UNIVERSITATEA DE STAT DIN MOLDOVA
FACULTATEA DE MATEMATICĂ ȘI INFORMATICĂ
SPECIALITATEA INFORMATICA

RAPORT

la disciplina „Algoritme, Structuri de Date și Complexitate”

Lucrarea de laborator nr. 2: Metodele de sortare

Conducătorul științific: Novac Ludmila, dr. conf. univ.
Autor: Cemîrtan Cristian, student din grupa I2101

CHIȘINĂU – 2023

Cuprins

Preliminări

Pentru a testa metodele de sortare, am specificat că fișierul .csv, din lucrarea precedentă, să conțină 10100 de înregistrări. Înainte de a rula o metodă de sortare, mixez ordinile rândurilor din tabel în așa fel ca ele să nu fie ordonate crescător.

În primul capitol, comparația între rânduri se bazează pe prima coloană (ID) a tabelei.

Rularea metodelor de sortare

Metoda inserției.

Este evident că această metodă este ideală pentru un număr mic de înregistrări, dar progresiv își pierde proprietatea odată cu incrementarea numărului de înregistrări.

```
Cazurile complexitatii temporale -- Bun: 10100.0000; Mediu: 102010000.0000; Rau: 102010000.0000
Nr. de comparari: 24806001
Nr. de interschimbari: 24795904
Timpul de executie: 1138.00 ms
```

Figura 1

Metoda lui Shell.

Donald Shell a venit cu o propunere să optimizeze metoda inserției prin introducerea secvențelor de salturi în algoritm.

Pentru secvența de salt, am utilizat versiunea lui Marcin Ciura: 1, 4, 10, 23, 57, 132, 301, 701. Această secvență a fost cercetată în mod empiric, conform autorului. Observăm că, în comparație cu metoda inserției, avem circa 5.000.000 mai puține comparări și interschimbări, și timpul de execuție înjumătățit.

```
Cazurile complexitatii temporale -- Bun: 10100.0000; Mediu: 1787144.5441; Rau: 1787144.5441
Nr. de comparari: 19723189
Nr. de interschimbari: 19708822
Timpul de executie: 434.00 ms
```

Figura 2

Metoda selecției.

Numărul de comparări este dublat față de metoda inserției, dar numărul de interschimbări este mai puțin cu circa 99,96%. Timpul de execuție este dublat în comparație cu metoda lui Shell.

```
Cazurile complexitatii temporale -- Bun: 102010000.0000; Mediu: 102010000.0000; Rau: 102010000.0000
Nr. de comparari: 50999950
Nr. de interschimbari: 10099
Timpul de executie: 941.00 ms
```

Figura 3

Metoda Heap.

Observăm că complexitățile temporale sunt identice pentru orice caz, deci timpul de execuție teoretic depinde numai de numărul de elemente din tablou, nu de modul cum sunt mixate ordinele elementelor.

Ca timpul de execuție să fie ideal în mod practic, am implementat această metodă să fie realizată deodată pe tablou, fără să generez temporar listele înlănțuite pentru un arbore Heap.

În general, metoda Heap are o performanță excelentă în comparație cu cele precedente.

```
Cazurile complexitatii temporale -- Bun: 134350.8835; Mediu: 134350.8835; Rau: 134350.8835
Nr. de comparari: 238188
Nr. de interschimbari: 125816
Timpul de executie: 8.00 ms
```

Figura 4

Metoda rapidă.

Denumirea sa vorbește de la sine.

În implementarea mea, am specificat că pivotul utilizat pentru partiționare va fi primul element. În comparație cu metoda Heap, timpul de execuție este înjumătățit, indiferent faptului că are complexitatea temporală pătratică în cel mai rău caz.

```
Cazurile complexitatii temporale -- Bun: 134350.8835; Mediu: 134350.8835; Rau: 102010000.0000
Nr. de comparari: 166101
Nr. de interschimbari: 53570
Timpul de executie: 4.00 ms
```

Figura 5

Metoda introspectivă.

Această metodă este hibridă (metoda inserției, Heap și rapidă), și ca rezultat - complexitatea temporală în cel mai rău caz s-a redus. Numărul de comparări și de interschimbări sunt mai mari decât cele prevăzute în metoda rapidă, dar timpul de execuție practic nu se deosebește.

```
Cazurile complexitatii temporale -- Bun: 134350.8835; Mediu: 134350.8835; Rau: 134350.8835
Nr. de comparari: 171457
Nr. de interschimbari: 66232
Timpul de executie: 4.00 ms
```

Figura 6

Metoda interclasării.

În comparație cu metoda introspectivă, complexitățile temporale sunt identice în orice caz, dar timpul de execuție s-a crescut cu 3 milisecunde. Această încetinire posibil s-a datorat alocării continue a zonei de memorie temporare utilizate pentru a interclasa elementele sortate din două subspații într-o una singură.

```
Cazurile complexitatii temporale -- Bun: 134350.8835; Mediu: 134350.8835; Rau: 134350.8835
Nr. de comparari: 121718
Nr. de interschimbari: 128925
Timpul de executie: 7.00 ms
```

Figura 7

Metoda lui Tim.

Această metodă de sortare a fost elaborată inițial pentru limbajul de programare Python. Complexitatea sa temporală în cel mai bun caz este liniar, înseamnă că teoretic trebuie să fie mai performant decât „metoda rapidă” de sortare.

Metoda lui Tim este un hibrid între metoda inserției și cea de interclasare. În deosebire de metoda de interclasare, timpul de execuție este mai optim. Acest fapt se explică prin: se sortează subspațiile cu lungimea mai mică decât 32 cu metoda inserției, și se interclasează toate subspațiile într-un singur spațiu sortat.

Numărul de comparări și de interschimbări sunt mai mari decât cele prevăzute în metoda interclasării.

```
Cazurile complexitatii temporale -- Bun: 10100.0000; Mediu: 134350.8835; Rau: 134350.8835
Nr. de comparari: 173752
Nr. de interschimbari: 165554
Timpul de executie: 5.00 ms
```

Figura 8

Metoda bazată pe bazele numerației.

Numărul de comparări este 0, și cel de interschimbări este minim comparat cu metodele precedente de sortare. Având complexitatea temporală liniară în orice caz, timpul de execuție tot nu este ideală (16 ms > 4 ms). Această încetinire posibil se datorează faptului că am utilizat listele înlănțuite pentru a implementa acest algoritm.

```
Baza numeratiei = 16
Cazurile complexitatii temporale -- Bun: 161600.0000; Mediu: 161600.0000; Rau: 161600.0000
Nr. de comparari: 0
Nr. de interschimbari: 40400
```

```
Timpul de executie: 16.00 ms
```

Figura 9

Metoda sertarelor (pigeonhole).

Această metodă tot nu se bazează pe comparări, dar este cea mai rapidă între toate metodele de sortare enumerate *(adio „sortarea rapidă”?).* Complexitățile temporale în orice caz sunt pur liniare. În schimb, avem consumul de memorie operativă mai înaltă.

```
Cazurile complexitatii temporale -- Bun: 10100.0000; Mediu: 10100.0000; Rau: 10100.0000
Nr. de comparari: 20188
Nr. de interschimbari: 10100
Timpul de executie: 4.00 ms
```

Figura 10

Metoda bulelor.

Cea mai simplă metodă de sortare, dar în schimb este cel mai lent între toate metodele de sortare.

```
Cazurile complexitatii temporale -- Bun: 10100.0000; Mediu: 102010000.0000; Rau:
102010000.0000
Nr. de comparari: 50996034
Nr. de interschimbari: 24795904
Timpul de executie: 1387.00 ms
```

Figura 11

Metoda prin paritate (odd/even).

Numărul de comparări și de interschimbări sunt similare cu metoda bulelor, dar timpul de execuție este mai puțin cu 89 ms datorită paralelizării metodei. Oricum, tot este lentă.

```
Cazurile complexitatii temporale -- Bun: 10100.0000; Mediu: 102010000.0000; Rau:
102010000.0000
Nr. de comparari: 50555594
Nr. de interschimbari: 24795904
Timpul de executie: 1298.00 ms
```

Figura 12

Complexitățile temporare pentru fiecare metodă de sortare

Tabela 1

Metoda	Cel mai bun caz	Caz mediu	Cel mai rău caz
Inserției	n	n^2	n^2
Shell	n	$n \log^2 n$	$n \log^2 n$
Selecției	n^2	n^2	n^2
Heap	$n \log n$	$n \log n$	$n \log n$
Rapidă	$n \log n$	$n \log n$	n^2
Introspectivă	$n \log n$	$n \log n$	$n \log n$
Interclasării	$n \log n$	$n \log n$	$n \log n$
Tim	n	$n \log n$	$n \log n$
Radix	$n+k$	$n+k$	$n+k$
Sertarelor	n	n	n
Bulelor	n	n^2	n^2
Parității	n	n^2	n^2

Unde k reprezintă baza numerației, introdusă de utilizator.

Stabilitatea fiecărei metode de sortare

În contextul programului nostru, stabilitatea înseamnă că ordinea relativă a elementelor cu aceeași coloană de sortat, rămâne nemodificată.

În program, rulez fiecare metodă de sortare cu comparatorul bazat pe a doua coloană (primul nume):

Metoda inserției

```
Cazurile complexitatii temporale -- Bun: 10100.0000; Mediu: 102010000.0000; Rau:
102010000.0000
Nr. de comparari: 25520632
Nr. de interschimbari: 25510539
Timpul de executie: 709.00 ms
```

Sortat: da; Stabil: da

Metoda lui Shell

```
Cazurile complexitatii temporale -- Bun: 10100.0000; Mediu: 1787144.5441; Rau:
1787144.5441
Nr. de comparari: 20261159
Nr. de interschimbari: 20246803
Timpul de executie: 536.00 ms
```

Sortat: da; Stabil: nu

Metoda selecției

```
Cazurile complexitatii temporale -- Bun: 102010000.0000; Mediu: 102010000.0000; Rau:
102010000.0000
Nr. de comparari: 50999950
Nr. de interschimbari: 10099
Timpul de executie: 1064.00 ms
```

Sortat: da; Stabil: nu

Metoda Heap

```
Cazurile complexitatii temporale -- Bun: 134350.8835; Mediu: 134350.8835; Rau:
134350.8835
Nr. de comparari: 237882
Nr. de interschimbari: 125500
Timpul de executie: 10.00 ms
```

Sortat: da; Stabil: nu

Metoda rapidă

```
Cazurile complexitatii temporale -- Bun: 134350.8835; Mediu: 134350.8835; Rau:
102010000.0000
Nr. de comparari: 157747
Nr. de interschimbari: 51401
Timpul de executie: 5.00 ms
```

Sortat: da; **Stabil: nu**

Metoda introspectivă

```
Cazurile complexitatii temporale -- Bun: 134350.8835; Mediu: 134350.8835; Rau:
134350.8835
Nr. de comparari: 159764
Nr. de interschimbari: 63789
Timpul de executie: 5.00 ms

Sortat: da; Stabil: nu
```

Metoda interclasării

```
Cazurile complexitatii temporale -- Bun: 134350.8835; Mediu: 134350.8835; Rau:
134350.8835
Nr. de comparari: 121711
Nr. de interschimbari: 128982
Timpul de executie: 9.00 ms

Sortat: da; Stabil: da
```

Metoda lui Tim

```
Cazurile complexitatii temporale -- Bun: 10100.0000; Mediu: 134350.8835; Rau:
134350.8835
Nr. de comparari: 173399
Nr. de interschimbari: 165236
Timpul de executie: 6.00 ms

Sortat: da; Stabil: da
```

Metoda bulelor

```
Cazurile complexitatii temporale -- Bun: 10100.0000; Mediu: 102010000.0000; Rau:
102010000.0000
Nr. de comparari: 50992200
Nr. de interschimbari: 25510539
Timpul de executie: 1902.00 ms

Sortat: da; Stabil: da
```

Metoda parității

```
Cazurile complexitatii temporale -- Bun: 10100.0000; Mediu: 102010000.0000; Rau:
102010000.0000
Nr. de comparari: 50383911
Nr. de interschimbari: 25510539
Timpul de executie: 2039.00 ms

Sortat: da; Stabil: da
```

Concluzii

Conform experimentelor realizate, pot să concluzionez că următorii doi algoritmi sunt ideali pentru sortarea cantităților mari de date: metoda interclasării și cea a lui Tim. Acești doi algoritmi sunt „câștigătorii” capitolului precedent, luând în considerare timpul de execuție și stabilitatea lor. Dacă stabilitatea elementelor nu importă, atunci metoda rapidă și cea introspectivă sunt într-adevăr cele mai rapide.

Anexe

Se anexează fișierul .csv și codurile sursă elaborate în limbajul de programare C, în care sunt implementate metodele de sortare cerute în sarcină.