**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра вычислительной техники.**

отчет

**по лабораторной работе №2**

**по дисциплине «Алгоритмы и структуры данных»**

Тема: Алгоритмы «разделяй и властвуй».

|  |  |  |
| --- | --- | --- |
| Студент гр. 1302 |  | Фомин В. В. |
| Преподаватель |  |  |

Санкт-Петербург

2022

Постановка задачи.

Реализовать следующие алгоритмы сортировки:

1. Сортировка вставками (Insertion sort)

2. Сортировка выбором (Selection sort)

3. Пузырьковая сортировка (Bubble sort)

4. Сортировка слиянием (Merge sort)

5. Сортировка Шелла (Shell sort)

6. Быстрая сортировка (Quick sort)

Для каждого алгоритма сортировки указать временную асимптотическую

сложность для лучшего, худшего случая и среднего случая, а также

пространственную сложность, подкрепив это логическими построениями (в

меру своих сил). Свести получившиеся результаты в таблицу. Построить

график зависимости времени выполнения от размера входных данных и

определить временную асимптотическую сложность для лучшего, худшего,

среднего случая практически. Определить наиболее быстрый алгоритм

сортировки. Сравнить скорость его выполнения с одним из алгоритмов

сортировки из базовых библиотек языка.

Теоретическая временная сложность.

|  |  |  |  |
| --- | --- | --- | --- |
| сортировка… | лучший случай | средний случай | худший случай |
| вставками | O(n) | O(n^2) | O(n^2) |
| выбором | O(n^2) | O(n^2) | O(n^2) |
| пузырьковая | O(n) | O(n^2) | O(n^2) |
| слиянием | O(n\*log(n)) | O(n\*log(n)) | O(n\*log(n)) |
| Шелла | O(n) | O(n\*log(n)^2) | O(n\*log(n)^2) |
| быстрая | O(n\*log(n)) | O(n\*log(n)) | O(n^2) |

Пространственная сложность.

|  |  |
| --- | --- |
| сортировка… |  |
| вставками | O(1) |
| выбором | O(1) |
| пузырьковая | O(1) |
| слиянием | O(n) |
| Шелла | O(1) |
| быстрая | O(log(n)) |

Эмпирическая оценка временной сложности



Пример работы

размер массива: 1422

InsertionSort, тест 0 корректен

SelectionSort, тест 0 корректен

BubbleSort, тест 0 корректен

MergeSort, тест 0 корректен

ShellSort, тест 0 корректен

QuickSort, тест 0 корректен

…

размер массива: 10574

InsertionSort, тест 34 корректен

SelectionSort, тест 34 корректен

BubbleSort, тест 34 корректен

MergeSort, тест 34 корректен

ShellSort, тест 34 корректен

QuickSort, тест 34 корректен

…

размер массива: 65409

StandartSort, тест 19 выполнен за 6 мс

InsertionSort, тест 19 выполнен за 3562 мс

SelectionSort, тест 19 выполнен за 2792 мс

BubbleSort, тест 19 выполнен за 15203 мс

MergeSort, тест 19 выполнен за 11 мс

ShellSort, тест 19 выполнен за 15 мс

QuickSort, тест 19 выполнен за 9 мс

Листинг

Среда разработки: Intellij Idea. Язык: Java.

import java.io.FileWriter;  
import java.io.IOException;  
import java.util.ArrayList;  
import java.util.Arrays;  
  
class Tester {  
 public static void main(String[] args) throws IOException {  
 ArrayList<Sortable> sortings = new ArrayList<>();  
 sortings.add(new Insertion());  
 sortings.add(new Selection());  
 sortings.add(new Bubble());  
 sortings.add(new Merge());  
 sortings.add(new Shell());  
 sortings.add(new Quick());  
 //sortings.add(new Incorrect());  
 *correctnessTest*(sortings);  
 int[][] result = *stressTest*(sortings);  
 FileWriter fw = new FileWriter("lab3.csv");  
 for (int i = 0; i < result.length; i++){  
 fw.write(i == 0 ? "StandartSort; ": sortings.get(i - 1) + "; ");  
 for (int j = 0; j < result[0].length; j++){  
 fw.write(result[i][j] + "; ");  
 }  
 fw.write('\n');  
 }  
 fw.write("time; ");  
 int n = 20;  
 for (int j = 0; j < result[0].length; j++){  
 n \*= 1.5;  
 fw.write(n + "; ");  
 }  
 fw.close();  
 }  
  
 public static void correctnessTest(ArrayList<Sortable> sortings) {  
 for (int i = 0; i < 35; i++) {  
 int n = (1 << (i / 10)) \* 1000;  
 int[] array = new int[n + (int) (n \* Math.*random*())];  
 for (int j = 0; j < array.length; j++) {  
 array[j] = (int) (Math.*random*() \* Integer.*MAX\_VALUE*);  
 }  
 int[] reference = Arrays.*copyOf*(array, array.length);  
 Arrays.*sort*(reference);  
 System.*out*.println("размер массива: " + array.length);  
 for (Sortable sorting : sortings) {  
 int[] temp = Arrays.*copyOf*(array, array.length);  
 sorting.sort(temp);  
 if (*isEqualsIntArrays*(reference, temp)) {  
 System.*out*.println(sorting + ", тест " + i + " корректен");  
 } else {  
 System.*out*.println(sorting + ", тест " + i + " НЕкорректен");  
 System.*out*.println("ожидалось: " + Arrays.*toString*(reference));  
 System.*out*.println("получено: " + Arrays.*toString*(temp));  
 }  
 }  
 }  
 }  
  
 public static int[][] stressTest(ArrayList<Sortable> sortings) {  
 int[][] res = new int[sortings.size() + 1][20];  
 int n = 20;  
 for (int i = 0; i < 20; i++) {  
 n \*= 1.5;  
 System.*out*.println("размер массива: " + n);  
 int[] array = new int[n + (int) (n \* Math.*random*())];  
 for (int j = 0; j < array.length; j++) {  
 array[j] = (int) (Math.*random*() \* Integer.*MAX\_VALUE*);  
 }  
 int[] reference = Arrays.*copyOf*(array, array.length);  
 long time = System.*currentTimeMillis*();  
 Arrays.*sort*(reference);  
 time = System.*currentTimeMillis*() - time;  
 System.*out*.println("StandartSort, тест " + i + " выполнен за " + time + " мс");  
 int k = 0;  
 res[k++][i] = (int) time;  
 for (Sortable sorting : sortings) {  
 int[] temp = Arrays.*copyOf*(array, array.length);  
 time = System.*currentTimeMillis*();  
 sorting.sort(temp);  
 time = System.*currentTimeMillis*() - time;  
 System.*out*.println(sorting + ", тест " + i + " выполнен за " + time + " мс");  
 res[k++][i] = (int) time;  
 }  
 }  
 return res;  
 }  
  
 public static boolean isEqualsIntArrays(int[] a, int[] b) {  
 if (a.length != b.length) {  
 return false;  
 }  
 for (int i = 0; i < a.length; i++) {  
 if (a[i] != b[i]) {  
 return false;  
 }  
 }  
 return true;  
 }  
}  
  
/\*  
class Manual {  
 public static void main(String[] args) {  
  
 }  
}  
 \*/  
  
interface Sortable {  
 public void sort(int[] array);  
}  
  
class Insertion implements Sortable {  
 @Override  
 public void sort(int[] array) {  
 for (int i = 1; i < array.length; i++) {  
 for (int j = i; j > 0; j--) {  
 if (array[j] < array[j - 1]) {  
 int t = array[j];  
 array[j] = array[j - 1];  
 array[j - 1] = t;  
 }  
 }  
 }  
 }  
  
 @Override  
 public String toString() {  
 return "InsertionSort";  
 }  
}  
  
class Selection implements Sortable {  
 @Override  
 public void sort(int[] array) {  
 for (int i = 0; i < array.length; i++) {  
 int min = i;  
 for (int j = i + 1; j < array.length; j++) {  
 if (array[j] < array[min]) min = j;  
 }  
 int t = array[i];  
 array[i] = array[min];  
 array[min] = t;  
 }  
 }  
  
 @Override  
 public String toString() {  
 return "SelectionSort";  
 }  
}  
  
class Bubble implements Sortable {  
 @Override  
 public void sort(int[] array) {  
 for (int i = 0; i < array.length; i++) {  
 for (int j = i + 1; j < array.length; j++) {  
 if (array[j] < array[i]) {  
 int t = array[j];  
 array[j] = array[i];  
 array[i] = t;  
 }  
 }  
 }  
 }  
  
 @Override  
 public String toString() {  
 return "BubbleSort";  
 }  
}  
  
class Merge implements Sortable {  
 @Override  
 public void sort(int[] array) {  
 int[] temp = new int[array.length];  
 mergeSort(array, temp, 0, array.length - 1);  
 }  
  
 private void mergeSort(int[] array, int[] temp, int left, int right) {  
 if (right - left == 0) {  
 return;  
 } else {  
 mergeSort(array, temp, left, (left + right) / 2);  
 mergeSort(array, temp, (left + right) / 2 + 1, right);  
 int i = left, j = (left + right) / 2 + 1;  
 for (int k = 0; k < right - left + 1; k++) {  
 if (i == (left + right) / 2 + 1) {  
 temp[k] = array[j++];  
 } else if (j == right + 1) {  
 temp[k] = array[i++];  
 } else if (array[i] < array[j]) {  
 temp[k] = array[i++];  
 } else {  
 temp[k] = array[j++];  
 }  
 }  
 for (int k = 0; k < right - left + 1; k++) {  
 array[left + k] = temp[k];  
 }  
 }  
 }  
  
 @Override  
 public String toString() {  
 return "MergeSort";  
 }  
}  
  
class Shell implements Sortable {  
 @Override  
 public void sort(int[] array) {  
 int h = 1;  
 while (h < array.length / 3) {  
 h = 3 \* h + 1;  
 }  
 while (h > 0) {  
 for (int i = 0; i < h; i++) {  
 for (int j = i + h; j < array.length; j += h) {  
 if (array[j] < array[j - h]) {  
 for (int k = j; k >= h; k -= h) {  
 if (array[k] > array[k - h]) break;  
 int t = array[k];  
 array[k] = array[k - h];  
 array[k - h] = t;  
 }  
 }  
  
 }  
 }  
 h /= 3;  
 }  
 }  
  
 @Override  
 public String toString() {  
 return "ShellSort";  
 }  
}  
  
class Quick implements Sortable {  
 @Override  
 public void sort(int[] array) {  
 quickSort(array, 0, array.length - 1);  
 }  
  
 private void swap(int[] array, int i, int j) {  
 int tmp = array[i];  
 array[i] = array[j];  
 array[j] = tmp;  
 }  
  
 public void quickSort(int[] array, int left, int right) {  
 int pivot = array[left + (right - left) / 2];  
 int i = left;  
 int j = right;  
 while (i <= j) {  
 while (array[i] < pivot) {  
 i++;  
 }  
 while (array[j] > pivot) {  
 j--;  
 }  
 if (i <= j) {  
 swap(array, i, j);  
 i++;  
 j--;  
 }  
 }  
 if (left < j)  
 quickSort(array, left, j);  
 if (i < right)  
 quickSort(array, i, right);  
 }  
  
  
 @Override  
 public String toString() {  
 return "QuickSort";  
 }  
}  
  
class Incorrect implements Sortable {  
  
 @Override  
 public void sort(int[] array) {  
  
 }  
  
 @Override  
 public String toString() {  
 return "Incorrect";  
 }