Algoritmi De Sortare

de Balanica Andrei

grupa 133

Algoritmi utilizati si implementati

Radix Sort
Merge Sort
Shell Sort
Bubble Sort
Quick Sort

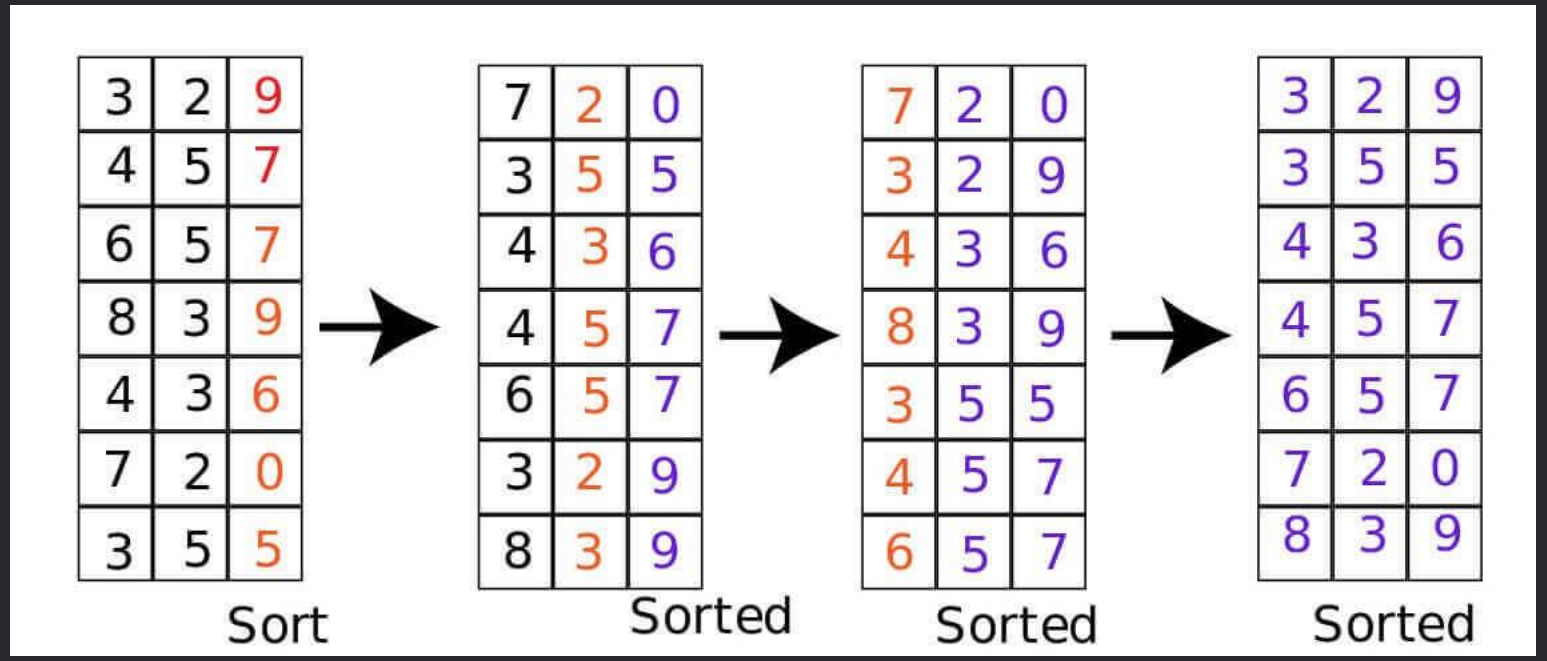
Radix Sort

Radix sort este un algoritm de sortare care sortează elementele prin gruparea mai întâi a cifrelor individuale cu aceeași valoare de loc. Apoi, sortați elementele în funcție de ordinea lor crescătoare/descrescătoare.

Complexitate O(n+max)

Radix Sort

Timpi de executie
baza 10
N = 10^3 0.000264600 sec
N = 10^5 0.012021700 sec
N = 10^8 22.960489800 sec
baza 16
N = 10^3 0.000168300 sec
N = 10^5 0.011529400 sec
N = 10^8 16.609678200 sec
baza 2^16
N = 10^3 0.000472000 sec
N = 10^5 0.006753200 sec
N = 10^8 6.957157700 sec



Shell Sort

Shell Sort este o versiune generalizată a algoritmului Insertion Sort. Mai întâi sortează elementele care sunt departe unele de altele și reduce succesiv intervalul dintre elementele de sortat.

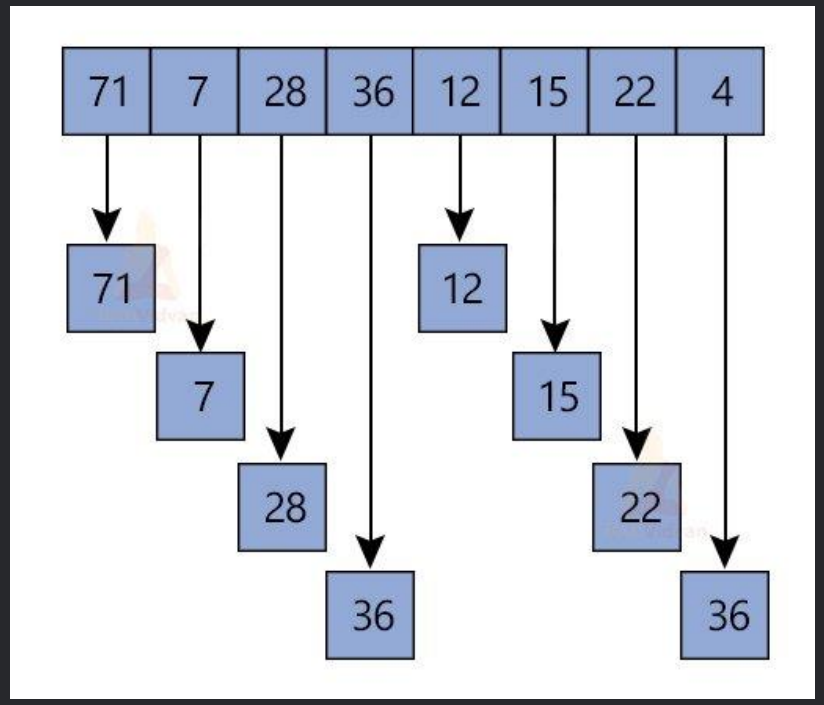
In acest proiect am utilizat secventa originala care imparte de fiecare data distanta la 2 incepand cu distanta n (lungimea array-ului)

Complexitate O(n^2)

Shell Sort

Timpi de executie

N = 10^3 0.000083000 sec
N = 10^5 0.019916300 sec
N = 10^8 54.217781400 sec



Merge Sort

Merge Sort este unul dintre cei mai populari algoritmi de sortare care se bazează pe principiul algoritmului Divide et Impera.

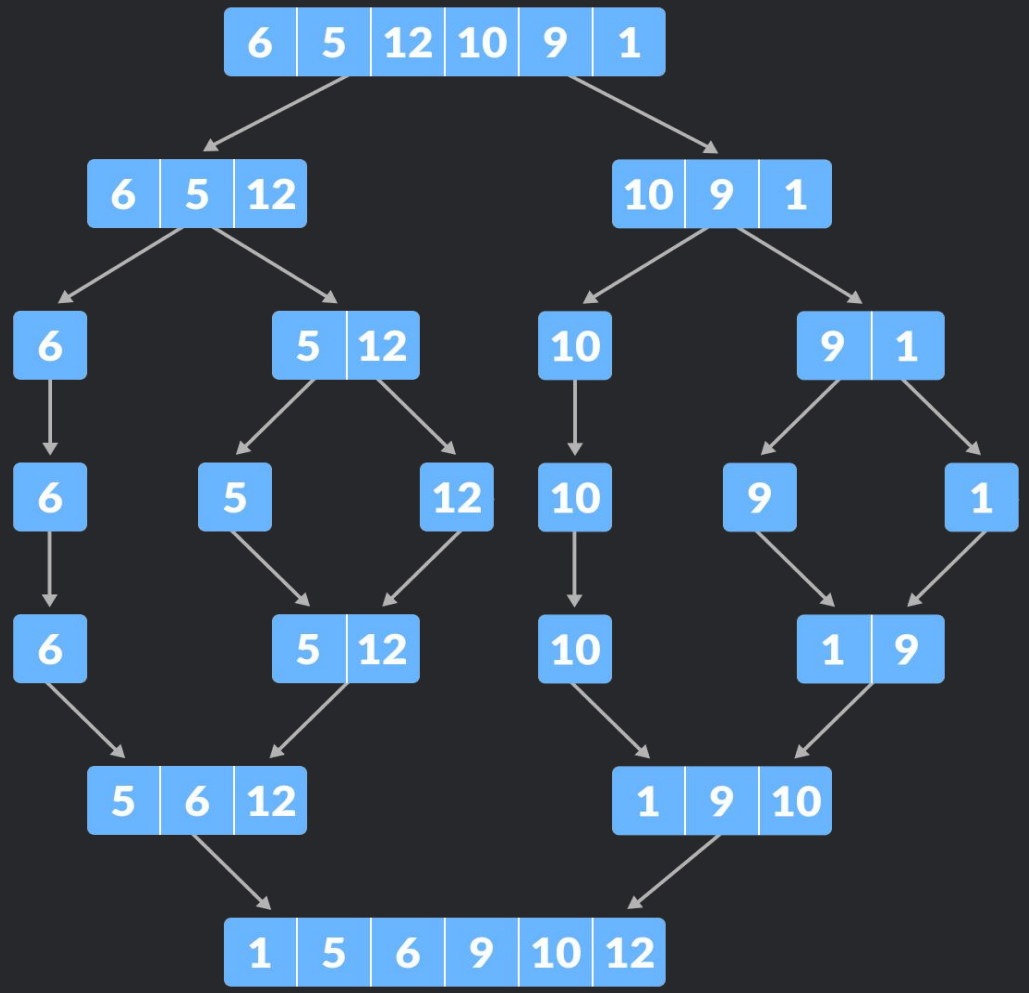
Acest desparte array-ul in sub array-uri care sunt si ele despartite pna se ajung la sub array-uri de 1 element care sunt sortate si care se merge-uiesc cu jumatatea lor

Complexitate O(n*logn)

Merge Sort

Timpi de executie

N = 10^3 0.000135200 sec
N = 10^5 0.016530000 sec
N = 10^8 32.297690700 sec



Bubble Sort

Bubble Sort este unul dintr cele mai usoare algoritme de implementat, dar nu si eficient

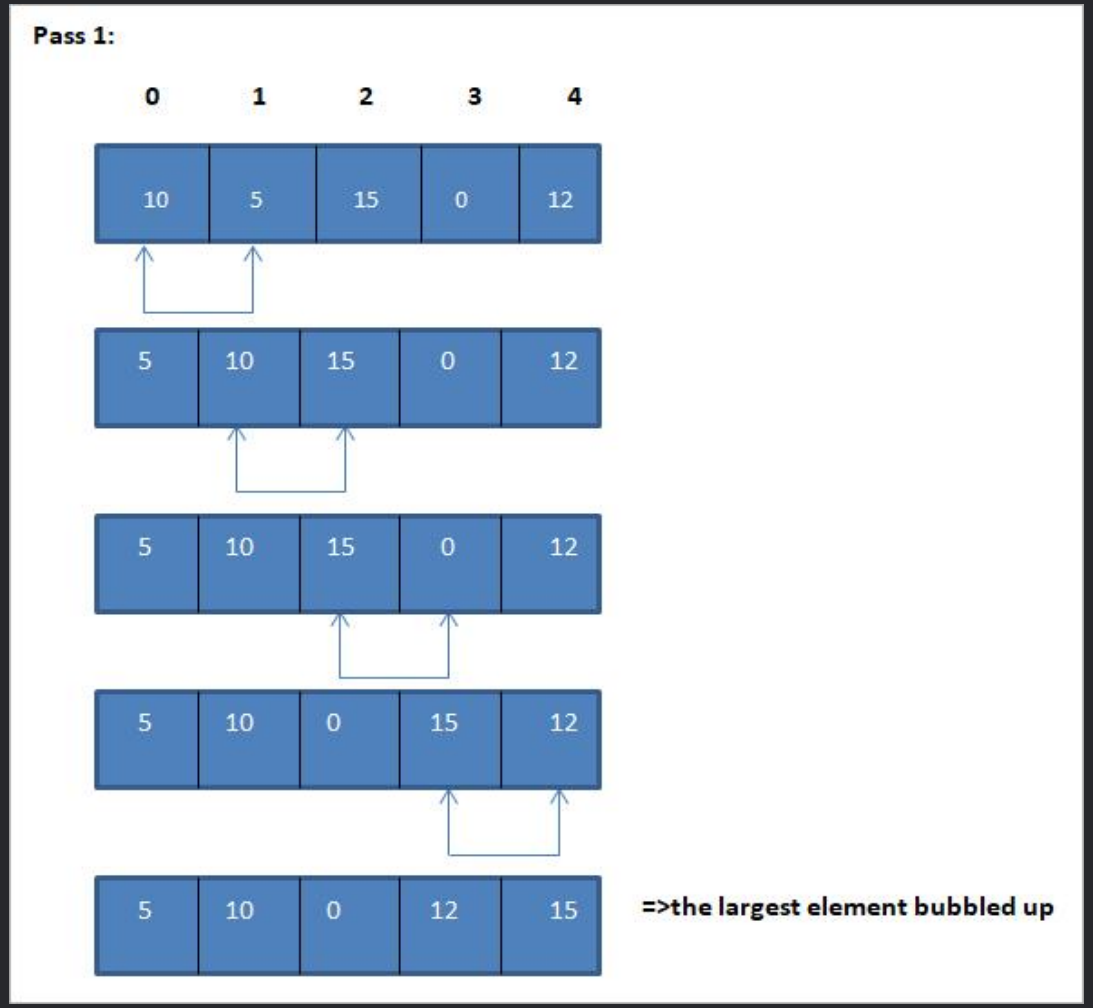
Acesta compara elemente consecutive si le interschimba daca primul este mai mare decat al doilea pana ajunge cel mai mare element in capat si tot asa

Complexitate O(n^2)

Bubble Sort

Timpi de executie

N = 10^3 0.001725100 sec
N = 10^5 26.578492800 sec
N = 10^8 Nu a putut fi calculate(timp de executie prea mare)



Quick Sort

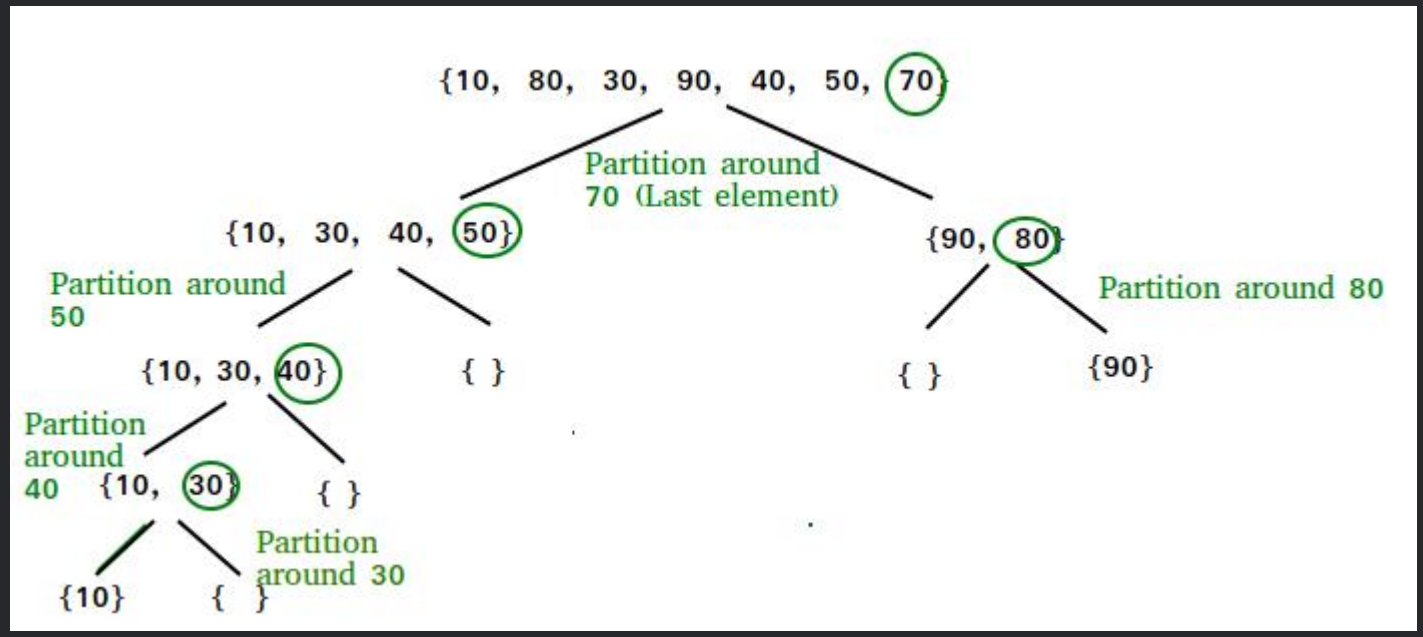
Quick Sort este un algoritm bazat pe principiul Divide et Impera care creaza sub array-uri in functie de un element pivot(eu am ales ultimul element) . Numerele mai mari ca piv formeaza un array si cele mai mic altul.

Complexitate O(n*logn)

Quick Sort

Timpi de executie

N = 10^3 0.000097300 sec
N = 10^5 0.011318200 sec
N = 10^8 17.792714100 sec



Comparatie

Am comparat toate functiile de sortare prezentate anterior, testandu-le pe diverse inputuri cu marimi si dimensiuni diverse. Dupa cum se observa, pentru array-uri cu dimensiuni mari, preferam sa utilizam Radix Sort, acesta devenind din ce in ce mai buna dupa ce am crescut baza.

```
N = 100000000  Maxim = 1000000
Time taken by stl sort is : 21.477833100 sec
Time taken by radixsort 10 is : 24.737428300 sec Sorted
Time taken by radixsort 16 is : 14.650499600 sec Sorted
Time taken by radixsort 2^16 is : 8.406350100 sec Sorted
Time taken by mergesort is : 31.644220400 sec Sorted
Time taken by shellsort is : 63.858135100 sec Sorted
Time taken by quicksort is : 26.120623800 sec Sorted
```

Comparatie

De Asemenea, putem observa ca si Sortul din C++ si QuickSortul fac o treaba buna, chiar daca sunt de 3 ori mai lente decat RadixSortul in baza 2^16. MergeSorul se afla pe penultimul loc(4 ori mai incet decat RS 2^16) urmat de Shell Sort(de 8 ori mai lent).

```
N = 100000000  Maxim = 1000000
Time taken by stl sort is : 21.477833100 sec
Time taken by radixsort 10 is : 24.737428300 sec Sorted
Time taken by radixsort 16 is : 14.650499600 sec Sorted
Time taken by radixsort 2^16 is : 8.406350100 sec Sorted
Time taken by mergesort is : 31.644220400 sec Sorted
Time taken by shellsort is : 63.858135100 sec Sorted
Time taken by quicksort is : 26.120623800 sec Sorted
```

Ineficienta Bubble Sortului

Bubble Sortul nu a fost inclus in testul anterior din pricina timpului de executare nemasurabil. Dupa minute bune de asteptare, acesta nu reusea sa sorteze 10^8 elemente intr-un timp decent si de aceea nu am putut sa testez aceasta. As putea sa aproximez ca acesta ar fi de n/log n mai ineficient decat restul algoritmelor cu complexitate nlogn(Merge, Quick) cea ce este o diferenta uriasa cand testam pe 10^8elemente. Bubble Sort este, pe departe cel mai ineficient dintre acestea dupa cum arata si testul 5.

```
Test 5
N = 100000  Maxim = 100000
Time taken by stl sort is : 0.014865900 sec
Time taken by radixsort 10 is : 0.018212000 sec Sorted
Time taken by radixsort 16 is : 0.014492000 sec Sorted
Time taken by radixsort 2^16 is : 0.009090400 sec Sorted
Time taken by mergesort is : 0.021800600 sec Sorted
Time taken by shellsort is : 0.025245500 sec Sorted
Time taken by bubblesort is : 32.954423700 sec Sorted
Time taken by quicksort is : 0.016251700 sec Sorted
```

Rezultatele testarii

```
Test 1
N = 1000  Maxim = 1000
Time taken by stl sort is : 0.000147 sec
Time taken by radixsort 10 is : 0.000135000 sec Sorted
Time taken by radixsort 16 is : 0.000099200 sec Sorted
Time taken by radixsort 2^16 is : 0.000272200 sec Sorted
Time taken by mergesort is : 0.000169400 sec Sorted
Time taken by shellsort is : 0.000097400 sec Sorted
Time taken by bubblesort is : 0.002099500 sec Sorted
Time taken by quicksort is : 0.000085200 sec Sorted
Test 2
N = 1000  Maxim = 100000
Time taken by stl sort is : 0.000091400 sec
Time taken by radixsort 10 is : 0.000193500 sec Sorted
Time taken by radixsort 16 is : 0.000136000 sec Sorted
Time taken by radixsort 2^16 is : 0.000565500 sec Sorted
Time taken by mergesort is : 0.000153000 sec Sorted
Time taken by shellsort is : 0.000099500 sec Sorted
Time taken by bubblesort is : 0.002056800 sec Sorted
Time taken by quicksort is : 0.000085200 sec Sorted
Test 3
N = 1000  Maxim = 100000000
Time taken by stl sort is : 0.000092100 sec
Time taken by radixsort 10 is : 0.000368200 sec Sorted
Time taken by radixsort 16 is : 0.000200600 sec Sorted
Time taken by radixsort 2^16 is : 0.000550000 sec Sorted
Time taken by mergesort is : 0.000184700 sec Sorted
Time taken by shellsort is : 0.000112800 sec Sorted
Time taken by bubblesort is : 0.002178200 sec Sorted
Time taken by quicksort is : 0.000086800 sec Sorted
```

```
Test 4
N = 100000  Maxim = 1000
Time taken by stl sort is : 0.013053900 sec
Time taken by radixsort 10 is : 0.013609300 sec Sorted
Time taken by radixsort 16 is : 0.008610300 sec Sorted
Time taken by radixsort 2^16 is : 0.004073700 sec Sorted
Time taken by mergesort is : 0.019950600 sec Sorted
Time taken by shellsort is : 0.020234000 sec Sorted
Time taken by bubblesort is : 34.751301000 sec Sorted
Time taken by quicksort is : 0.016697000 sec Sorted
Test 5
N = 100000  Maxim = 100000
Time taken by stl sort is : 0.014865900 sec
Time taken by radixsort 10 is : 0.018212000 sec Sorted
Time taken by radixsort 16 is : 0.014492000 sec Sorted
Time taken by radixsort 2^16 is : 0.009090400 sec Sorted
Time taken by mergesort is : 0.021800600 sec Sorted
Time taken by shellsort is : 0.025245500 sec Sorted
Time taken by bubblesort is : 32.954423700 sec Sorted
Time taken by quicksort is : 0.016251700 sec Sorted
Test 6
N = 100000  Maxim = 10000000
Time taken by stl sort is : 0.014665100 sec
Time taken by radixsort 10 is : 0.024672000 sec Sorted
Time taken by radixsort 16 is : 0.018181500 sec Sorted
Time taken by radixsort 2^16 is : 0.008816900 sec Sorted
Time taken by mergesort is : 0.022854800 sec Sorted
Time taken by shellsort is : 0.025018700 sec Sorted
Time taken by bubblesort is : 33.170081100 sec Sorted
Time taken by quicksort is : 0.015009100 sec Sorted
```

Thanks

Add your text