**Universitatea de Vest din Timișoara**

Facultatea de Matematică și Informatică

**Analiză teoretică și practică**

**Temă**:coparații între tipuri de sortari

Student

Tudor-Bogdan Stănescu

Cuprins:

I.Partea teoretica................................................................................... 2

a. Quicksort………………………………………………………....2

b. Merge sort………………………………………………………..3

c. Insertion sort……………………………………………………..4

d.Bubble sort………………………………………………………..5

e.Selection sort……………………………………………………...6

f.Shell sort…………………………………………………………..6

g.Shaker sort………………………………………………………..7

h.Radix sort…………………………………………………………8

II.Partea practica.................................................................................... 2

1.Generatorul de date………………………………………………10

2.Algoritmi de sortare

a. Quicksort……………………………………………………..11

b. Merge sort……………………………………………………15

c. Insertion sort………………………………………………….18

d.Bubble sort……………………………………………………20

e.Selection sort………………………………………………….22

f.Shell sort………………………………………………………25

g.Shaker sort……………………………………………………28

h.Radix sort…………………………………………………….30

III.Diagrame……..………………………………………………….34

IV.Bibliografie………………………………………………………36

I.Partea teoretica

1. Quicksort:

(Extras si rezumar din [1] și [2] - Bibliografie)

Cum funcționează:

1.Se alege un element al listei, denumit [pivot](https://ro.wikipedia.org/w/index.php?title=Pivot&action=edit&redlink=1)

2.Se reordonează lista astfel încât toate elementele mai mici decât pivotul să fie plasate înaintea pivotului și toate elementele mai mari să fie după pivot. După această partiționare, pivotul se află în poziția sa finală.

3.Se sortează [recursiv](https://ro.wikipedia.org/wiki/Recursivitate" \o "Recursivitate) sublista de elemente mai mici decât pivotul și sublista de elemente mai mari decât pivotul.

Algoritm implementat:

O imagine care conține text

Descriere generată automat

-complexitate : O(n \* ln n)

-În teorie acest algoritm e mai rapid decât ceilalți algoritmi de sortare de complexitate O(n\*logn) deoarece în majoritatea implementărilor practice,la proiectare, se pot lua decizii ce ajută la evitarea cazului în care algoritmul ajunge la complexitatea O(n\*n)

-Functionează mai bine pe liste mai mici

-Ineficient pentru liste mari

b. Merge sort:

(Extras si rezumar din [3] și [4] - Bibliografie)

Cum funcționează:

1.Dacă lista este de lungime 0 sau 1, atunci este deja sortată. Altfel:

2.Împarte lista nesortată în două subliste aproximativ egale.

3.Sortează fiecare sublistă [recursiv](https://ro.wikipedia.org/wiki/Recursivitate" \o "Recursivitate) prin reaplicarea algoritmului merge sort.

4.Se [interclasează](https://ro.wikipedia.org/w/index.php?title=Interclasare&action=edit&redlink=1" \o "Interclasare — pagină inexistentă) cele două liste și se obține lista inițială sortată.

Algoritm implementat:

O imagine care conține text

Descriere generată automat

-complexitate : O(n \* ln n)

-Functionează bine pe orice dimensiune de lista

c. Insertion sort:

(Extras si rezumar din [5] și [6] - Bibliografie)

Cum funcționează:

1.dacă secvența cu indici 0, 1, …, i-1 este ordonată, atunci putem insera

2. elementul X[i] în această secvență astfel încât să fie ordonată secvența cu indici 0, 1, …, i-1, i.

3.luăm pe rând fiecare element X[i] și îl inserăm în secvența din stânga sa

Algoritm implementat:

O imagine care conține text

Descriere generată automat

-complexitate : O(n \* n)

-eficient pe liste mici

d. Bubble sort:

(Extras si rezumar din [7] și [8] - Bibliografie)

Cum funcționează:

1.parcurgem vectorul și pentru oricare două elemente învecinate care nu sunt în ordinea dorită, le interschimbăm valorile

2.după o singură parcurgere, vectorul nu se va sorta, dar putem repeta parcurgerea

3.dacă la o parcurgere nu se face nicio interschimbare, vectorul este sortat

Algoritm implementat:

O imagine care conține text

Descriere generată automat

-complexitate: O(n \* n)

-algoritm foarte simplu

-lent

e. Selection sort:

(Extras si rezumar din [9] - Bibliografie)

Cum funcționează:

1.fie un vector X[] cu n elemente;

2.plasăm în X[0] cea mai mică valoare din vector;

3.plasăm în X[1] cea mai mică valoare rămasă;

4.etc.

Algoritm implementat:

O imagine care conține text

Descriere generată automat

-complexitate: O (n \* n)

-functionează bine pe liste mici

-nu necesita mult spatiu pentru sortare

f. Shell sort:

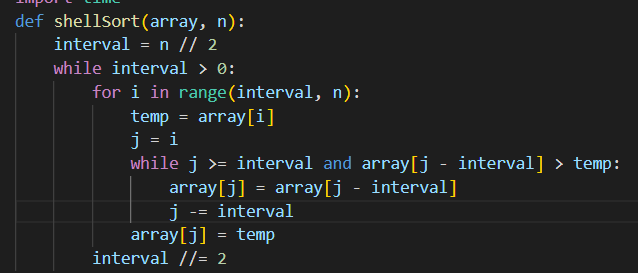
(Extras si rezumar din [10], [11] - Bibliografie)

Cum funcționează:

Sortarea Shell este adesea denumită o îmbunătățire față de sortarea prin inserție. În sortarea prin inserție, luăm creșteri cu 1 pentru a compara elementele și a le pune în poziția lor corectă.

În sortare shell, lista este sortată prin descompunerea ei într-un număr de subliste mai mici. Nu este necesar ca listele să conțină elemente contigue. În schimb, tehnica sortării shell utilizează incrementul i, care este, de asemenea, numit „gap” și îl folosește pentru a crea o listă de elemente care sunt „i” separate.

Algoritm implementat:



-complexitate: O (n \* n) (in cel mai rau caz)

-E mult mai eficient atunci când lista este aproape sortata

g. Shaker sort:

(Extras si rezumar din [12], [13] - Bibliografie)

Cum funcționează:

Cocktail Sort este o variantă de bubble sort. Bubble sort traversează întotdeauna elementele de la stânga și mută cel mai mare element în poziția corectă în prima iterație și al doilea cel mai mare în a doua iterație și așa mai departe. Cocktail Sort traversează o matrice dată în ambele direcții alternativ. Cocktail Sort nu trece prin iterația inutilă, făcându-l eficient pentru matrice mari.

Algoritm implementat:

O imagine care conține text

Descriere generată automat

-complexitate: O (n \* n)

-functionează pe princiul de la Bubble sort dar este mai eficient

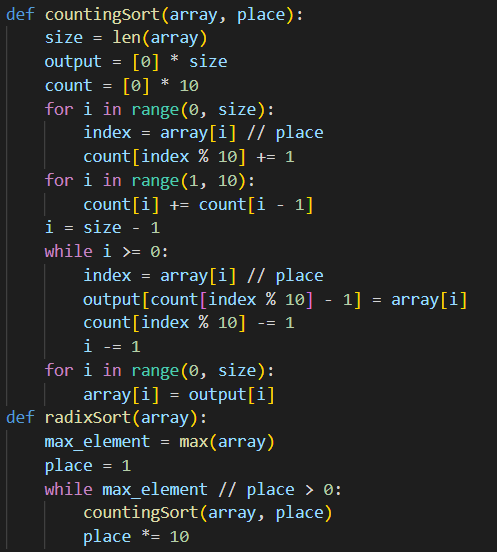
d.Radix sort

(Extras si rezumar din [14], [15] - Bibliografie)

Cum funcționează:

Algoritmul Radixsort tratează cheile ca numere reprezentate într-o bază *M*, pentru diferite valori ai lui *M*(radixul) și lucrează cu cifrele individuale ale numerelor.(De ex: caracterele pot fi reprezentate în bază 128) Astfel 10 grămezi corespunzând celor 10 cifre zecimale când M=10. Fiecare coloană este sortată începând cu rangul cel mai puțin semnificativ (prima din dreapta) al cheilor, mutând înregistrările din aria de intrare într-o arie auxiliară. Apoi se sortează după următorul rang semnificativ, până când pasul final stabilește toate înregistrările în ordinea dorită.

Algoritm implementat:



-complexitate: O (n \* n)(in cel mai rau caz)

-stabil

-functionează rapid atunci cand elementele listei sunt scurte si listele sunt relativ scurte

II.Partea practica

# **1.GENERATORUL DE DATE:**

Inainte de a începe orice teste ,pentru acestea să fie relevante , avem nevoie de date , multe si diferite ,astfel primul pas in realizarea acestui experiment este de a creea un “generator” de date.

In cazul nostru ,datele de care avem nevoie, sunt reprezentate de **liste** .O imagine care conține text

Descriere generată automat

(Atașat mai sus este codul de bază pentru generatorul de liste )

OBSERVATII:

1. Initial am incercat sa folosesc PyCharm pentru generarea de date insa am fost limitat la 89 mb .Am incercat sa maresc limita dar nu m-a ajutat cu mult deoarece ordata ce treceam de 89 mb aplicatia ingheta si trebuia restartata.
2. Asadar am ales sa folosec VsCode .Aici am reusit sa generez liste cu pana la 10^8 elemente .Am incercat sa mai generez si liste cu 10^9 elem insa aici m-am lovit din nou de pragul de memorie (MemoryError).
3. La lista de 10^7 elem am reusit sa fac cu elemente ce variaza intre -10^7 si 10^7 insa cand am inceput sa generez liste de 10^8 elem am observat ca acestea nu sunt constante (1 din 3 liste dadea eorare si era nevoie sa respornesc aplicatia cu totul)
4. Asadar voi folosi,pentru sortarile cu complexitatea o(n\*log(n)) liste de maxim 10^7 deoarece sunt cele mai mari liste stabile pentru PC ul meu

# **2.STOCAREA DATELOR:**

Pentru a usura procesul de utilizare a datelor am ales sa le sa stochez in fisiere de tip **JSON.**

De asemenea am stocat timpul de genereare a unei liste pentru a analiza ulterior daca un timp mai ridicat de generare a unei liste afecteaza in vreun fel sortarea unei liste insa rezultatele au fost prea variate si nu au dus nicaieri.

# **3.ALGORITMI DE SORTARE:**

1. Quicksort - **O(n\*log(n))**

**Setul 1 de date : Timp total stortare liste:**

**-nr liste :1000** 

**-marile liste:1000**

**-range elemente: [-1000,1000]**

**Timp mediu de sortare: 0.0009970664978027344**

**Setul 2 de date: Timp total stortare liste:**

**-nr liste :1000** 

**-marime liste:2000**

**-range elemente: [-1000,1000]**

**Timp mediu de sortare: 0.001987457275390625**

**Setul 3 de date : Timp total stortare liste:**

**-nr liste :1000**  

**-marime liste:4000**

**-range elemente: [-1000,1000]**

**Timp mediu de sortare: 0.006982088088989258**

**Setul 4 de date: Timp total stortare liste:**

**-nr liste :1000**  

**-marime liste:8000**

**-range elemente: [-1000,1000]**

**Timp mediu de sortare:** **0.013962030410766602**

**Setul 5 de date: Timp total stortare liste:**

**-nr liste :1000**  

**-marime liste:10000**

**-range elemente: [-1000,1000]**

**Timp mediu de sortare: 0.01795172691345215**

**Setul 6 date: Timp total stortare liste:**

**-nr liste :1000**  

**-marime liste:50000**

**-range elemente: [-1000,1000]**

**Timp mediu de sortare:** **0.14760994911193848**

**Setul 7 de date: Timp total stortare liste:**

**-nr liste :1000**  

**-marime liste:100000**

**-range elemente: [-1000,1000]**

**Timp mediu de sortare:** **0.42987895011901855**

**Setul 8 de date: Timp total stortare liste:**

**-nr liste :250**  

**-marime liste:1000000**

**-range elemente: [-1000,1000]**

**Timp mediu de sortare:** **29.052233457565308**

**Setul 9 de date:**

**-nr liste :10**

**-marime liste:2000000**

**-range elemente: [-1000,1000]**

O imagine care conține text

Descriere generată automat

1. Merge sort - **O(n\*log(n))**

**Setul 1 date: Timp total stortare liste:**

**-nr liste :1000** 

**-marile liste:1000**

**-range elemente: [-1000,1000]**

**Timp mediu de sortare:** **0.0009970664978027344**

**Setul 2 de date: Timp total stortare liste:**

**-nr liste :1000** 

**-marime liste:2000**

**-range elemente: [-1000,1000]**

**Timp mediu de sortare:** **0.003989458084106445**

**Setul 3 de date: Timp total stortare liste:**

**-nr liste :1000**  

**-marime liste:4000**

**-range elemente: [-1000,1000]**

**Timp mediu de sortare:** **0.009972572326660156**

**Setul 4 de date: Timp total stortare liste:**

**-nr liste :1000**  

**-marime liste:8000**

**-range elemente: [-1000,1000]**

**Timp mediu de sortare:** **0.020943880081176758**

**Setul 5 de date: Timp total stortare liste:**

**-nr liste :1000**  

**-marime liste:10000**

**-range elemente: [-1000,1000]**

**Timp mediu de sortare: 0.02717447280883789**

**Setul 6 de date: Timp total stortare liste:**

**-nr liste :1000**  

**-marime liste:50000**

**-range elemente: [-1000,1000]**

**Timp mediu de sortare:** **0.14461350440979004**

**Setul 7 de date: Timp total stortare liste:**

**-nr liste :1000**  

**-marime liste:100000**

**-range elemente: [-1000,1000]**

**Timp mediu de sortare:** **0.33107566833496094**

**Setul 8 de date: Timp total stortare liste:**

**-nr liste :250**  

**-marime liste:1000000**

**-range elemente: [-1000,1000]**

**Timp mediu de sortare:** **4.182673692703247**

**Setul 9 de date: Timp total stortare liste:**

**-nr liste :10**  

**-marime liste:10000000**

**-range elemente: [-1000,1000]**

**Timp mediu de sortare:** **46.198055028915405**

1. Insertion sort - **O(n²)**

**Setul 1 de date: Timp total stortare liste:**

**-nr liste :1000**

**-marime liste:1000**

**-range elemente: [-1000,1000]**

**O imagine care conține text

Descriere generată automat**

**Timp mediu de sortare: 0,02062394285**

**Setul 2 de date : Timp total stortare liste:**

**-nr liste :1000** ****

**-marime liste:2000**

**-range elemente: [-1000,1000]**

**Timp mediu de sortare: 0,0861537377834**

**Setul 3 de date : Timp total stortare liste:**

**-nr liste :1000**  ****

**-marime liste:4000**

**-range elemente: [-1000,1000]**

**Timp mediu de sortare: 0,3816633183956146**

**Setul 4 de date: Timp total stortare liste:**

**-nr liste :500**  ****

**-marime liste:8000**

**-range elemente: [-1000,1000]**

**Timp mediu de sortare: 1.492495691776276**

**Setul 5 de date: Timp total stortare liste:**

**-nr liste :250**  

**-marime liste:10000**

**-range elemente: [-1000,1000]**

**Timp mediu de sortare: 2.561999844551086**

**Setul 6 de date: Timp total stortare liste:**

**-nr liste :50**  ****

**-marime liste:50000**

**-range elemente: [-1000,1000]**

**Timp mediu de sortare:54.87652492523**

**Setul 7 de date: Timp total stortare liste:**

**-nr liste :10**  

**-marime liste:100000**

**-range elemente: [-1000,1000]**

**Timp mediu de sortare:** **222.5555636882782**

d. Bubble sort - **O(n²)**

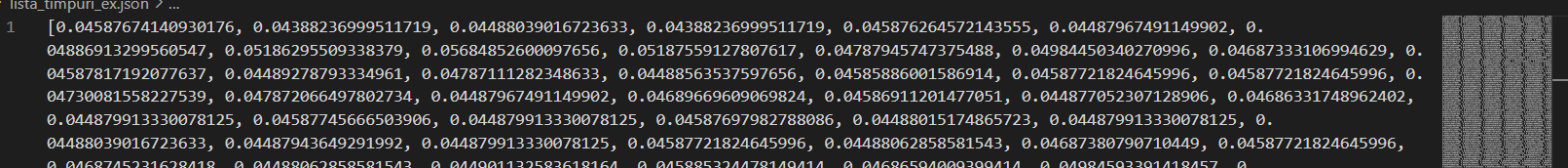
**Setul 1 de date: Timp total stortare liste:**

**O imagine care conține text

Descriere generată automat-nr liste :1000**

**-marme liste:1000**

**-range elemente: [-1000,1000]**

****

**Timp mediu de sortare: 0,0495765049457550**

**Setul 2 de date: Timp total stortare liste:**

**-nr liste :1000** ****

**-marime liste:2000**

**-range elemente: [-1000,1000]**

**Timp mediu de sortare: 0,** **2047689635753631**

**Setul 3 de date : Timp total stortare liste:**

**-nr liste :1000**  

**-marime liste:4000**

**-range elemente: [-1000,1000]**

**Timp mediu de sortare: 0.8516645431518555**

**Setul 4 de date: Timp total stortare liste:**

**-nr liste :500**  

**-marime liste:8000**

**-range elemente: [-1000,1000]**

**Timp mediu de sortare: 3.4119935035705566**

**Setul 5 de date: Timp total stortare liste:**

**-nr liste :250**  

**-marime liste:10000**

**-range elemente: [-1000,1000]**

**Timp mediu de sortare: 5.1487603187561035**

**Setul 6 de date: Timp total stortare liste:**

**-nr liste :50**  

**-marime liste:50000**

**-range elemente: [-1000,1000]**

**Timp mediu de sortare:** **128.90262389183**

**Setul 7 de date: Timp total stortare liste:**

**-nr liste :10**  

**-marime liste:100000**

**-range elemente: [-1000,1000]**

**Timp mediu de sortare:** **598.0480008125305**

e. Selection sort - **O(n²)**

**Setul 1 de date: Timp total stortare liste:**

**-nr liste :1000**

**-marile liste:1000**

**-range elemente: [-1000,1000]**

**O imagine care conține text

Descriere generată automat**

**Timp mediu de sortare: 0,020606288433074**

**Setul 2 de date : Timp total stortare liste:**

**-nr liste :1000** 

**-marime liste:2000**

**-range elemente: [-1000,1000]**

**Timp mediu de sortare: 0, 0** **941766142845153**

**Setul 3 de date : Timp total stortare liste:**

**-nr liste :1000**  

**-marime liste:4000**

**-range elemente: [-1000,1000]**

**Timp mediu de sortare: 0.3311142921447754**

**Setul 4 de date: Timp total stortare liste:**

**-nr liste :500**  

**-marime liste:8000**

**-range elemente: [-1000,1000]**

**Timp mediu de sortare: 1.3825514316558838**

**Setul 5 de date: Timp total stortare liste:**

**-nr liste :250**  

**-marime liste:10000**

**-range elemente: [-1000,1000]**

**Timp mediu de sortare: 2.406078338623047**

**Setul 6 de date: Timp total stortare liste:**

**-nr liste :50**  

**-marime liste:50000**

**-range elemente: [-1000,1000]**

**Timp mediu de sortare:** **53.99167323112488**

**Setul 7 de date: Timp total stortare liste:**

**-nr liste :10**  

**-marime liste:100000**

**-range elemente: [-1000,1000]**

**Timp mediu de sortare:** **235.62679266929626**

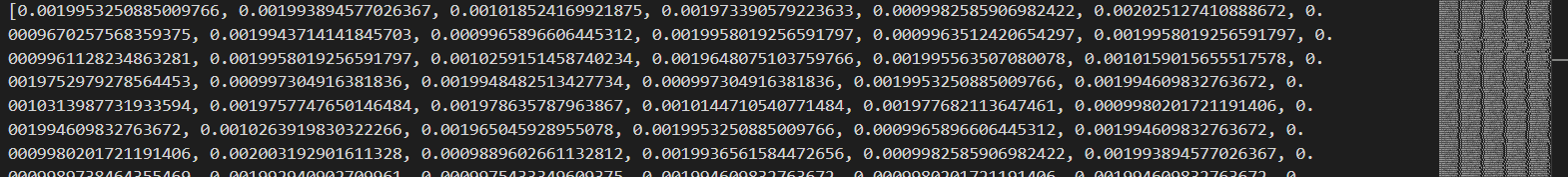
f. Shell sort - **O(n²)**

**Setul 1 de date : Timp total stortare liste:**

**-nr liste :1000**

**-marile liste:1000**

**-range elemente: [-1000,1000]**

****

**Timp mediu de sortare: 0,0015507733821868**

**Setul 2 de date : Timp total stortare liste:**

**-nr liste :1000** 

**-marime liste:2000**

**-range elemente: [-1000,1000]**

**Timp mediu de sortare: 0, 0036688454151153564**

**Setul 3 de date : Timp total stortare liste:**

**-nr liste :1000**  

**-marime liste:4000**

**-range elemente: [-1000,1000]**

**Timp mediu de sortare: 0.007978677749633789**

**Setul 4 de date: Timp total stortare liste:**

**-nr liste :1000**  

**-marime liste:8000**

**-range elemente: [-1000,1000]**

**Timp mediu de sortare: 0.020961523056030273**

**Setul 5 de date: Timp total stortare liste:**

**-nr liste :1000**  

**-marime liste:10000**

**-range elemente: [-1000,1000]**

**Timp mediu de sortare: 0.027927398681640625**

**Setul 6 de date: Timp total stortare liste:**

**-nr liste :1000**  

**-marime liste:50000**

**-range elemente: [-1000,1000]**

**Timp mediu de sortare:** **0.1685490608215332**

**Setul 7 de date: Timp total stortare liste:**

**-nr liste :1000**  

**-marime liste:100000**

**-range elemente: [-1000,1000]**

**Timp mediu de sortare:** **0.3999481201171875**

**Setul 8 de date: Timp total stortare liste:**

**-nr liste :250**  

**-marime liste:1000000**

**-range elemente: [-1000,1000]**

**Timp mediu de sortare:** **6.9294679164886475**

**Setul 9 de date: Timp total stortare liste:**

**-nr liste :10**  

**-marime liste:10000000**

**-range elemente: [-1000,1000]**

**Timp mediu de sortare:** **94.54498767852783**

g. Shaker sort

**Setul 1 de date: Timp total stortare liste:**

**-nr liste :1000**

**-marile liste:1000**

**-range elemente: [-1000,1000]**

**Timp mediu de sortare: 0,043226726293563**

**Setul 2 de date : Timp total stortare liste:**

**-nr liste :1000** 

**-marime liste:2000**

**-range elemente: [-1000,1000]**

**Timp mediu de sortare: 0,17103596830368**

**Setul 3 de date : Timp total stortare liste:**

**-nr liste :1000**  

**-marime liste:4000**

**-range elemente: [-1000,1000]**

**Timp mediu de sortare: 0.7260582447052002**

**Setul 4 de date: Timp total stortare liste:**

**-nr liste :500**  

**-marime liste:8000**

**-range elemente: [-1000,1000]**

**Timp mediu de sortare: 2.9619617462158203**

**Setul 5 de date: Timp total stortare liste:**

**-nr liste :250**  

**-marime liste:10000**

**-range elemente: [-1000,1000]**

**Timp mediu de sortare:** **4.458128452301025**

**Setul 6 de date: Timp total stortare liste:**

**-nr liste :50**  

**-marime liste:50000**

**-range elemente: [-1000,1000]**

**Timp mediu de sortare:** **121.556556224823**

**Setul 7 de date: Timp total stortare liste:**

**-nr liste :10**  

**-marime liste:100000**

**-range elemente: [-1000,1000]**

**Timp mediu de sortare:** **462.77883982658386**

h. Radix sort

**Setul 1 de date: Timp total stortare liste:**

**-nr liste :1000**

**-marile liste:1000**

**-range elemente: [-1000,1000]**

**Timp mediu de sortare: 0,0012506544589996**

**Setul 2 set de date : Timp total stortare liste:**

**-nr liste :1000** 

**-marime liste:2000**

**-range elemente: [-1000,1000]**

**Timp mediu de sortare: 0,002524934768676758**

**Setul 3 de date: Timp total stortare liste:**

**-nr liste :1000**  

**-marime liste:4000**

**-range elemente: [-1000,1000]**

**Timp mediu de sortare: 0.0049855709075927734**

**Setul 4 de date: Timp total stortare liste:**

**-nr liste :1000**  

**-marime liste:8000**

**-range elemente: [-1000,1000]**

**Timp mediu de sortare: 0.009974479675292969**

**Setul 5 de date: Timp total stortare liste:**

**-nr liste :1000**  

**-marime liste:10000**

**-range elemente: [-1000,1000]**

**Timp mediu de sortare: 0.012964963912963867**

**Setul 6 de date: Timp total stortare liste:**

**-nr liste :1000**  

**-marime liste:50000**

**-range elemente: [-1000,1000]**

**Timp mediu de sortare:** **0.06582355499267578**

**Setul 7 de date: Timp total stortare liste:**

**-nr liste :1000**  

**-marime liste:100000**

**-range elemente: [-1000,1000]**

**Timp mediu de sortare:** **0.1296687126159668**

**Setul 8 de date: Timp total stortare liste:**

**-nr liste :250**  

**-marime liste:1000000**

**-range elemente: [-1000,1000]**

**Timp mediu de sortare:** **2.007628917694092**

**Setul 9 de date: Timp total stortare liste:**

**-nr liste :10**  

**-marime liste:10000000**

**-range elemente: [-1000,1000]**

**Timp mediu de sortare:** **18.712099313735962**

III.Diagrame

VI.Bibliografie:

[1] https://en.wikipedia.org/wiki/Quicksort#cite\_note-alg64-2

# [2] **Tony Hoare**(1961). "Algorithm 64: Quicksort".  [doi](https://en.wikipedia.org/wiki/Doi_(identifier)):[10.1145/366622.366644](https://doi.org/10.1145%2F366622.366644).

[3] https://en.wikipedia.org/wiki/Merge\_sort#cite\_note-3

[4] "Algorithms and Complexity". *Proceedings of the 3rd Italian Conference on Algorithms and Complexity*. Italian Conference on Algorithms and Complexity. Lecture Notes in Computer Science. Vol. 1203. Rome. pp. 217–228. [CiteSeerX](https://en.wikipedia.org/wiki/CiteSeerX_(identifier)) [10.1.1.86.3154](https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.86.3154). [doi](https://en.wikipedia.org/wiki/Doi_(identifier)):[10.1007/3-540-62592-5\_74](https://doi.org/10.1007%2F3-540-62592-5_74). [ISBN](https://en.wikipedia.org/wiki/ISBN_(identifier)) [978-3-540-62592-6](https://en.wikipedia.org/wiki/Special:BookSources/978-3-540-62592-6).

[5] Bentley, Jon (2000). ["Column 11: Sorting"](https://books.google.com/books?id=kse_7qbWbjsC&pg=PA116). *Programming Pearls* (2nd ed.). ACM Press / Addison-Wesley. pp. 115–116. [ISBN](https://en.wikipedia.org/wiki/ISBN_(identifier)) [978-0-201-65788-3](https://en.wikipedia.org/wiki/Special:BookSources/978-0-201-65788-3). [OCLC](https://en.wikipedia.org/wiki/OCLC_(identifier)) [1047840657](https://www.worldcat.org/oclc/1047840657).

[6] https://en.wikipedia.org/wiki/Insertion\_sort#cite\_note-pearls-1

[7] https://en.wikipedia.org/wiki/Bubble\_sort

[8] Iverson, K: A Programming Language. John Wiley, 1962.

[9] https://en.wikipedia.org/wiki/Selection\_sort

[10] <https://en.wikipedia.org/wiki/Shellsort>

[11] Shell, D. L. (1959). ["A High-Speed Sorting Procedure"](http://penguin.ewu.edu/cscd300/Topic/AdvSorting/p30-shell.pdf) (PDF). *Communications of the ACM*. **2** (7): 30–32. [doi](https://en.wikipedia.org/wiki/Doi_(identifier)):[10.1145/368370.368387](https://doi.org/10.1145%2F368370.368387). [S2CID](https://en.wikipedia.org/wiki/S2CID_(identifier)) [28572656](https://api.semanticscholar.org/CorpusID:28572656).

[12] <https://en.wikipedia.org/wiki/Cocktail_shaker_sort>

[13] Professor Jerry Thomas

[14] https://en.wikipedia.org/wiki/Radix\_sort

[15] [Donald Knuth](https://en.wikipedia.org/wiki/Donald_Knuth). *The Art of Computer Programming*, Volume 3: *Sorting and Searching*, Third Edition. Addison-Wesley, 1997. [ISBN](https://en.wikipedia.org/wiki/ISBN_(identifier)) [0-201-89685-0](https://en.wikipedia.org/wiki/Special:BookSources/0-201-89685-0). Section 5.2.5: Sorting by Distribution, pp. 168–179.