Министерство образования и науки Украины

Национальный аэрокосмический университет

им. Н.Е. Жуковского

Кафедра компьютерных систем и сетей

Лабораторная работа № 1

«**Исследование простых алгоритмов сортировки**»

по курсу «Модели и структуры данных»

ХАІ.503.525Б.19O.050102, 1805059 ПЗ

|  |  |  |  |
| --- | --- | --- | --- |
|  | Выполнил: | студент гр. 525Б | Озеров А.И.. |
|  |  | подпись | дата |
|  | Проверил: | к.т.н., доцент | Шостак А. В. |
|  |  | подпись | дата |

Харьков 2019

**Математическая постановка задачи**: дана последовательность из n элементов

a1,a2, … an, принадлежащих множеству A, на котором задано отношение всеобщего порядка «≤». Найти перестановку этих элементов в таком порядке: ak1, ak2,… akn, при котором будет справедливо отношение ak1 ≤ ak2 … ≤ akn.

**Вариант 13.**

Диапазон [0, 600]

Алгоритмы: сортировка вставками, сортировка пузырьком.

**Текст программы на языке JavaScript:**

// Реализация сортировки пузырькем (возвращает кол-во сравнений и обменов)

function BubbleSort(ARR) {

let arr = ARR.slice(0);

let n = arr.length;

let sravn = 0;

let obmen = 0;

for (let i = 0; i < n - 1; i++) {

for (let j = 0; j < n - 1 - i; j++) {

if (sravn++ && (arr[j + 1] < arr[j])) {

obmen++;

let t = arr[j + 1];

arr[j + 1] = arr[j];

arr[j] = t;

}

}

}

return {

sravn: sravn,

obmen: obmen

};

}

// Реализация сортировки вставками (возвращает кол-во сравнений и обменов)

function InsertionSort(ARR) {

let arr = ARR.slice(0);

let length = arr.length;

let sravn = 0;

let obmen = 0;

for (let i = 1; i < length; i++) {

let key = arr[i];

let j = i - 1;

while (sravn++ && j >= 0 && arr[j] > key) {

obmen++;

arr[j + 1] = arr[j];

j--;

}

arr[j + 1] = key;

}

return {

sravn: sravn,

obmen: obmen

};

}

// Получение случайного числа от 0, по {max}

function getRandomInt(max) {

return Math.floor(Math.random() \* Math.floor(max));

}

//Генерация случаного массива [0..600]

function generage(n) { // random

var arr = [];

for (let i = 0; i < n; i++) {

arr[i] = getRandomInt(600);

}

return arr;

}

// Кол-во повторений для каждого из кол-ва элементов массива

const REPEATINGS = 5000;

// Массив с результатами, для каждого прогона

var answers = [];

for (let i = 0; i < REPEATINGS; i++) {

answers[i] = [];

for(let j = 0; j < 11; j++){

answers[i][j] = [];

}

}

// Общий цикл повторений

for (let repeating = 0; repeating < REPEATINGS; repeating++) {

let ARRAY = [];

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

ARRAY[i] = [];

}

// Цикл генерации массивов заданных размеров ( 5..50 с шагом 5)

for (let i = 1; i < 11; i++) {

let tmparr = generage(i\*5);

ARRAY[0] = tmparr.slice(0);

ARRAY[1] = tmparr.sort((a, b) => a - b).slice(0);

ARRAY[2] = tmparr.sort((a, b) => b - a).slice(0);

let sort\_insert = [

InsertionSort(ARRAY[0]),

InsertionSort(ARRAY[1]),

InsertionSort(ARRAY[2])

];

let sort\_bubble = [

BubbleSort(ARRAY[0]),

BubbleSort(ARRAY[1]),

BubbleSort(ARRAY[2])

];

answers[repeating][i][0] = [

[

sort\_insert[0].sravn, // sravn

sort\_insert[0].obmen, // obmen

],

[

sort\_insert[1].sravn,

sort\_insert[1].obmen,

],

[

sort\_insert[2].sravn,

sort\_insert[2].obmen,

],

];

answers[repeating][i][1] = [

[

sort\_bubble[0].sravn,

sort\_bubble[0].obmen,

],

[

sort\_bubble[1].sravn,

sort\_bubble[1].obmen,

],

[

sort\_bubble[2].sravn,

sort\_bubble[2].obmen,

]

];

}

}

// Нормализация результатов, методом подсчета среднего арифметического

var summ = []

for(let i = 0; i < 11; i++){

summ[i] = {

INSsravn: [0, 0, 0],

INSobmen: [0, 0, 0],

INSsumm: [0, 0, 0],

BUBsravn: [0, 0, 0],

BUBobmen: [0, 0, 0],

BUBsumm: [0, 0, 0],

};

}

for(let i = 1; i < 11; i++){

for(let j = 0; j < REPEATINGS; j++){

for(let k = 0; k < 3; k++){

summ[i].INSsravn[k] += answers[j][i][0][k][0] // sravn // random -> sorted -> sorted & reversed

summ[i].INSobmen[k] += answers[j][i][0][k][1] // obmen

summ[i].BUBsravn[k] += answers[j][i][1][k][0] // sravn // random -> sorted -> sorted & reversed

summ[i].BUBobmen[k] += answers[j][i][1][k][1] // obmen

}

}

}

for(let i = 1; i < 11; i++){

for(let k = 0; k < 3; k++){

summ[i].INSsravn[k] /= REPEATINGS;

summ[i].INSobmen[k] /= REPEATINGS;

summ[i].INSsumm[k] = summ[i].INSsravn[k] + summ[i].INSobmen[k]

summ[i].BUBsravn[k] /= REPEATINGS;

summ[i].BUBobmen[k] /= REPEATINGS;

summ[i].BUBsumm[k] = summ[i].BUBsravn[k] + summ[i].BUBobmen[k]

}

// Вывод результатов

console.log(`${i\*5} elements: ${JSON.stringify(summ[i])}`);

}

**Примеры пошаговой работы сортировок:**

Сортировка вставкой:

step |array |N sr |N obm |Summ|

0 |(3)(4 5 6 10 2 3 |- |- |-

1 |(3 4)(5 6 10 2 3 |1 |0 |1

2 |(3 4 5)(6 10 2 3 |1 |0 |1

3 |(3 4 5 6)(10 2 3) |1 |0 |1

4 |(3 4 5 6 10)(2 3) |1 |0 |1

5 |(2 3 4 5 6 10)(3) |5 |5 |10

6 |(2 3 3 4 5 6 10) |5 |5 |10

--/14 /10 /24

Сортировка пузырьком:

step |array |Nsr |Nobm |Summ

0 |3 4 5 6 (10 2) 3 |- |- |-

1 |3 4 5 6 2 (10 3) |5 |1 |6

2 |3 4 5 (6 2) 3 10 |1 |1 |2

3 |3 4 5 2 (6 3) 10 |4 |1 |5

4 |3 4 (5 2) 3 6 10 |1 |1 |2

5 |3 4 2 (5 3) 6 10 |3 |1 |4

6 |3 (4 2) 3 5 6 10 |1 |1 |2

7 |3 2 (4 3) 5 6 10 |2 |1 |3

8 |(3 2) 3 4 5 6 10 |1 |1 |2

9 |(2 3 3 4 5 6 10)

--/18 /8 /26

**Таблицы со статистическими данными:**

Таблица 1 Сортировка вставкой, массив упорядочен по возрастанию

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| N | 5 | 10 | 15 | 20 | 25 | 30 | 35 | 40 | 45 | 50 |
| Sr | 4 | 9 | 14 | 19 | 24 | 29 | 34 | 39 | 44 | 49 |
| Obm | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Summ | 4 | 9 | 14 | 19 | 24 | 29 | 34 | 39 | 44 | 49 |

Таблица 2 Сортировка пузырьком, массив упорядочен по возрастанию

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| N | 5 | 10 | 15 | 20 | 25 | 30 | 35 | 40 | 45 | 50 |
| Sr | 10 | 45 | 105 | 190 | 300 | 435 | 595 | 780 | 990 | 1225 |
| Obm | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Summ | 10 | 45 | 105 | 190 | 300 | 435 | 595 | 780 | 990 | 1225 |

Таблица 3 Сортировка вставкой, массив упорядочен по убыванию

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| N | 5 | 10 | 15 | 20 | 25 | 30 | 35 | 40 | 45 | 50 |
| Sr | 12.98 | 52.92 | 117.83 | 207.68 | 322.52 | 462.2 | 627 | 816.7 | 1031.4 | 1270.9 |
| Obm | 8.99 | 43.92 | 103.8 | 188.78 | 298.52 | 433.2 | 593 | 777.7 | 987.4 | 1222 |
| Summ | 21.97 | 96.83 | 221.66 | 396.36 | 621.04 | 895.5 | 1220 | 1594 | 2018 | 2492 |

Таблица 4 Сортировка пузырьком, массив упорядочен по убыванию

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| N | 5 | 10 | 15 | 20 | 25 | 30 | 35 | 40 | 45 | 50 |
| Sr | 10 | 45 | 105 | 190 | 300 | 435 | 595 | 780 | 990 | 1225 |
| Obm | 8.99 | 43.9 | 103.8 | 188.7 | 298.5 | 433.3 | 593 | 777.8 | 987.4 | 1222 |
| Summ | 19 | 88.9 | 208.8 | 378.7 | 598.5 | 868.3 | 1188 | 1667 | 1977 | 2447 |

Таблица 5 Сортировка вставкой, массив не упорядочен

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| N | 5 | 10 | 15 | 20 | 25 | 30 | 35 | 40 | 45 | 50 |
| Sr | 7.9 | 29.5 | 63.8 | 110.4 | 170.2 | 240.9 | 324.8 | 422.1 | 531 | 651 |
| Obm | 4 | 20.5 | 49.9 | 91.4 | 146.2 | 211.9 | 290.8 | 383.1 | 487 | 602 |
| Summ | 11.9 | 50 | 113.7 | 201.8 | 316.4 | 452.9 | 615.9 | 805.1 | 1018 | 1253 |

Таблица 6 Сортировка пузырьком, массив не упорядочен

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| N | 5 | 10 | 15 | 20 | 25 | 30 | 35 | 40 | 45 | 50 |
| Sr | 10 | 45 | 105 | 190 | 300 | 435 | 595 | 780 | 990 | 1225 |
| Obm | 4.79 | 22.3 | 52.5 | 95 | 150.4 | 217 | 296.6 | 390.1 | 494.6 | 610.4 |
| Summ | 14.8 | 67.3 | 157.5 | 285 | 350.4 | 652 | 891.6 | 1070 | 1484 | 1835 |

**Графики со сравнением сортировок и их качеств, в зависимости от качества исходных данных.**

График 1. Зависимость сортировки вставками от

качества исходных данных

График 2. Зависимость сортировки пузырьком от

качества исходных данных

График 3. Cравнение сортировок при отсортированном массиве

График 4. Cравнение сортировок при обратно-отсортированном массиве

График 5. Cравнение сортировок при неотсортированном массиве

**Выводы**: при сравнении сортировки вставками стало понятно, что при любом качестве массива, кроме обратно-отсортированного, сортировка простыми вставками менее ресурсозатратна, так как требует меньшее количество операций и, соответственно, времени, а в случае с обратно-отсортированным массивом эти две сортировки показывают сравнимые результаты, что следует из самих алгоритмов, а именно сортировка вставками, в данном случае, проводит те же операции в том-же количестве, что и алгоритм пузырьковой сортировки. Из этого следует, что во всех случаях предпочтительнее использовать алгоритм сортировки вставками, так как статистически затраченное время будет ниже.