Лабораторна робота № 6

ШВИДКІ МЕТОДИ СОРТУВАННЯ

Мета : реалізація швидких алгоритмів сортування та дослідження їх характеристик (швидкодія, необхідний обсяг пам'яті, застосування тощо).

5.1 Хід роботи

Зміст звіту

1. Опис алгоритму (словесна форма або блок-схема алгоритму).
2. Текст функцій сортування з коментарями.
3. Таблиця результатів вимірів часу.
4. Графіки результатів вимірів часу.
5. Висновки по роботі (опис досліджених характеристик кожного алгоритму, порівняння алгоритмів, відзначити переваги та недоліки).

Порядок виконання роботи

1. Реалізувати алгоритми сортування у відповідності за таблицею 6.1:
   а) пірамідальне сортування (структура даних – масив);
   б) сортування Шелла (структура даних – масив);
   в) сортування підрахунком (структура даних – масив).
2. Виміряти час сортування даних різної розмірності: 10, 100, 500, 1000, 2000, 5000, 10000. Дані сформувати з використанням генератора випадкових чисел.
3. За отриманими даними побудувати графіки залежностей часу сортування від кількості вхідних даних (з використанням Excel).

Варіант	Пірамідальне сортування		Сортування Шелла			Сортування підрахунком	
	Тип даних	Діапазон	Тип даних	Діапазон	Формула приросту	Тип даних	Діапазон
2, 17	double	[-10, 100]	float	[0, 200]	inc(s)=(3^s-1)/2	char	[-100, -10]

Змн.	Арк.	№ докум.	Підпис	Дата	ДУ«Житомирська політехніка».21.121.02.000–Лр 6		
Розроб.		Маньківський В.В			Звіт з лабораторної роботи	Літ.	Арк. 1 / Аркушів 14
Перевір.		Локтікова Т.М.					
Керівник						ФІКТ Гр. ВТ-21-1[2]	
Н. контр.							
Зав. каф.							

6.1.1

Завдання:

Лістинг:

```
using System.Diagnostics;
namespace lab_5
{
    class Program
    {
        static void Main()
        {
            Console.OutputEncoding = System.Text.Encoding.Unicode;
            Console.InputEncoding = System.Text.Encoding.Unicode;
            Stopwatch stopwatch = new Stopwatch();
            Random random = new Random();
            Console.WriteLine("а) пірамідальне сортування (структура даних – масив);");
            Console.WriteLine("б) сортування Шелла(структура даних – масив);");
            Console.WriteLine("в) сортування підрахунком(структура даних – масив);\n");

            double[] piramida_10 = new double[10];
            double[] piramida_100 = new double[100];
            double[] piramida_500 = new double[500];
            double[] piramida_1000 = new double[1000];
            double[] piramida_2000 = new double[2000];
            double[] piramida_5000 = new double[5000];
            double[] piramida_10000 = new double[10000];

            float[] szela_10 = new float[10];
            float[] szela_100 = new float[100];
            float[] szela_500 = new float[500];
            float[] szela_1000 = new float[1000];
            float[] szela_2000 = new float[2000];
            float[] szela_5000 = new float[5000];
            float[] szela_10000 = new float[10000];

            char[] liczba_10 = new char[10];
            char[] liczba_100 = new char[100];
            char[] liczba_500 = new char[500];
            char[] liczba_1000 = new char[1000];
            char[] liczba_2000 = new char[2000];
            char[] liczba_5000 = new char[5000];
```

```
                char[] liczba_10000 = new char[10000];

                HeapSort heapSort = new HeapSort();
                ShellSort shellSort = new ShellSort();
                CountSort countSort = new CountSort();



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

                int count = 10;
                Console.WriteLine($"Розмірність: {count}");
                for (int i = 0; i < count; i++)
                {
                    piramida_10[i] = random.NextDouble() * (100 - (-10)) - 10;
                }
                stopwatch.Start();
                heapSort.sort(piramida_10);
                stopwatch.Stop();
                Console.WriteLine($"а) {stopwatch.Elapsed.TotalMilliseconds} мс");
                stopwatch.Reset();
                for (int i = 0; i < count; i++)
                {
                    szela_10[i] = (float)random.NextDouble() * (200 - 0) - 0;
                }
                stopwatch.Start();
                shellSort.Sort(szela_10);
                stopwatch.Stop();
                Console.WriteLine($"б) {stopwatch.Elapsed.TotalMilliseconds} мс");
                stopwatch.Reset();
                for (int i = 0; i < count; i++)
                {
                    liczba_10[i] = (char)random.Next(-100, -10);
                }
                stopwatch.Start();
                countSort.Sort(liczba_10);
                stopwatch.Stop();
                Console.WriteLine($"в) {stopwatch.Elapsed.TotalMilliseconds} мс");
                stopwatch.Reset();
```

```
////////////////////////////////////////////////////////////////////////////////
/////////////////

                count = 100;
                Console.WriteLine($"\nРозмірність: {count}");
                for (int i = 0; i < count; i++)
                {
                    piramida_100[i] = random.NextDouble() * (100 - (-10)) - 10;
                }
                stopwatch.Start();
                heapSort.sort(piramida_100);
                stopwatch.Stop();
                Console.WriteLine($"а) {stopwatch.Elapsed.TotalMilliseconds} мс");
                stopwatch.Reset();
                for (int i = 0; i < count; i++)
                {
                    szela_100[i] = (float)random.NextDouble() * (200 - 0) - 0;

                }
                stopwatch.Start();
                shellSort.Sort(szela_100);
                stopwatch.Stop();
                Console.WriteLine($"б) {stopwatch.Elapsed.TotalMilliseconds} мс");
                stopwatch.Reset();
                for (int i = 0; i < count; i++)
                {
                    liczba_100[i] = (char)random.Next(-100, -10);
                }
                stopwatch.Start();
                countSort.Sort(liczba_100);
                stopwatch.Stop();
                Console.WriteLine($"в) {stopwatch.Elapsed.TotalMilliseconds} мс");
                stopwatch.Reset();


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

                count = 500;
```

```
                Console.WriteLine($"\nРозмірність: {count}");
                for (int i = 0; i < count; i++)
                {
                    piramida_500[i] = random.NextDouble() * (100 - (-10)) - 10;
                }
                stopwatch.Start();
                heapSort.sort(piramida_500);
                stopwatch.Stop();
                Console.WriteLine($"а) {stopwatch.Elapsed.TotalMilliseconds} мс");
                stopwatch.Reset();
                for (int i = 0; i < count; i++)
                {
                    szela_500[i] = (float)random.NextDouble() * (200 - 0) - 0;

                }
                stopwatch.Start();
                shellSort.Sort(szela_500);
                stopwatch.Stop();
                Console.WriteLine($"б) {stopwatch.Elapsed.TotalMilliseconds} мс");
                stopwatch.Reset();
                for (int i = 0; i < count; i++)
                {
                    liczba_500[i] = (char)random.Next(-100, -10);
                }
                stopwatch.Start();
                countSort.Sort(liczba_500);
                stopwatch.Stop();
                Console.WriteLine($"в) {stopwatch.Elapsed.TotalMilliseconds} мс");
                stopwatch.Reset();


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

                count = 1000;
                Console.WriteLine($"\nРозмірність: {count}");
                for (int i = 0; i < count; i++)
                {
                    piramida_1000[i] = random.NextDouble() * (100 - (-10)) - 10;
                }
                stopwatch.Start();
```

```
heapSort.sort(piramida_1000);
stopwatch.Stop();
Console.WriteLine($"a) {stopwatch.Elapsed.TotalMilliseconds} мс");
stopwatch.Reset();
for (int i = 0; i < count; i++)
{
    szela_1000[i] = (float)random.NextDouble() * (200 - 0) - 0;

}
stopwatch.Start();
shellSort.Sort(szela_1000);
stopwatch.Stop();
Console.WriteLine($"б) {stopwatch.Elapsed.TotalMilliseconds} мс");
stopwatch.Reset();
for (int i = 0; i < count; i++)
{
    liczba_1000[i] = (char)random.Next(-100, -10);
}
stopwatch.Start();
countSort.Sort(liczba_1000);
stopwatch.Stop();
Console.WriteLine($"в) {stopwatch.Elapsed.TotalMilliseconds} мс");
stopwatch.Reset();



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

count = 2000;
Console.WriteLine($"\nРозмірність: {count}");
for (int i = 0; i < count; i++)
{
    piramida_2000[i] = random.NextDouble() * (100 - (-10)) - 10;
}
stopwatch.Start();
heapSort.sort(piramida_2000);
stopwatch.Stop();
Console.WriteLine($"а) {stopwatch.Elapsed.TotalMilliseconds} мс");
stopwatch.Reset();
for (int i = 0; i < count; i++)
{
```

```
            szela_2000[i] = (float)random.NextDouble() * (200 - 0) - 0;

        }
        stopwatch.Start();
        shellSort.Sort(szela_2000);
        stopwatch.Stop();
        Console.WriteLine($"6) {stopwatch.Elapsed.TotalMilliseconds} мс");
        stopwatch.Reset();
        for (int i = 0; i < count; i++)
        {
            liczba_2000[i] = (char)random.Next(-100, -10);
        }
        stopwatch.Start();
        countSort.Sort(liczba_2000);
        stopwatch.Stop();
        Console.WriteLine($"в) {stopwatch.Elapsed.TotalMilliseconds} мс");
        stopwatch.Reset();



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

        count = 5000;
        Console.WriteLine($"\nРозмірність: {count}");
        for (int i = 0; i < count; i++)
        {
            piramida_5000[i] = random.NextDouble() * (100 - (-10)) - 10;
        }
        stopwatch.Start();
        heapSort.sort(piramida_5000);
        stopwatch.Stop();
        Console.WriteLine($"а) {stopwatch.Elapsed.TotalMilliseconds} мс");
        stopwatch.Reset();
        for (int i = 0; i < count; i++)
        {
            szela_5000[i] = (float)random.NextDouble() * (200 - 0) - 0;

        }
        stopwatch.Start();
        shellSort.Sort(szela_5000);
        stopwatch.Stop();
```

```
                Console.WriteLine($"б) {stopwatch.Elapsed.TotalMilliseconds} мс");
                stopwatch.Reset();
                for (int i = 0; i < count; i++)
                {
                    liczba_5000[i] = (char)random.Next(-100, -10);
                }
                stopwatch.Start();
                countSort.Sort(liczba_5000);
                stopwatch.Stop();
                Console.WriteLine($"в) {stopwatch.Elapsed.TotalMilliseconds} мс");
                stopwatch.Reset();



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


                count = 10000;
                Console.WriteLine($"\nРозмірність: {count}");
                for (int i = 0; i < count; i++)
                {
                    piramida_10000[i] = random.NextDouble() * (100 - (-10)) - 10;
                }
                stopwatch.Start();
                heapSort.sort(piramida_10000);
                stopwatch.Stop();
                Console.WriteLine($"а) {stopwatch.Elapsed.TotalMilliseconds} мс");
                stopwatch.Reset();
                for (int i = 0; i < count; i++)
                {
                    szela_10000[i] = (float)random.NextDouble() * (200 - 0) - 0;

                }
                stopwatch.Start();
                shellSort.Sort(szela_10000);
                stopwatch.Stop();
                Console.WriteLine($"б) {stopwatch.Elapsed.TotalMilliseconds} мс");
                stopwatch.Reset();
                for (int i = 0; i < count; i++)
                {
                    liczba_10000[i] = (char)random.Next(-100, -10);
                }
```

```
            stopwatch.Start();
            countSort.Sort(liczba_10000);
            stopwatch.Stop();
            Console.WriteLine($"в) {stopwatch.Elapsed.TotalMilliseconds} мс");
            stopwatch.Reset();
        }
    }
    public class HeapSort
    {
        public void sort(double[] arr)
        {
            int n = arr.Length;
            //перегрупування масиву
            for (int i = n / 2 - 1; i >= 0; i--)
                heapify(arr, n, i);
            for (int i = n - 1; i >= 0; i--)
            {
                var temp = arr[0];
                arr[0] = arr[i];
                arr[i] = temp;
                heapify(arr, i, 0);//найбільше в кінець
            }
        }
        void heapify(double[] arr, int n, int i)
        {
            int largest = i;
            int l = 2 * i + 1;//ліва частина
            int r = 2 * i + 2;//права частина
            if (l < n && arr[l] > arr[largest])//якщо лівий елемент більше
                largest = l;
            if (r < n && arr[r] > arr[largest])//якщо правий елемент більше
                largest = r;
            if (largest != i)//якщо неайбільше не корінь
            {
                var swap = arr[i];
                arr[i] = arr[largest];
                arr[largest] = swap;
                heapify(arr, n, largest);
            }
        }
    }
    public class ShellSort
    {
```

```
        static void Swap(ref float a, ref float b)//заміна
        {
            var t = a;
            a = b;
            b = t;
        }
        public void Sort(float[] array)
        {
            var d = array.Length / 2;//відстань між елементами, які порівнюються
            while (d >= 1)
            {
                for (var i = d; i < array.Length; i++)
                {
                    var j = i;
                    while ((j >= d) && (array[j - d] > array[j]))//пошук більшого
                    {
                        Swap(ref array[j], ref array[j - d]);
                        j = j - d;
                    }
                }
                d = d / 2;//зменшення відстані
            }
        }
    }
    public class CountSort
    {
        public void Sort(char[] inputArray)
        {
            int[] countArray = new int[inputArray.Max() + 1];//мазив з кінцевим
індексом максимального елемента
            for (int i = 0; i < inputArray.Length; i++)//елемент стає на індекс,
відповідно свого числа
            {
                countArray[inputArray[i]]++;
            }

            int sortedArrayIndex = 0;//індекс відсортованого елемента
            for (int i = countArray.Length - 1; i >= 0; i--)//сортування
            {
                for (int j = 0; j < countArray[i]; j++)
                {
                    inputArray[sortedArrayIndex++] = (char)i;
                }
```

```
            }
        }
    }
}
```

```
а) пірамідальне сортування (структура даних - масив);
б) сортування Шелла(структура даних - масив);
в) сортування підрахунком(структура даних - масив);

Розмірність: 10
а) 0,5155 мс
б) 0,3777 мс
в) 3,295 мс

Розмірність: 100
а) 0,0272 мс
б) 0,0156 мс
в) 0,3546 мс

Розмірність: 500
а) 0,1282 мс
б) 0,12 мс
в) 0,5443 мс

Розмірність: 1000
а) 0,3839 мс
б) 0,422 мс
в) 0,4845 мс

Розмірність: 2000
а) 0,9182 мс
б) 0,6573 мс
в) 0,3426 мс

Розмірність: 5000
а) 1,8241 мс
б) 1,9742 мс
в) 0,3923 мс

Розмірність: 10000
а) 3,9212 мс
б) 4,1629 мс
в) 0,8003 мс
```

Рисунок 6.1 – Результат виконання завдання

Словесний опис алгоритмів:

Пірамідальне сортування - алгоритм сортування, який використовує бінарну кучу. Куча (англ. Heap) – структура даних типу дерево, яка задовольняє наступній властивості: для будь-якого заданого вузла B, який є нащадком вузла A виконується умова: ключ (A) ≥ ключ (B). Таким чином, кореневий вузол кучи буде зберігати найбільше значення, тому іноді таку кучу називають max-кучою. Якщо змінити порівняння на протилежне, то

кореневий вузол буде зберігати найменше значення і таку кучу називають min-кучою. Поширеною реалізацією кучи є бінарна куча (англ. Binary heap), в якій дерево є двійковим деревом (рисунок 6.1 та рисунок 6.2). Двійкова куча задовольняє трьом умовам:

- Значення в будь-якій вершині не менш, ніж значення її нащадків (max-куча);
- Глибина всього листя відрізняється не більше ніж на 1 шар;
- Останній шар заповнюється зліва направо. Висота кореневого вузла (дерева): int(log2 N).

Сортування Шелла є модифікацією алгоритму сортування вставками та класифікується як сортування вставками з убиваючим кроком. Ефективність алгоритму полягає в тому, що на кожному з проміжних кроків сортується або невелике число елементів, або вже досить добре впорядковані набори елементів. Впорядкованість масиву зростає після кожного проходу. В ході вивчення алгоритму досліджувалася залежність середнього числа перестановок від розміру масивів при N від 100 до 60000 для декількох типів послідовностей кроків, для яких були отримані відповідні залежності часу роботи від розміру масиву.

Сортування підрахунком - алгоритм впорядкування, що застосовується при малій кількості різних елементів (ключів) у масиві даних. Час його роботи лінійно залежить як від загальної кількості елементів у масиві так і від кількості різних елементів.

Графіки та таблиця:

Кількість елементів	Пірамідальне масив (мс)	Шелла масив (мс)	Підрахунком масив (мс)
10	0,5155	0,3777	3,295
100	0,0272	0,0156	0,3546
500	0,1282	0,12	0,5443
1000	0,3839	0,422	0,4845
2000	0,9182	0,6573	0,3426
5000	1,8241	1,9742	0,3923
10000	3,9212	4,9742	0,8003

Рисунок 6.2 – Таблиця

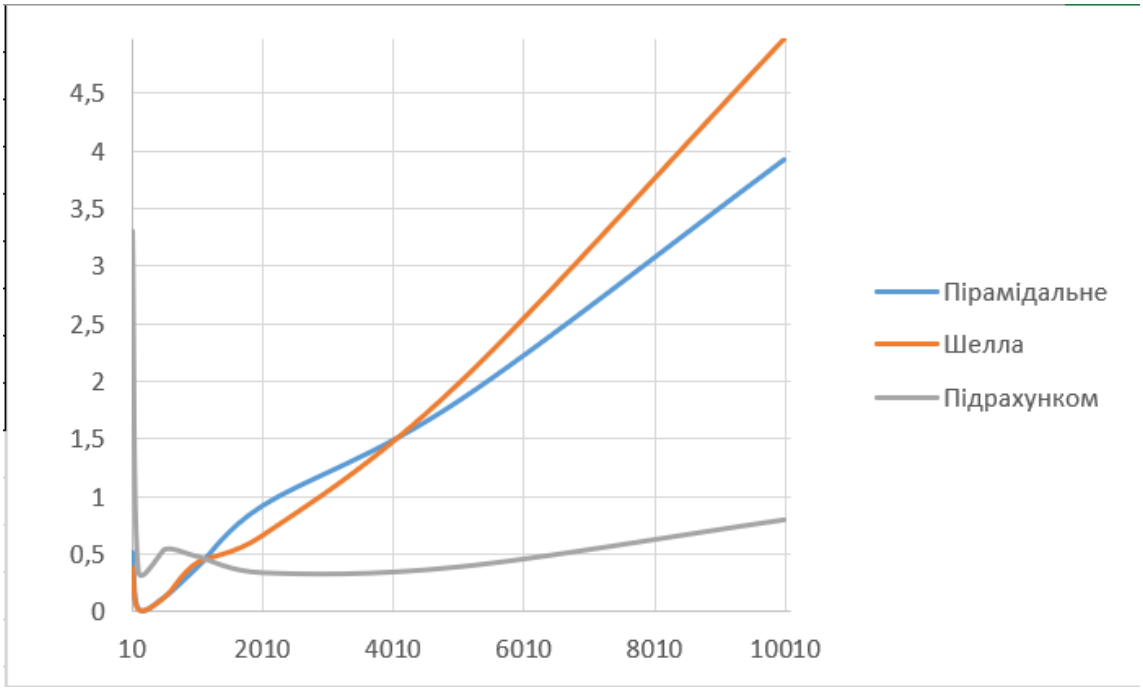

Рисунок 6.3 – Графіки

Висновки: я дослідив та порівняв характеристики кожного з алгоритмів, та дійшов висновку, що на різних проміжках, швидкість сортування різна, наприклад, Шелла найшвидше від 10 до 500, пірамідальне на 1000, підрахунком від 2000 до 10000. Перш за все, ці результати не точні, окрім підрахунку, так як в різних масивах були різні числа. Отже необхідно проаналізувати кожен алгоритм. Пірамідальне сортування залежить від кількості першин під коренем, та чи відсортованні вже елементи, якщо так, то складність алгоритму збільшується до квадрату.Сортування Шелла навпаки більш ефективне для більш впорядкованих

елементів. В свою чергу метод підрахунку не подібний на інші, так як він повністю залежить від кількості елементів, а не від їх значення.