**ЛАБОРАТОРНА РОБОТА № 5**

**Методи сортування**

***Мета:*** реалізація алгоритмів сортування та дослідження їх характеристик (швидкодія, необхідний об'єм пам'яті, галузь застосування тощо.)

**Зміст роботи**

Реалізував алгоритми сортування вибором, вставками (масив та список), Шелла (варіант Р. Седжвіка).

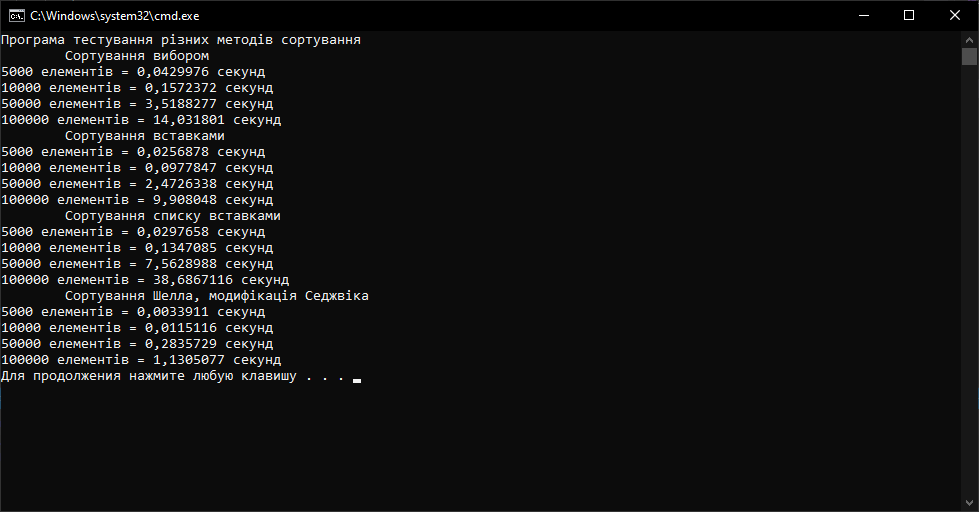


Рис.5.1 – результат виконання роботи (без виводу масивів)

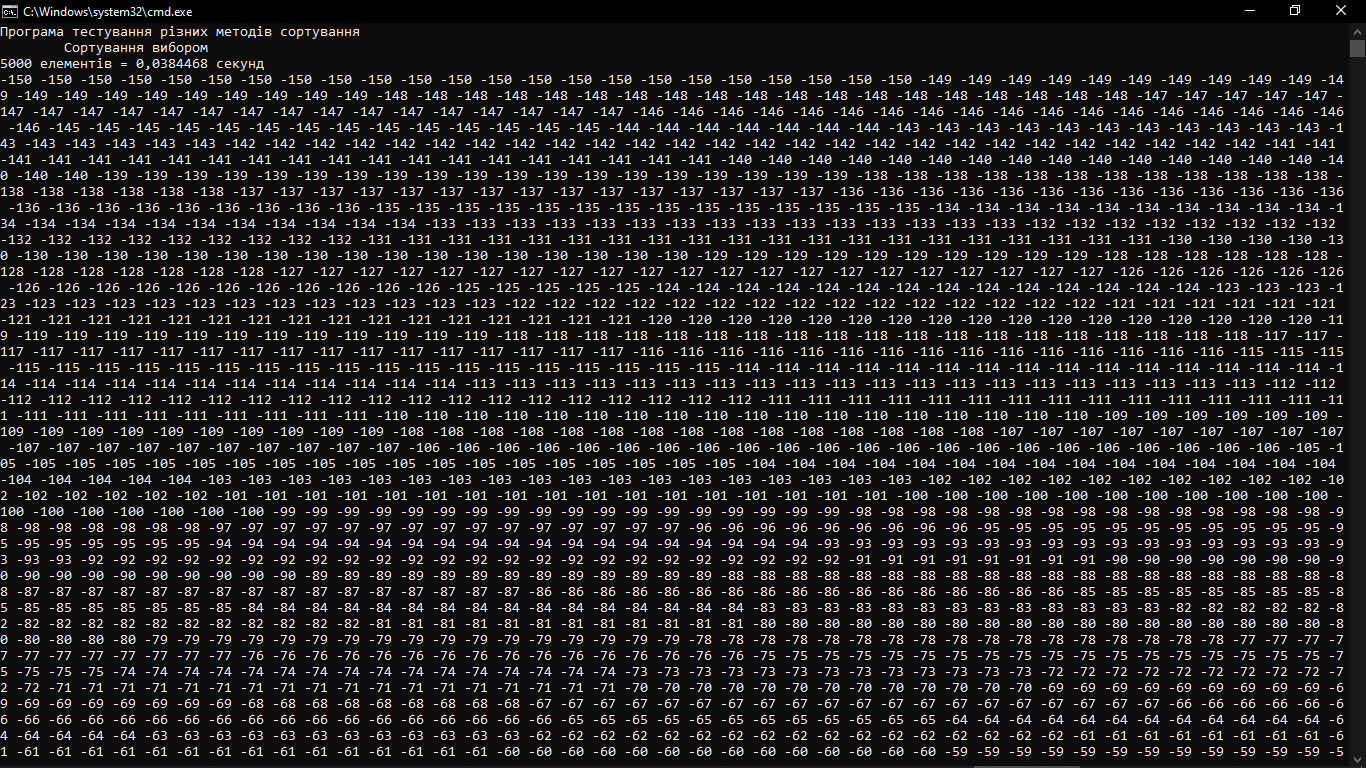


Рис.5.2 – приклад відсортованого масиву методом вибору

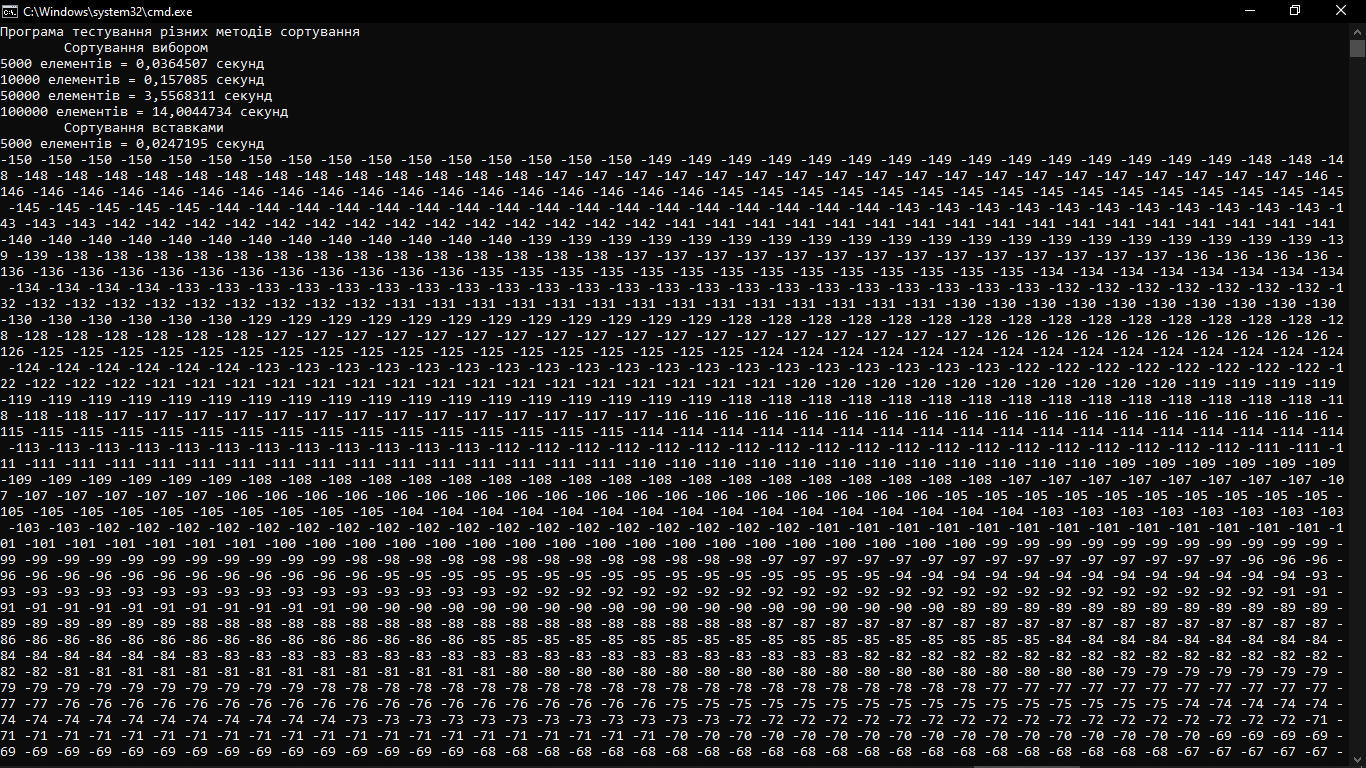


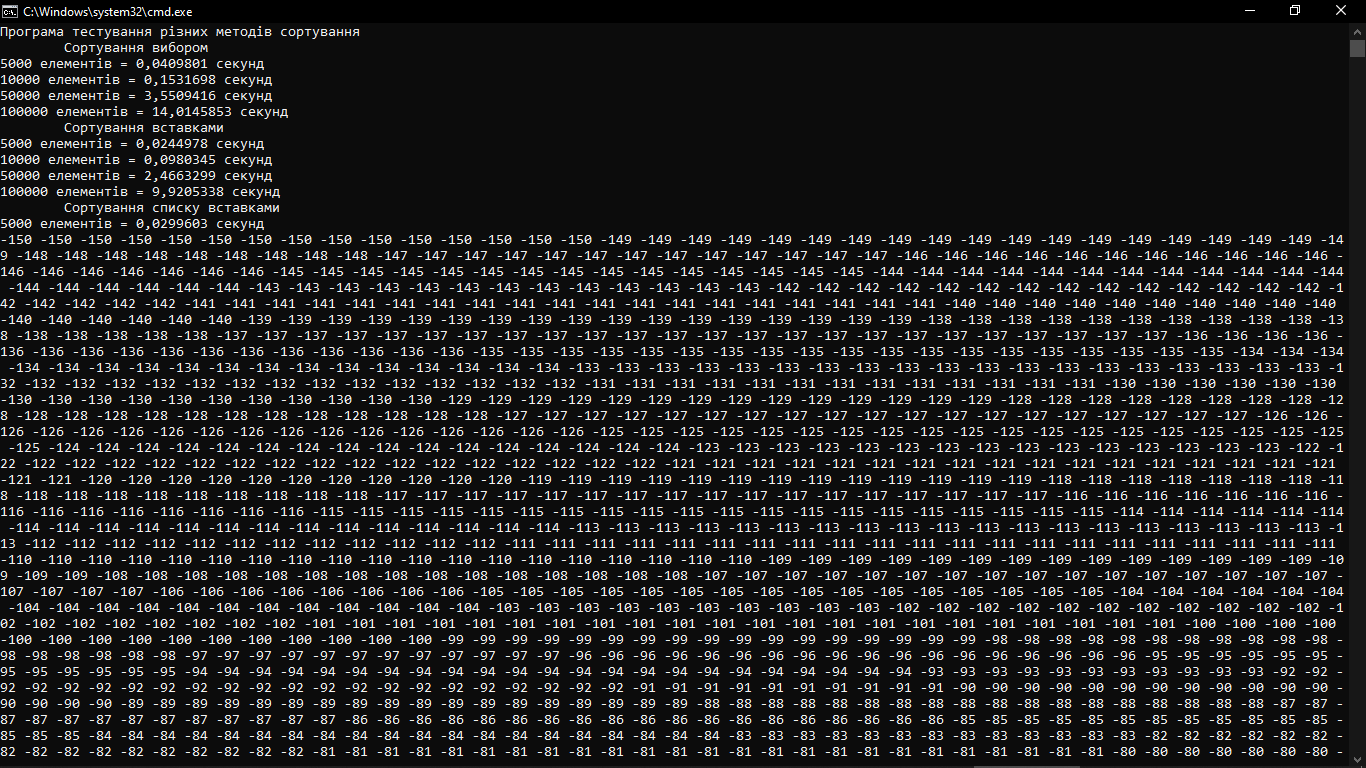
Рис.5.3 – приклад відстортованого масиву методом вставок

Рис.5.3 – приклад відсортованого списку методом вставок

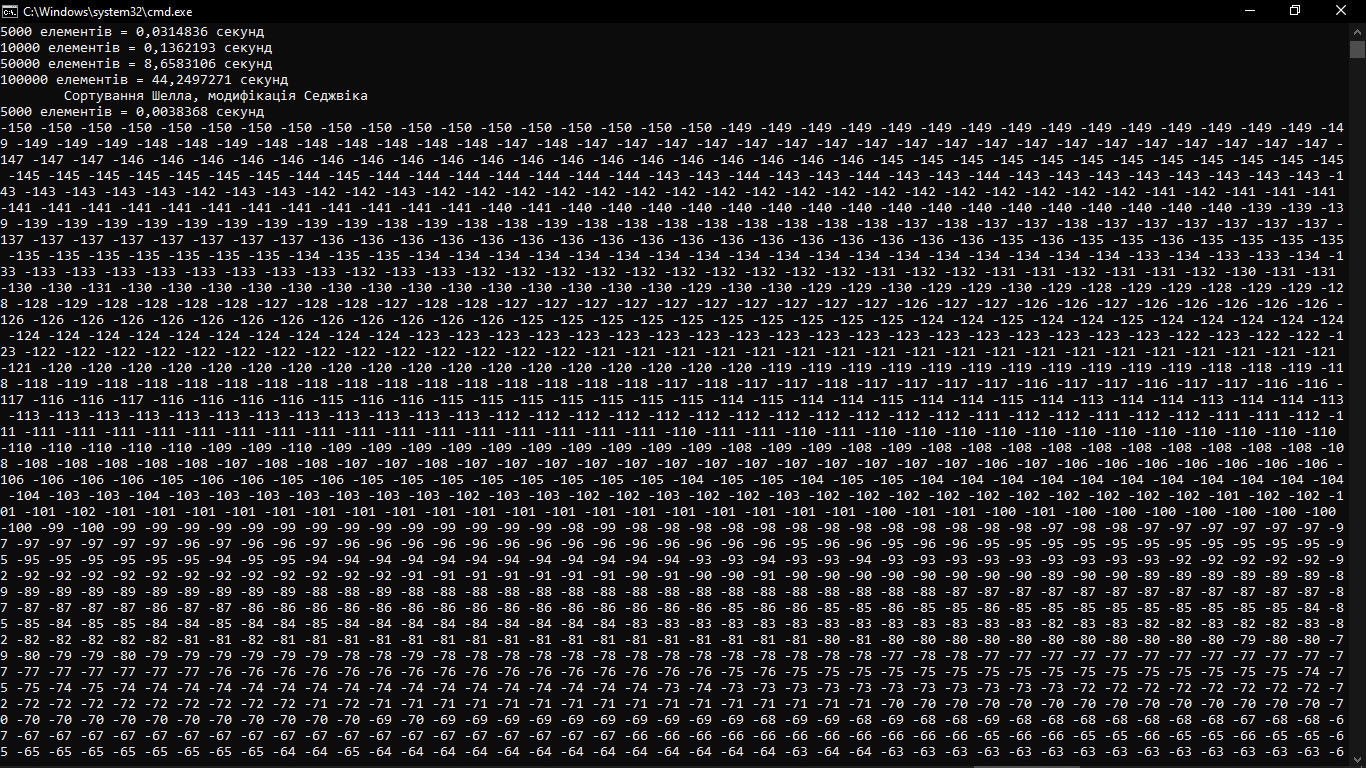


Рис.5.4 – приклад відсортованого масиву методом Шелла, модифікації Седжвіка

Для виводу масиву, розміром у 5000 елементів, користувався циклом:

for (int i = 0; i < firstArray.Length; i++)

{

Write($"{firstArray[i]} ");

if (i == firstArray.Length - 1)

{

WriteLine();

}

}

Для списку модифікував заміною змінної типу Node на наступний член списку після виведення значення.

Метод вибору працює наступним чином: за допомоги вкладеного циклу по змінній j у циклі по і шукається найменший елемент масиву та після роботи другого циклу міняється місцями з i-им членом масиву, після чого i інкрементується (з 0-го члену переходимо до 1-го і т.д) та знову запускається вкладений цикл. Таким чином швидко та зручно стають на свої місця менші елементи масиву один за одним, коли більші потроху виходять на останні місця у масиві.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Крок масиву | Порядок елементу та власне числа масиву | | | | | | | | | |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| Заповенння | 5 | 15 | 2 | 19 | 4 | 4 | 10 | 11 | 16 | 1 |
| I = 0 | 1 | 15 | 2 | 19 | 4 | 4 | 10 | 11 | 16 | 5 |
| I = 1 | 1 | 2 | 15 | 19 | 4 | 4 | 10 | 11 | 16 | 5 |
| I = 2 | 1 | 2 | 4 | 19 | 15 | 4 | 10 | 11 | 16 | 5 |
| I = 3 | 1 | 2 | 4 | 4 | 15 | 19 | 10 | 11 | 16 | 5 |
| I = 4 | 1 | 2 | 4 | 4 | 5 | 19 | 10 | 11 | 16 | 15 |
| I = 5 | 1 | 2 | 4 | 4 | 5 | 10 | 19 | 11 | 16 | 15 |
| I = 6 | 1 | 2 | 4 | 4 | 5 | 10 | 11 | 19 | 16 | 15 |
| I = 7 | 1 | 2 | 4 | 4 | 5 | 10 | 11 | 15 | 16 | 19 |
| Кінець: | 1 | 2 | 4 | 4 | 5 | 10 | 11 | 15 | 16 | 19 |

Метод сортування вставками розглядає масив у вкладеному циклі у зворотному напряму від індексу i зовнішьного циклу та переміщує елемент j покроково до свого місця, порівнюючи з іншими відсортованими членами масиву, після чого i переходить до нового числа. У випадку зі списком сортування проводиться кожний раз, коли додається новий вузол.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Крок масиву | Порядок елементу та власне числа масиву | | | | | | | | | |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| Заповенння | 5 | 15 | 2 | 19 | 4 | 4 | 10 | 11 | 16 | 1 |
| I = 1 | 5 | 15 | 2 | 19 | 4 | 4 | 10 | 11 | 16 | 1 |
| I = 2 | 2 | 5 | 15 | 19 | 4 | 4 | 10 | 11 | 16 | 1 |
| I = 3 | 2 | 5 | 15 | 19 | 4 | 4 | 10 | 11 | 16 | 1 |
| I = 4 | 2 | 4 | 5 | 15 | 19 | 4 | 10 | 11 | 16 | 1 |
| I = 5 | 2 | 4 | 4 | 5 | 15 | 19 | 10 | 11 | 16 | 1 |
| I = 6 | 2 | 4 | 4 | 5 | 10 | 15 | 19 | 11 | 16 | 1 |
| I = 7 | 2 | 4 | 4 | 5 | 10 | 11 | 15 | 19 | 16 | 1 |
| I = 8 | 2 | 4 | 4 | 5 | 10 | 11 | 15 | 16 | 19 | 1 |
| I = 9 | 1 | 2 | 4 | 4 | 5 | 10 | 11 | 15 | 16 | 19 |
| Кінець: | 1 | 2 | 4 | 4 | 5 | 10 | 11 | 15 | 16 | 19 |

Метод сортування Шелла у варіанті Седжвіка сортує масив по крокам, що записані у окремий масив за допомоги формул за парного i або за непарного i, при цьому починаємо з , якщо (розміру). За кроку 1 виконується сортування вставками

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Крок масиву | Порядок елементу та власне числа масиву | | | | | | | | | |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| Заповенння | 5 | 15 | 2 | 19 | 4 | 4 | 10 | 11 | 16 | 1 |
| Крок: 5 | 4 | 15 | 2 | 19 | 4 | 5 | 10 | 11 | 16 | 1 |
|  | 4 | 10 | 2 | 19 | 4 | 5 | 15 | 11 | 16 | 1 |
|  | 4 | 10 | 2 | 16 | 4 | 5 | 15 | 11 | 19 | 1 |
|  | 4 | 10 | 2 | 16 | 1 | 5 | 15 | 11 | 19 | 4 |
| Крок: 1 (сортування вставками | 2 | 4 | 10 | 16 | 1 | 5 | 15 | 11 | 19 | 4 |
|  | 1 | 2 | 4 | 10 | 16 | 5 | 15 | 11 | 19 | 4 |
|  | 1 | 2 | 4 | 5 | 10 | 16 | 15 | 11 | 19 | 4 |
|  | 1 | 2 | 4 | 5 | 10 | 15 | 16 | 11 | 19 | 4 |
|  | 1 | 2 | 4 | 5 | 10 | 11 | 15 | 16 | 19 | 4 |
|  | 1 | 2 | 4 | 4 | 5 | 10 | 11 | 15 | 16 | 19 |
| Кінець: | 1 | 2 | 4 | 4 | 5 | 10 | 11 | 15 | 16 | 19 |

Лістинг програми:

using System;

using static System.Console;

using System.Diagnostics;

using static System.Diagnostics.Stopwatch;

using System.Text;

namespace Sorting

{

class Program

{

public class Node //власне список

{

public int Data;

public Node Next, Previous;

}

static Node NewNode(int data) //функція створення нового вузла

{

Node newNode = new Node();

newNode.Data = data;

newNode.Previous = newNode.Next = null;

return newNode;

}

static void RandomFiller(int[] arr) //заповнює масив випадковими числами

{

Random random = new Random();

for (int i = 0; i < arr.Length; i++)

{

arr[i] = random.Next(-150, 150);

}

}

static void SelectionSorting(int[] arr) //функція сортування вибором

{

for (int i = 0; i < arr.Length - 1; i++)

{

int min = i;

for (int j = i + 1; j < arr.Length; j++)

{

if (arr[j] < arr[min])

min = j;

}

int tmp = arr[i];

arr[i] = arr[min];

arr[min] = tmp;

}

}

static void InsertionSortingArray(int[] arr) //функція сортування вставками (масив)

{

for (int i = 1; i < arr.Length; i++)

{

int j;

int tmp = arr[i];

for (j = i - 1; j >= 0; j--)

{

if (arr[j] < tmp)

break;

arr[j + 1] = arr[j];

}

arr[j + 1] = tmp;

}

}

static Node InsertionSortingList(Node Node, Node NewNode) //функція сортування вставками (список з додаванням вузлів)

{

Node current;

if (Node == null)

{

Node = NewNode;

}

else if (Node.Data >= NewNode.Data)

{

NewNode.Next = Node;

NewNode.Next.Previous = NewNode;

Node = NewNode;

}

else

{

current = Node;

while (current.Next != null && current.Next.Data < NewNode.Data)

{

current = current.Next;

}

NewNode.Next = current.Next;

if (current.Next != null)

{

NewNode.Next.Previous = NewNode;

}

current.Next = NewNode;

NewNode.Previous = current;

}

return Node;

}

static void Shell\_SedgewickSorting(int[] arr) //функція сортування Шелла у модифікації Седжвіка

{

int step, i, j;

int[] inc = new int[40];

int p1 = 1, p2 = 1, p3 = 1, st = -1;

do

{

if (++st % 2 != 0)

inc[st] = 8 \* p1 - 6 \* p2 + 1;

else

{

inc[st] = 9 \* p1 - 9 \* p3 + 1;

p2 \*= 2;

p3 \*= 2;

}

p1 \*= 2;

} while (3 \* inc[st] < inc.Length);

st = st > 0 ? --st : 0;

while (st >= 0)

{

step = inc[st--];

for (i = step; i < arr.Length; i++)

{

int temp = arr[i];

for (j = i - step; (j >= 0) && (arr[j] > temp); j -= step)

arr[j + step] = arr[j];

arr[j + step] = temp;

}

}

}

static void Main()

{

InputEncoding = Encoding.Unicode;

OutputEncoding = Encoding.Unicode;

Stopwatch timer;

int[] firstArray = new int[5000], secondArray = new int[10000], thirdArray = new int[50000], fourthArray = new int[100000];

/////////////////////////////////////////////////////////////////////////////////////////////////////

WriteLine("Програма тестування різних методів сортування");

{

WriteLine("\tСортування вибором");

RandomFiller(firstArray);

timer = StartNew();

SelectionSorting(firstArray);

timer.Stop();

WriteLine($"{firstArray.Length} елементів = {timer.Elapsed.TotalSeconds} секунд");

RandomFiller(secondArray);

timer = StartNew();

SelectionSorting(secondArray);

timer.Stop();

WriteLine($"{secondArray.Length} елементів = {timer.Elapsed.TotalSeconds} секунд");

RandomFiller(thirdArray);

timer = StartNew();

SelectionSorting(thirdArray);

timer.Stop();

WriteLine($"{thirdArray.Length} елементів = {timer.Elapsed.TotalSeconds} секунд");

RandomFiller(fourthArray);

timer = StartNew();

SelectionSorting(fourthArray);

timer.Stop();

WriteLine($"{fourthArray.Length} елементів = {timer.Elapsed.TotalSeconds} секунд");

}

/////////////////////////////////////////////////////////////////////////////////////////////////////

{

WriteLine("\tСортування вставками");

RandomFiller(firstArray);

timer = StartNew();

InsertionSortingArray(firstArray);

timer.Stop();

WriteLine($"{firstArray.Length} елементів = {timer.Elapsed.TotalSeconds} секунд");

RandomFiller(secondArray);

timer = StartNew();

InsertionSortingArray(secondArray);

timer.Stop();

WriteLine($"{secondArray.Length} елементів = {timer.Elapsed.TotalSeconds} секунд");

RandomFiller(thirdArray);

timer = StartNew();

InsertionSortingArray(thirdArray);

timer.Stop();

WriteLine($"{thirdArray.Length} елементів = {timer.Elapsed.TotalSeconds} секунд");

RandomFiller(fourthArray);

timer = StartNew();

InsertionSortingArray(fourthArray);

timer.Stop();

WriteLine($"{fourthArray.Length} елементів = {timer.Elapsed.TotalSeconds} секунд");

}

/////////////////////////////////////////////////////////////////////////////////////////////////////

{

WriteLine("\tСортування списку вставками");

Random random = new Random();

Node node = null;

timer = StartNew();

for (int i = 0; i < firstArray.Length; i++)

{

node = InsertionSortingList(node, NewNode(random.Next(-150, 150)));

}

timer.Stop();

WriteLine($"{firstArray.Length} елементів = {timer.Elapsed.TotalSeconds} секунд");

node = null;

timer = StartNew();

for (int i = 0; i < secondArray.Length; i++)

{

node = InsertionSortingList(node, NewNode(random.Next(-150, 150)));

}

timer.Stop();

WriteLine($"{secondArray.Length} елементів = {timer.Elapsed.TotalSeconds} секунд");

node = null;

timer = StartNew();

for (int i = 0; i < thirdArray.Length; i++)

{

node = InsertionSortingList(node, NewNode(random.Next(-150, 150)));

}

timer.Stop();

WriteLine($"{thirdArray.Length} елементів = {timer.Elapsed.TotalSeconds} секунд");

node = null;

timer = StartNew();

for (int i = 0; i < fourthArray.Length; i++)

{

node = InsertionSortingList(node, NewNode(random.Next(-150, 150)));

}

timer.Stop();

WriteLine($"{fourthArray.Length} елементів = {timer.Elapsed.TotalSeconds} секунд");

}

/////////////////////////////////////////////////////////////////////////////////////////////////////

{

WriteLine("\tСортування Шелла, модифікація Седжвіка");

RandomFiller(firstArray);

timer = StartNew();

Shell\_SedgewickSorting(firstArray);

timer.Stop();

WriteLine($"{firstArray.Length} елементів = {timer.Elapsed.TotalSeconds} секунд");

RandomFiller(secondArray);

timer = StartNew();

Shell\_SedgewickSorting(secondArray);

timer.Stop();

WriteLine($"{secondArray.Length} елементів = {timer.Elapsed.TotalSeconds} секунд");

RandomFiller(thirdArray);

timer = StartNew();

Shell\_SedgewickSorting(thirdArray);

timer.Stop();

WriteLine($"{thirdArray.Length} елементів = {timer.Elapsed.TotalSeconds} секунд");

RandomFiller(fourthArray);

timer = StartNew();

Shell\_SedgewickSorting(fourthArray);

timer.Stop();

WriteLine($"{fourthArray.Length} елементів = {timer.Elapsed.TotalSeconds} секунд");

}

}

}

}

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Кількість елементів | Час сортування, сек | | | |
| Вибором | Вставками, масив | Вставками, список | Шелла (Седжвік) |
| 5000 | 0,04 | 0,03 | 0,03 | 0,006 |
| 10000 | 0,23 | 0,1 | 0,13 | 0,025 |
| 50000 | 3,78 | 2,65 | 8,05 | 0,58 |
| 100000 | 14,91 | 10,62 | 44,42 | 2,34 |

Таблиця.5.1 – час сортування для різних методів різної кількості елементів

Рис.5.4 – діаграма-графік результатів вимірів

***Висновок:*** після дослідження результатів роботи, можна вказати на метод сортування Шелла (Седжвіка) як на найефективніший з методів, особливо за надвеликої кількості елементів (більше, ніж 100 тис.). Складність алгоритмів сортування: найгірший та найкращий вибором – O(n2); найгірший вставками– O(n2), найкращий – O(n); найгірший Шелла (Седжвіка) – O(n4/3), найкращий – O(n7/6).