**Московский автомобильно-дорожный государственный технический университет (МАДИ)**

**Кафедра «Высшая математика»**

**Практическая работа №2**

**по дисциплине**

**«Структуры и алгоритмы обработки данных»**

«Методы сортировки»

Выполнил: студент группы 1бПМ1

Грицук Максим Вадимович

Вариант 5

Преподаватель: Кутейников И. А.

Москва 2021

Цель работы

Реализовать заданный метод сортировки числового массива в соответствии с индивидуальным заданием. Для всех вариантов добавить реализацию быстрой сортировки (quicksort). Оценить время работы каждого алгоритма сортировки и сравнить его со временем стандартной функции сортировки, используемой в выбранном языке программирования.

Вариант 5. Сортировка Слиянием.

Ход работы

В соответствии с заданием, реализовала алгоритм быстрой сортировки на языке Java:

public class QuickSort {

public static void sortArray(int[] \_array, int \_low, int \_high) {

if (\_array.length == 0)

return;

if (\_low >= \_high)

return;

int \_middle = \_low + (\_high - \_low) / 2;

int \_support = \_array[\_middle];

int i = \_low, j = \_high;

while (i <= j) {

while (\_array[i] < \_support) {

i++;

}

while (\_array[j] > \_support) {

j--;

}

if (i <= j) {

int \_temp = \_array[i];

\_array[i] = \_array[j];

\_array[j] = \_temp;

i++;

j--;

}

}

if (\_low < j)

sortArray(\_array, \_low, j);

if (\_high > i)

sortArray(\_array, i, \_high);

}

}

Также реализовала алгоритм сортировки Слиянием:

public class MergeSort{

private static int[] mergeArray(int[] \_arrayA, int[] \_arrayB){

int[] \_arrayC = new int[\_arrayA.length + \_arrayB.length];

int \_posA = 0 , \_posB = 0;

for (int i = 0; i< \_arrayC.length; i++){

if (\_posA == \_arrayA.length){

\_arrayC[i] = \_arrayB[\_posB];

\_posB++;

} else if (\_posB == \_arrayB.length) {

\_arrayC[i] = \_arrayA[\_posA];

\_posA++;

} else if (\_arrayA[\_posA] < \_arrayB[\_posB]) {

\_arrayC[i] = \_arrayA[\_posA];

\_posA++;

} else {

\_arrayC[i] = \_arrayB[\_posB];

\_posB++;

}

}

return \_arrayC;

}

public static int[] sortArray(int[] \_arrayA){

if (\_arrayA == null)

return null;

if (\_arrayA.length < 2)

return \_arrayA;

int[] \_arrayB = new int[\_arrayA.length / 2];

System.*arraycopy*(\_arrayA, 0, \_arrayB, 0, \_arrayB.length);

int[] \_arrayC = new int[\_arrayA.length - \_arrayB.length];

System.*arraycopy*(\_arrayA, \_arrayB.length, \_arrayC, 0,\_arrayC.length);

\_arrayB = *sortArray*(\_arrayB);

\_arrayC = *sortArray*(\_arrayC);

return *mergeArray*(\_arrayB,\_arrayC);

}

}

Сравнение времени выполнения алгоритмов для массива размером 10.000:

|  |  |  |  |
| --- | --- | --- | --- |
| N | Merge sort | Quick sort | Java sort |
| 1 | 10 | 5 | 0 |
| 2 | 12 | 6 | 1 |
| 3 | 10 | 6 | 1 |
| 4 | 11 | 7 | 1 |
| 5 | 10 | 7 | 1 |
| Среднее время(мс) | 10,6 | 6,2 | 0,8 |

Сравнение времени выполнения алгоритмов для массива размером 100.000:

|  |  |  |  |
| --- | --- | --- | --- |
| N | Merge sort | Quick sort | Java sort |
| 1 | 46 | 14 | 3 |
| 2 | 25 | 12 | 2 |
| 3 | 30 | 12 | 2 |
| 4 | 24 | 13 | 2 |
| 5 | 43 | 13 | 2 |
| Среднее время(мс) | 33,6 | 12,8 | 2,2 |

Сравнение времени выполнения алгоритмов для массива размером 1.000.000:

|  |  |  |  |
| --- | --- | --- | --- |
| N | Merge sort | Quick sort | Java sort |
| 1 | 174 | 60 | 5 |
| 2 | 147 | 57 | 5 |
| 3 | 146 | 63 | 5 |
| 4 | 186 | 67 | 5 |
| 5 | 150 | 66 | 5 |
| Среднее время(мс) | 160,6 | 62,6 | 5 |

Лучший результат показала встроенная в java сортировка, худший - сортировка слиянием.

При сортировках использовался одинаковый массив случайно сгенерированных чисел.

Код программы:

import java.util.Arrays;

import java.util.Scanner;

public class Main {

public static int keyboardInput(String \_text) {

Scanner \_in = new Scanner(System.*in*);

System.*out*.print(\_text);

return (\_in.nextInt());

}

private static void firstMenu(int[] \_array) {

System.*out*.println("1 - Enter values manually");

System.*out*.println("2 - Use random values");

int \_choice = *keyboardInput*("Choose an action: ");

if (\_choice==1){

for (int i = 0; i< \_array.length; i++){

\_array[i] = *keyboardInput*("Enter " + (i+1) + " number: ");

}

}else if (\_choice == 2){

int \_min = *keyboardInput*("Enter the minimum value: ");

int \_max = *keyboardInput*("Enter the maximum value: ");

for ( int i = 0; i < \_array.length; i++){

\_array[i] = \_min + (int) (Math.*random*() \* (\_max-\_min+1));

}

}else{

System.*out*.println("You have selected a non-existent menu item");

*firstMenu*(\_array);

}

}

private static void speedSort(int[] \_array) {

int[][] \_copyArrays = new int[3][\_array.length];

for (int i = 0; i<3; i++){

\_copyArrays[i] = \_array;

}

long \_startTime = System.*currentTimeMillis*();

MargeSort.*sortArray*(\_copyArrays[0]);

long \_endTime = System.*currentTimeMillis*();

System.*out*.println("That merge sort took " + (\_endTime - \_startTime) + " milliseconds");

\_startTime = System.*currentTimeMillis*();

QuickSort.*sortArray*(\_copyArrays[1], 0, \_copyArrays[1].length-1);

\_endTime = System.*currentTimeMillis*();

System.*out*.println("That quick sort took " + (\_endTime - \_startTime) + " milliseconds");

\_startTime = System.*currentTimeMillis*();

Arrays.*sort*(\_copyArrays[2]);

\_endTime = System.*currentTimeMillis*();

System.*out*.println("That java sort took " + (\_endTime - \_startTime) + " milliseconds");

}

private static void numberB(int[] \_array) {

int[] \_min = new int[3];

System.*arraycopy*(\_array, 0, \_min, 0, 3);

Arrays.*sort*(\_min);

for (int i = 3; i<\_array.length; i++){

if(\_min[2]>\_array[i]){

\_min[2]=\_array[i];

Arrays.*sort*(\_min);

}

}

System.*out*.println(\_min[0] + " " + \_min[1] + " " + \_min[2]);

System.*out*.println((\_min[0] + \_min[1] + \_min[2]));

}

private static int[] secondMenu(int[] \_array) {

System.*out*.println("1 - Merge sort");

System.*out*.println("2 - Quick sort");

System.*out*.println("3 - Java sort");

System.*out*.println("4 - Speed sort test");

System.*out*.println("5 - Three minimum numbers and their sum");

System.*out*.println("6 - Exit");

int \_choice = *keyboardInput*("Choose an action: ");

if (\_choice==1) {

\_array = MergeSort.*sortArray*(\_array);

\_array = *secondMenu*(\_array);

}else if (\_choice==2){

QuickSort.*sortArray*(\_array, 0, \_array.length-1);

\_array = *secondMenu*(\_array);

}else if (\_choice==3){

Arrays.*sort*(\_array);

\_array = *secondMenu*(\_array);

}else if (\_choice==4){

*speedSort*(\_array);

\_array = *secondMenu*(\_array);

}else if (\_choice==5){

*numberB*(\_array);

\_array = *secondMenu*(\_array);

}else if (\_choice==6){

System.*out*.println("Exit");

}else {

System.*out*.println("You have selected a non-existent menu item");

\_array= *secondMenu*(\_array);

}

return \_array;

}

public static void main(String[] args) {

System.*out*.println("Hello!");

int[] \_array = new int[*keyboardInput*("Enter the length of the array: ")];

*firstMenu*(\_array);

System.*out*.println("Array: " + Arrays.*toString*(\_array));

*secondMenu*(\_array);

System.*out*.println("Array: " + Arrays.*toString*(\_array));

}

}

**Сортировка** **Слиянием**(**Merge** **sort**) — **Сложность** алгоритма: O (NlogN);

**Сложность** **быстрой** **сортировки** – O (NlogN)