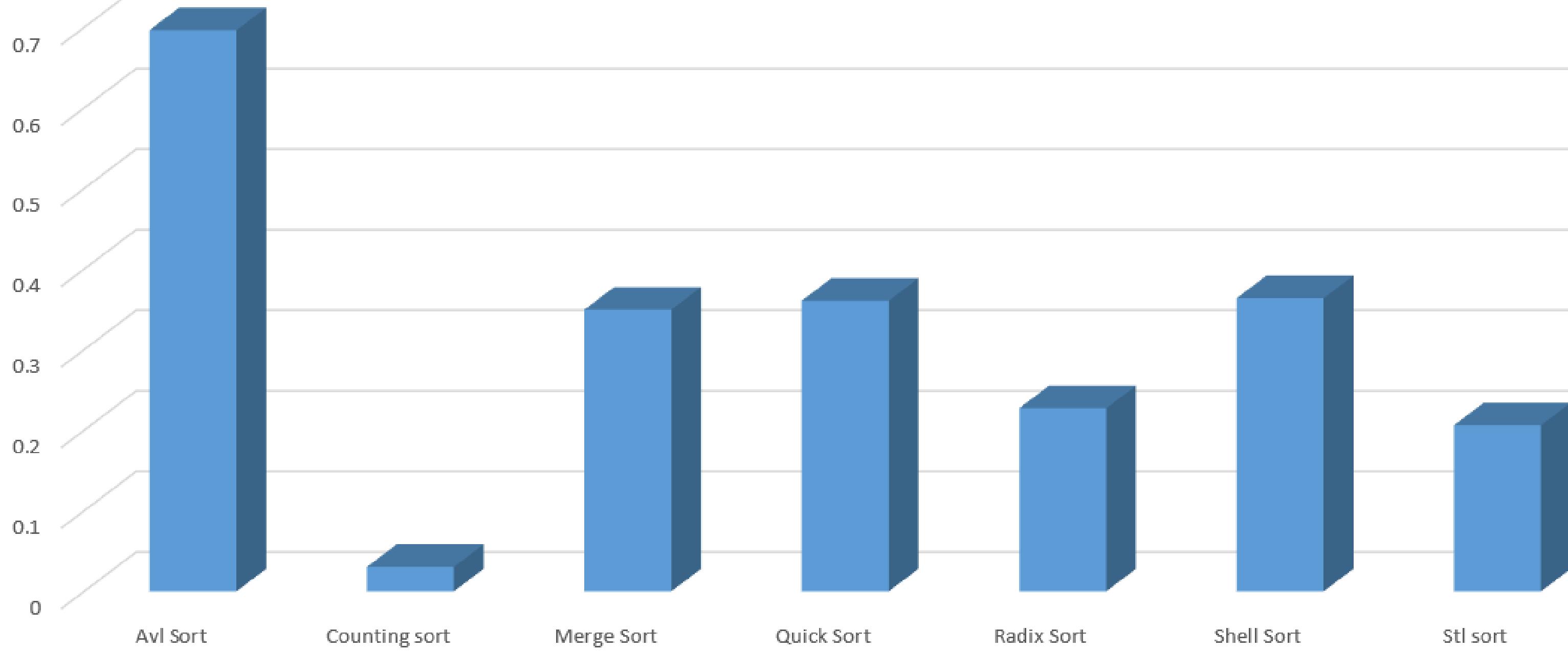
Best Case



e doar un pivot...



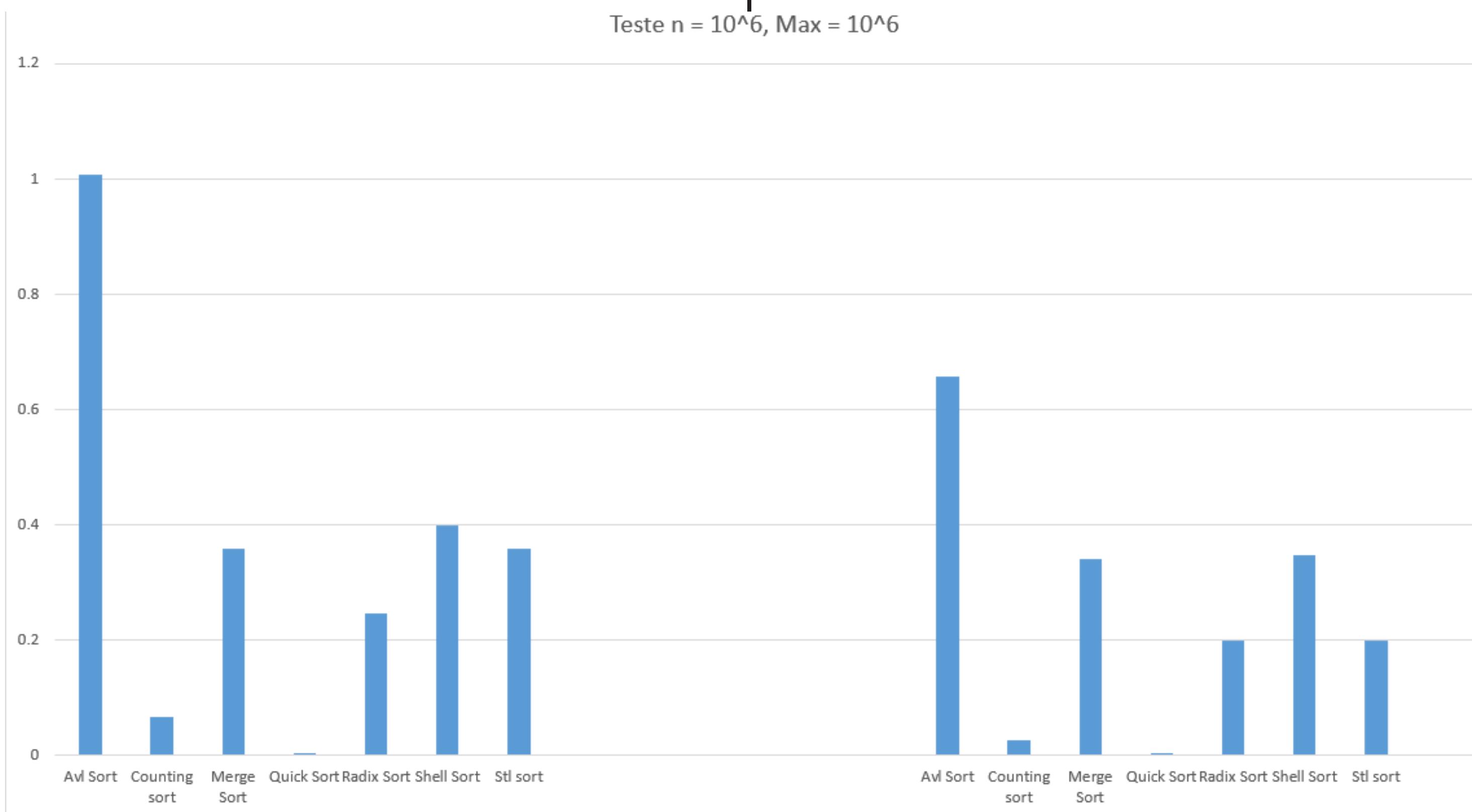
Teste $n = 10^6$, Max = 10^6

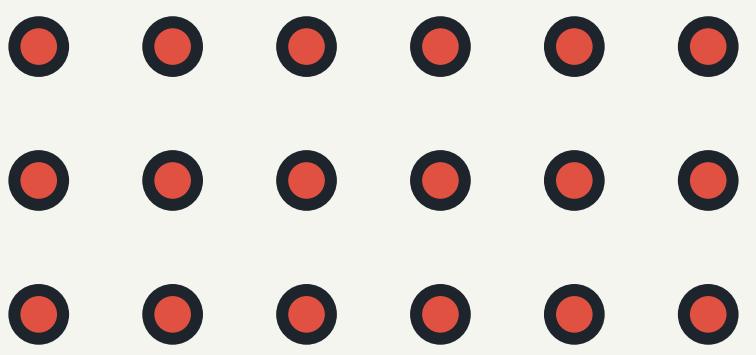


Worst case

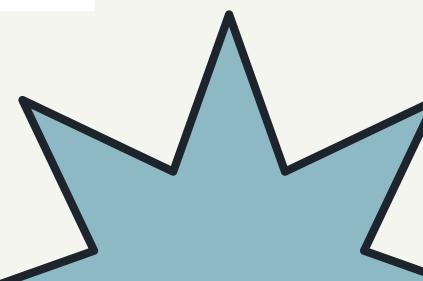
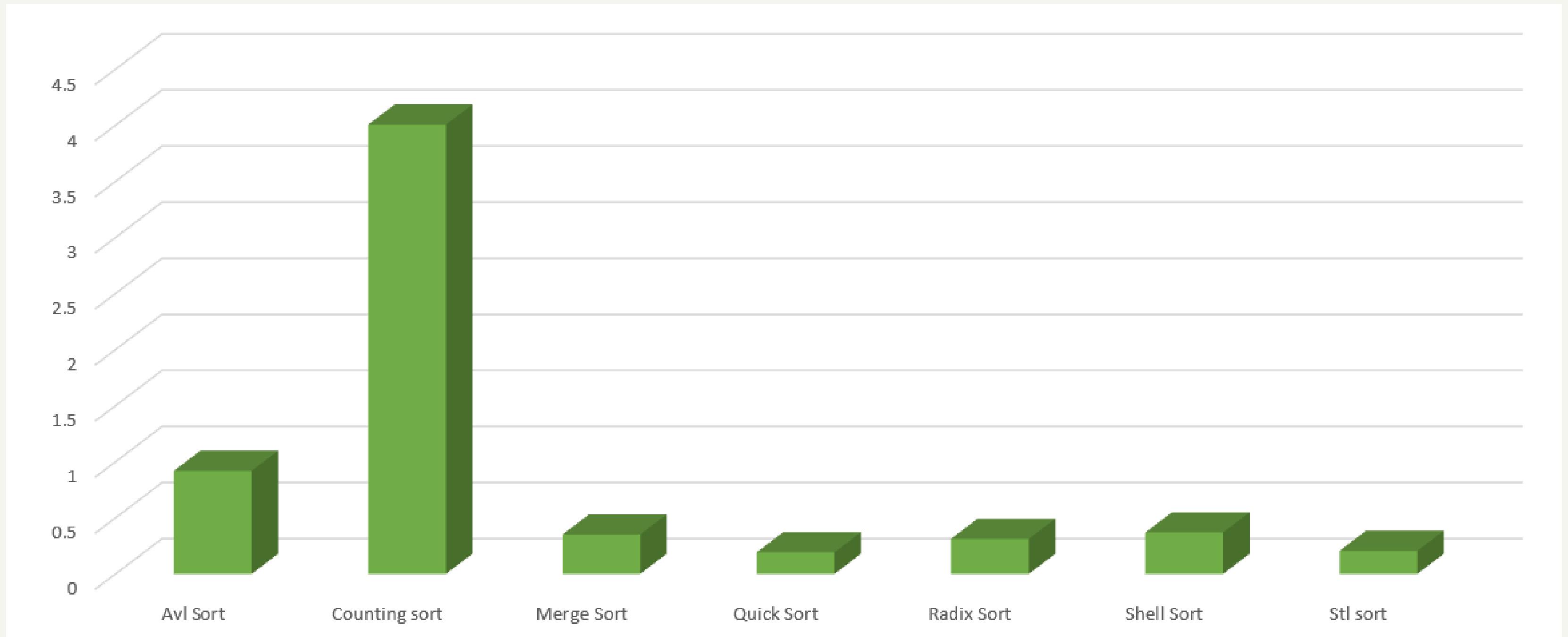
Best Case

Teste $n = 10^6$, Max = 10^6



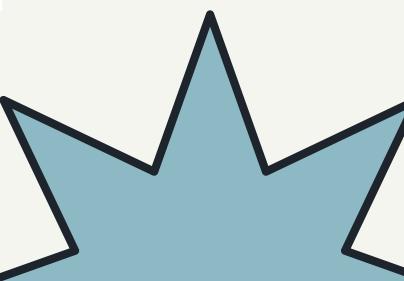
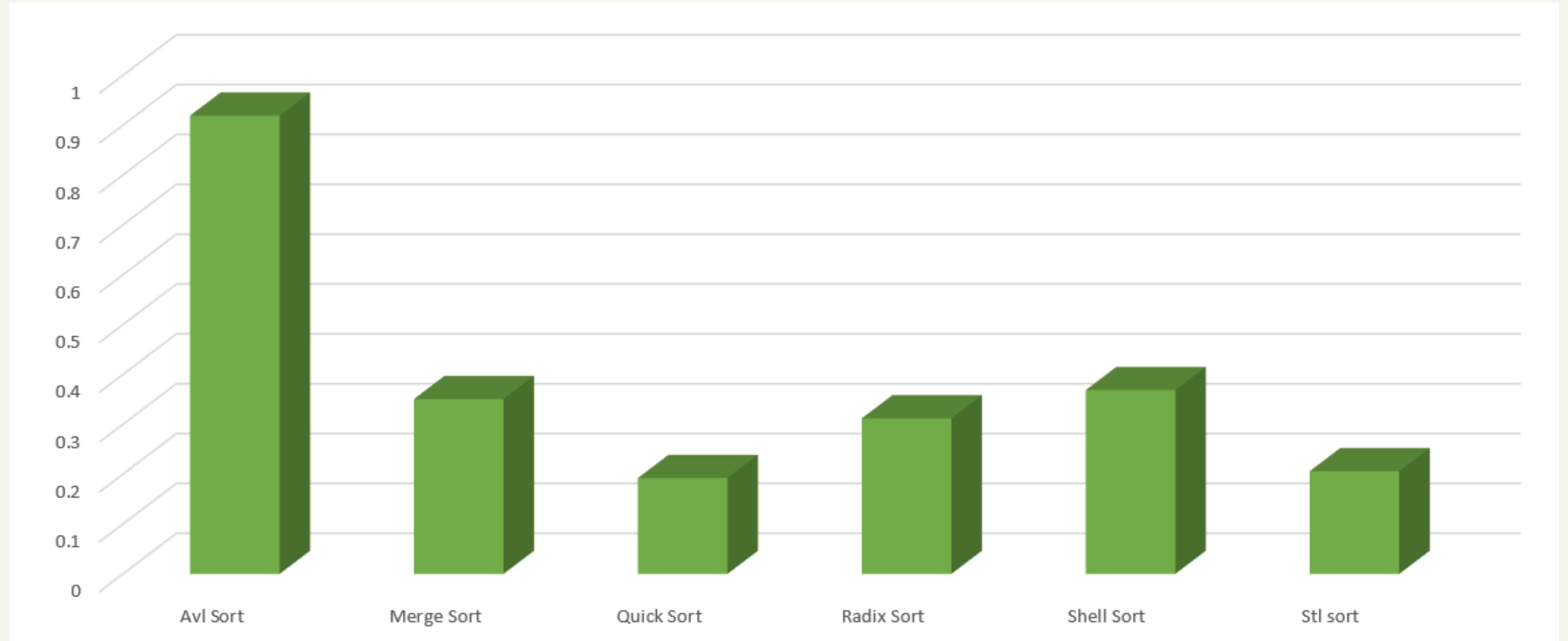
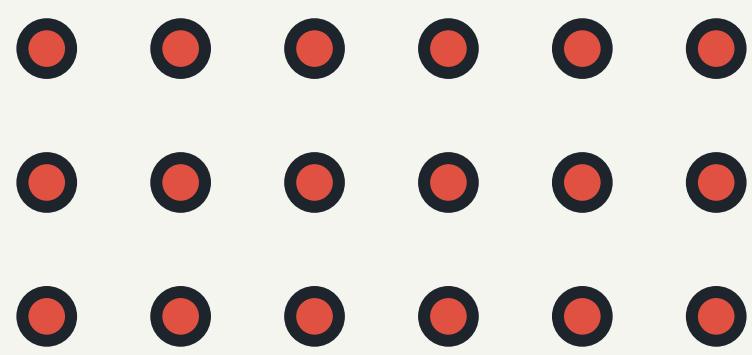


Teste $n = 10^6$, Max = 10^9



fără counting Sort

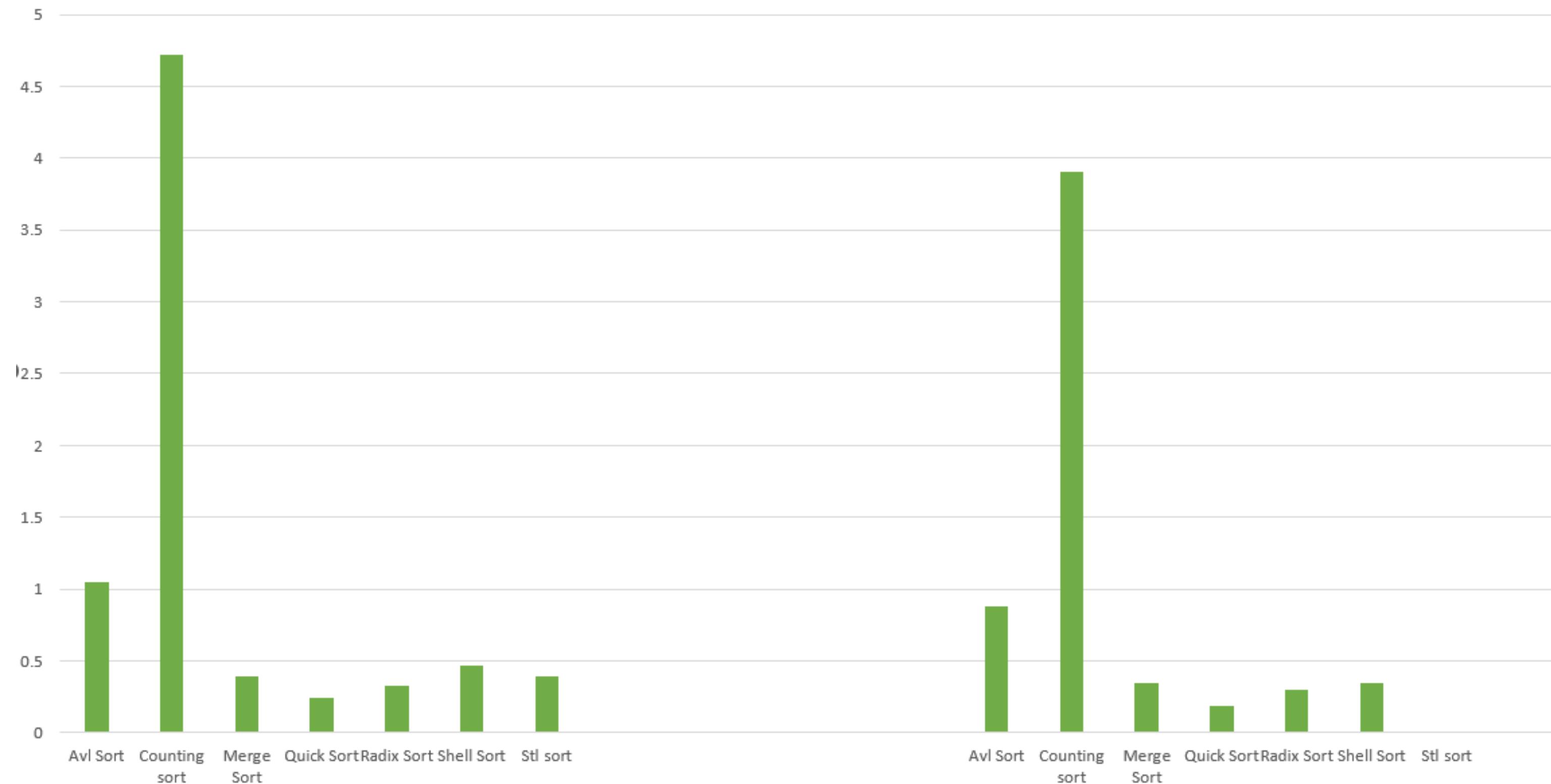
Teste $n = 10^6$, Max = 10^9



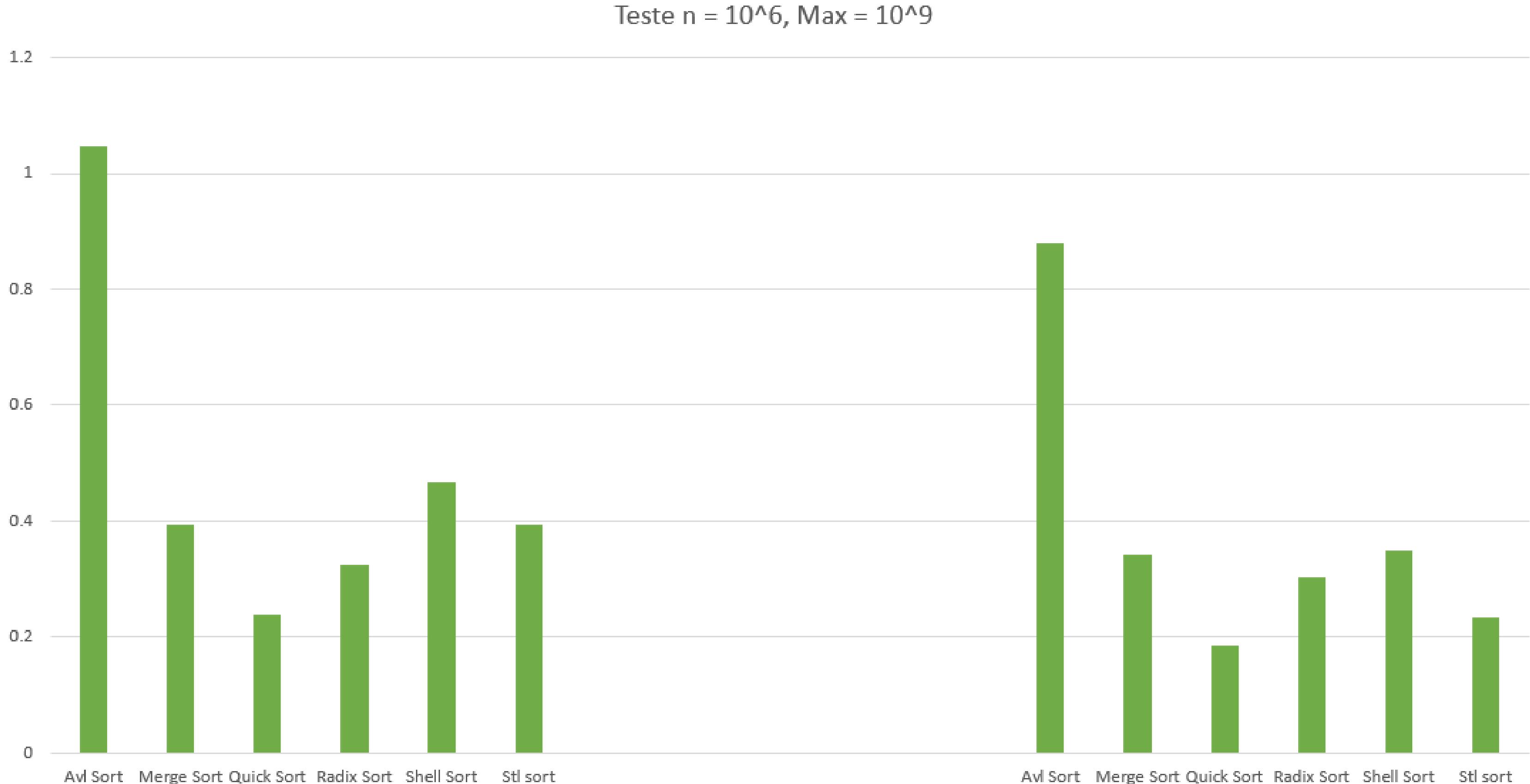
Worst case

Best Case

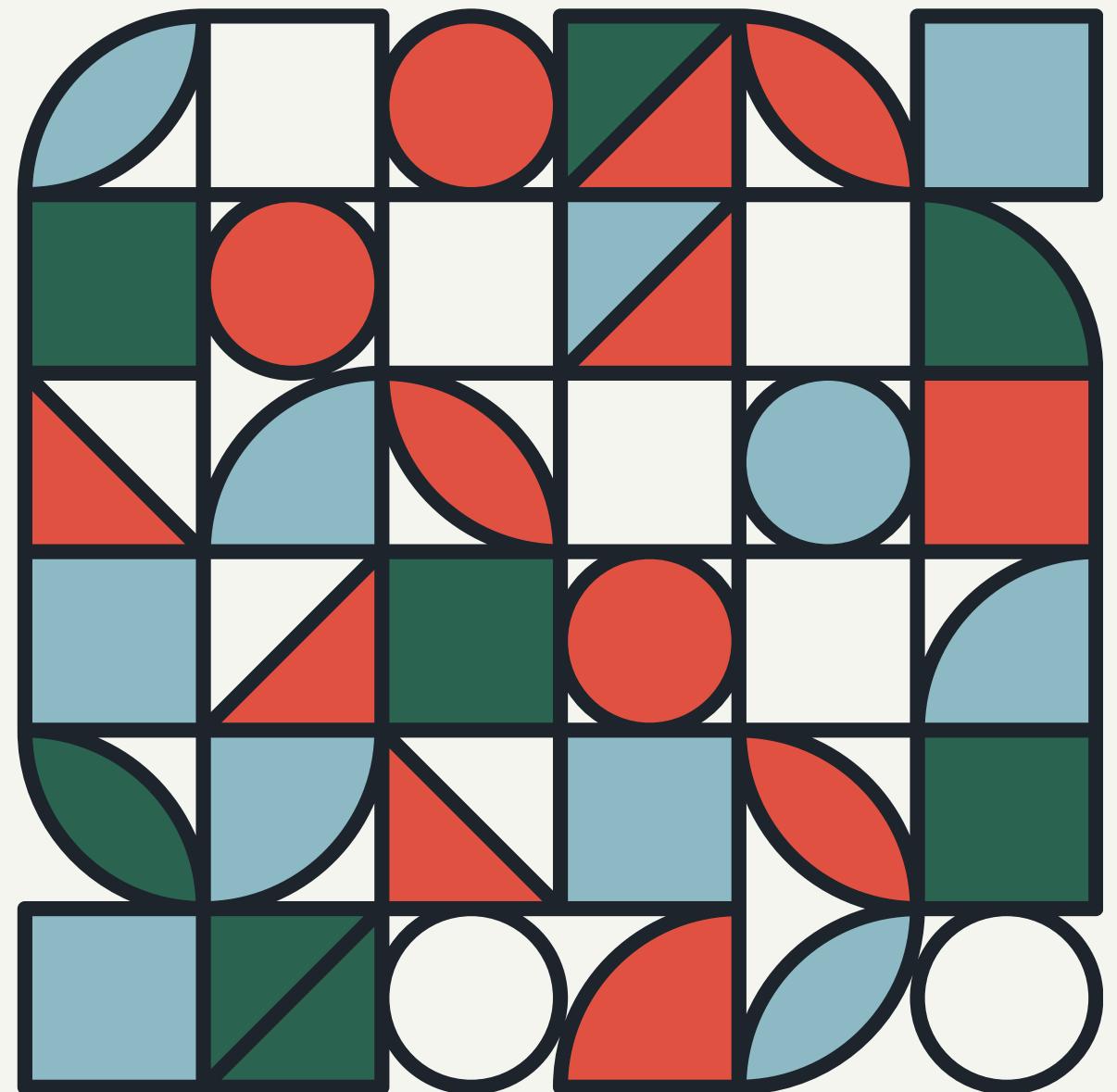
Teste n = 10^6 , Max = 10^9



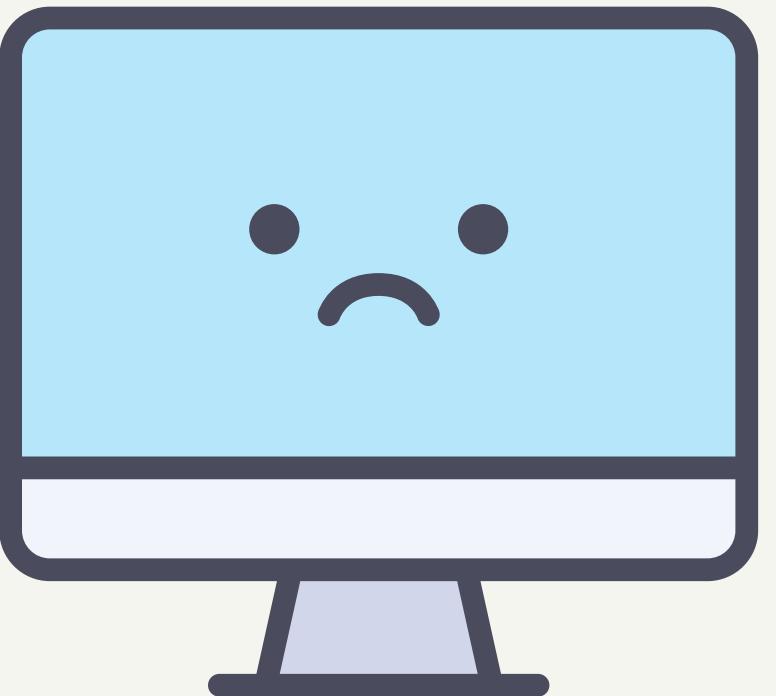
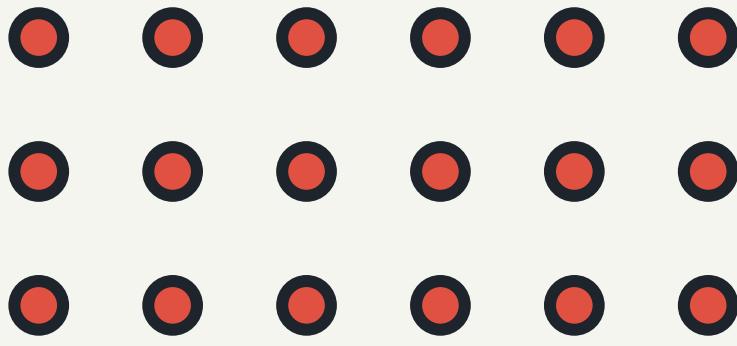
Worst case

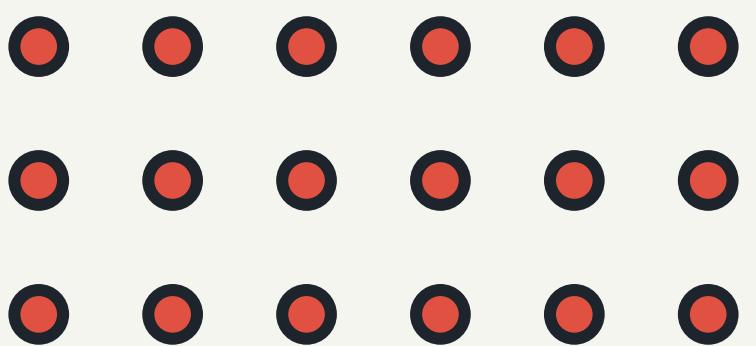


Best Case

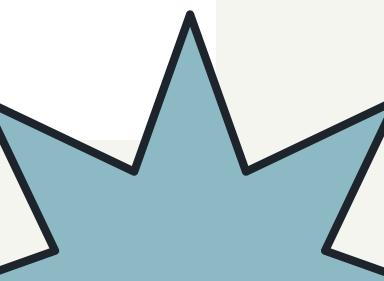
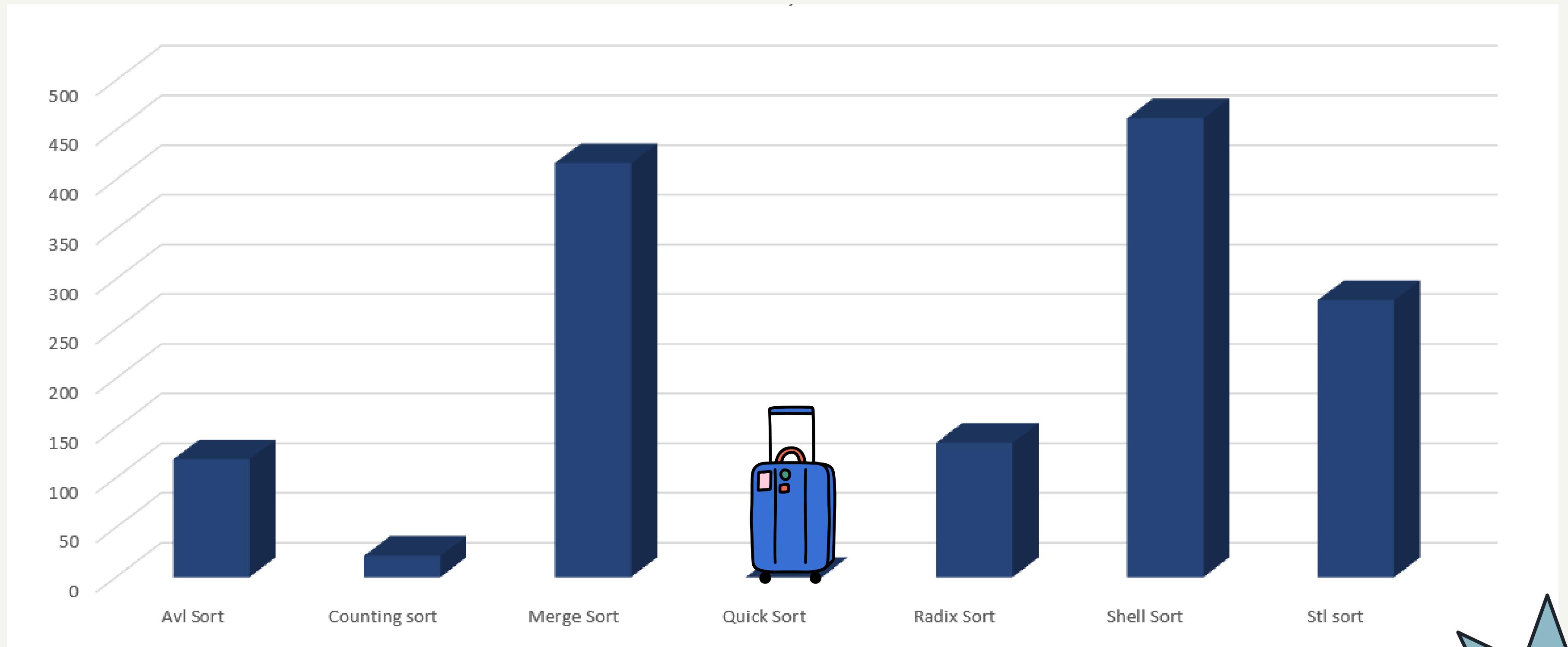


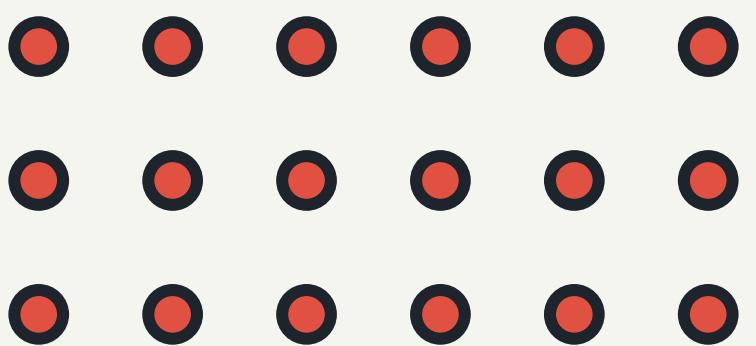
SI MAI
MULTE
NUMERE...



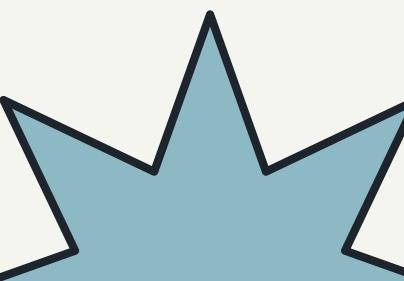
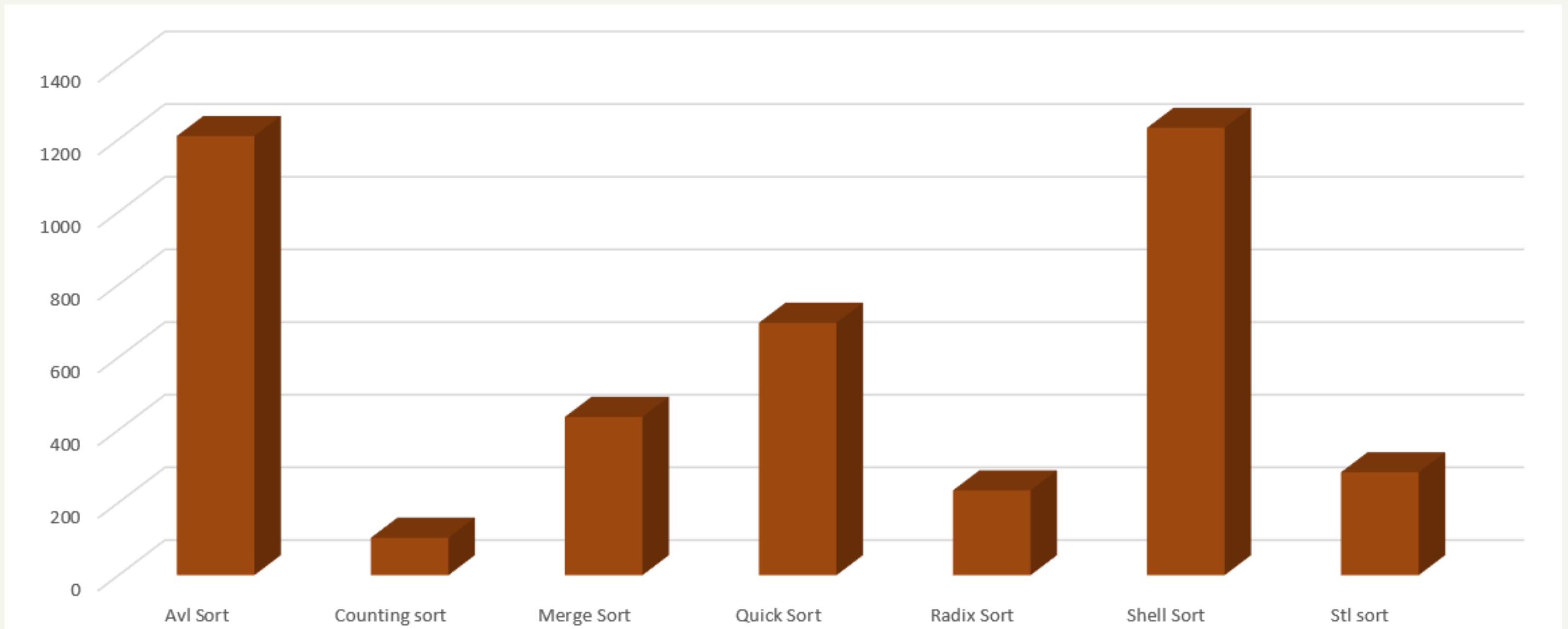


Teste $n = 10^9$, Max = 10^3

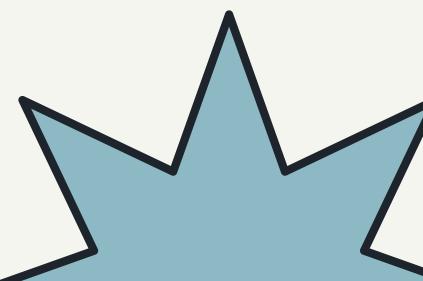
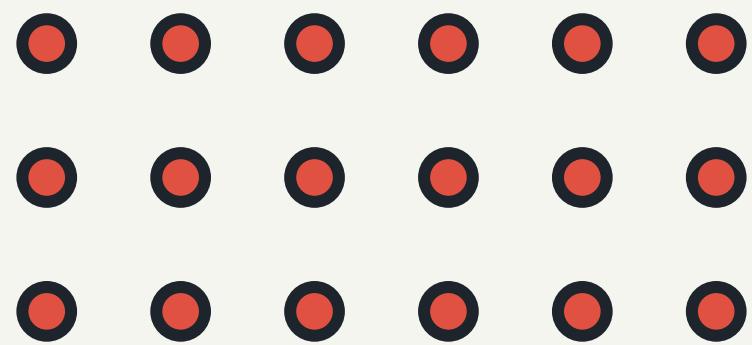
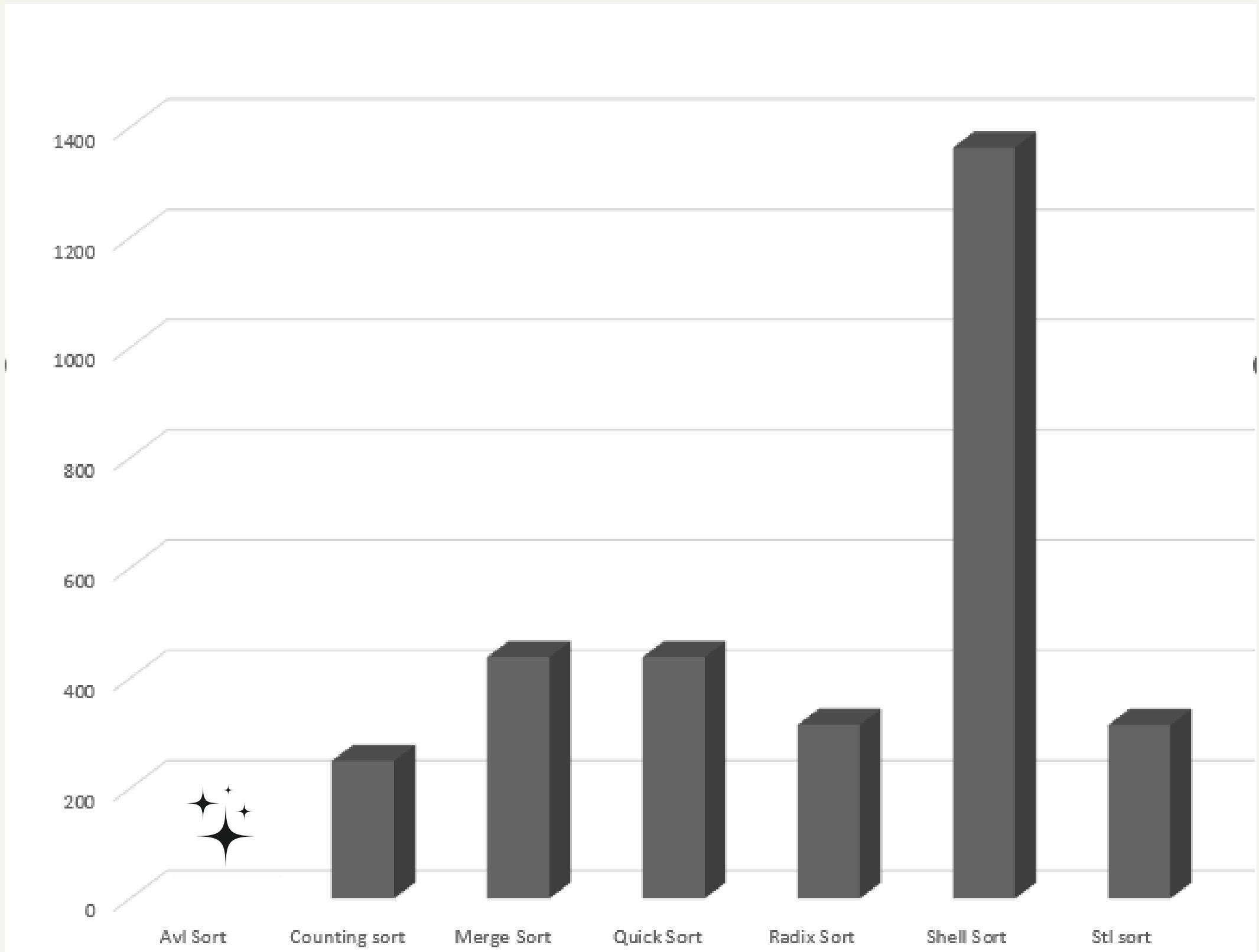




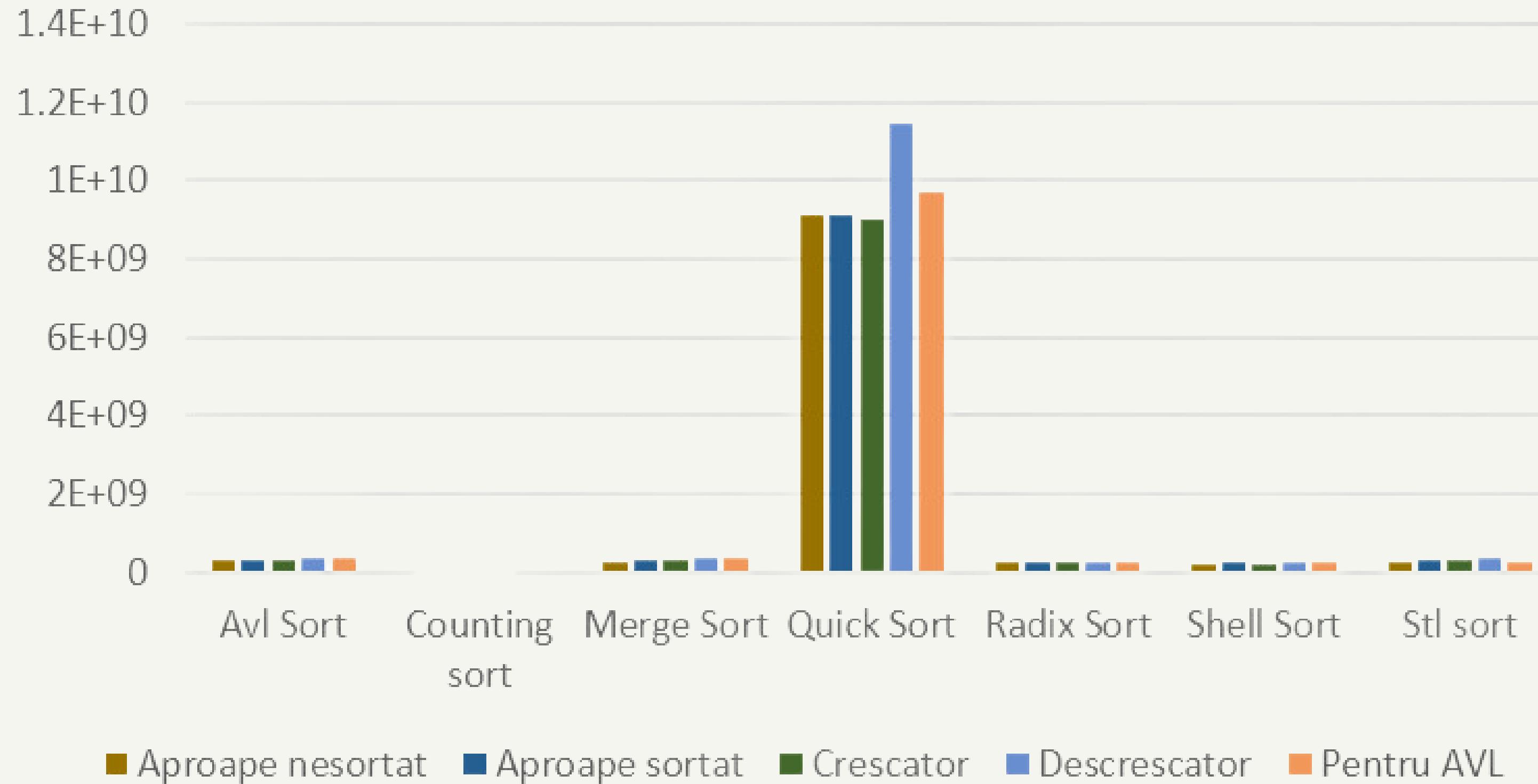
Teste $n = 10^9$, Max = 10^6



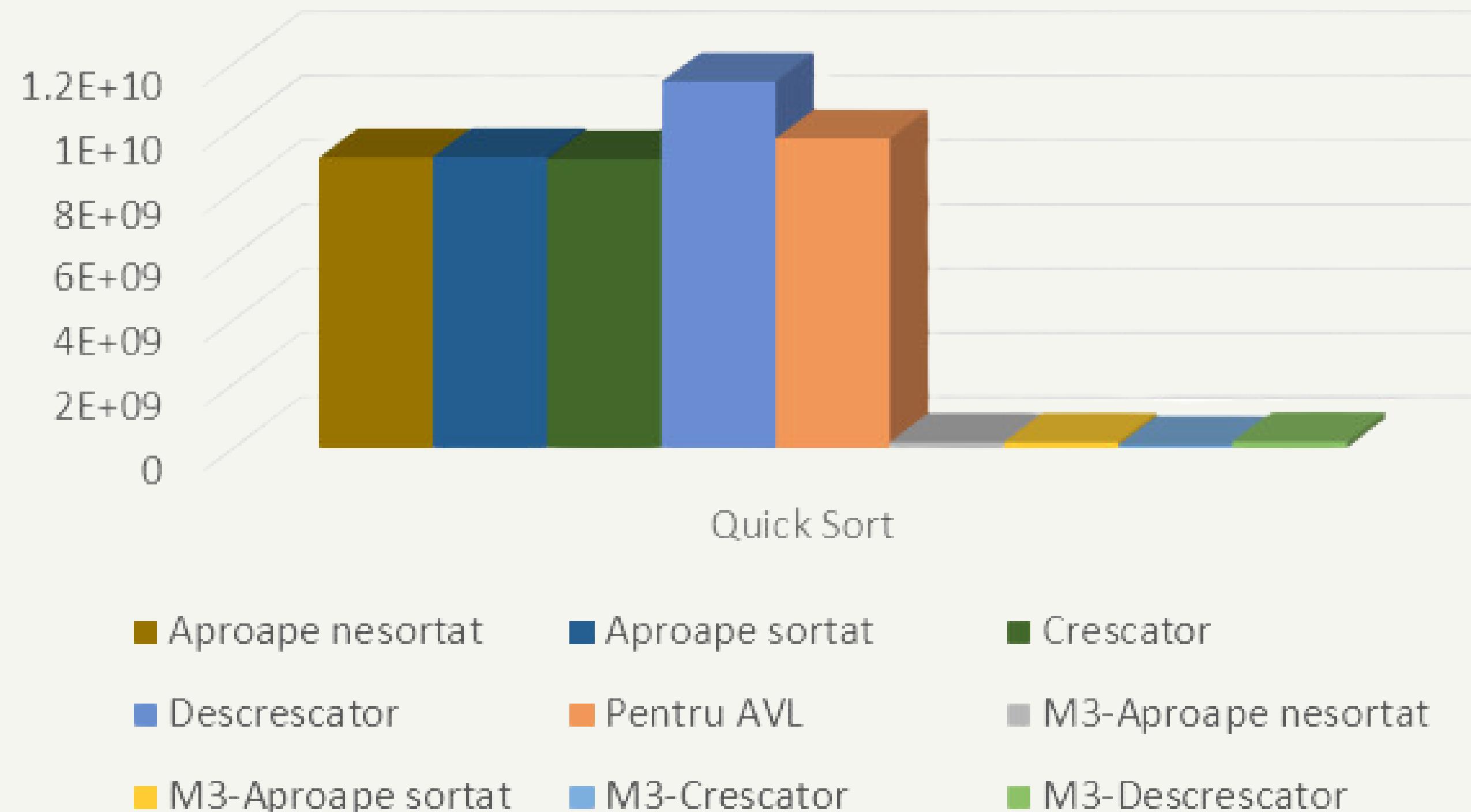
Teste n = 10^9, Max = 10^9



CAZURI SPECIALE



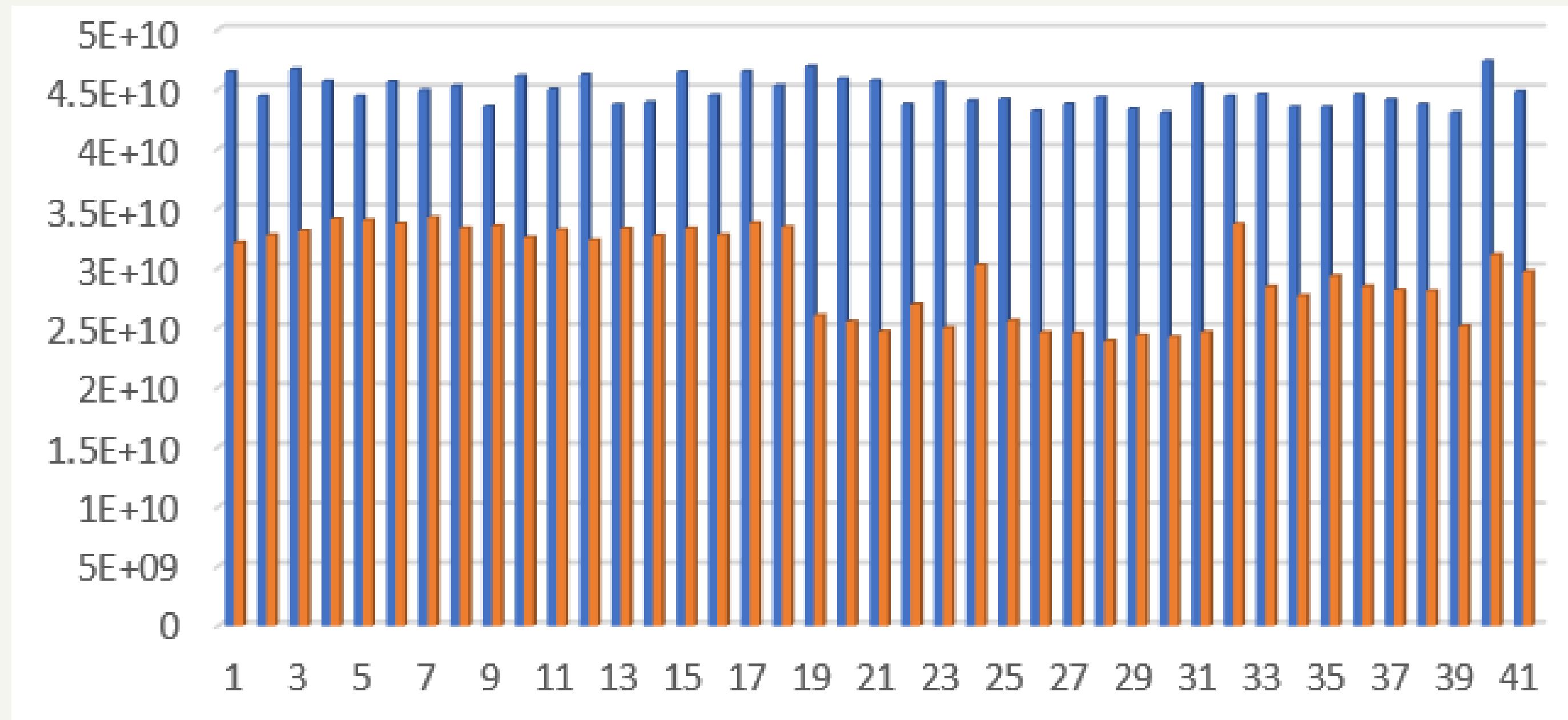
QUICK SORT PIVOT CAPĂT VS M3



TESTE N=10⁶, MAX=10⁶

Quick Sort

$n=10^8$ max= 10^9

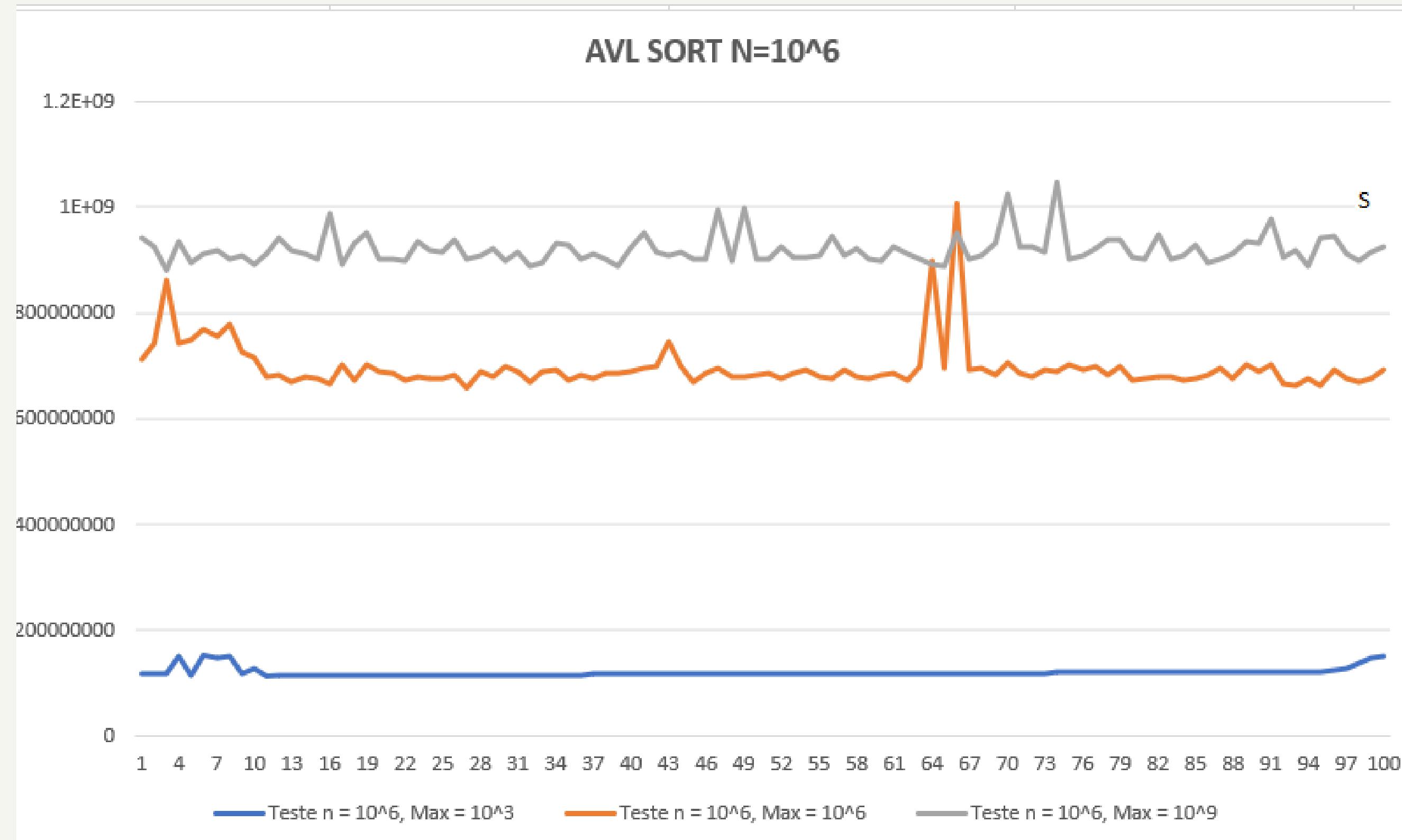


Merge Sort

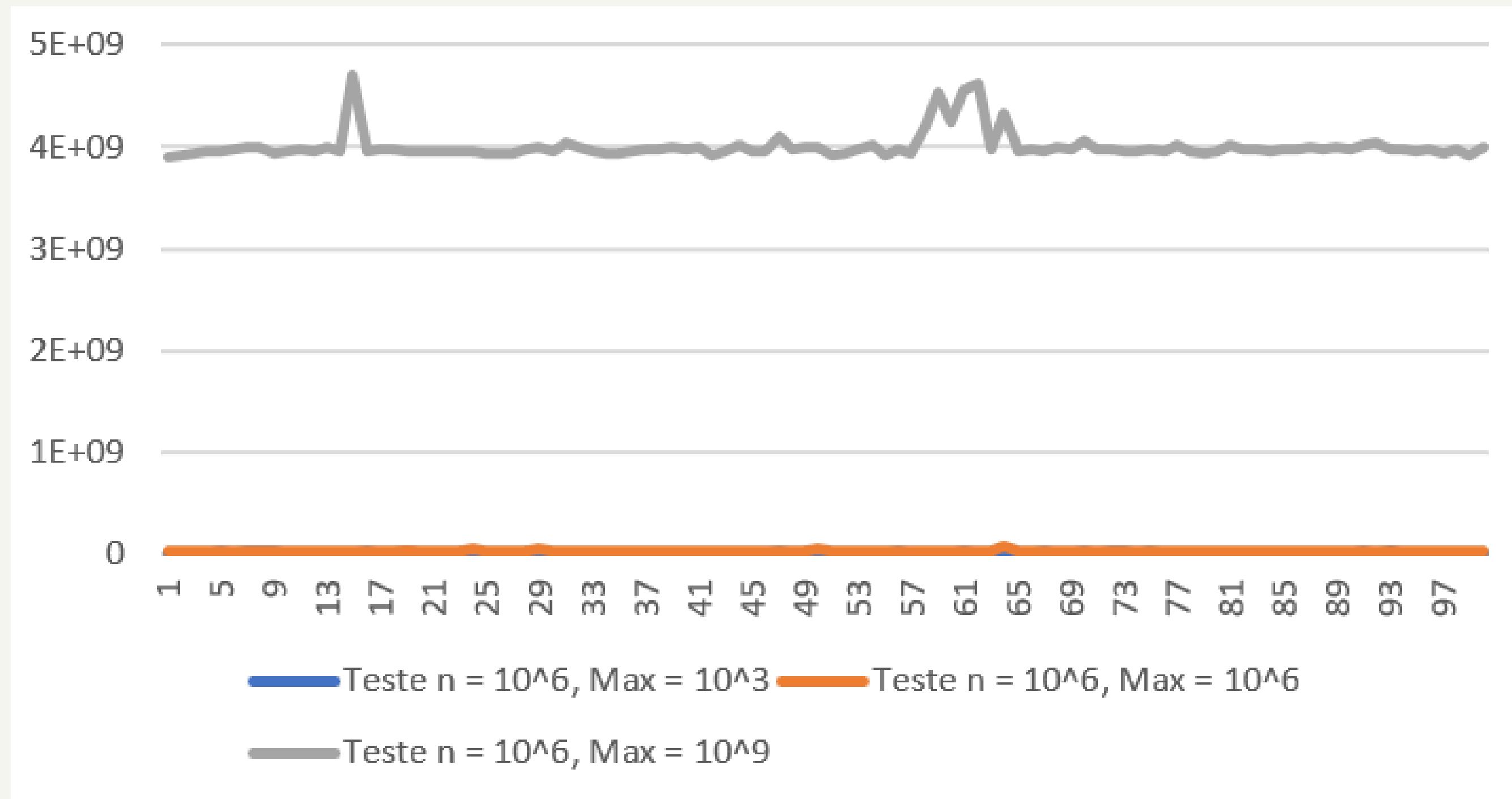


Quick Sort

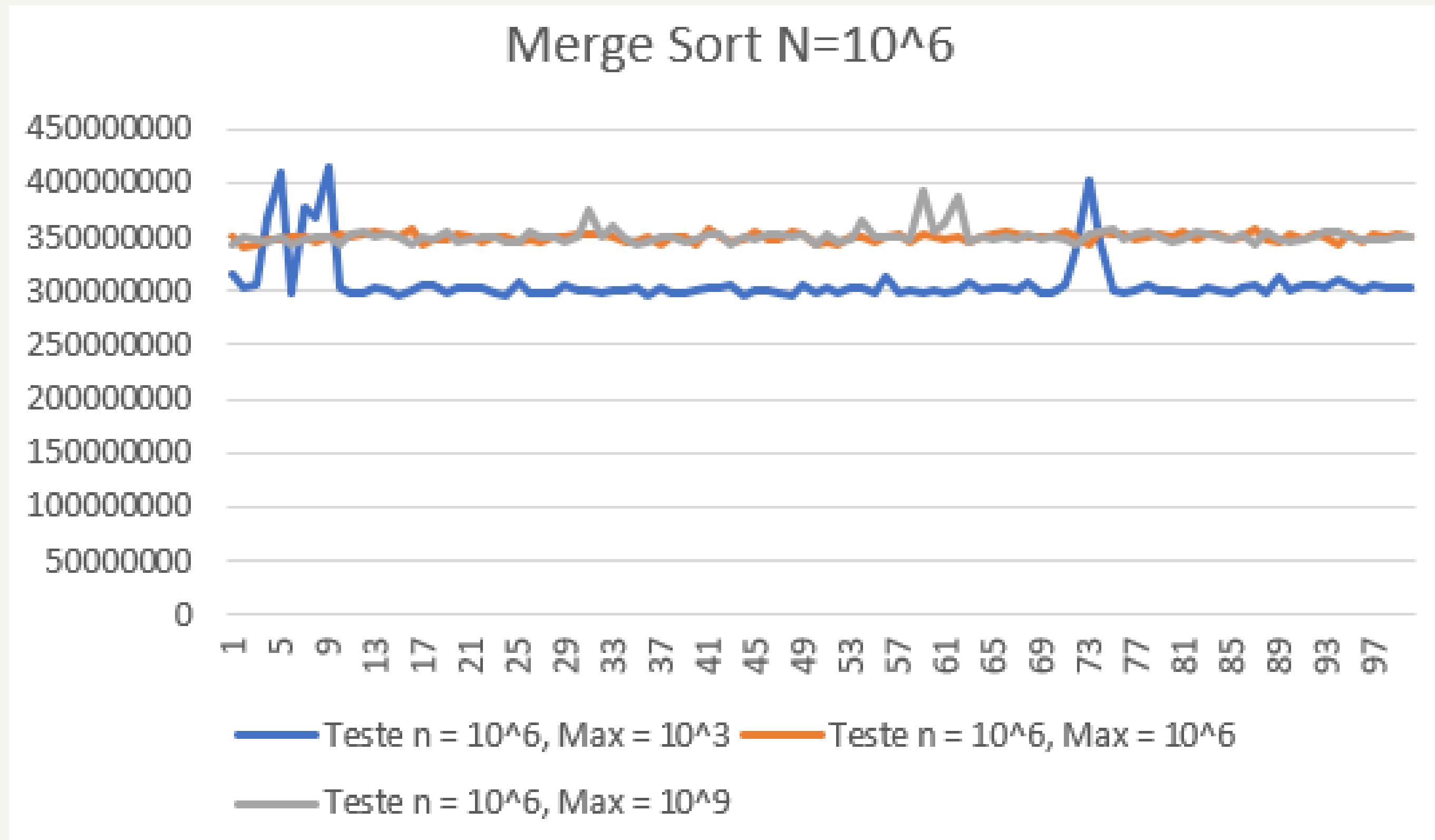
AVL SORT



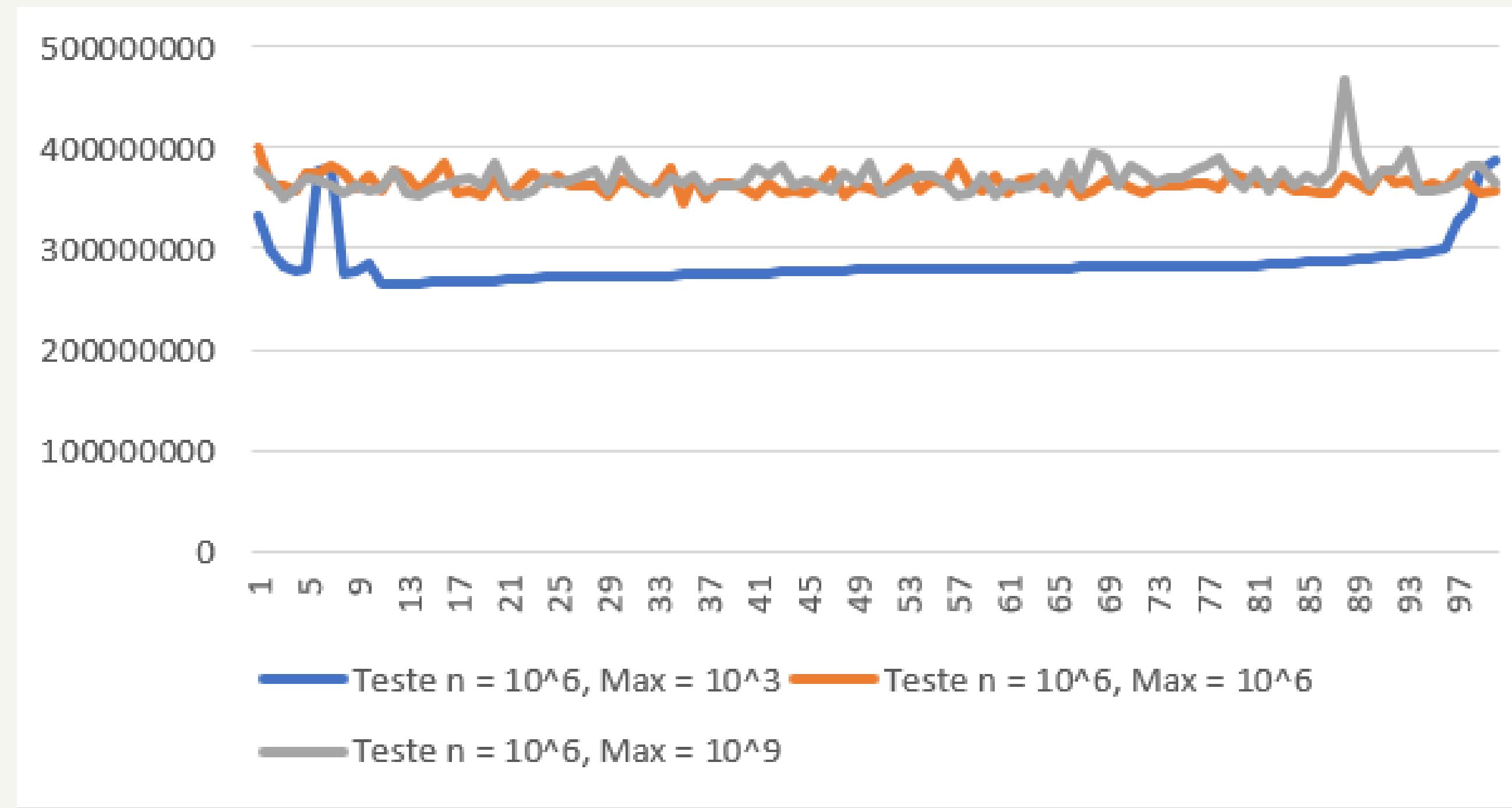
COUNTING SORT



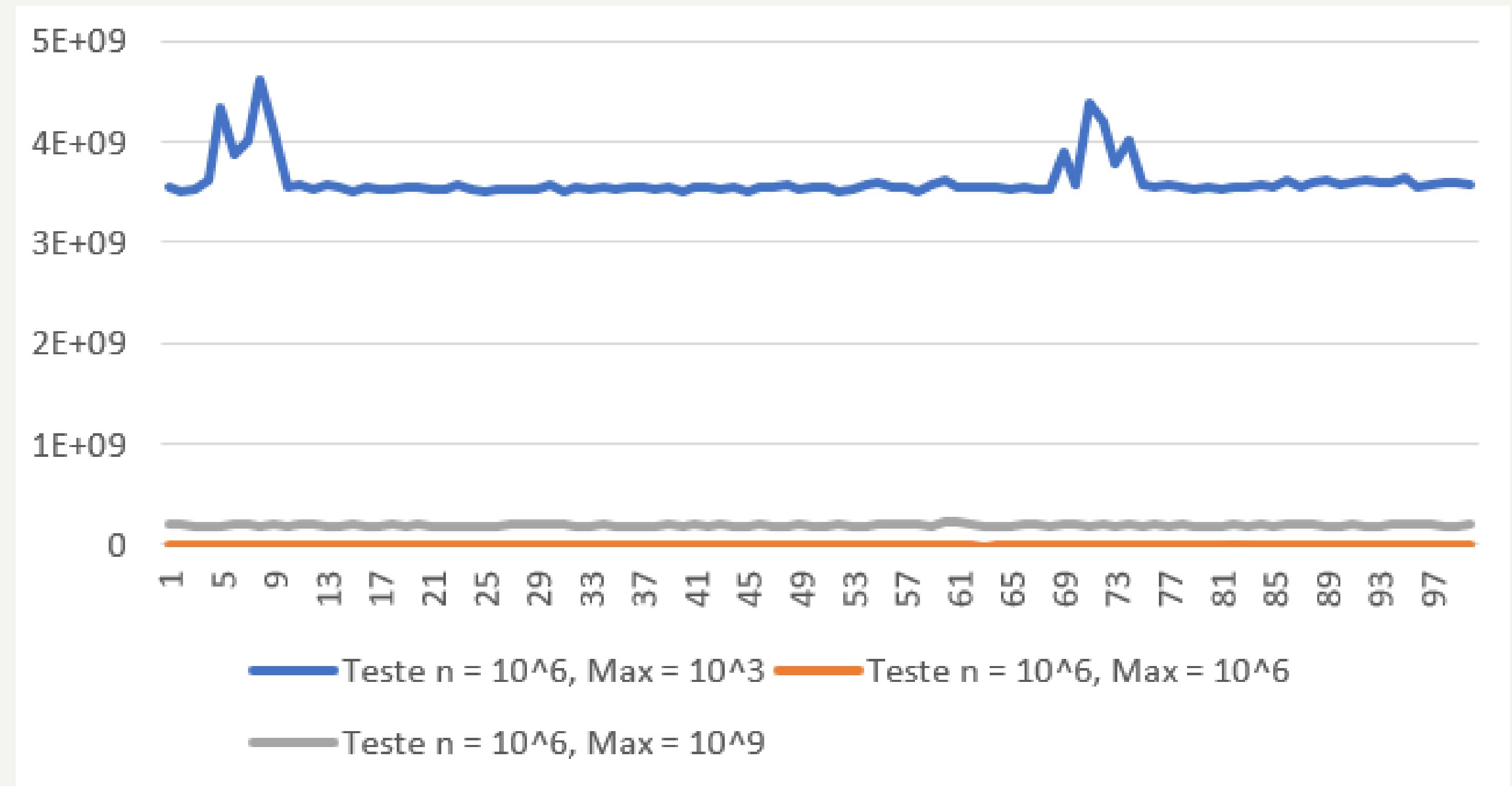
MERGE SORT



SHELL SORT

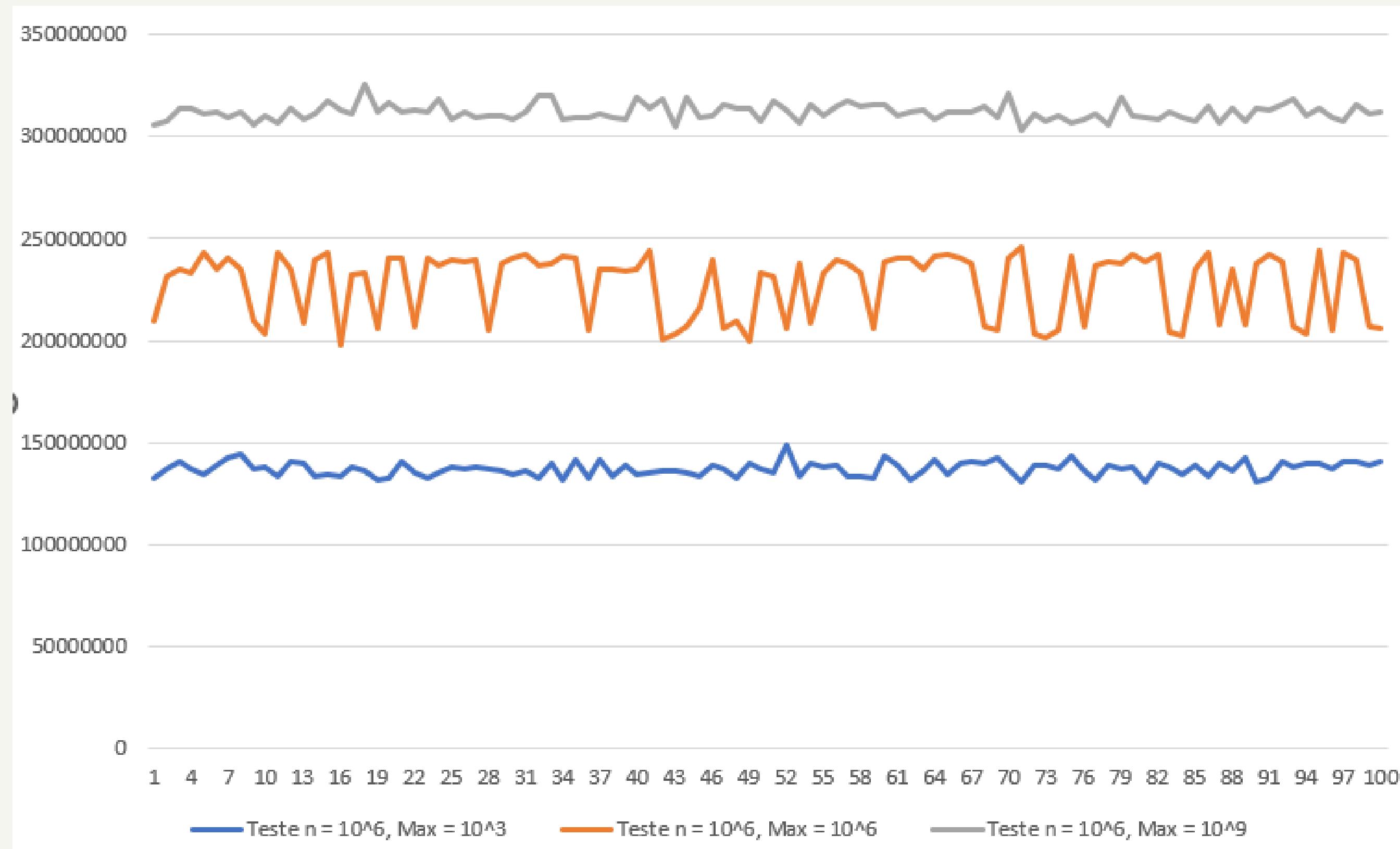


QUICK SORT



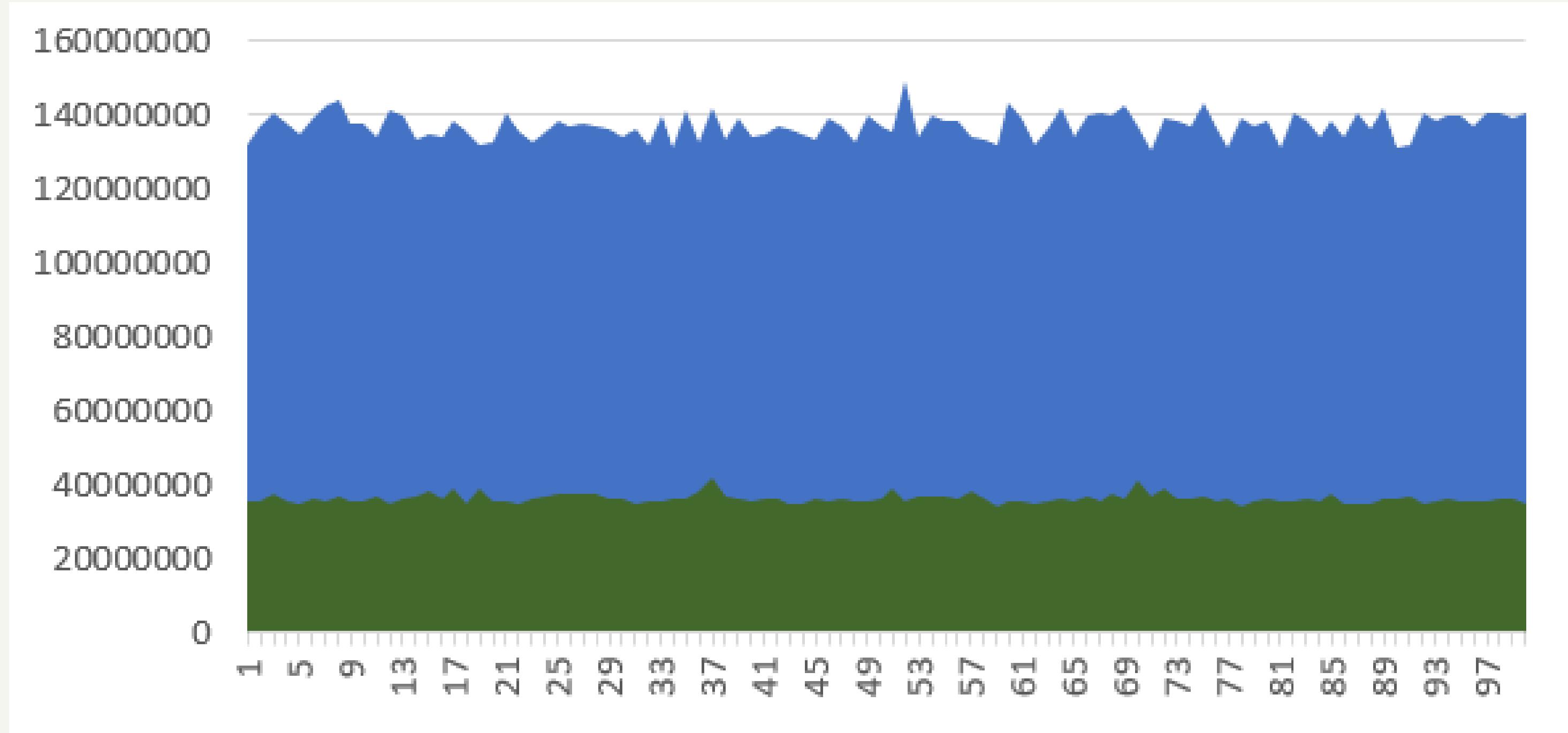
RADIX SORT

RADIX SORT



Radix sort

$n=10^6$ max= 10^3



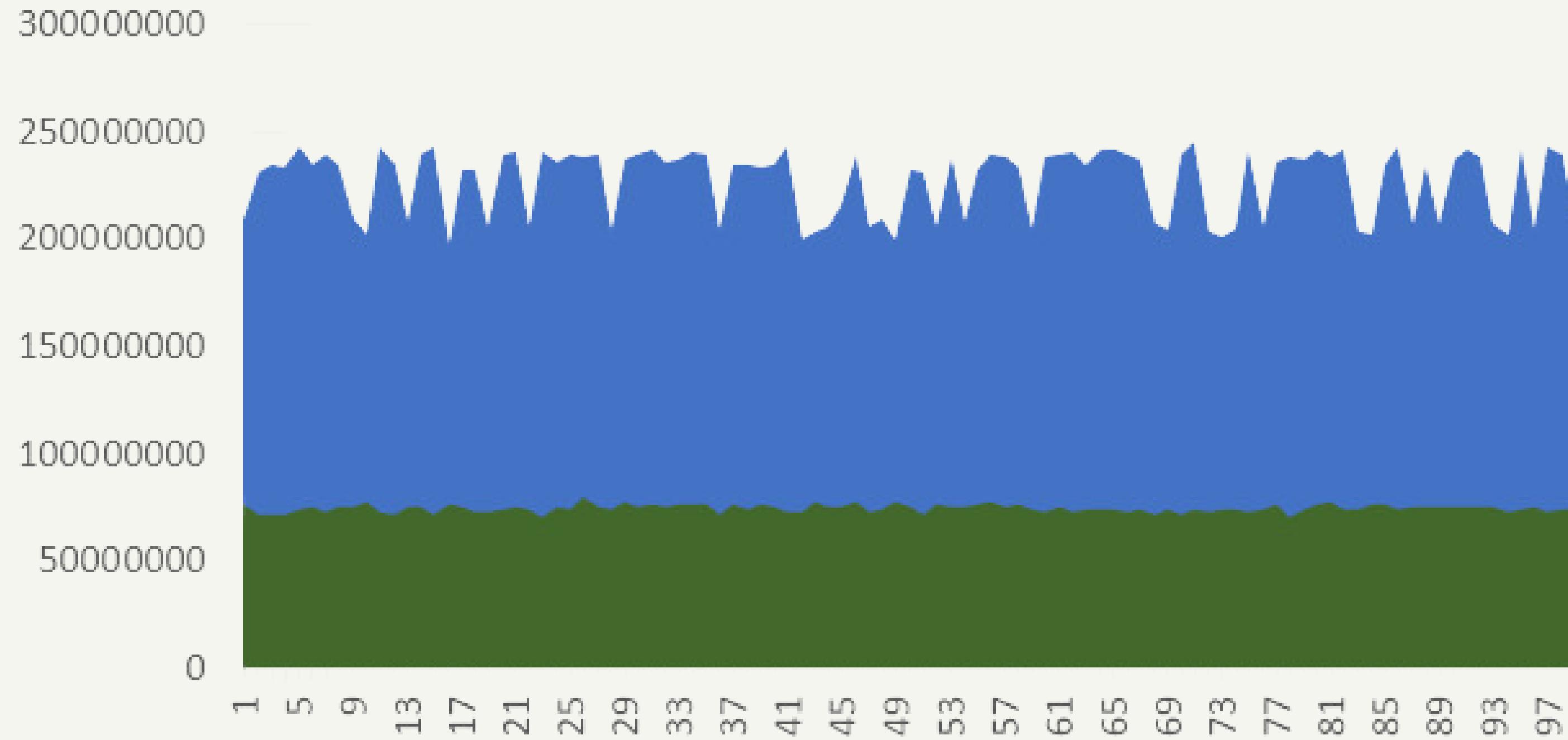
Baza 10



Baza 2^{16}

Radix sort

$n=10^6$ max= 10^6



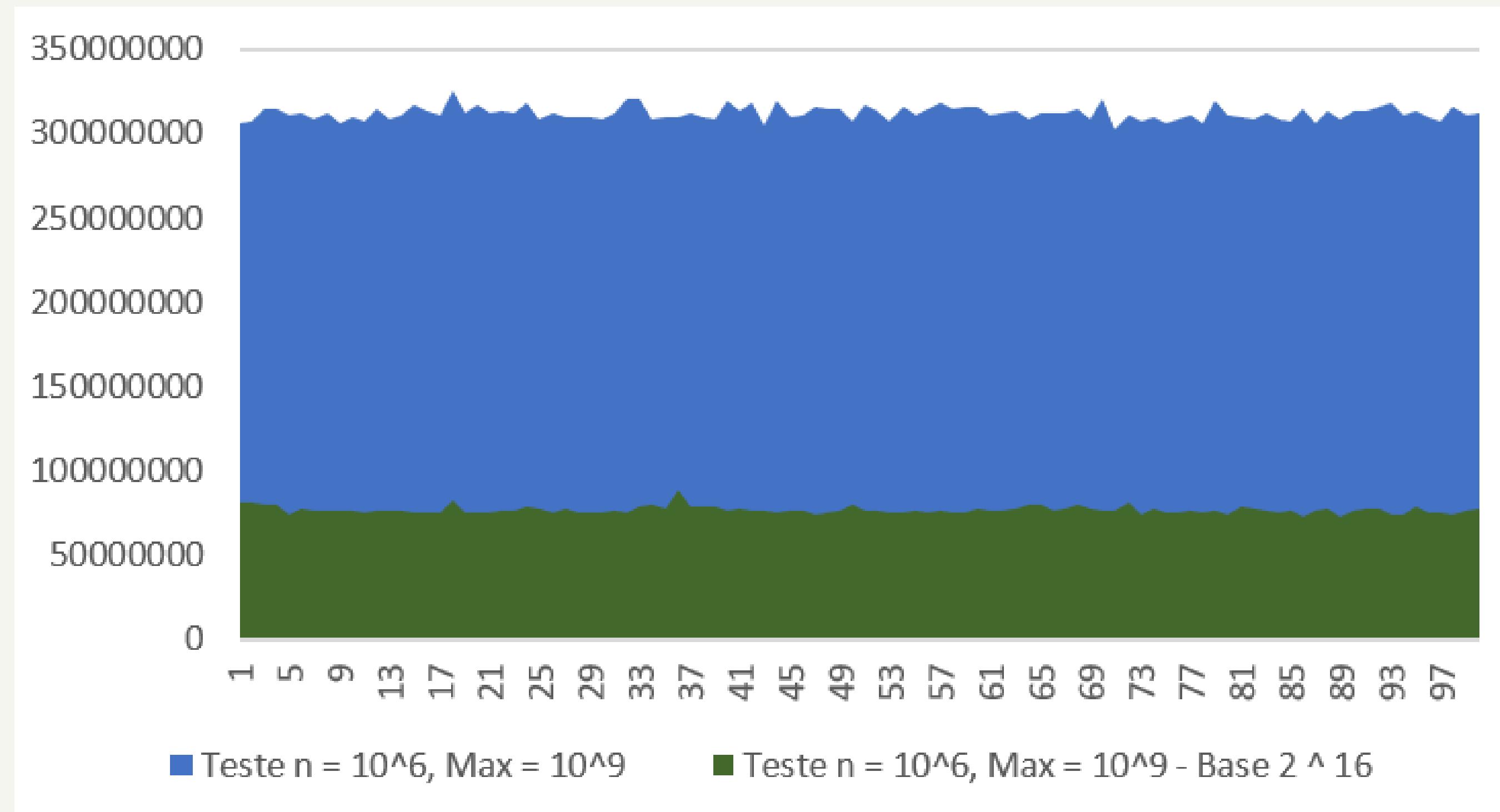
Baza 10



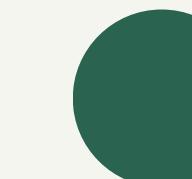
Baza 2^{16}

Radix sort

$n=10^6$ max= 10^9



Baza 10



Baza 2^{16}