Prezentare timpi de rulare pentru diferite sortari

**MergeSort**

1. Random Array:
2. SIZE = 1000000, MAX\_VALUE = 10000 => 0.2970 SEC
3. SIZE = 10000000, MAX\_VALUE = 10000 => 3.1520 SEC
4. Ascending Array:
5. SIZE = 1000000, MAX\_VALUE = 1000 => 0.2360 SEC
6. SIZE = 10000000, MAX\_VALUE = 1000 => 2.2280 SEC
7. Descending Array:
8. SIZE = 1000000, MAX\_VALUE = 1000 => 0.1870 SEC
9. SIZE = 10000000, MAX\_VALUE = 1000 => 2.0070 SEC
10. Constant Array:
11. SIZE = 1000000, MAX\_VALUE = 1000 => 0.2250 SEC
12. SIZE = 10000000, MAX\_VALUE = 1000 => 2.2960 SEC

*! Complexitate: O(nlogn).*

*Max\_Value nu este relevant, ci numarul de elemente din array (SIZE)*

**QuickSort**

1. Random Array:
2. SIZE = 1000000, MAX\_VALUE = 10000 => 0.1730 SEC
3. SIZE = 10000000, MAX\_VALUE = 10000 => 1.6140 SEC
4. Ascending Array:
5. SIZE = 1000000, MAX\_VALUE = 1000 => 0.0470 SEC
6. SIZE = 10000000, MAX\_VALUE = 1000 => 0.5380 SEC
7. Descending Array:
8. SIZE = 1000000, MAX\_VALUE = 1000 => 0.0500 SEC
9. SIZE = 10000000, MAX\_VALUE = 1000 => 0.5310 SEC
10. Constant Array:
11. SIZE = 1000000, MAX\_VALUE = 1000 => 0.1010 SEC
12. SIZE = 10000000, MAX\_VALUE = 1000 => 1.2860 SEC

*! Complexitate: O(nlogn).*

*Max\_Value nu este relevant, ci numarul de elemente din array (SIZE).*

*Algoritmul este optim in cazul sortarii listelor nesortate, de preferat, cand numarul de elemente nu este foarte mare.*

**CountSort**

1. Random Array:
2. SIZE = 1000000, MAX\_VALUE = 100000 => 0.0100 SEC
3. SIZE = 10000000, MAX\_VALUE = 10000000 => 0.1370 SEC
4. Ascending Array:
5. SIZE = 100000, MAX\_VALUE = 100000 => 0.0010 SEC
6. SIZE = 10000000, MAX\_VALUE = 10000000 => 0.1580 SEC
7. Descending Array:
8. SIZE = 1000000, MAX\_VALUE = 1000000 => ? SEC
9. SIZE = 1000, MAX\_VALUE = 10000 => 2.0000 SEC
10. Constant Array:
11. SIZE = 1000000, MAX\_VALUE = 100000 => 0.1260 SEC
12. SIZE = 10000000, MAX\_VALUE = 10000000 => 0.0100 SEC

*! Complexitate: O(n).*

*Max\_Value este relevant, determinand lungimea vectorului de sortare.*

*Este optim atunci cand vrem sa sortam liste ordonate cu multe elemente.*

**RadixSort**

1. Random Array:
2. SIZE = 10000000, MAX\_VALUE = 100000 => 1.2550 SEC
3. SIZE = 10000000, MAX\_VALUE = 10000000 => 1.3120 SEC
4. Ascending Array:
5. SIZE = 10000000, MAX\_VALUE = 10000000 => 1.6700 SEC
6. SIZE = 10000000, MAX\_VALUE = 1000000 => ? SEC
7. Descending Array:
8. SIZE = 10000000, MAX\_VALUE = 10000 => 1.6950 SEC
9. SIZE = 10000000, MAX\_VALUE = 10000000 => 1.6910 SEC
10. Constant Array:
11. SIZE = 10000000, MAX\_VALUE = 10000 => 0.4780 SEC
12. SIZE = 10000000, MAX\_VALUE = 10000000 => 0.5010 SEC

*! Complexitate: O(n + baza).*

*Max\_Value este relevant.*

*Algoritmul este optim in sortarea listelor ci un numar foarte mare de elemente.*

**BubbleSort**

1. Random Array:
2. SIZE = 100, MAX\_VALUE = 1000 => 0.0000 SEC
3. SIZE = 10000, MAX\_VALUE = 1000 => 0.5430 SEC
4. Ascending Array:
5. SIZE = 100, MAX\_VALUE = 1000 => 0.0000 SEC
6. SIZE = 10000, MAX\_VALUE = 1000 => ? SEC
7. Descending Array:
8. SIZE = 100, MAX\_VALUE = 1000 => 0.0000 SEC
9. SIZE = 10000, MAX\_VALUE = 1000 => 0.5590 SEC
10. Constant Array:
11. SIZE = 100, MAX\_VALUE = 1000 => 0.0000 SEC
12. SIZE = 10000, MAX\_VALUE = 1000 => 0.0000 SEC

*! Complexitate: O(nlogn).*

*Max\_Value nu este relevant.*

*Algoritmul in sine, din cauza complexitatii, nu este foarte optim in sortarea listelor cu prea multe elemente.*