Algoritmi de sortare C++

de Biciușcă Matei-Alexandru

Algoritmii de sortare prezentați

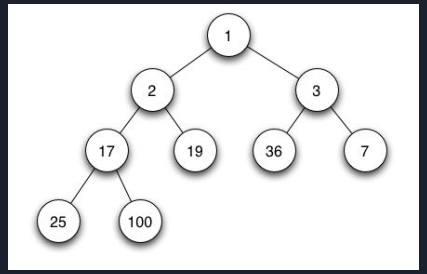
Heap Sort

Merge Sort

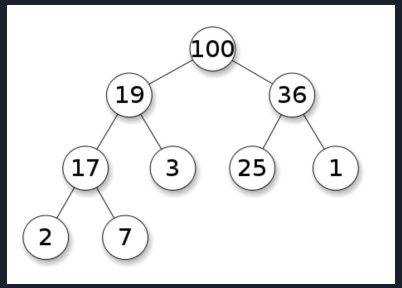
Radix Sort

Shell Sort

Bubble Sort



Min Heap



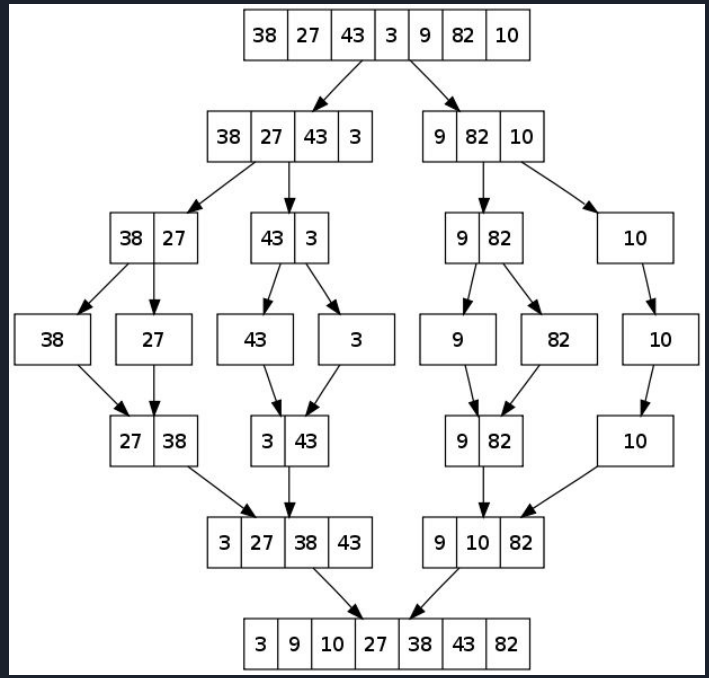
Max Heap

Heap Sort

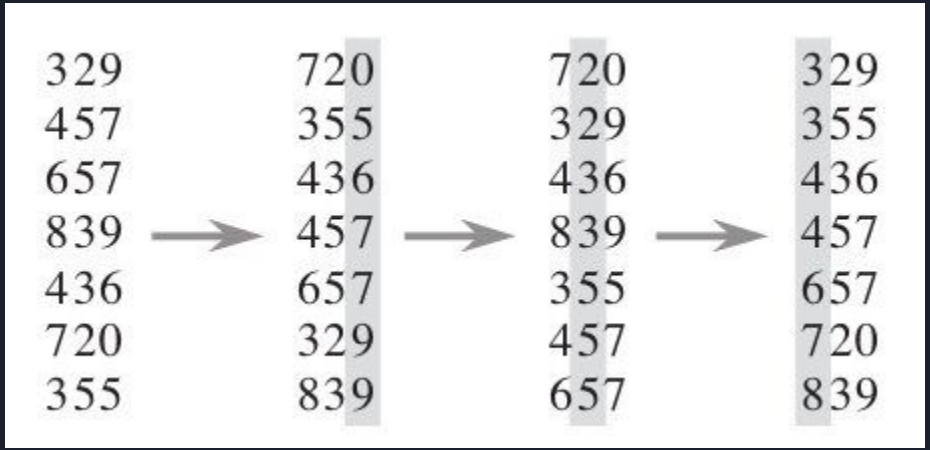
Heap Sort este un algoritm de sortare prin comparare care are la bază o structură heap (mai exact max heap). Complexitatea algoritmului este O(n log n). Nu are un best-case sau un worst-case diferit față de complexitatea generală. Ca avantaj nu folosește memorie suplimentară. În schimb, nu este o sortare stabilă.

Merge Sort

Merge Sort (sortare prin interclasare) este un algoritm de tip Divide et Impera care are complexitatea O(n log n). La fel ca Heap Sort, best-case și worst-case sunt tot O (n log n). Este o sortare stabilă, bazându-se pe comparări, dar folosește memorie suplimentară.



Radix Sort


329
457
657
839
436
720
355
720
355
436
457
657
329
839
720
329
436
839
355
457
657
329
355
436
457
657
720
839

Radix Sort

Radix Sort este o sortare stabilă, bazată pe numărare, ce poate fi implementată în mai multe baze și care este cunoscută sub două forme:

- MSD (Most Significant Digit)
- LSD (Least Significant DIgit)

Dintre cele două variante am ales să implementez LSD-ul. LSD Radix Sort este stabil, dar folosește memorie în plus.

În acest caz am implementat Radix Sort-ul în bazele 10, 2^8, 2^{10} și 2^{16}.

Shell Sort

Shell Sort este o versiune generalizată și mai eficientă a Insertion Sort, care sortează elementele aflate la o anumită distanță ce se modifică de-a lungul algoritmului. Secvențele de distanță pot varia, existând diverse șiruri încercate pentru determinarea unui Shell Sort mai eficient. Shell Sort-ul nu are memorie în plus, dar nu este stabil.

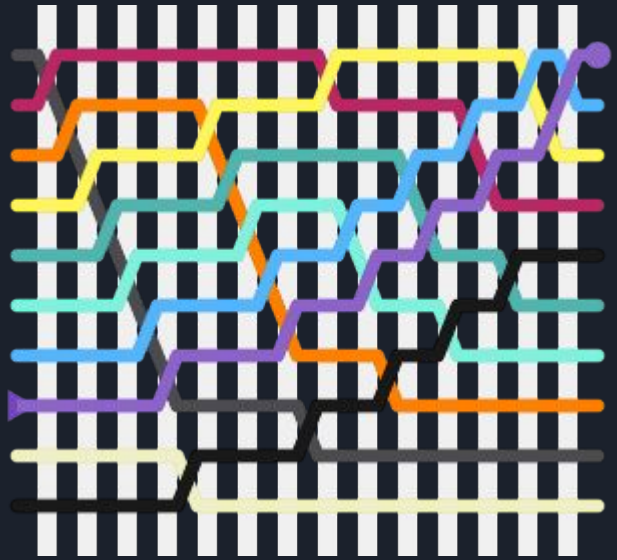
Am luat în considerare 5 șiruri pentru compararea eficienței Shell Sort, raportat la distanțele alese:

- secvența lui Donald Shell: $N/2^k$
- secvența lui Ciura: 1, 4, 10, 23, 57, 132, 301, 701, 1750, ...
- secvența lui Hibbard: $2^k - 1$
- secvența lui Tokuda: ceil((9 * (9 / 4)^k - 4) / 5)
- secvența lui Sedgewick (1982): $4^k + 3 * 2^{k-1} + 1$

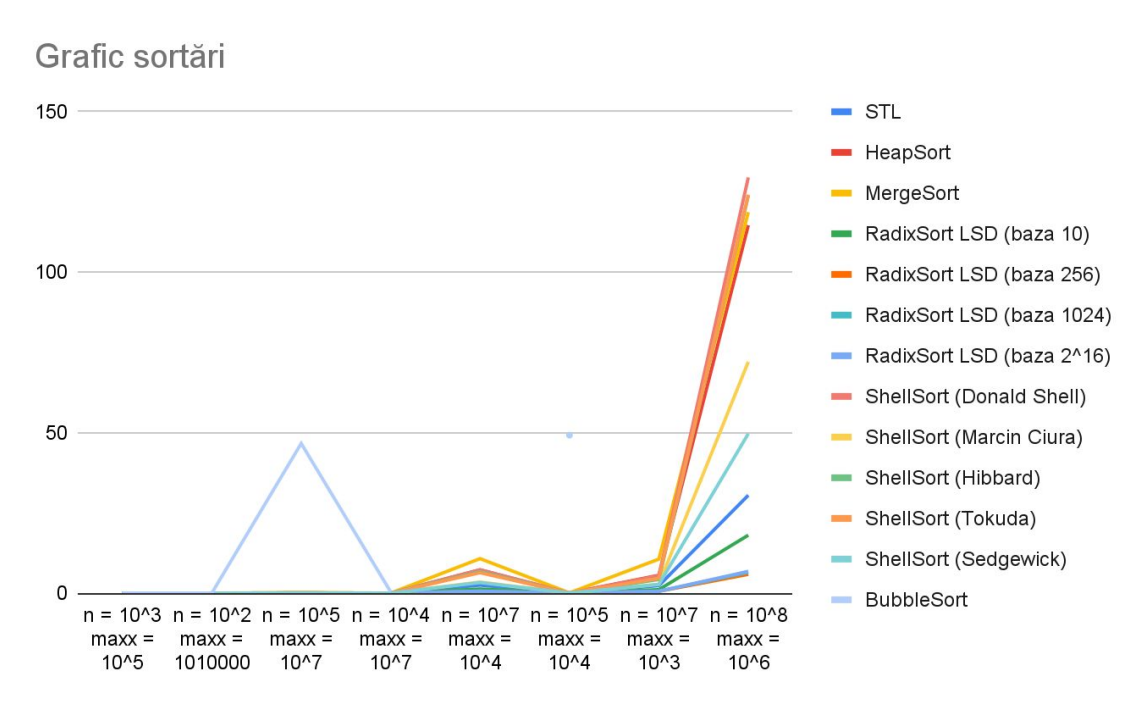
Bubble Sort

Bubble Sort este un algoritm simplu de sortare care parcurge în mod repetat lista, compară elementele adiacente și le schimbă dacă sunt în ordinea greșită.

Această este o sortare stabilă, ce nu solicită memorie suplimentară. Best-case-ul este O(n), iar worst-case (și case-ul general) este $O(n^2)$.



Graficul general al sortărilor date



Idei generale ce reies din grafic și output.txt

1. Așa cum se aștepta, Bubble Sort-ul este cea mai lentă sortare, întrucât are cea mai mare complexitate generală. De la 10^6 a fost de preferat să se renunțe la afișarea timpului, deoarece cronometrarea ar fi durat prea mult.
2. Până în 10^7, Heap Sort-ul este de 2-3 ori mai lent decât sortarea din STL, în schimb se observă cum la 10^8, Heap Sort-ul e de 4 ori mai lent ca STL-ul.
3. La rândul lui, până în 10^7, Merge Sort-ul este de 2-3 ori mai lent decât Heap Sort-ul, în schimb cei doi timpi se apropie pentru valori ale lui n mai mari (Test 8: Heap în 114, Merge în 118).

Idei generale ce reies din grafic și output.txt

4. Așa cum se aștepta, toate Radix Sort-urile sunt mai rapide decât sort-ul din STL, cu următoarele precizări: cea în baza 256 este cea mai rapidă, apoi cea în baza 1024, baza 2^{16} și apoi baza 10, care e cam de 3 ori mai lentă față de celelalte.

5. La Shell Sort-uri, se poate observa că Sedgewick și Ciura au timp asemănători pe toate testele, fiind chiar destul de apropiați de STL, cu un plus pentru Sedgewick pe testul mare, ceilalți fiind în ordinea următoare: Hibbard < Tokuda (foarte apropiați totuși) < Donald Shell

Idei generale ce reies din grafic și output.txt

6. Shell Sort-urile cu Sedgewick și Ciura se poate observa că sunt cam de 1.2 - 1.5 ori mai rapizi decât Heap Sort-ul, în timp ce restul Shell Sort-urile ori au timp apropiați cu Heap Sort-ul, ori sunt chiar puțin mai lenți decât acesta, în special pe testele mari.

7. După complexitate, sortările sunt sub următorul fel:

Radix Sort < STL < Shell Sort < Heap Sort < Merge Sort << Bubble Sort

Sfârșit!