**Лабораторная работа №4 (Отчет)**

**Марченко Максим 9 группа**

**9 ВАРИАНТ**

Составить две программы, которые реализуют алгоритмы ускоренной сортировки «пузырьком» и выбором. Исходные данные задавать с помощью датчика случайных чисел.

**1.Название и цель работы.**

ИССЛЕДОВАНИЕ И ОЦЕНКА АЛГОРИТМОВ СОРТИРОВКИ

***Цель работы***: Разработка программ, реализующих различные алгоритмы сортировки, и оценка их временной и пространственной сложности.

**2. Словесное описание заданных алгоритма сортировки.**

***Ускоренный алгоритм сортировки пузырьком:***

1. Задать массив из n чисел.

2.1 Номер просмотра (k) = 1.

2.2. Повторять

2.2.1. Перестановка = ложь.

2.2.2. Для i от 1 до n-k выполнить

Если элементы i-тый и (i+1)-вый стоят неправильно, то

a) Поменять местами i-тый и (i+1)-вый элементы;

b) Перестановка = истина.

2.2.3. k = k +1

Пока Перестановка.

3. Вывести отсортированный массив.

4. Закончить.

***Сортировка вставками:***

1. Задать исходный массив А из n элементов с помощью генератора случайных чисел.

2. Для i от 0 до n

2.1. Индекс элемента в конце массива j = i

2.2. Пока (j > 0) И (Аj-1 > Ai)

2.2.1. Сдвиг большего элемента, чтобы было место для вставки Аj = Аj-1

2.2.2. j = j - 1

2.3. Вставка j - го элемента на его место Аj = Ai

3. Вывести массив А.

**3. Текст программ.**

***Ускоренный алгоритм сортировки пузырьком:***

*start = System.nanoTime();  
SpeedSortBubble(numbersOne, 1, true);  
end = System.nanoTime();*

*//Ускоренный алгоритм сортировки пузырьком  
public static int[] SpeedSortBubble(int[] numbers, int k, boolean flag){  
while(flag){  
 flag = false;  
 for (int j = 0; j < numbers.length - k; j++)  
 if(numbers[j] > numbers[j + 1]){  
 numbers[j] = numbers[j] + numbers[j + 1] - (numbers[j + 1] = numbers[j]);  
 flag = true;  
 }  
 k++;  
 }  
 return numbers;  
}*

***Сортировка вставками:***

*start = System.nanoTime();  
SortChoice(numbersTwo);  
end = System.nanoTime();*

*//Сортировка вставками  
public static int[] SortChoice(int[] numbers){  
 for(int i = 1; i < numbers.length; i++)  
 for(int j = i; j > 0 && numbers[j - 1] > numbers[j]; j--){  
 numbers[j] = numbers[j] + numbers[j - 1] - (numbers[j - 1] = numbers[j]);  
 }  
 return numbers;  
}*

**4. Формулы верхней оценки временной и емкостной сложности заданного алгоритма.**

***Ускоренный алгоритм сортировки пузырьком:***

***Верхняя оценка алгоритма:*** В среднем: =.

Т.к. Если массив уже отсортирован, то он проработает со сложностью , если все по убыванию, то .

***Асимптотическая оценка алгоритма для лучшего случая:*** Лучший случай будет тогда, когда исходный массив уже отсортирован, тогда:

Θ(n) = 1(Вызов функции SpeedSortBubble) +3(передача 3 параметров) + 1(проверка условия flag) + 1(присваивание флагу false) + 1(инициализация j) + (n-1)\*(3(присваивание, проверка условия, увеличение j на один) + 1(проверка условия) + (2+2)(две одномерные индексации) + 1(увеличение на 1 в индексации)) + 1(вернуть готовый массив) = 9n - 1

***Асимптотическая оценка алгоритма в худшем случае:*** Худший случай будет тогда, когда исходный массив стоит по убыванию, тогда:

Θ(n) = 1(Вызов функции SpeedSortBubble) + 3(передача 3 параметров) + 1(вернуть отсортированный массив) + (4(проверка условия flag, присваивание флагу false, инициализация j, увеличиваем k на 1) + (1(проверка условия) + (2 + 2)(две индексации одномерного массива) + 1(увеличение на 1 в индексации) + 2(два приравнивания) + 3(три суммы) + 1(разность) + (5\*2)(пять одномерных индексаций) + 1(приравниваем флаг истине)) +( 4 + ) + … + (4 + )) = 5 + 4(n-1) + =

11.5 – 7.5n + 1

***Сортировка вставками:***

***Верхняя оценка алгоритма:*** Аналогично предыдущему алгоритму = =.

***Асимптотическая оценка алгоритма для лучшего случая:***

Θ(n) =

***Асимптотическая оценка алгоритма в худшем случае:***

Θ(n) =

**5.Результаты экспериментальной оценки временной и емкостной сложности заданного алгоритма**.

***A) Для 10000 элементов:***

Уск. алгоритм сортировки пузырьком проработал 0.33 секунд!

Алгоритм сортировки вставками проработал 0.08 секунд!

***Б) Для 100000 элементов:***

Уск. алгоритм сортировки пузырьком проработал 32.54 секунд!

Алгоритм сортировки вставками проработал 6.17 секунд!

***B) Для 1000000 элементов:***

Оба алгоритма проработали очень долго (не дождался)