Міністерство освіти і науки України
Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського"
Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації і управління

Звіт

з лабораторної роботи № 5 з дисципліни
«Алгоритми та структури даних 2. Структури даних»

«Спискові структури даних»

Виконав(ла) *ІП-13 Шиманська Ганна Артурівна* _____
(шифр, прізвище, ім'я, по батькові)

Перевірив *Сопов Олексій Олександрович* _____
(прізвище, ім'я, по батькові)

Київ 2022

ЗМІСТ

1 МЕТА ЛАБОРАТОРНОЇ РОБОТИ

Мета роботи – вивчити основні підходи формалізації евристичних алгоритмів і вирішення типових задач з їх допомогою.

2 ЗАВДАННЯ

Розробити алгоритм розв’язання задачі відповідно до варіанту. Виконати програмну реалізацію задачі. Не використовувати вбудовані спискові структури даних (контейнери). Зробити висновок по лабораторній роботі.

Варіанти завдань.

1. Заданий текст, що має декілька рядків. Використовуючи стек, елементами якого є літери, надрукувати текст, в якому літери кожного рядка містяться у зворотному порядку.

2. Побудувати список, елементами якого є дійсні числа. Знайти їх середнє арифметичне і розмістити отримане значення останнім елементом списку. Надрукувати початковий і змінений списки.

3. Заданий рядок слів, які відокремлюються одне від одного пробілами. Побудувати список, елементами якого є відповідні слова. Вилучити із списку всі слова, що починаються та закінчуються на задану користувачем літеру. Надрукувати початковий і змінений списки.

4. Задане натуральне число n та послідовність дійсних чисел $x_1, ..., x_n$. Використовуючи двозв'язний список, елементами якого є задані дійсні числа, визначити $(x_1 - x_n) + (x_2 - x_{n-1}) + ... + (x_n - x_1)$.

5. Заданий текст, що містить декілька рядків слів, розділених пробілами. Використовуючи стек, елементами якого є слова, надрукувати текст, в якому слова кожного рядка містяться у зворотному порядку.

6. Заданий рядок слів, які відокремлюються одне від одного комами. Побудувати список, елементами якого є відповідні слова. Вилучити із списку всі слова заданої користувачем довжини. Надрукувати початковий і змінений списки.

7. Побудувати список, елементами якого є цілі числа. Знайти суму останнього і передостаннього його елементів і замістити відповідним значенням ці елементи списку. Надрукувати початковий і змінений списки.

8. Заданий текст, що має декілька рядків. Використовуючи чергу або стек,

елементами яких є літери, надрукувати тексти, що знаходяться між кожною парою дужок заданого користувачем виду.

9. Заданий рядок слів, які відокремлюються одне від одного будь-якою кількістю пробілів. Побудувати список відповідних слів. Поміняти місцями найдовше та найкоротше слово цього списку. Надрукувати початковий і змінений списки.

10. Одне з можливих представлень тексту — це розділити його на рядки і створити список рядків, додавши ознаку кінця тексту. Використовуючи дане представлення тексту, визначити, чи входить задана літера у текст, і, якщо входить, то вивести «координати» першого входження цієї літери у текст (номер рядка і номер позиції в цьому рядку).

11. Заданий рядок слів, які відокремлюються одне від одного символами ";". Побудувати список слів, що містяться у цьому рядку. Вилучити із спискувсі однакові слова. Надрукувати початковий і змінений списки.

12. Побудувати список символів. Визначити найбільше значення списку і помістити його на початок списку. Надрукувати початковий і змінений списки.

13. Задане натуральне число n та послідовність дійсних чисел $x_1, ..., x_n$. Використовуючи двозв'язний список, елементами якого є задані дійсні числа, визначити $x_1 \cdot x_n + x_2 \cdot x_{n-1} + ... + x_n \cdot x_1$.

14. Заданий список символів. Перенести в кінець цього списку його перший елемент. Надрукувати початковий та змінений списки.

15. Задана послідовність натуральних чисел. Використовуючи стек, надрукувати у зворотному порядку усі числа, які містяться між найбільшим та найменшим числами послідовності.

16. Текстовий файл містить вираз, записаний у звичайній (інфіксній) формі. Використавши стек, перекласти заданий вираз в постфіксну форму і записати в новий текстовий файл. Інфіксна форма виразу: a-b, a*b; постфіксна форма: ab-, ab+.

17. Задана послідовність цілих чисел. Використовуючи стек, надрукувати у зворотному порядку усі її числа, що не кратні 5.

18. Створити кільцевий список, елементами якого є числа. Послідовно вилучати кожне третє число. Підрахувати кількість вилучень. Вивести початковий список та вилучені елементи у порядку їх вилучення.

19. Задана послідовність дійсних чисел, що містить від'ємні елементи. Побудувати список, елементами якого є відповідні числа. Вилучити всі від'ємні елементи цього списку. Надрукувати початковий та новий списки.

20. Задана послідовність цілих чисел, що містить від'ємні елементи. Використовуючи стек, елементами якого є цілі числа, надрукувати у зворотному порядку всі додатні елементи послідовності.

21. Задане натуральне число n та послідовність дійсних чисел x_1, ..., x_n. Використовуючи двозв'язний список, елементами якого є задані дійсні числа, побудувати список, що містить елементи $x_1 \cdot x_n$, $x_2 \cdot x_n$, ..., $x_{n-1} \cdot x_n$.

22. Створити список цілих чисел. Вилучити із списку усі парні числа, підрахувавши їх кількість. Надрукувати початковий, змінений список та визначену величину.

23. Задана послідовність цілих чисел. Використовуючи чергу, елементами якої є цілі числа, вивести на друк спочатку парні елементи послідовності, а потім – непарні.

24. Побудувати кільцевий список, елементами якого є цілі числа. Послідовно вилучати кожне його парне число, заносячи його до стеку. Вивести початковий список та вміст стеку.

25. Створити двозв'язний список, елементами якого є слова тексту.Вивести слова, які стоять на парних місцях при проході по списку в одному напрямку, та слова, які стоять на непарних позиціях при проході по списку в зворотньому напрямку.

26. Задана послідовність цілих чисел. Побудувати список, у якому числа впорядковані у порядку зростання. Надрукувати послідовність і впорядкований список.

27. Заданий текст, що містить декілька рядків слів, розділених пробілами. Використовуючи чергу, елементами якої є слова, та стек, елементами якого є

літери, надрукувати текст, в якому літери слів кожного рядка містяться у зворотному порядку.

28. Одне з можливих представлень тексту — це розділити його на рядки і створити список рядків, додавши ознаку кінця тексту. Використовуючи дане представлення тексту, визначити номер рядка з максимальною кількістю входжень заданої літери.

29. Задане натуральне число n та послідовність дійсних чисел x_1, ..., x_n. Використовуючи двозв'язний список, елементами якого є задані дійсні числа, побудувати список, що містить елементи $x_1 - x_n$, $x_2 - x_n$, ..., $x_{n-1} - x_n$.

30. Заданий список, елементами якого символи. Вставити в кінець списку новий елемент, що введений з клавіатури, та вилучити із списку перший його елемент. Надрукувати початковий та змінений списки.

31. Реалізуйте структуру "черга з пріоритетом", яка підтримує наступні операції: додавання елемента в чергу; видалення з черги елемента з найбільшим пріоритетом; зміна пріоритету для довільного елемента, що знаходиться в черзі.

32. Створіть двозв'язний список груп факультету інформатики. Кожна група представляє собою однозв'язний список студентів. Надрукувати дані двозв'язного списку.

33. Створити двозв'язний список, елементами якого є цілі числа. Упорядкувати елементи списку двома способами (зміна покажчиків, зміна значень елементів).

34. Створити двозв'язний список елементами якого є цілі числа, видалити зі списку елементи, значення яких вже зустрічалися раніше. Надрукувати початковий та змінений списки.

35. Створити двозв'язний список із цілих значень та визначити чи можна видалити зі списку якихось два елементи так, щоб новий список виявився упорядкованим.

3 ВИКОНАННЯ

3.1 Псевдокод алгоритмів

Для values:

```
Функція SwapValues(node1, node2)
    (node1.Value, node2.Value) = (node2.Value, node1.Value)
все функція

Функція BubbleSort(first, last)
    якщо first == last
    то
        все функція
    все якщо
    swapCounter = 0
    для current від first до last повторити
        якщо current.Value > current.Next.Value
        то
            SwapValues(current, current.Next)
            swapCounter++
        все якщо
    все повторити
    якщо swapCounter>0
    то
        BubbleSort(first, last.Previous)
    все якщо
все функція
```

Для nodes:

```
Функція SwapNodes(node1,  node2)
    якщо node1.Next = node2
    то
        first = node1
    все якщо
    інакше
        first = node2
    все інакше
    якщо node1.Next = node2
    то
         second = node2
    все якщо
    інакше
        second = node1
    все інакше
    якщо Head == first
    то
        Head = second
    все якщо
    якщо Tail == second
```

```
            то
                Tail = first
            все якщо
            якщо first.Previous is not null
            то
                first.Previous.Next = second
            все якщо
            якщо second.Next is not null
            то
                second.Next.Previous = first
            все якщо
            second.Previous = first.Previous
            first.Previous = second
            first.Next = second.Next
            second.Next = first
        все функція

Функція BubbleSort(first, last)
            якщо first == last
            то
                все функція
            все якщо
            swapCounter = 0
            для current  від first до last повторити
                якщо current.Value > current.Next.Value
                то
                    якщо current == first
                    то
                        first = current.Next
                    все якщо
                    якщо current.Next == last
                    то
                        last = current
                    все якщо
                    SwapNodes(current, current.Next)
                    current = current.Previous
                    swapCounter++
                все якщо
            все повторити
            якщо swapCounter>0
            то
                BubbleSort(first, last.Previous)
            все якщо
        все функція
```

Таблиця базових функцій двозв'язного списку

Ім'я	Призначення	Вихідний результат
PushBack(int value)	Додає значення у кінець списку	Нічого не повертає, змінює список
PushFront(int value)	Додає значення у початок списку	Нічого не повертає, змінює список
PopBack()	Обробляє елемент з кінця списку	Повертає значення обробленого елемента та модифікує список
PopFront()	Обробляє елемент з початку списку	Повертає значення обробленого елемента та модифікує список

3.2 Програмна реалізація

3.2.1 Вихідний код

```
namespace Lab5
{
    public class Node
    {
        public int Value { get; set; }
        public Node Next { get; set; }
        public Node Previous { get; set; }

        public Node(int value)
        {
            Value = value;
            Next = Previous = null;
        }
    }
}
```

```
using System;

namespace Lab5
{
    public class DoubleLinkedList
    {
        public Node Head, Tail;
        public int Count { get; private set; }

        public DoubleLinkedList()
        {
            Head = null;
            Tail = null;
            Count = 0;
        }

        public void PushBack(int value)
        {
```

```
            Node node = new Node(value);
            if (Count == 0)
            {
                Head = Tail = node;
            }
            else
            {
                Tail.Next = node;
                node.Previous = Tail;
                Tail = node;
            }
            Count++;
        }

        public void PushFront(int value)
        {
            Node node = new Node(value);
            if (Count == 0)
            {
                Head = Tail = node;
            }
            else
            {
                Head.Previous = node;
                node.Next = Head;
                Head = node;
            }
            Count++;
        }

        public int PopBack()
        {
            if (Count != 0)
            {
                int value = Tail.Value;
                Tail.Previous.Next = null;
                Tail = Tail.Previous;
                Count--;
                return value;
            }
            throw new Exception("List is empty");
        }

        public int PopFront()
        {
            if (Count != 0)
            {
                int value = Head.Value;
                Head.Next.Previous = null;
                Head = Head.Next;
                Count--;
                return value;
            }
            throw new Exception("List is empty");
        }

        public void SwapValues(Node node1, Node node2)
        {
            (node1.Value, node2.Value) = (node2.Value, node1.Value);
        }

        public void SwapNodes(Node node1, Node node2)
        {
```

```
            Node first = node1.Next == node2 ? node1 : node2;
            Node second = node1.Next == node2 ? node2 : node1;
            if (Head == first) Head = second;
            if (Tail == second) Tail = first;
            if (first.Previous is not null) first.Previous.Next = second;
            if (second.Next is not null) second.Next.Previous = first;
            second.Previous = first.Previous;
            first.Previous = second;
            first.Next = second.Next;
            second.Next = first;
        }

        public override string ToString()
        {
            string result = "";
            if (Count > 0) result += Head.Value;

            for (Node current = Head.Next; current is not null; current =
current.Next)
            {
                result += $", {current.Value}";
            }

            return result;
        }
```

Bubble sort для values:

```
public void BubbleSort(Node first, Node last)
{
    if (first == last) return;
    int swapCounter = 0;
    for (Node current = first; current != last; current = current.Next)
    {
        if (current.Value > current.Next.Value)
        {
            SwapValues(current, current.Next);
            /*if (current == first) first = current.Next;
            if (current.Next == last) last = current;
            SwapNodes(current, current.Next);
            current = current.Previous;*/

            swapCounter++;
        }
    }
    if (swapCounter>0) BubbleSort(first, last.Previous);
}
```

Bubble sort для nodes:

```
public void BubbleSort(Node first, Node last)
{
    if (first == last) return;
    int swapCounter = 0;
    for (Node current = first; current != last; current = current.Next)
    {
        if (current.Value > current.Next.Value)
        {
           /* SwapValues(current, current.Next); */
            if (current == first) first = current.Next;
            if (current.Next == last) last = current;
            SwapNodes(current, current.Next);
```

```
            current = current.Previous;

            swapCounter++;
        }
    }
    if (swapCounter>0) BubbleSort(first, last.Previous);
}
```

Main() для values:

```
using System;
using System.Diagnostics;

namespace Lab5
{
    class Program
    {
        static void Main(string[] args)
        {
            DoubleLinkedList list = new DoubleLinkedList();
            Random rand = new Random();
            for (int i = 0; i < 500; i++)
            {
                list.PushBack(rand.Next(300));
            }

            Console.WriteLine("Initial array:");
            Console.WriteLine(list);
            Console.WriteLine();
            Console.WriteLine();

            Stopwatch sw = new Stopwatch();
            sw.Start();

            list.BubbleSort(list.Head, list.Tail);
            sw.Stop();
            Console.WriteLine("Sorted values");
            Console.WriteLine("time in milliseconds: " +  sw.ElapsedMilliseconds);

            Console.WriteLine(list);
        }
    }
}
```

Main() для nodes:

```
using System;
using System.Diagnostics;

namespace Lab5
{
    class Program
    {
        static void Main(string[] args)
        {
            DoubleLinkedList list = new DoubleLinkedList();
            Random rand = new Random();
            for (int i = 0; i < 500; i++)
            {
```

```
                list.PushBack(rand.Next(300));
            }

            Console.WriteLine("Initial array:");
            Console.WriteLine(list);
            Console.WriteLine();
            Console.WriteLine();

            Stopwatch sw = new Stopwatch();
            sw.Start();

            list.BubbleSort(list.Head, list.Tail);
            sw.Stop();
            Console.WriteLine("Sorted nodes");
            Console.WriteLine("time in milliseconds: " +  sw.ElapsedMilliseconds);

            Console.WriteLine(list);
        }
    }
}
```

3.2.2 Приклади роботи

На рисунках 3.1 і 3.2 показані приклади роботи програми.

```
Initial array:
0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 2
9, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55,
56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82,
 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99


Sorted values
time in milliseconds: 0
0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 2
9, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55,
56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82,
 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99

Process finished with exit code 0.
```

Рисунок 3.1.1 – Час роботи програми при масиві, ініціалізованому у зростаючому порядку для сортування через values

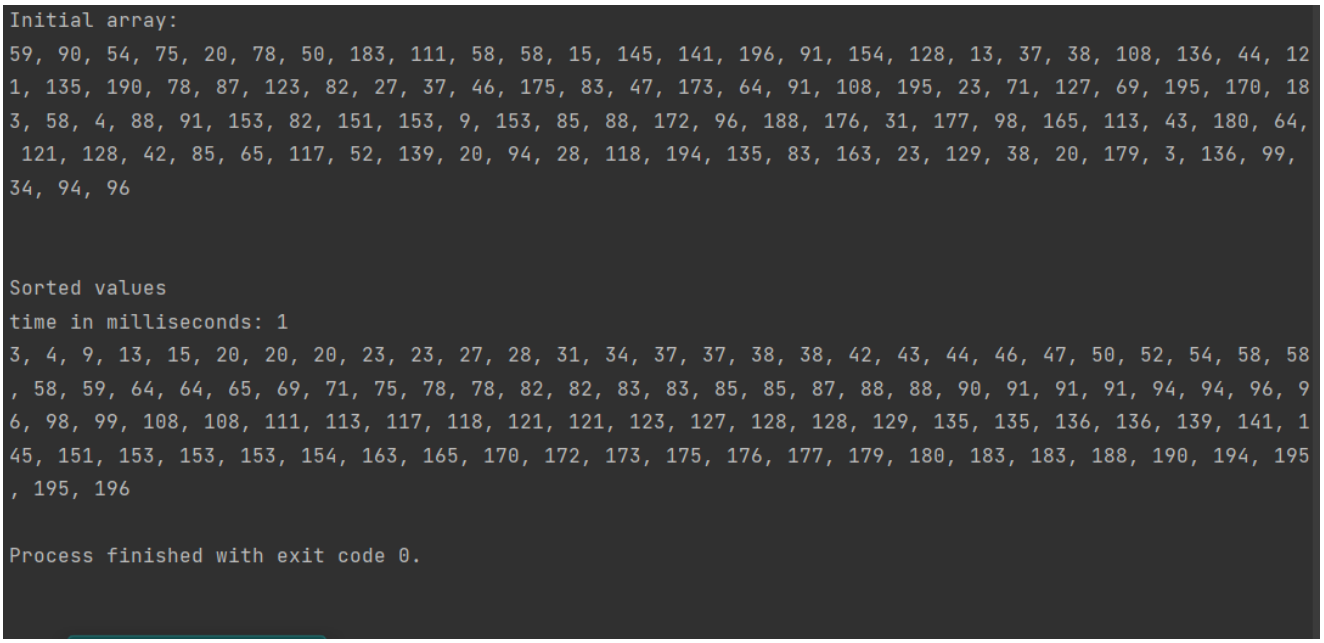

Рисунок 3.1.2 – Час роботи програми при масиві, ініціалізованому рандомізованими значеннями для сортування через values

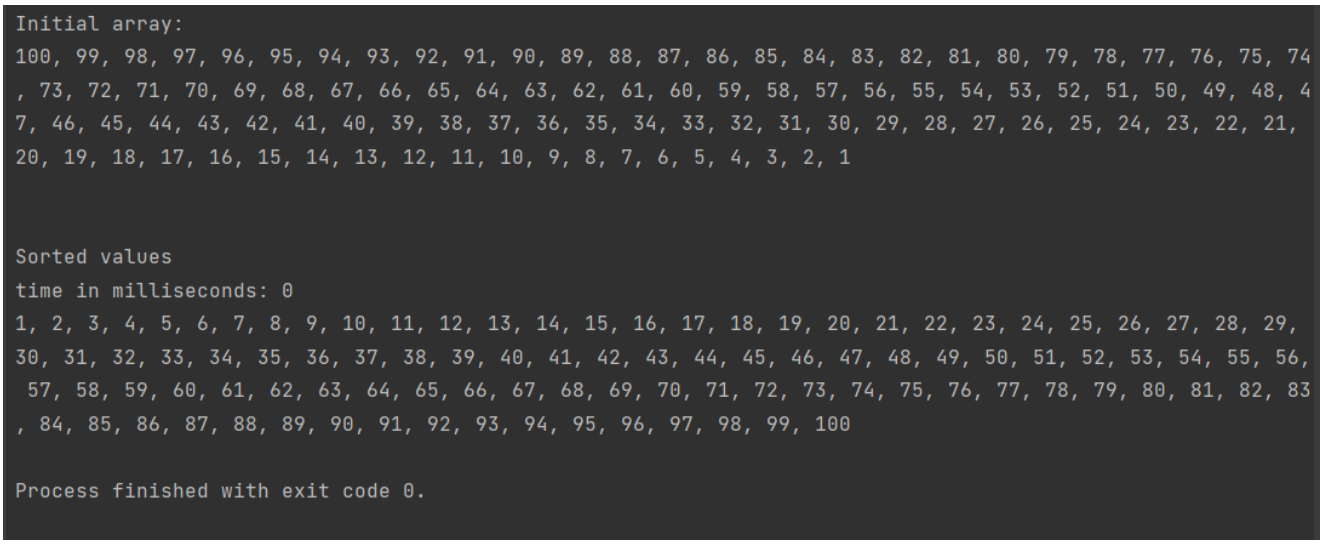

Рисунок 3.1.3 – Час роботи програми при масиві, ініціалізованому у спадаючому порядку для сортування через values

```
Initial array:
0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 2
9, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55,
56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82,
 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99


Sorted nodes
time in milliseconds: 0
0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 2
9, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55,
56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82,
 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99

Process finished with exit code 0.
```

Рисунок 3.2.1– Час роботи програми при масиві, ініціалізованому у зростаючому порядку для сортування через nodes

```
Initial array:
191, 195, 24, 189, 145, 111, 98, 108, 144, 135, 60, 62, 94, 104, 148, 116, 169, 166, 36, 187, 155, 128, 198
, 0, 149, 65, 35, 133, 123, 91, 58, 140, 35, 158, 101, 106, 6, 197, 25, 179, 141, 91, 129, 152, 11, 192, 79
, 190, 130, 162, 92, 53, 80, 155, 120, 62, 196, 24, 39, 79, 76, 164, 185, 53, 77, 120, 27, 135, 2, 42, 109,
 172, 141, 57, 46, 64, 198, 46, 188, 157, 185, 45, 2, 160, 198, 4, 59, 80, 38, 39, 109, 182, 157, 145, 102,
 176, 106, 76, 74, 91


Sorted nodes
time in milliseconds: 1
0, 2, 2, 4, 6, 11, 24, 24, 25, 27, 35, 35, 36, 38, 39, 39, 42, 45, 46, 46, 53, 53, 57, 58, 59, 60, 62, 62,
64, 65, 74, 76, 76, 77, 79, 79, 80, 80, 91, 91, 91, 92, 94, 98, 101, 102, 104, 106, 106, 108, 109, 109, 111
, 116, 120, 120, 123, 128, 129, 130, 133, 135, 135, 140, 141, 141, 144, 145, 145, 148, 149, 152, 155, 155,
157, 157, 158, 160, 162, 164, 166, 169, 172, 176, 179, 182, 185, 185, 187, 188, 189, 190, 191, 192, 195, 19
6, 197, 198, 198, 198

Process finished with exit code 0.
```

Рисунок 3.2.2 – Час роботи програми при масиві, ініціалізованому рандомізованими значеннями для сортування через nodes

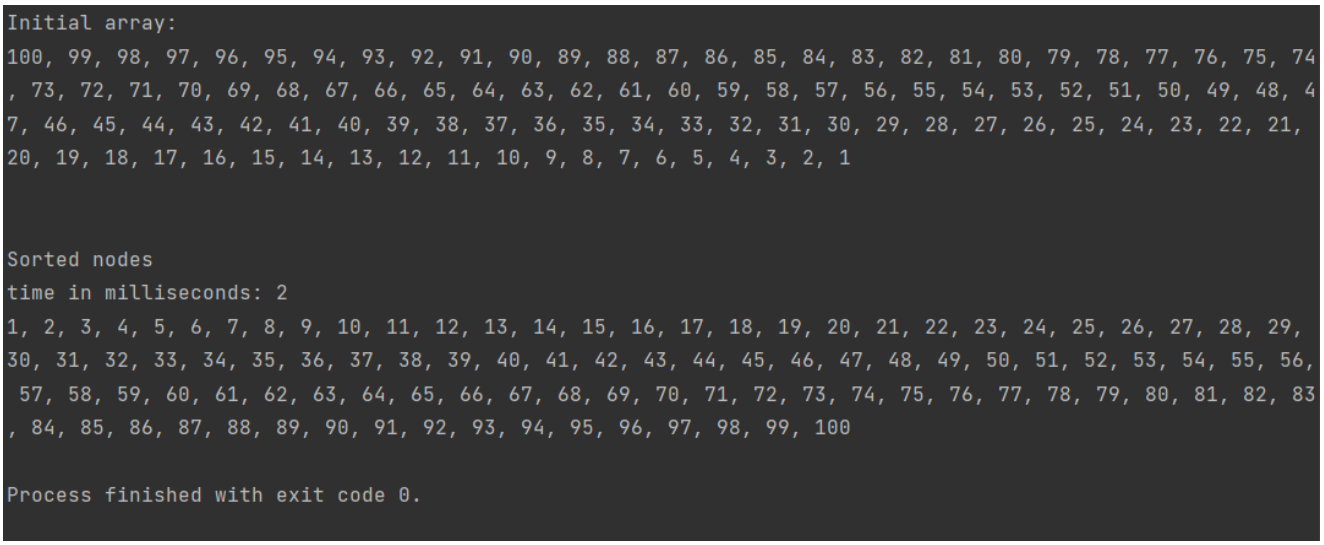

Рисунок 3.2.3 – Час роботи програми при масиві, ініціалізованому у спадаючому порядку для сортування через nodes

ВИСНОВОК

При виконанні даної лабораторної роботи я створила двозв’язний список та упорядкувала його елементи двома способами – за значеннями і за покажчиками на них. Я зрозуміла, що упорядкування за значеннями працює швидше та ефективніше, а також вивчила основні підходи формалізації евристичних алгоритмів і вирішення типових задач з їх допомогою.

КРИТЕРІЇ ОЦІНЮВАННЯ

За умови здачі лабораторної роботи до 20.04.2022 включно максимальний бал дорівнює – 5. Після 20.04.2022 максимальний бал дорівнює – 1.

Критерії оцінювання у відсотках від максимального балу:

– псевдокод алгоритму – 10%;

– програмна реалізація алгоритму – 80%;

– висновок – 10%.