Национальный технический университет Украины

«Киевский политехнический институт имени Игоря Сикорского»

Факультет информатики и вычислительной техники

Кафедра автоматизации и управления в технических система

**Лабораторная работа №**

с дисциплины «СТП-1»

|  |  |
| --- | --- |
| Выполнил:  студент группыИТ-51  Кульчицкий Алексей Андреевич  Дата сдачи\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Защищенобаллом \_\_\_\_\_\_\_\_\_\_\_ | Проверил:  ac. Федорчук В. В |

Киев 2017

Завдання: реалізувати алгоритми сортування bubble sort, selection sort, quick sort, odd-even sort, merge sort;

package com.alex;

import java.util.ArrayList;

import java.util.Arrays;

import java.util.Collections;

import java.util.Comparator;

public class SortLibrary<T extends Comparable<T>> {

public T[] BubbleSort(T[] arr){

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

for(int j = 0;j<arr.length-1-i;j++){

if(arr[j].compareTo(arr[j+1])>0){

swap(arr,j,j+1);

}

}

}

return arr;

}

public T[] SelectionSort(T[] arr){

for(int i=0;i<arr.length-1;i++){

int min=i;

for(int j=i+1;j<arr.length;j++){

if(arr[j].compareTo(arr[min])<0){

min=j;

}

}

swap(arr,i,min);

}

return arr;

}

public T[] QuickSort(T[] arr){

directQuickSort(arr,0,arr.length-1);

return arr;

}

private void directQuickSort(T[] arr,int low,int high){

if(low>=high){

return;

}

int separator=partition(arr,low,high);

directQuickSort(arr,low,separator);

directQuickSort(arr,separator+1,high);

}

private int partition(T[] arr,int low,int high){

int i=low;

int j=high;

int separator=(high+low)/2;

while(i<j){

while(i<separator && (arr[i].compareTo(arr[separator])<0 || arr[i].equals(arr[separator]))){

i++;

}

while (j>separator && (arr[j].compareTo(arr[separator])>0 || arr[j].equals(arr[separator]))){

j--;

}

if(i<j) {

swap(arr, i, j);

if (i == separator)

separator = j;

else if (j == separator)

separator = i;

}

}

return separator;

}

public T[] OddEvenSort(T[] arr){

boolean sorted = false;

while (!sorted){

sorted=true;

for(int i=1;i<=arr.length-2;i+=2){

if(arr[i].compareTo(arr[i+1])>0){

swap(arr,i,i+1);

sorted=false;

}

}

for(int i=0;i<=arr.length-2;i+=2){

if(arr[i].compareTo(arr[i+1])>0){

swap(arr,i,i+1);

sorted=false;

}

}

}

return arr;

}

public T[] MergeSort(T[] arr){

doMergeSort(arr,0,arr.length);

return arr;

}

private void merge(T[] arr,int left,int mid,int right){

int iter1=0,iter2=0;

ArrayList<T> result = new ArrayList<>();

while(left+iter1<mid && mid+iter2<right){

if(arr[left+iter1].compareTo(arr[mid+iter2])<0){

result.add(arr[left+iter1]);

iter1++;

}else{

result.add(arr[mid+iter2]);

iter2++;

}

}

while(left+iter1<mid){

result.add(arr[left+iter1]);

iter1++;

}

while (mid+iter2<right){

result.add(arr[mid+iter2]);

iter2++;

}

for(int i=0;i<iter1+iter2;i++){

arr[left+i]=result.get(i);

}

}

private void doMergeSort(T[] arr,int left,int right){

if(left+1>=right)

return;

int mid = (right+left)/2;

doMergeSort(arr,left,mid);

doMergeSort(arr,mid,right);

merge(arr,left,mid,right);

}

public T[] DefaultSort(T[] arr){

Arrays.sort(arr);

return arr;

}

public T[] ShellSort(T[]arr){//FROM INTERNET

int j;

for( int gap = arr.length / 2; gap > 0; gap /= 2 ){

for(int i=gap;i<arr.length;i++){

T temp = arr[i];

for (j = i; j >= gap && arr[j - gap].compareTo(temp)>0; j -= gap)

{

arr[j] = arr[j - gap];

}

arr[j] = temp;

}

}

return arr;

}

private void swap(T[] arr,int i,int j){

T tmp = arr[j];

arr[j]=arr[i];

arr[i]=tmp;

}

}

package com.alex;

import org.omg.PortableInterceptor.SYSTEM\_EXCEPTION;

import java.util.Random;

import java.lang.Integer;

public class Main {

private final static int size=1500;

private static Integer[] generateArray(int n){

Integer[] array = new Integer[n];

Random rand = new Random();

for(int i =0;i<n;i++)

array[i]=rand.nextInt(50000)-25000;

return array;

}

private static void showArray(Integer[] arr){

for(Integer var:arr)

System.out.print(var+" ");

System.out.println();

}

// private static void showMemory(){

// Runtime rt = Runtime.getRuntime();

// double usedMB = (rt.totalMemory() - rt.freeMemory());

// System.out.println(usedMB);

// System.gc();

// }

public static void main(String[] args) {

SortLibrary<Integer> sorter = new SortLibrary<Integer>();

Integer[] arr = generateArray(size);

long startClock=0;

long endClock=0;

showArray(arr);

startClock=System.nanoTime();

sorter.OddEvenSort(arr.clone());

endClock=System.nanoTime();

System.out.println("Odd Even Sort :" + String.format("%,12d",(endClock-startClock)/1000000) + " ms");

startClock=System.nanoTime();

sorter.QuickSort(arr.clone());

endClock=System.nanoTime();

System.out.println("Quick Sort :" + String.format("%,12d",(endClock-startClock)/1000000) + " ms");

startClock=System.nanoTime();

sorter.DefaultSort(arr.clone());

endClock=System.nanoTime();

System.out.println("Default JAVA(timsort) Sort :" + String.format("%,12d",(endClock-startClock)/1000000) + " ms");

startClock=System.nanoTime();

sorter.BubbleSort(arr.clone());

endClock=System.nanoTime();

System.out.println("Bubble Sort :" + String.format("%,12d",(endClock-startClock)/1000000) + " ms");

startClock=System.nanoTime();

sorter.SelectionSort(arr.clone());

endClock=System.nanoTime();

System.out.println("Selection Sort :" + String.format("%,12d",(endClock-startClock)/1000000) + " ms");

startClock=System.nanoTime();

sorter.MergeSort(arr.clone());

endClock=System.nanoTime();

System.out.println("Merge Sort :" + String.format("%,12d",(endClock-startClock)/1000000) + " ms");

startClock=System.nanoTime();

sorter.ShellSort(arr.clone());

endClock=System.nanoTime();

System.out.println("Shell Sort :" + String.format("%,12d",(endClock-startClock)/1000000) + " ms");

}

}

Висновок: виконуючи цю лабораторну роботу я закріпив знання алгоритмів сортування.