Structuri de date – Tema 1

Analiza mai multor algoritmi de sortare

Neacsu-Tranciuc Sasa-Andrei Grupa 133

In urma mai multor teste se poate deduce eficienta algoritmilor in functie de cantitatea numerelor sortate si de numarul maxim posibil. Metodele de sortare testate au fost Count Sort, Radix Sort, Merge Sort, Quick Sort, Bubble Sort si Tim Sort (varianta nativa din python).

Dupa mai multe teste, timpul mediu(in secunde) in functie de test al algoritmilor este:

N = Numere sortate Max = Cel mai mare numar posibil

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Count | Radix | Merge | Quick | Bubble | Tim |
| N = 10000  Max =1000 | 0.001 | 0.016 | 0.024 | 0.005 | 5.8 | 0.0007 |
| N =1000  Max =10000 | 0.009 | 0.1 | 0.33 | 0.03 | 0.05 | 0.005 |
| N =100000  Max =100 | 0.0004 | 0.016 | 0.0018 | 0.0012 | 622 | 7.3 |
| N =100000  Max =10000 | 0.01 | 0.28 | 0.32 | 0.14 | 620 | 0.011 |
| N = 10000  Max =100000 | 0.1 | 2.45 | 4.15 | 1.4 | 5.9 | 0.13 |
| N =1000000  Max =1000 | 0.036 | 0.028 | 0.026 | 0.018 | +++ | 0.01 |
| N =1000000 Max =100000 | 0.11 | 4 | 4.3 | 2.6 | +++ | 0.21 |
| N =100000000 Max =1000 | 11 | 256 | 586 | 253 | +++ | 13.5 |
| N =1000000 Max =1000000 | 3.3 | 0.03 | 0.027 | 0.19 | +++ | 0.001 |

In zonele cu +++ timpul de procesare era atat de mare incat nu mai fost masurat, fiind evidenta ineficienta algoritmului la acel test.

Observatii:

Bubble sort este extrem de ineficient, fiind bun doar pentru liste deja sortate sau aproape sortate si pentru liste cu un numar extrem de mic de numere

Count sort este foarte rapid cand este utilizat pe liste cu o cantitate mare de numere cu un maxim mic. In situatii inverse, este mai incet comparativ cu ceilalti algoritmi.

Avand in vedere toate tipurile de liste sortabile, algoritmul nativ este cea mai buna solutie in majoritatea cazurilor.

Merge sort are o viteza decenta pana la un anumit punct, unde devine mai incet comparativ cu celelalte metode de sortare.