Министерство Образования, Культуры и Исследований  
Молдавский Государственный Университет  
Факультет Математики и Информатики  
Департамент Информатики

Лабораторная работа №2

По предмету: «Алгоритмы, структуры данных и сложность».  
ТЕМА: Методы сортировки. Оценка сложности.

Преподаватель: Кроитор М.  
Студент: Журба Дарья.  
Группа: IA2102.

Кишинёв 2023

Цель работы:

Требуется реализовать программу на любом удобном вам языке программирования, в которой реализовано **3 способа сортировки** массивов данных из созданного файла по ключу. Для каждого метода сортировки проанализировать теоретическую и практическую сложности.

Ход работы:

Сложность:

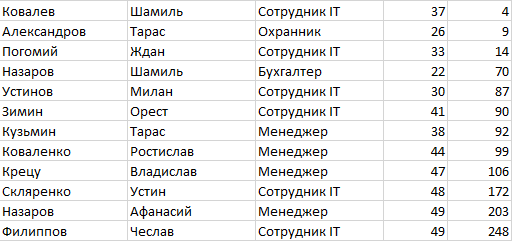
|  |  |  |
| --- | --- | --- |
| Лучший время | Среднее время | Худшее время |
| O(n) | O(n2) | O(n2) |

Память: O(1)  
Проходы повторяются n-1 раз или до тех пор,пока на проходе не будет получен  
отсортированный массив.  
Реализация:public static void bubbleS(ArrayList<String> list)  
{  
System.out.println("Bubble Sort");  
int numIterations=0;  
int numSwap = 0;  
boolean isSorted = false;  
int temp ;  
while(!isSorted)  
{  
isSorted=true;  
for(int j = 0;j<list.size()-1;j++)  
{  
if(Integer.parseInt(list.get(j).substring(0,list.get(j).indexOf("\t")))>  
Integer.parseInt(list.get(j+1).substring(0,list.get(j+1).indexOf("\t"))))  
{  
isSorted = false;  
toSwap(list, j, j+1);  
numSwap++;  
}  
numIterations++;  
}  
}  
System.out.println("The number of iterations : "+ numIterations);  
System.out.println("The number of swaps : "+ numSwap);  
}  
//Метод toSwapstatic void toSwap(ArrayList<String> list, int i, int j)  
{  
String temp = list.get(i);  
list.set(i,list.get(j));  
list.set(j,temp);  
}  
Результат работы:

Bubble Sort

The number of iterations: 1666

The number of swaps: 626



Реализация N2:

Замечание: После каждой итерации наибольший элемент из неотсортированных становится на

свое место

**public static void** bubbleSort(ArrayList<String> list)  
{  
System.***out***.println(**"Bubble Sort"**);  
**int** numIterations=0;  
**int** n = list.size();

**for** (**int** i = 0; i <n-1; i++)

**for** (**int** j = 0; j <n-1-i; j++)

{

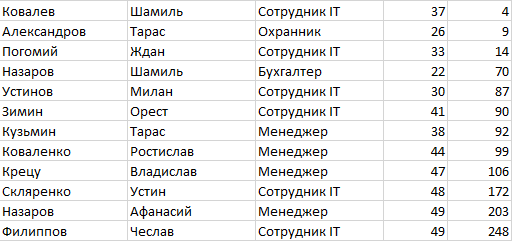
**if**(Integer.*parseInt*(list.get(j).substring(0,list.get(j).indexOf(**"\t"**)))>  
Integer.*parseInt*(list.get(j+1).substring(0,list.get(j+1).indexOf(**"\t"**))))  
{  
*toSwap*(list, j, j+1);  
}  
numIterations++;  
}  
System.***out***.println(**"The number of iterations : "**+ numIterations);  
} }

Результат работы:

Bubble Sort

The number of iterations: 1225

The number of swaps: 626



**Сортировка выбором (Selection Sort)**

Алгоритм:  
Исходный массив длиной N разбивается на две части: итог и остаток. Итог – упорядоченный участок массива, располагающийся в начале массива. Остаток - участок массива, располагающийся вплотную за итогом и содержащий исходные числа не отсортированной части исходного массива.

Алгоритм:

Шаг 1. Полагается i=0, т.е. считается, что итоговый участок - пуст.  
Шаг 2. В остатке массива ищется минимальный элемент и он меняется местом с первым элементом остатка (i-ым элементом массива). После чего значение i увеличивается на единицу, тем самым расширяя итоговый участок массива ( отсортированную часть исходного массива).  
Шаг 3. Если i < N, то повторяется Шаг 2. В противном случае - конец алгоритма, т.к. итог становится равным всему массиву.

Конец алгоритма.

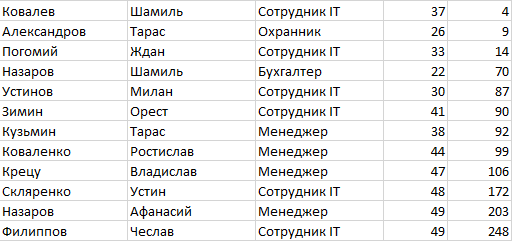
|  |  |  |
| --- | --- | --- |
| Лучшее время | Среднее время | Худшее время |
| O(n2) | O(n2) | O(n2) |

Реализация:public class SelectionSort {  
static void toSwap(ArrayList<String> list, int i, int j)  
{  
String temp = list.get(i);  
list.set(i,list.get(j));  
list.set(j,temp);  
}  
public static void selectionSort(ArrayList <String> list)  
{  
System.out.println("Selection Sort");  
int num =0;  
int numSwap =0;  
for(int i=0;i<list.size()-1;i++)  
{  
int temp\_position = i;  
for(int j=i+1;j<list.size();j++)  
{  
num++;  
if(Integer.parseInt(list.get(j).substring(0,list.get(j).indexOf("\t")))<  
Integer.parseInt(list.get(temp\_position).substring(0,list.get(temp\_position).  
indexOf("\t"))))  
{  
temp\_position=j;  
}  
}  
toSwap(list,i,temp\_position);  
numSwap++;  
}  
System.out.println("The number of iterations : "+ num);  
System.out.println("The number of swaps : "+ numSwap);  
}  
}  
Результат работы:

Selection Sort

The number of iterations: 1225

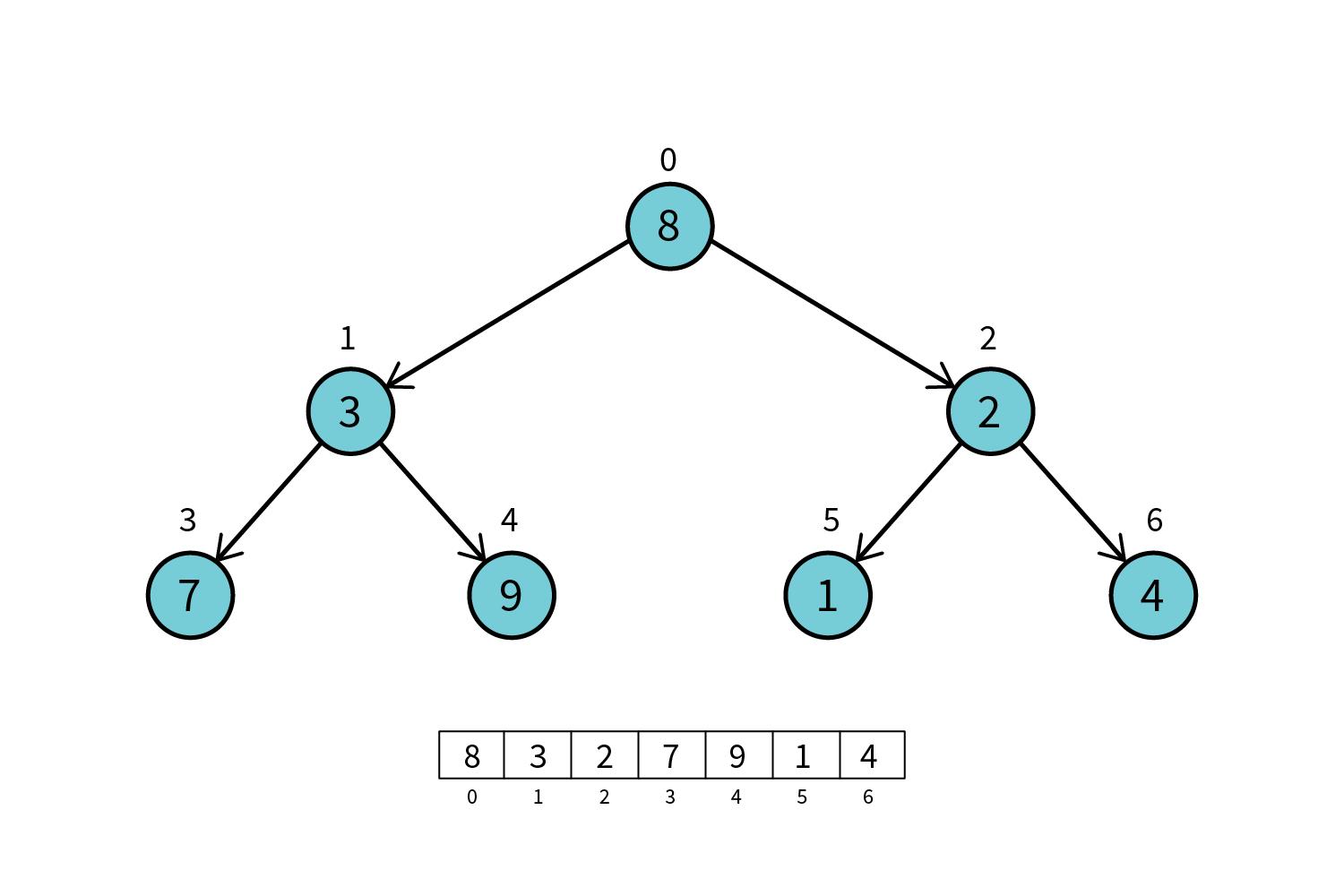
The number of swaps: 49

****

**Пирамидальная сортировка (HeapSort)**

Метод пирамидальной сортировки, изобретенный Д. Уилльямсом, является улучшением традиционных сортировок с помощью дерева.

Пирамидой (кучей) называется двоичное дерево такое, что:



Общая идея пирамидальной сортировки заключается в том, что сначала строится пирамида из элементов исходного массива, а затем осуществляется сортировка элементов.

Выполнение алгоритма разбивается на два этапа:

1. Построение пирамиды. Определяем правую часть дерева, начиная с n/2-1 (нижний уровень дерева). Берем элемент левее этой части массива и просеиваем его сквозь пирамиду по пути, где находятся меньшие его элементы, которые одновременно поднимаются вверх; из двух возможных путей выбираете путь через меньший элемент.

2. Сортировка на построенной пирамиде. Берем последний элемент массива в качестве текущего. Меняем верхний (наименьший) элемент массива и текущий местами. Текущий элемент (он теперь верхний) просеиваем сквозь n-1 элементную пирамиду. Затем берем предпоследний элемент и т.д.

|  |  |  |
| --- | --- | --- |
| Лучшее время | Среднее время | Худшее время |
| O(n log(n)) | O(n log(n)) | O(n log(n)) |

Реализация:public class HeapSort {  
static int it =0;  
static int s=0;  
public static void sort(ArrayList<String> list)  
{  
System.out.println("Heap Sort");  
int n = list.size();

for (int i = n / 2 - 1; i >= 0; i--) heapf(list, n, i);

for (int i = n - 1; i > 0; i--) {it++;

String temp = list.get(0);  
list.set(0,list.get(i));  
list.set(i,temp);  
s++;  
heapf(list, i, 0);  
}  
System.out.println("The number of iterations : "+ it);  
System.out.println("The number of swaps : "+ s);  
}  
static void heapf(ArrayList<String> list, int n, int i)  
{  
int largest = i;  
int l = 2 \* i + 1;  
int r = 2 \* i + 2;  
it++;  
if (l < n &&  
Integer.parseInt(list.get(l).substring(0,list.get(l).indexOf("\t"))) >  
Integer.parseInt(list.get(largest).substring(0,list.get(largest).indexOf("\t"))))  
largest = l;  
if (r < n &&  
Integer.parseInt(list.get(r).substring(0,list.get(r).indexOf("\t"))) >  
Integer.parseInt(list.get(largest).substring(0,list.get(largest).indexOf("\t"))))  
largest = r;  
if (largest != i) {  
String swap = list.get(i);  
list.set(i,list.get(largest));  
list.set(largest,swap);s++;  
heapf(list, n, largest);  
}  
}

Результат работы:

Heap Sort

The number of iterations: 318

The number of swaps: 244

