Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут»
Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

ЛАБОРАТОРНА РОБОТА № 2
З дисципліни «Алгоритми та методи обчислень»

На тему «Обчислювальна складність алгоритмів сортування»

Виконав:
студент 2 курсу ФІОТ
групи ІВ-71
Мазан Я. В.
Залікова – 7109

Перевірив:
ст.вик. Порєв В. М.

Київ – 2019

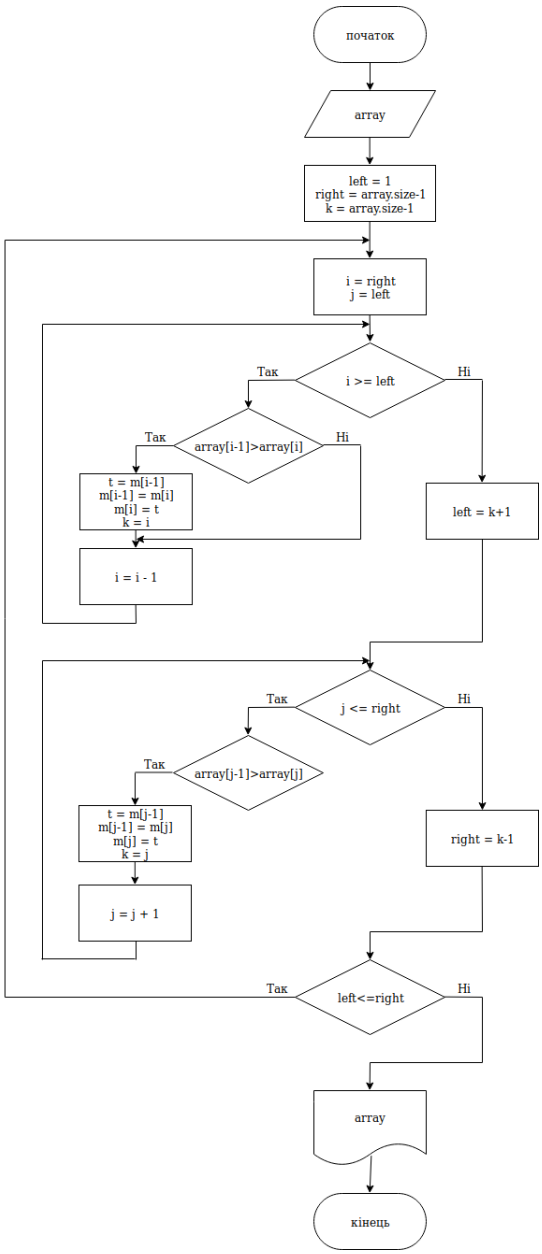
Мета:
Закріплення навичок практичної оцінки алгоритмічної складності логічних алгоритмів на прикладі алгоритмів сортування.

Завдання:
Використовуючи відповідний до варіанту алгоритм сортування написати програму сортування масиву даних. Застосовуючи дану програму, дослідити часову складність алгоритму сортування та порівняти її з теоретичною алгоритмічною складністю.

Завдання за номером у списку:

9	Алгоритм 4. Сортування Шейкером	$O(n^2)$

Блок-схема алгоритму:



Код програми:

Клас ShakerSort.kt: (містить алгоритм сортування та рахує час його виконання)

```
package com.example.amclab2
class ShakerSort {
    val time: Double
    val arr: Array<Double>
    val sorted:  Array<Double>
    constructor (array:  Array<Double>) {
        this.arr = array
        val beginTime: Long = System.currentTimeMillis()
        /**
         * sort array
         */
        this.sorted = sort()
        val endTime: Long = System.currentTimeMillis()
        this.time = (endTime-beginTime)/Math.pow(10.0,3.0)
    }
    //sort by ascending
    fun sort() : Array<Double> {
        var sortedArr:  Array<Double> = arr
        var left: Int = 1
        var right: Int = arr.size - 1
        var k: Int = arr.size - 1
        do {
            for (j: Int in right downTo left) {
                if (sortedArr[j - 1] > sortedArr[j]) {
                    val t: Double = sortedArr[j - 1]
                    sortedArr[j - 1] = sortedArr[j]
                    sortedArr[j] = t
                    k = j
                }
                left = k + 1
            }
            for (j: Int in left..right) {
                if (sortedArr[j - 1] > sortedArr[j]) {
                    val t: Double = sortedArr[j - 1]
                    sortedArr[j - 1] = sortedArr[j]
                    sortedArr[j] = t
                    k = j
                }
                right = k - 1
            }
        } while(left<=right)
        return sortedArr
    }
}
```

Функція генерування тестових 10 масивів:

```
/**
 * @input Unit
 * @return 10 arrays of different size with random double numbers in range [-100, 100] rounded to 0 decimal points
 */
fun generateArr() : Array<Array<Double>> {
    return Array(10, {i -> Array(5+i*i*20, {j -> Math.round((Math.random()-0.5)*200).toDouble()})})
}
```

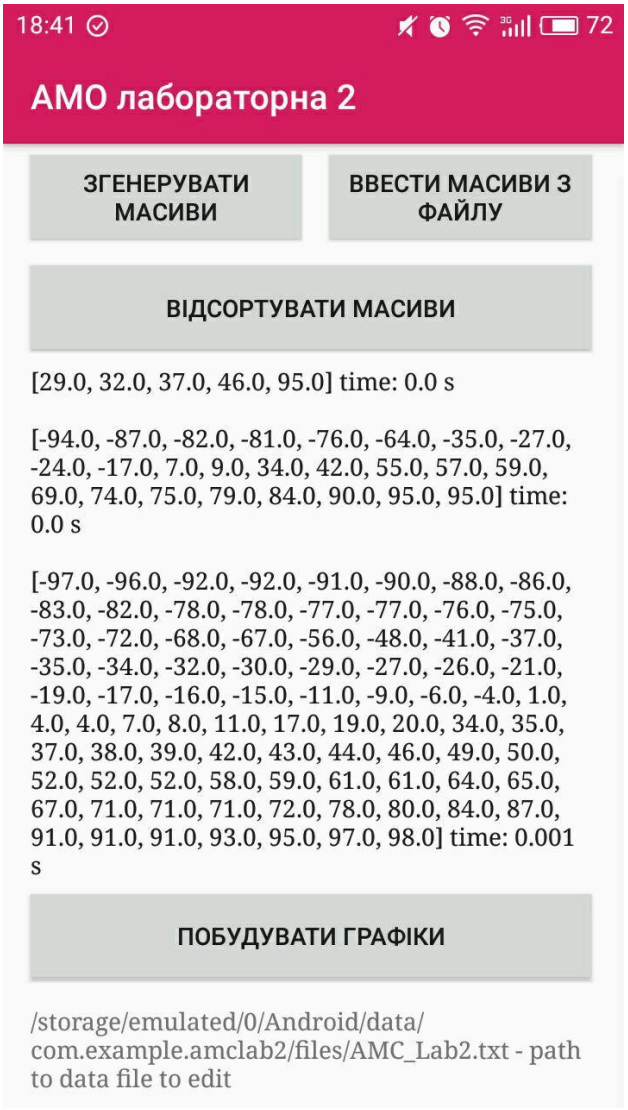
Функції зчитування та обробки текстового файлу з масивами:

```
/**
```

```
 * receive information from our datafile
 */
fun generateFromFile(view: View) {
    try {
        val sdPath = File(filePath + "/" + fileName)
        if (!sdPath.exists())
            initialFileCreate()
        val stringSdcardPath: String = File(sdPath.absolutePath.toString().plus("/")).toString()
        val file: File = File(stringSdcardPath)
        try {
            val reader = BufferedReader(FileReader(file))
            var str = ""
            var line: String? = ""
            while (line != null) {
                line = reader.readLine()
                str += line.plus("\n")
            }
            this.arrays = parseInputFromFile(str)
            showOnCanvas(view, parseForTextView(arrays))
            button3.setOnClickListener{ v -> generateSortedArr(v) }
        } catch (e: Exception) {
            e.printStackTrace()
        }
    } catch (e: FileNotFoundException) {
        Toast.makeText(this, "The input file has been deleted!", Toast.LENGTH_SHORT).show()
    }

/**
 * @input string representation of a file with lines connected by \n
 * @return 10 arrays of different size with random double numbers in range [0,1] rounded to 4 decimal points
 */
fun parseInputFromFile(src: String) : Array<Array<Double>> {
    val split: List<String> = src.split("\n").filter{i -> i != ""}
    val lines_length: Int = split.size
    val comment_line_begin: Int?
    if (split.contains("/*"))
        comment_line_begin = split.indexOf("/*")+1
    else
        comment_line_begin = split.size
    //delete everything after comment sign
    val array: List<List<Double>> = split.dropLast(lines_length-comment_line_begin+1).map{ s -> s.split(" ").map { i
-> i.toDouble() } }
    return array.map{e -> e.toTypedArray()}.toTypedArray()
}
```

Результати виконання програми:



Висновок:

Під час виконання лабораторної роботи мною були закріплені знання з базових понять алгоритмів, навички практичної оцінки алгоритмічної складності логічних алгоритмів на прикладі алгоритму шейкерного сортування.