ALGORITMI DE SORTARE

Calota Stefan-Daniel

Grupa 134

ALGORITMI IMPLEMENTATI

- bubbleSort
- quickSort
- mhellSort
- mergeSort
- radixSort

MODALITATEA DE TESTARE

2 tipuri de teste:

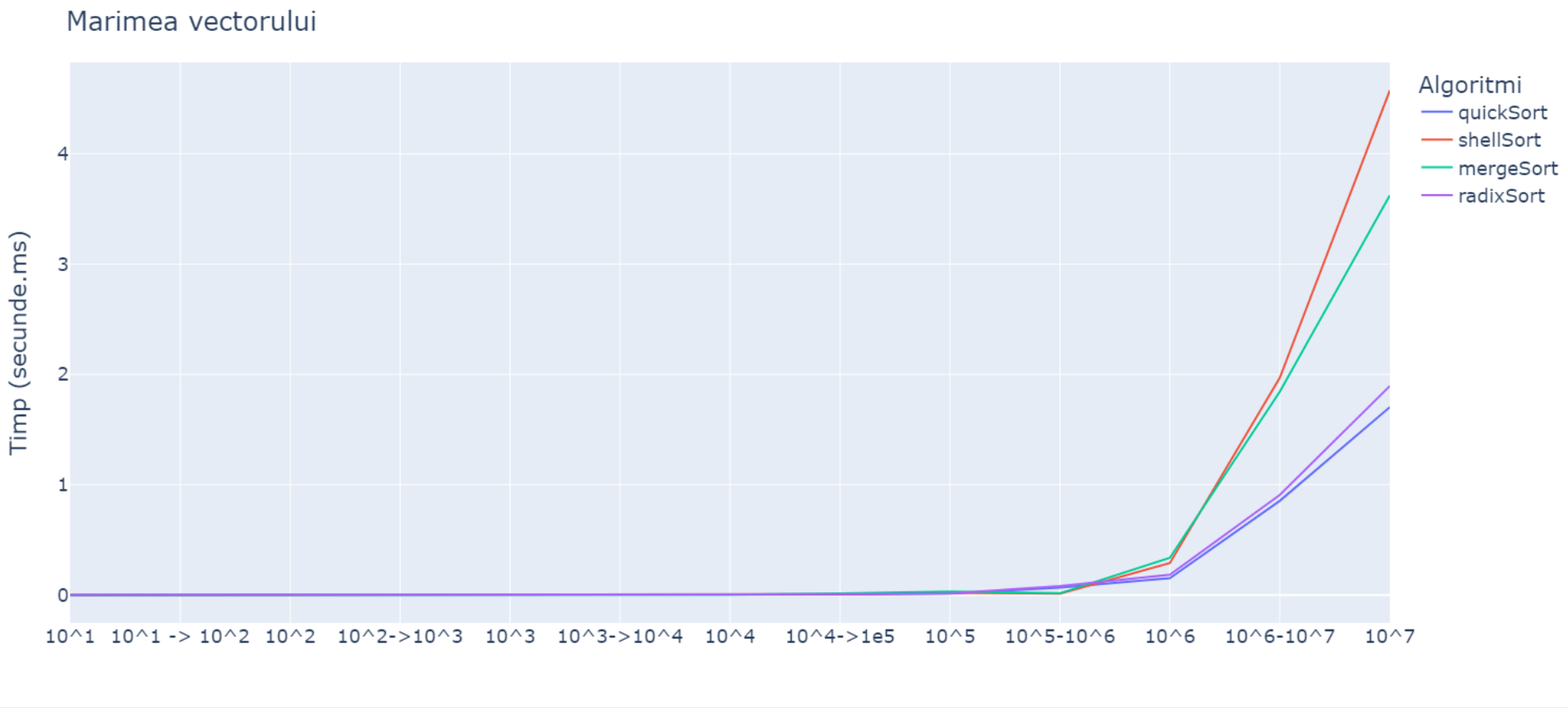
- numere aleatorii: elementele vor apartine intervalului [0,1000000000)
- numere pana in 8 cifre (pentru a observa avantajul radixSort-ului)

Fiecare algoritm va fi supus a câtor 100 de teste generate prin acest cod.

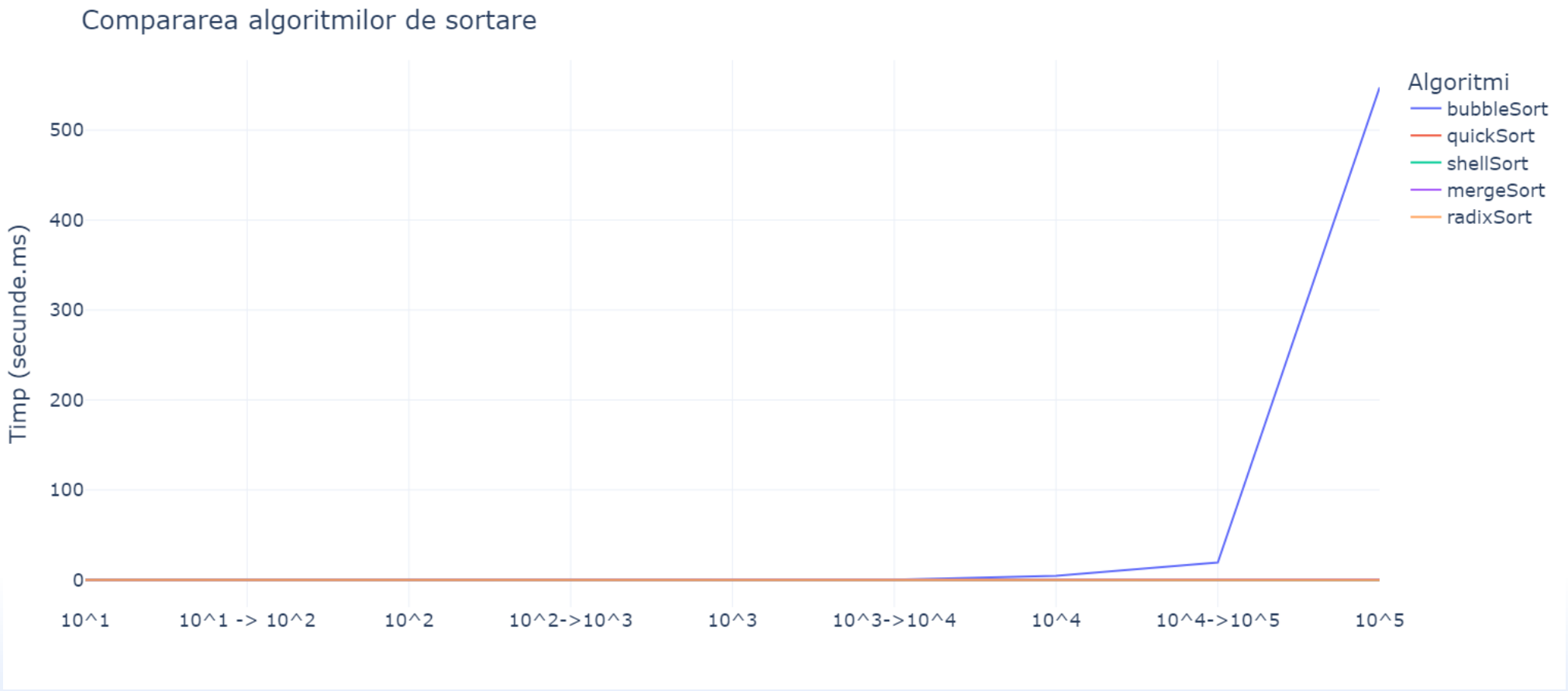
```
void generareVector(int v[], int range, int size)
{
    for (int i = 0; i < size; i++)
    {
        v[i] = nrfabi() % range;
    }
}
```

```
int nrfabi()
{
    int nr = 0;
    int p = 1;
    for (int i = 0; i < 9; i++)
    {
        nr += p * (rand() % 10);
        p *= 10;
    }

    return nr;
}
```

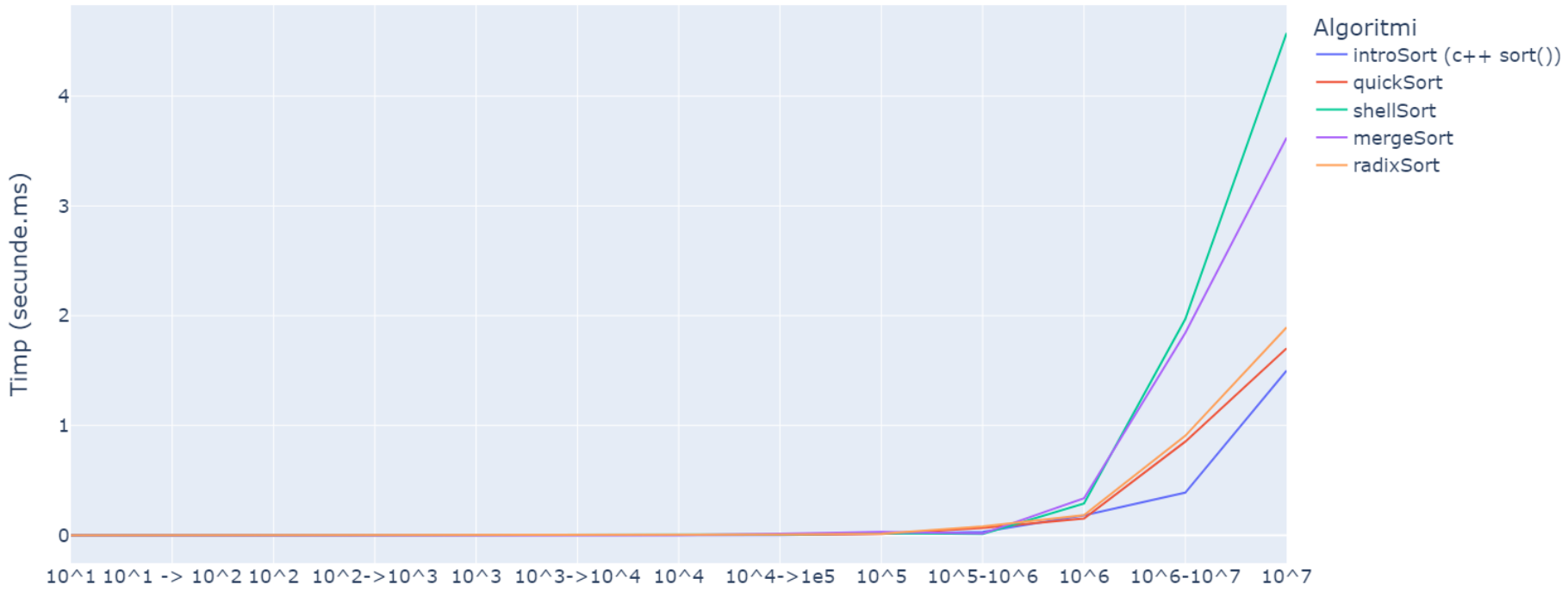


Pe cazul general, quickSort-ul ajunge sa fie cel mai eficient, foarte apropiat totusi de radixSort

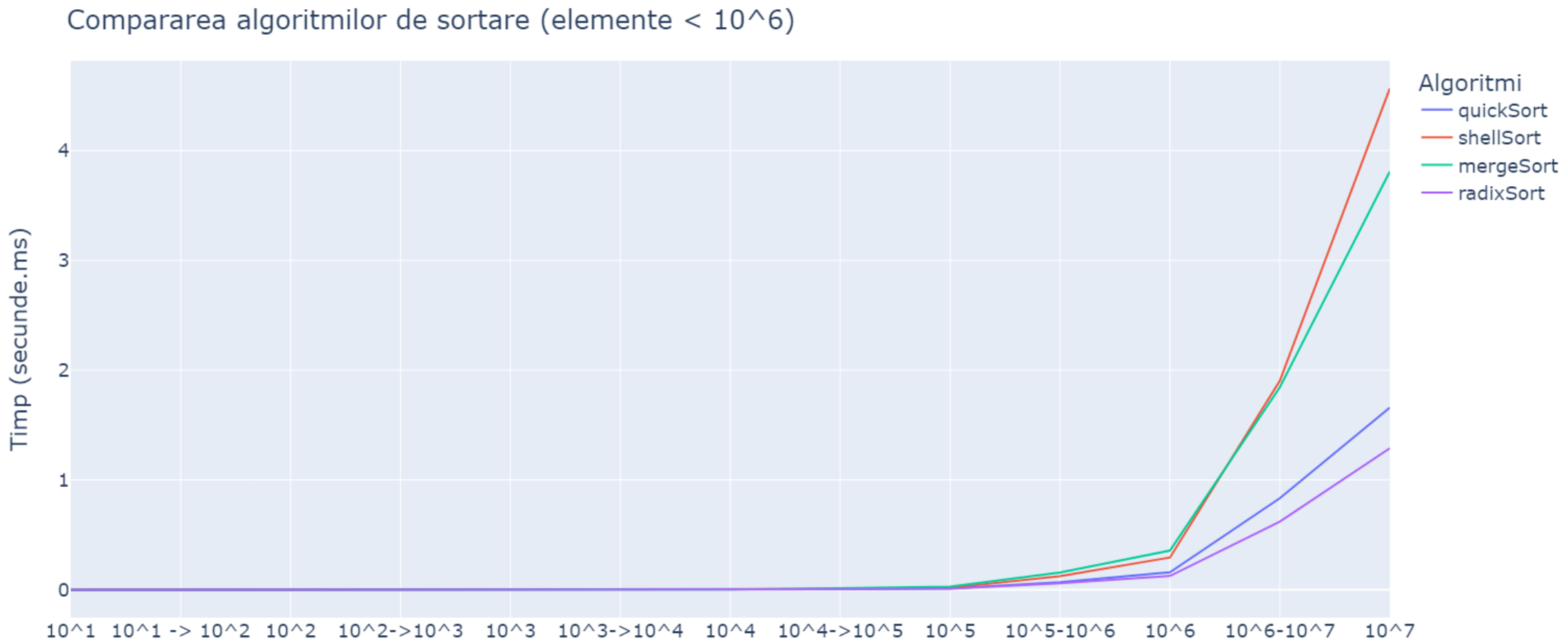


Aici putem vedea ineficienta unui algoritm in genul BubbleSort, un timp de executare astronomic comparativ cu restul.

Compararea algoritmilor de sortare



Observam ca sortarea default din c++, (introsortul) ramane mai buna decat toti ceilalti algoritmi implementati de catre mine.



Vedem totusi ca daca elementele nu depasesc valoarea 10^6, radixSort-ul este ramane cel mai eficient din cele implementate de catre mine, ceea ce mi-a aratat cel putin mie importanta de a lua in considerare si tipul de date in alegerea algoritmului folosit