Лабораторная работа. Алгоритмы сортировки.
Часть 1. Сортировка выбором, пузырьком, вставками, Шелла, расческой и другие за O(n^2)

Замечания

1) *Базовые реализации алгоритмов сортировки выбором, пузырьком, вставками приведены в конце этого текста, а также даны в файле BaseSorting.java*
2) *Все собственные решения **отдельных задач в этой работе***
   *удобнее собирать как отдельные статические методы **в рамках одного класса***
   Самым ленивым можно взять содержимое BaseSorting.java и творить дальше в нем

3) *В файле **large.txt** дан набор из 1000000 случайных целых из диапазона [0, 999999]*
4) *Для определения времени работы фрагмента программы, без использования специальных профайлеров можно использовать следующую схему*

```
// есть метод, длительность работы которого надо посмотреть
public static void myProc() {
    // .... какие-то полезные действия, например, сортировка
}

// метод, возвращающий продолжительность работы для method
// при необходимости функциональный интерфейс Runnable
// меняем на подходящий к задаче (по сигнатуре myProc)
public static long timeLine(Runnable method) {
    long startTime = System.nanoTime();
    method.run();
    long duration = System.nanoTime() - startTime;
    return duration;
}

    // ... используем по мере необходимости
    //  вывод длительности работы метода myProc() в наносекундах
   System.out.println( timeLine( MyClass::myProc)); //ссылкой на него
   System.out.println( timeLine( ()->myProc()));   // или лямбдой
```

Задачи

1. Говорят, что массив отсортирован по max-min значениям, если первый элемент массива является максимальным, второй - минимальным, третий - вторым максимумом и так далее.
Используя принцип сортировки выбором реализуйте сортировку max-min.

Пример:
Входные данные: 1 2 3 4 5
Выходные: 5 1 4 2 3

2. Реализуйте двунаправленный вариант сортировки выбором, который находит минимальное и максимальное значения в массиве за каждый проход и устанавливает их на свои места.
Алгоритм делит массив на три подмассива:
1) отсортированные минимумы;
2) несортированные;
3) отсортированные максимумы.

Определите минимальное количество итераций этого алгоритма для сортировки массива { 5, 4, 3, 2, 1 } в порядке возрастания

3. Массив целых чисел сортируется вставками в порядке убывания. Выведите количество элементов, сдвинутых со своего изначального места в ходе этой сортировки.

Пояснение (см рис.->):
а) число 21 не сдвигается
б) для числа 24 делается перемещение,
значит +1 к счетчику сдвинутых чисел
(задача из https://hyperskill.org/learn/step/3173).

Пример 1:
ввод: 50 40 30 10 20
вывод: 1
Пример 2:
ввод: 30 40 20 5 10
вывод: 2
Пример 3
ввод: 5 2 9 1 2 4 9 5
вывод: 5

4. Числовой массив сортируется пузырьком в порядке убывания. Выведите количество перестановок (обмен двух чисел, swap), которое надо выполнить, чтобы полностью его отсортировать.
(задача из https://hyperskill.org/learn/step/3826)
Пример 1:
ввод: 5 4 3 1 2
вывод: 1
Пример 2:
ввод: 8 3 4 6 5 2 1
вывод: 5

5. Проведите сравнение производительности алгоритмов сортировки
- выбором, двунаправленным выбором
- пузырьком
- вставками

Тестирование провести
a) на пустом массиве
b) на массиве из одного элемента
c) на массиве из 100 элементов
d) на массиве из 1000000 элементов
В двух последних случаях (c-d) массив должен быть заполнен
1) одинаковыми числами
2) случайными числами
3) натуральными в порядке возрастания { 1, 2,…,n}
4) натуральными в порядке убывания { n, n-1,…,1}
5) *предложите свои тестовые примеры

Каждый тест для каждого метода имеет смысл провести несколько раз, получив некоторые средние показатели

6*. Реализуйте алгоритмы сортировки
- Шелла (Shell sort) на основе вставок. Вывести на экран все перемещаемые со своей позиции элементы и на какое расстояние они сдвигаются (количество индексов влево).
- расческой (comb sort), на основе пузырьковой. Вывести на экран все перемещаемые пары элементов (для каждого swap те элементы, которые перемещаем).
- шейкерную (cocktail sort), на основе пузырьковой

Включите их в общее тестирование и анализ алгоритмов в задаче 5.

Дополнительные материалы

Базовые реализации алгоритмов

```
public static int[] bubbleSort1(int[] array) {
    for (int i = 0; i < array.length - 1; i++) {
        for (int j = 0; j < array.length - i - 1; j++) {
            if (array[j] > array[j + 1]) {
                int t = array[j];
                array[j] = array[j + 1];
                array[j + 1] = t;
            }
        }
    }
    return array;
}

public static int[] bubbleSort2(int[] array) {
    boolean f = false;
    int i = 0;
    while ( !f ) {
        f = true;
        for (int j = 0; j < array.length - i - 1; j++) {
            if (array[j] > array[j + 1]) {
                int t = array[j];
                array[j] = array[j + 1];
                array[j + 1] = t;
                f = false;
            }
        }
        i++;
    }
    return array;
}

public static int[] selectionSort(int[] array) {
    for (int i = 0; i < array.length - 1; i++) {
        int index = i;
        for (int j = i + 1; j < array.length; j++) {
            if (array[j] < array[index]) {
                index = j;
            }
        }
        int min = array[index];
        array[index] = array[i];
        array[i] = min;
    }
    return array;
}

public static int[] insertSort(int[] array) {
    for (int i = 1; i < array.length; i++) {
        int elem = array[i];
        int j = i - 1;
        while (j >= 0 && array[j] > elem) {
            array[j + 1] = array[j];
            j--;
        }
        array[j + 1] = elem;
    }
    return array;
}
```

Сортировка Шелла

Общий принцип – как у сортировки вставками, но элементы перемещаются влево не на 1 шаг, а «большими прыжками».

Каждый проход в алгоритме характеризуется смещением h_i, таким, что сортируются элементы отстающие друг от друга на h_i позиций. Шелл предлагал использовать $h_t=N/2$, $h_{t-1}=h_t/2$, … , $h_0=1$. Возможны и другие смещения, но $h_0=1$ (на последнем этапе получается сортировка вставками)

Псевдокод:

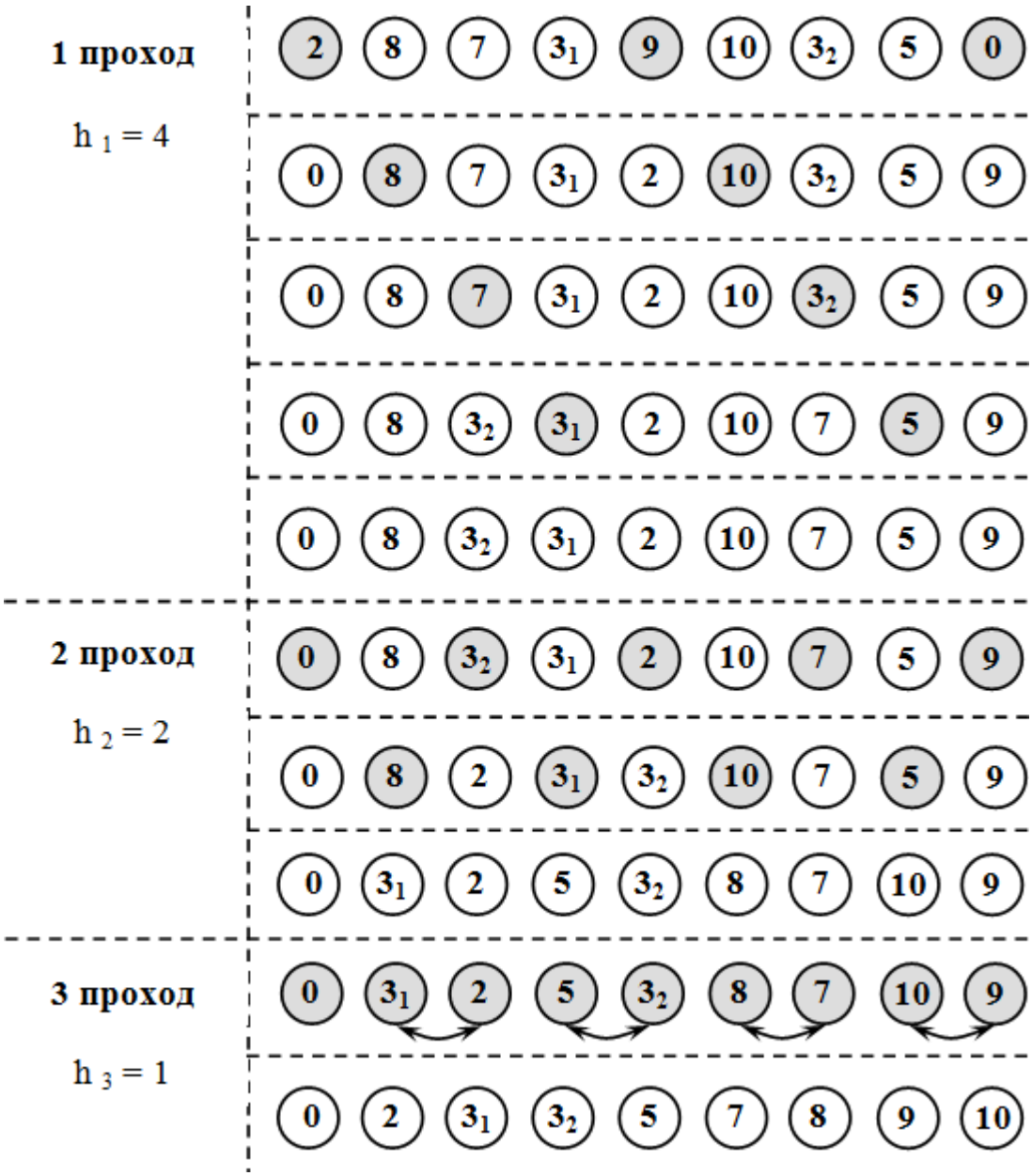
Шаг 0. $i=t$

Шаг 1. Разобьем массив на наборы элементов, отстающих друг от друга на h_i.

Шаг 2. Отсортируем элементы каждого набора сортировкой вставками.

Шаг 3. Объединим наборы обратно в массив. Уменьшим i. Если i неотрицательно — вернемся к шагу 1

Пример работы алгоритма Шелла



Пример работы сортировки Шелла с альтернативным вариантом выбора h (https://www.math.spbu.ru/user/jvr/DA_html/_lec_1_14.html)

Будем выбирать для размера шага числа вида h= $2^r + 1$. Так как n = 14 , самое большое из таких значений — это 9. Для каждого шага сделаем отдельную таблицу.

Шаг h=9:

it	1	2	3	4	5	6	7	8	9	10	11	12	13	14	comm
1	18	42	51	18	16	18	44	22	31	11	63	23	9	14	переставим
2	11	42	51	18	16	18	44	22	31	18	63	23	9	14	оставим
3	11	42	51	18	16	18	44	22	31	18	63	23	9	14	переставим
4	11	42	23	18	16	18	44	22	31	18	63	51	9	14	переставим
5	11	42	23	9	16	18	44	22	31	18	63	51	18	14	переставим
	11	42	23	9	14	18	44	22	31	18	63	51	18	16	пора менять шаг

Теперь, уменьшаем шаг вдвое. Шаг h=5:

it	1	2	3	4	5	6	7	8	9	10	11	12	13	14	comm
1	11	42	23	9	14	18	44	22	31	18	63	51	18	16	оставим
2	11	42	23	9	14	18	44	22	31	18	63	51	18	16	оставим
3	11	42	23	9	14	18	44	22	31	18	63	51	18	16	переставим
4	11	42	22	9	14	18	44	23	31	18	63	51	18	16	оставим
5	11	42	22	9	14	18	44	23	31	18	63	51	18	16	оставим
6	11	42	22	9	14	18	44	23	31	18	63	51	18	16	оставим
7	11	42	22	9	14	18	44	23	31	18	63	51	18	16	оставим
8	11	42	22	9	14	18	44	23	31	18	63	51	18	16	переставим, продолжим
9	11	42	22	9	14	18	44	18	31	18	63	51	23	16	переставим
10	11	42	18	9	14	18	44	22	31	18	63	51	23	16	переставим, продолжим
11	11	42	18	9	14	18	44	22	16	18	63	51	23	31	оставим
	11	42	18	9	14	18	44	22	16	18	63	51	23	31	пора менять шаг

Обратите внимание на четыре последних итерации. Здесь видно, что значение продвигается вперед не один раз, а столько, сколько понадобится.

Снова уменьшаем шаг вдвое. Шаг h=3:

it	1	2	3	4	5	6	7	8	9	10	11	12	13	14	comm
1	11	42	18	9	14	18	44	22	16	18	63	51	23	31	переставим
2	9	42	18	11	14	18	44	22	16	18	63	51	23	31	переставим
3	9	14	18	11	42	18	44	22	16	18	63	51	23	31	оставим
4	9	14	18	11	42	18	44	22	16	18	63	51	23	31	оставим
5	9	14	18	11	42	18	44	22	16	18	63	51	23	31	переставим, продолжим
6	9	14	18	11	22	18	44	42	16	18	63	51	23	31	оставим
7	9	14	18	11	22	18	44	42	16	18	63	51	23	31	переставим, продолжим
8	9	14	18	11	22	16	44	42	18	18	63	51	23	31	переставим
9	9	14	16	11	22	18	44	42	18	18	63	51	23	31	переставим, продолжим
10	9	14	16	11	22	18	18	42	18	44	63	51	23	31	оставим

11	9	14	16	11	22	18	18	42	18	44	63	51	23	31	оставим
12	9	14	16	11	22	18	18	42	18	44	63	51	23	31	оставим
13	9	14	16	11	22	18	18	42	18	44	63	51	23	31	переставим, продолжим
14	9	14	16	11	22	18	18	42	18	23	63	51	44	31	оставим
15	9	14	16	11	22	18	18	42	18	23	63	51	44	31	переставим, продолжим
16	9	14	16	11	22	18	18	42	18	23	31	51	44	63	переставим, продолжим
17	9	14	16	11	22	18	18	31	18	23	42	51	44	63	оставим
	9	14	16	11	22	18	18	31	18	23	42	51	44	63	пора менять шаг

Дальше нужно тоже самое выполнить с шагом h=2 и с шагом h=1.