

#### Sortarile alese:

- Counting Sort
- Heap Sort
- Radix Sort
- Shell Sort
- Merge Sort

Sortarile au fost implementate in limbajul Python si au fost testate pe 5 seturi de date distincte pentru a evalua performanta acestora.

Pentru generarea seturilor de date a fost folosita biblioteca random.

	Setul 1	Setul 2	Setul 3	Setul 4	Setul 5
Tip valori	Numere naturale	Numere naturale	Numere naturale	Numere naturale	Numere intregi
Interval	0 -> 100	0 -> 5.000.000	150.000 -> 800.000	0 => 42. 200.000	-50.000 -> 50.000
Numar valori	100.000	1.224.000	126.400	132.080	11.040

Au fost alese multe numere naturale dintr-un interval mic.

=> multe variabile identice.

Numar foarte mare de valori de sortat.

Numere naturale din intervalul 150.000 -> 800.000.

Majoritatea s-au situat in jurul valorii de 500.000.

Numerele naturale au fost foarte mari.

S-au considerat pana la 7 cifre.

Printre numerele date apar si valori negative.

#### **ALGORITM PROPRIU**

Setul 1 0.25499868392944336

Setul 2 9.61899733543396

Setul 3 6.000998497009277

Setul 4 55.713154315948486

Setul 5

0

### Count Sort

## Heap Sort

#### **ALGORITM PROPRIU**

Setul 1 17.95197343826294

Setul 2 293.1143527030945

Setul 3 26.420048236846924

Setul 4 26.438997507095337

Setul 5 2.7319939136505127

# Merge Sort

#### **ALGORITM PROPRIU**

Setul 1 13.028003454208374

Setul 2 184.5362513065338

Setul 3 16.007372856140137

Setul 4 16.693798303604126

Setul 5 1.3199901580810547

#### **ALGORITM PROPRIU**

Setul 1

83.17249774932861

Setul 2

Setul 3

Setul 4

Setul 5

### Radix Sort

#### **ALGORITM PROPRIU**

Setul 1

Setul 2

Setul 3

Setul 4

Setul 5

### **Shell Sort**