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

Кафедра інформатики та програмної інженерії

Звіт

з лабораторної роботи № 1 з дисципліни
«Проектування алгоритмів»

«Проектування алгоритмів зовнішнього сортування»

Варіант 4

Виконав студент ІП-15, Буяло Дмитро Олександрович
(шифр, прізвище, ім'я, по батькові)

Перевірив Соколовський Владислав Володимирович
(прізвище, ім'я, по батькові)

Київ 2022

ЗМІСТ

1 МЕТА ЛАБОРАТОРНОЇ РОБОТИ

Мета роботи – вивчити основні алгоритми зовнішнього сортування та способи їх модифікації, оцінити поріг їх ефективності.

Індивідуальне завдання

Варіант 4

2 ЗАВДАННЯ

Для алгоритму багатофазного сортування розробити та записати алгоритм зовнішнього сортування за допомогою псевдокоду (чи іншого способу за вибором).

Виконати програмну реалізацію алгоритму на будь-якій мові програмування та відсортувати випадковим чином згенерований масив цілих чисел, що зберігається у файлі (розмір файлу має бути не менше 10 Мб, можна значно більше).

Здійснити модифікацію програми і відсортувати випадковим чином згенерований масив цілих чисел, що зберігається у файлі розміром не менше ніж 32Гб. Досягти швидкості сортування з розрахунку 1Гб на 3хв або менше.

Рекомендується попередньо впорядкувати серії елементів довжиною, що займає не менше 100Мб або використати інші підходи для пришвидшення процесу сортування.

Зробити узагальнений висновок з лабораторної роботи, у якому порівняти базову та модифіковану програми. У висновку деталізувати, які саме модифікації було виконано і який ефект вони дали.

3 ВИКОНАННЯ

3.1 Псевдокод алгоритму

```
POLYPHASE_STEP(s, N)
    (l, F, d) ← HORIZONT_DISTR_M(s, N)
    while l > 0 do
        while d[N – 1] ≠ < > И F[N – 1] ≠ 0 do
            k1 ← k2 ← 0
            for i ← 1 to N – 1 do
                if F[i] > 0 then  k1 ← k1 + 1, m1[k1] ← i
                else              k2 ← k2 + 1, m2[k2] ← i
            if k1 = N – 1 then
                F[N] ← F[N] + 1
            else
                MERGERUN_N(d[N], k2, d[m2[1]], …, d[m2[k2]])
            for k ← 1 to k1 do
                F[m1[k]] ← F[m1[k]] – 1
        l ← l – 1
        (d[1], d[2], d[3],…, d[N]) ← (d[N], d[1], d[2], …, d[N – 1])
    return d[1]
```

© Кафедра вычислительных систем ФГОБУ ВПО «СибГУТИ»


```
HORIZONT_DISTR_M(s, N)
    for i = 1 to N – 1 do
        A[i] ← F[i] ← 1, d[i] ← < >
    A[N] ← F[N] ← 1, d[N] ← < >, l ← j ← 1
    while a ≠ < > do
        last[j] ← COPYRUN(a, d[j]),
        F[j] ← F[j] – 1
        if  F[j] < F[j + 1] then  j ← j + 1
        else if F[j] ≠ 0 then j ← 1
        else
            l ← l + 1, x ← A[1]
            for i = 1 to N – 1 do
                F[i] ← x + A[i + 1] – A[i]
                A[i] ← x + A[i + 1]
            j ← 1
        CHECKMERGE(a, d[j], last[j])
    return (l, F, d)
```

федра вычислительных систем ФГОБУ ВПО «СибГУТИ»

```
COPYRUN(a, b)
    cur ← first(a), a ← rest(a)
    if a ≠ < > then
        next ← first(a)
    else
        next ← cur – 1
    while a ≠ < > И next ≥ cur do
        b ← b & cur
        cur ← next
        a ← rest(a)
        if a ≠ < > then
            next ← first(a)
    b ← b & cur
    return cur
```

```
CHECKMERGE(a, d, last)
    if a ≠ < > then
        if first(a) > last then
            COPYRUN(a, d)
```

```
MERGERUN_N(d, N, s_1, …, s_N )
    k ← 0
    for i = 1 to N do
        if s_i ≠ < > then   c_i ← first(s_i), k ← k + 1, m_k ← i
    k' ← (k > 0)
    while k > 0 do
        j' ← 1, i' ← m_1, c' ← c_i'
        for j = 2 to k do
            i ← m_j
            if  c_i < c' then j' ← j, i' ← m_j, c' ← c_i'
        d ← d & c',   s_i' ← rest(s_i')
        if s_i' = < > ИЛИ c_i' ≥ first(s_i')  then
            m_k ↔ m_j' , k ← k – 1
        else c_i' ← first(s_i')
    return (k', d)
```

m – карта непустых серий
m_1 – первая непустая серия
m_k – последняя непустая серия

для удаления последовательности с пустой серией достаточно поменять ее местами с последней непустой серией и уменьшить количество k непустых серий

3.2 Програмна реалізація алгоритму

```
import java.io.*;
import java.nio.charset.StandardCharsets;
import java.nio.file.Files;
import java.nio.file.Path;
import java.util.*;

import static java.lang.Integer.MAX_VALUE;

public class Main {
    public static void main(String[] args) throws FileNotFoundException {
        Scanner scan = new Scanner(System.in);
        System.out.print("Enter the name of file: ");
        String name = scan.nextLine() + ".txt";
        System.out.print("In which number system to create a file? (Kb/Mb/Gb): ");
        String system = scan.nextLine().toUpperCase();
        System.out.print("Enter file size in " + system + ": ");
        long size = scan.nextLong();
        long time = System.currentTimeMillis();
        try {
            long count = generateFile(name, system, size);
            System.out.print("A file with the name \"" + name + "\" and size of " + size + system +
                    " was successfully generated in ");
            System.out.println((double) (System.currentTimeMillis() - time) + "ms!");
            System.out.print("\nHow many files to use for sorting? N = ");
            int n = scan.nextInt();
//          time = System.currentTimeMillis();
//          ArrayList<ArrayList<Integer>> arr = fibonacci(6, n-1);
//          System.out.println(arr);
//          System.out.println((double) (System.currentTimeMillis() - time) + "ms!");
            int i = polyPhaseSort(name, n, count, system); // отсортированный файл
        } catch (Exception e) {
            System.err.println(e);
        }
//      File file = new File(name);
//      Scanner reader = new Scanner(file);
    }
    public static ArrayList<ArrayList<Integer>> fibonacci(int rows, int cols) {
//      int rows = 6;
        int sum;
        int res = 2;
        ArrayList<ArrayList<Integer>> arr = new ArrayList<>();
        for (int i = 0; i < rows; i++) {
            res--;
            if(res==0) {
                res=cols+1;
```

```
            }
            sum = 0;
            arr.add(new ArrayList<>());
            for (int j = 0; j < cols; j++) {
                if(i == 0 && j == 0) {
                    arr.get(i).add(1);
                }
                else if(i==0) {
                    arr.get(i).add(0);
                }
                else if (j == cols - 1) {
                    arr.get(i).add(j, arr.get(i - 1).get(0));
                }
                else {
                    arr.get(i).add(j, arr.get(i - 1).get(0) + arr.get(i - 1).get(j + 1));
                }
                sum +=arr.get(i).get(j);
            }
            arr.get(i).add(sum);
            arr.get(i).add(res);
        }
        return arr;
    }

    public static int polyPhaseSort(String name, int n, long count, String system) throws IOException {
        for(int i=1;i<n;i++) {
            File file = new File("T" + i + ".txt");
            file.createNewFile();
        }
        ArrayList<ArrayList<Integer>> fibList = fibonacci(20, n-1);
        BufferedReader reader = Files.newBufferedReader(Path.of(name), StandardCharsets.UTF_8);
        File file = new File(name);
        long time = System.currentTimeMillis();
        int gig = 500000000;//java -XX:+PrintFlagsFinal -version | findstr /i "HeapSize PermSize ThreadStackSize"
        int k=0;
//      switch (system) {
//          case "GB":
//              size *= 1024 * 1024 * 1024;
//              break;
//          case "MB":
//              size *= 1024 * 1024;
//              break;
//          case "KB":
//              size *= 1024;
//              break;
//      }
        while (count/fibList.get(k).get(n-1)>gig) {
            k++;
        }
        System.out.println(k);
```

```
        int[] arr = new int[500000000];
        for(int i=0;i<50000000;i++) {
            arr[i] = Integer.parseInt((reader.readLine()));
        }
        System.out.println("\nArray read = "+ (double) (System.currentTimeMillis() - time) + "ms!");
        time = System.currentTimeMillis();
        Arrays.parallelSort(arr);
        System.out.println("Array.parallelSort = "+ (double) (System.currentTimeMillis() - time) + "ms!");
//        for(int i=0;i<10;i++) {
//            System.out.println(arrAm[i]);
//        }
//        System.out.println();
//        for(int i=990;i<1000;i++) {
//            System.out.println(arr[i]);
//        }
        return 1;
    }

    public static long generateFile(String name, String system, long size) throws IOException {
        File file = new File(name);
        file.createNewFile();
        FileWriter in = new FileWriter(file);
        Random rand = new Random();
        StringBuilder s = new StringBuilder();
        long count =0;
        switch (system) {
            case "GB":
                size *= 1024 * 1024 * 1024;
                break;
            case "MB":
                size *= 1024 * 1024;
                break;
            case "KB":
                size *= 1024;
                break;
        }
        while (file.length() < size) {
            for (int i = 0; i < 10240; i++) {
                s.append(rand.nextInt(MAX_VALUE));
                s.append('\n');
            }
            in.write(s.toString());
            s.setLength(0);
            count++;
        }
        in.close();
        return count*10240;
    }
}
//for (int i=0;i<10;i++) {
```

```
//for (int j=0;j<n+1;j++) {
//System.out.print(fibList.get(i).get(j)+" ");
//}
//System.out.println();
//}
```

3.2.1 Вихідний код