Prezentare generală a sortărilor

RADIX SORT

- Este un algoritm de sortare de complexitate O(d*(n+b)), deoarece este format din câte un counting sort O(n+k) pentru fiecare cifră din număr.

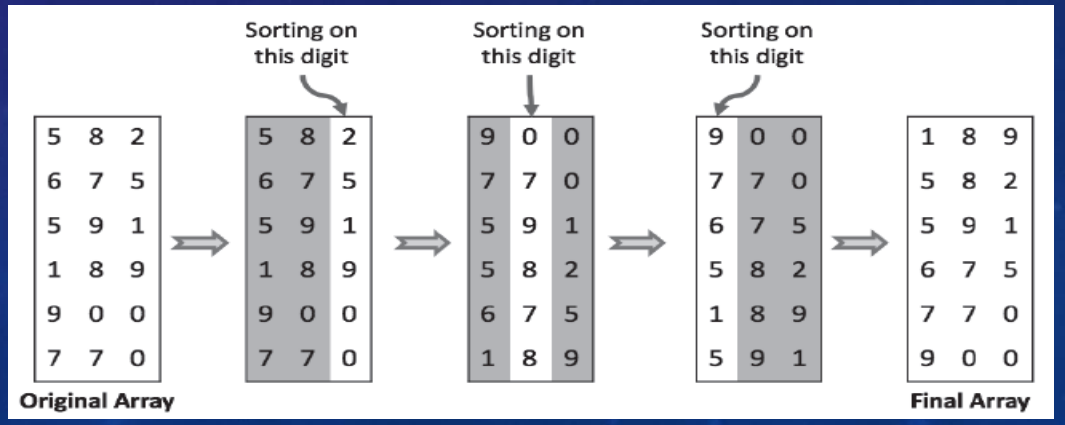
d = numărul de cifre;

n = numărul de elemente din listă;

b = baza în care facem sortarea.

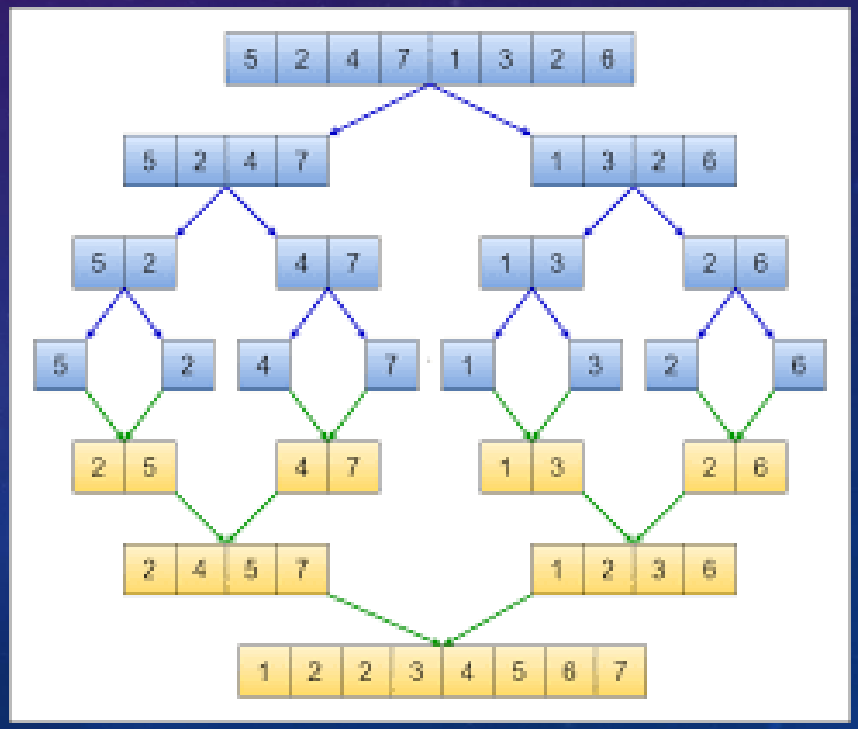
- Există două tipuri de Radix Sort:
  - LSD – sortarea începe de la Least Significant Digit;
  - MSD – sortarea începe de la Most Significant Digit.

Exemplu Radix Sort LSD:



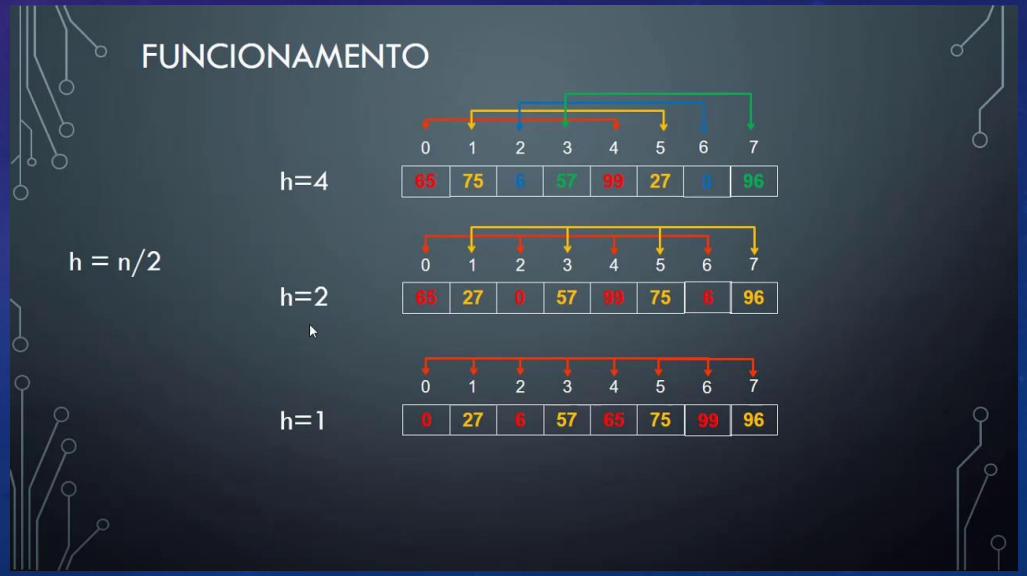
MERGE SORT

- Complexitatea acestui algoritm este O(n*logn).
- Sortarea listei de numere se realizează prin împărțirea ei repetată în vectori de lungime înjumătățită cât timp este posibil, iar la refacerea ei, de fiecare dată când combinăm doi sub-vectori într-un vector, o sa ținem cont și de mărimea elementelor.



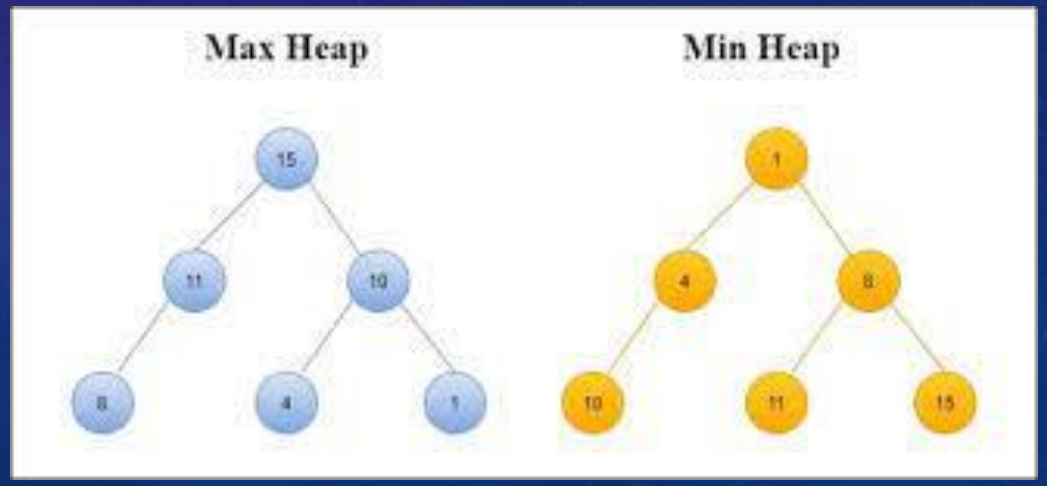
SHELL SORT

- Acesta este un algoritm de sortare instabil cu complexitatea O(n*logn) în cel mai bun caz și O(n^2) în cel mai rău caz.
- Functionarea algoritmului este asemănătoarea cu cea a unui Insertion Sort , diferența este că comparăm elementele de la o anumită distanță pe care o micșorăm după fiecare parcurgere. La momentul în care distanța devine 0, vectorul este sortat.



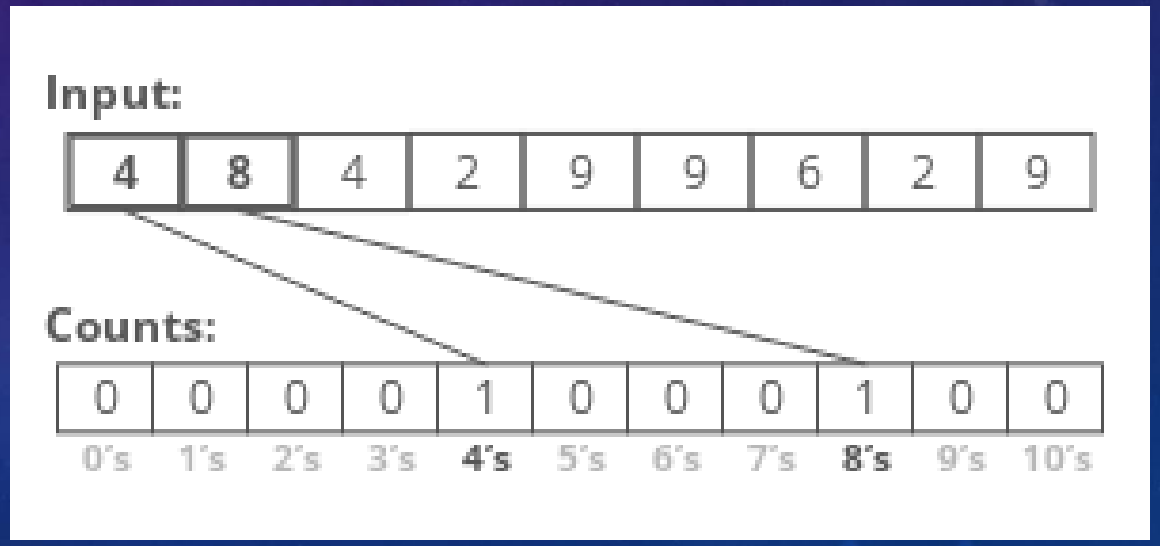
HEAP SORT

- Algoritmul este unul instabil și prezintă complexitate O(n*logn).
- Metoda presupune, așa cum îi sugerează și numele, utilizarea de heap-uri. Creăm un arbore binar, iar prin comparări îl transformăm într-un Max Heap, iar la final interschimbăm ultimul element din el cu primul, eliminând elementul maxim. Repetăm pașii până nu mai avem elemente de sortat. Algoritmul se poate realiza si cu Min Heap.



COUNTING SORT

- Algoritmul este unul stabil de complexitate O(n+k).
  - k = numărul maxim din vector.
- Sortarea se realizează cu ajutorul unui vector de aparițiii. Dezavantajul extrem al acestei metode, pe lângă faptul că ocupă multă memorie cu cât am elemente mai mari, este că nu pot sorta elemente mai mari de 10^9 deoarece ar trebui să folosesc un vector de lungime > 10^9.



TABELE ȘI COMPARAȚII PE DIFERITE TESTE

N M (RANDOM)	STL	MERGE	HEAP	SHELL	COUNTING	RADIX(2^3)	RADIX(10)	RADIX(2^16)
10^8 10^3	29.96050s	57.71300s	57.744s	39.5325s	0.74950s	12.37450s	9.48850s	3.511s
10^8 10^6	30.51225s	78.18525s	126.64525s	78.18525s	1.29200s	23.46625s	20.58225s	8.67425s
10^8 10^9	30.56725s	161.97000s	132.40275s	81.20000s	17.145s	37.02625s	33.64775s	9.7415s
10^7 10^3	2.74800s	5.44900s	5.40400s	3.43100s	0.06800s	1.22700s	0.94500s	0.34700s
10^7 10^9	2.67800s	5.76400s0s	8.29300s	5.96000s	17.50700s	3.67600s	3.33600s	0.90600s
10^7 10^12	2.67400s	5.74600s	8.08900s	5.93100s	--	5.48900s	4.73300s	1.30800s

N M (vector)	STL	MERGE	HEAP	SHELL	COUNTING	RADIX(2^3)	RADIX(10)	RADIX(2^16)
10^8 10^9 (crescator)	16.86400s	51.05600s	42.21800s	12.33900s	11.93100s	37.23800s	33.65700s	7.85600s
10^8 10^9 (descrescator)	18.83600s	50.23100s	42.63800s	16.04000s	7.62700s	37.33600s	33.85000s	7.92900s
10^8 10^9 (aproape nuli)	29.24200s	50.67400s	1.80800s	12.31000s	7.22800s	28.40200s	25.65000s	6.09300s
10^8 10^6 (descrescator)	19.72200s	51.93900s	43.72800s	15.92100s	0.72900s	24.28200s	21.28000s	6.95100s
10^8 10^4 (elem egale)	30.04000s	52.22900s	1.82200s	12.83000s	0.70300s	13.23400s	13.25300s	3.77500s

OBSERVATII:

- Valorile din tabel au fost realizate prin media mai multor rezultate de teste cu valori random.
- Un test de N=10^8 si M=10^6 random a rezultat valoarea 600s pentru Merge Sort , chiar dacă în circumstanțe normale pentru datele inserate are valoarea aprox 161s.
- Counting Sort-ul domină în testele cu multe numere, dar de dimensiuni mici.
- Există o diferență vizibilă între Radix Sort în diferite baze.
- STL functionează de 2 ori mai rapid pe vectori deja sortati.
- Merge si Heap functionează de 3 ori mai rapid pe vectori deja sortati.
- Shell functionează de 5 ori mai rapid pe vectori deja sortati.
- Nu se observă o diferență considerabilă dacă vectorii sunt sortati initial descrescători , comparativ când sunt inițial crescători.
- Heap Sort este cel mai rapid pe vectori în care se repetă de multe ori valorile.
- Compilatorul folosit este cel din Clion.