Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н. Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н. Э. Баумана)

ФАКУЛЬТЕТ «Информатика, искусственный интеллект и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

ОТЧЕТ

по лабораторной работе № 3

по курсу «Анализ Алгоитмов»

на тему: «Трудоемкость сортировок»

Студент ИУ7-55Б _____ Романов С. К.
(Группа) (Подпись, дата) (И. О. Фамилия)

Преподаватель _____ Волкова. Л. Л.
(Подпись, дата) (И. О. Фамилия)

2022 г.

Оглавление

1 ВВЕДЕНИЕ

Цель лабораторной работы: изучить, реализовать и протестировать алгоритмы сортировки, оценить их трудоемкость.

Задачи лабораторной работы:

1. Изучить алгоритмы сортировки.
   - Пузырьковая сортировка.
   - Сортировка вставками.
   - Итеративная быстрая сортировка.
2. Оценить трудоемкость алгоритмов и сравнивать их временые характеристики экспериментально.
3. Сделать выводы на основе полученных результатов.

Для достижения поставленных целей и задач необходимо:

1. Изучить теоретические основы алгоритмов сортировки.
2. Реализовать алгоритмы сортировки.
3. Протестировать корректность алгоритмов сортировки.
4. Провести экспериментальное исследование.

В ходе работы будут затронуты следующие темы:

1. Алгоритмы сортировки.
2. Сложность алгоритмов.
3. Тестирование алгоритмов

2 Аналитическая часть

2.1 Пузырьковая сортировка

2.1.1 Описание алгоритма

procedure BUBBLESORT(A)
    $n \leftarrow \texttt{len}(A)$
    for $i \leftarrow 1$ **to** n **do**
        for $j \leftarrow 1$ **to** $n - i$ **do**
            if $A[j] > A[j + 1]$ **then**
                $SWAP(A[j], A[j + 1])$
            end if
        end for
    end for
end procedure

2.1.2 Анализ алгоритма

- Время выполнения алгоритма: $O(n^2)$
- Память: $O(1)$

2.2 Сортировка вставками

2.2.1 Описание алгоритма

procedure INSERTIONSORT(A)
    $n \leftarrow \texttt{len}(A)$
    for $i \leftarrow 1$ **to** n **do**
        $x \leftarrow A[i]$
        $j \leftarrow i - 1$
        while $j \geq 0$ **and** $A[j] > x$ **do**
            $A[j + 1] \leftarrow A[j]$
            $j \leftarrow j - 1$
        end while

$A[j+1] \leftarrow x$
end for
end procedure

2.2.2 Анализ алгоритма

- Время выполнения алгоритма: $O(n^2)$
- Память: $O(1)$

2.3 Итеративная быстрая сортировка

2.3.1 Описание алгоритма

procedure QUICKSORT(A)
$n \leftarrow \texttt{len}(A)$
$S \leftarrow \{\}$ ▷ Стек
$S \leftarrow S \cup \{0, n-1\}$
while $S \neq \{\}$ **do**
$r \leftarrow S[n_S - 1]$
$l \leftarrow S[n_S - 2]$
$S \leftarrow S \setminus \{r, l\}$
$p \leftarrow \texttt{partition}(A, l, r)$
$S \leftarrow S \cup \{l, p-1\}$
$S \leftarrow S \cup \{p+1, r\}$
end while
end procedure

2.3.2 Анализ алгоритма

- Время выполнения алгоритма: $O(n \log n)$
- Память: $O(\log n)$

3 Конструкторская часть

3.1 Разработка алгоритмов

На рисунках 3.1 3.2 3.3 приведены схемы алгоритмов сортировки пузырьком, сортировки вставками и быстрой сортировки соответственно.

3.2 Трудоёмкость алгоритмов

TBD

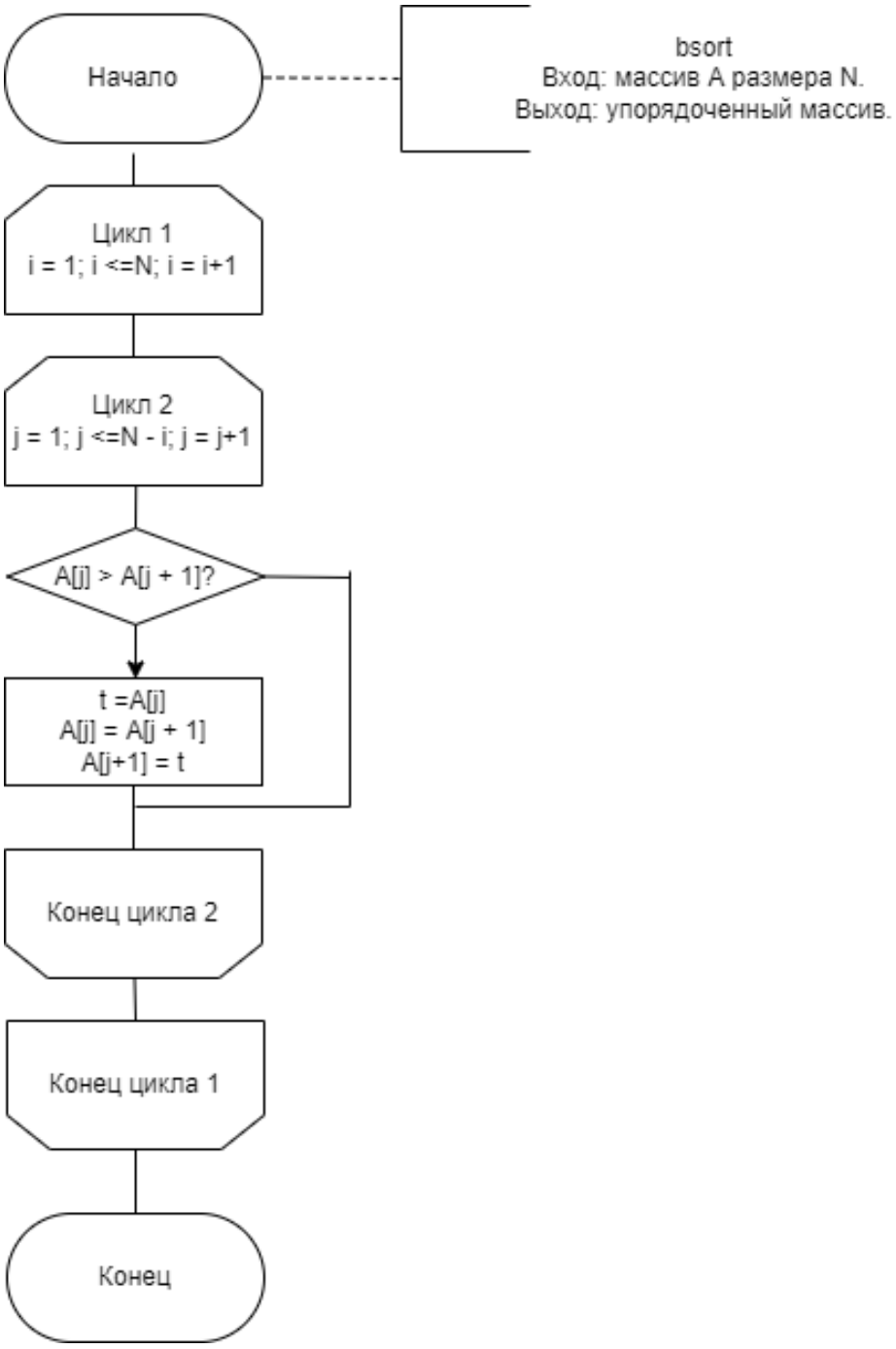


Рисунок 3.1 – Сортирвка пузырьком

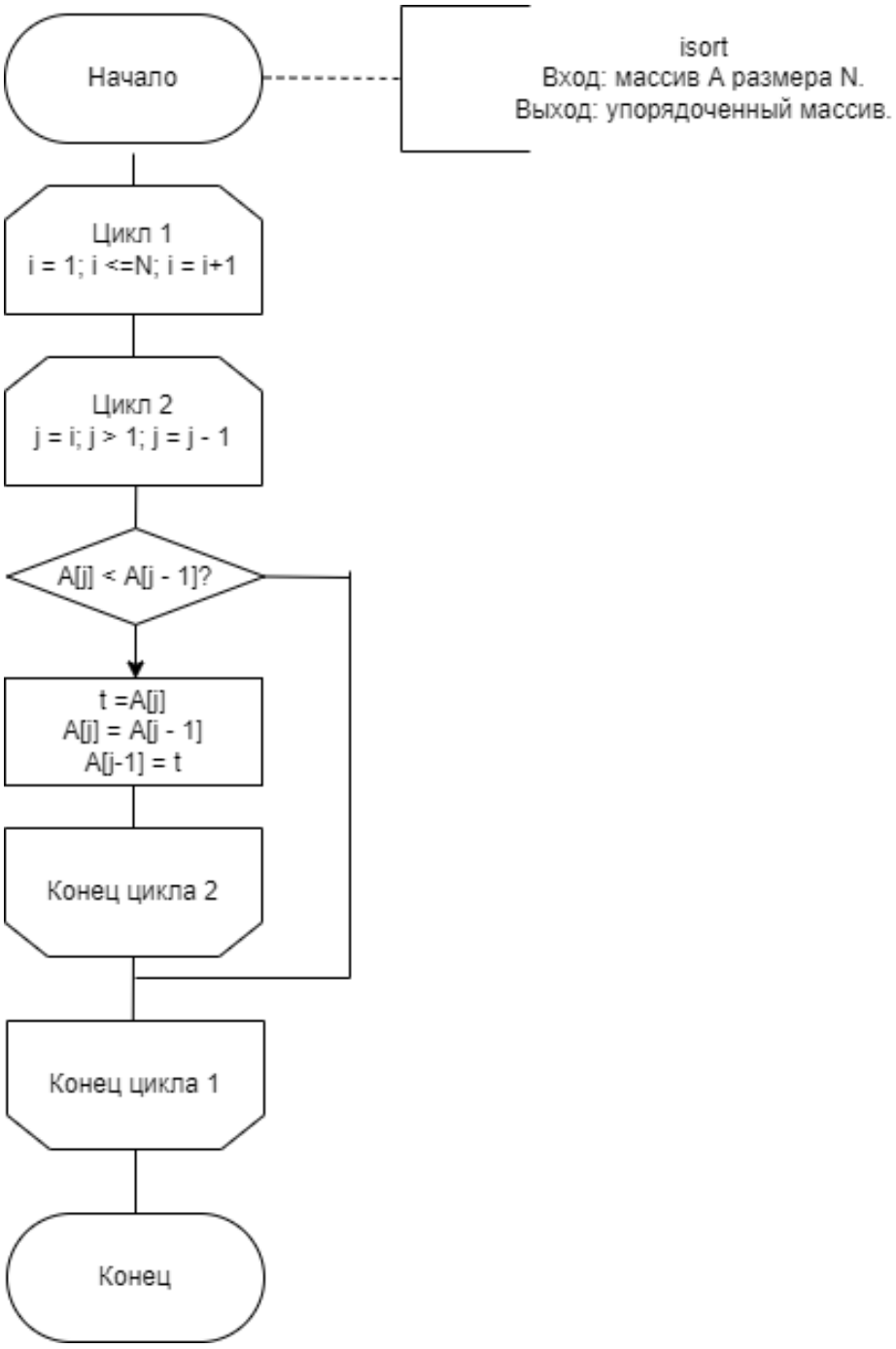


Рисунок 3.2 – Сортировка вставками

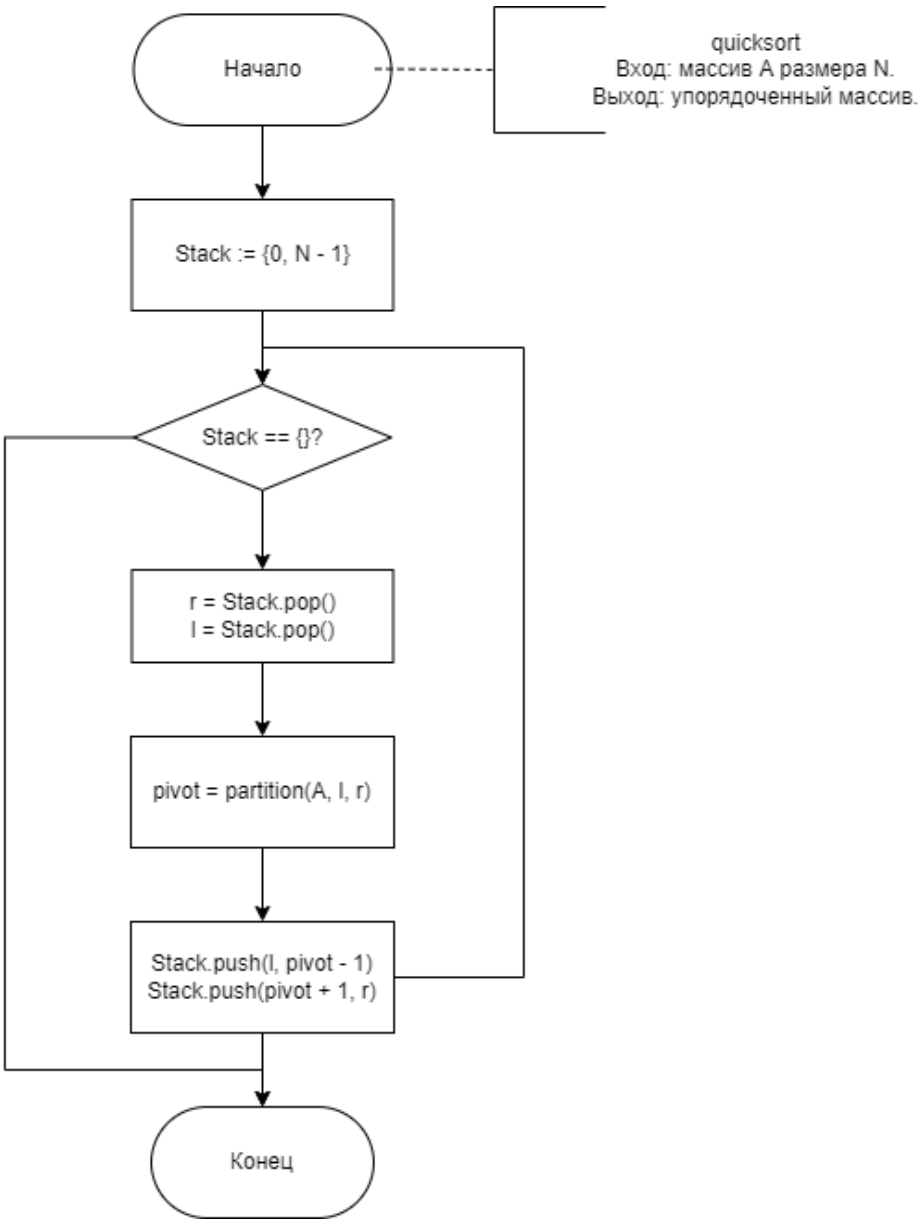


Рисунок 3.3 – Быстрая сортировка

4 Технологическая часть

В данном разделе приведены требования к программному обеспечению, средства реализации и листинги кода.

4.1 Требования к программному обеспечению

К программе предъявляется ряд условий:

– на вход программе подается размер массива (целое число) и массив (целые числа);

– на выходе программа должна выдать отсортированный массив одним из 3 способов;

4.2 Средства реализации

Для реализации данной лабораторной работы необходимо установить следующее программное обеспечение:

- Golang 1.19.1 - язык программирования
- Benchdraw 0.1.0 - Средство визуализации данных
- LaTeX - система документооборота

4.3 Листинги кода

В листингах 4.1, 4.2 и 4.3 приведены реализации алгоритмов сортировки пузырьком, вставками и итеративная быстрая соответственно.

Листинг 4.1 – Сортировка массива пузырьком

```
func BubbleSort[T constraints.Ordered](array []T) {
    n := len(array) - 1
    for i := 0; i < n; i++ {
        for j := 0; j < n-i; j++ {
            if array[j] > array[j+1] {
                array[j], array[j+1] = array[j+1], array[j]
            }
        }
    }
}
```

Листинг 4.2 – Сортировка массива вставками

```
func InsertionSort[T constraints.Ordered](array []T) {
    l := len(array)
    for i := 0; i < l; i++ {
        x := array[i]
        j := i
        for j > 0 && array[j-1] > x {
            array[j] = array[j-1]
            j--
        }
        array[j] = x
    }
}
```

Листинг 4.3 – Итеративная быстрая сортировка массива

```

func partition[T constraints.Ordered](arr []T, left int, right int) int {
    pivot := arr[right]
    i := left - 1
    for j := left; j <= right-1; j++ {
        if arr[j] <= pivot {
            i++
            arr[i], arr[j] = arr[j], arr[i]
        }
    }
    arr[i+1], arr[right] = arr[right], arr[i+1]

    return i + 1
}

func Quicksort[T constraints.Ordered](array []T) {
    if len(array) < 2 {
        return
    }

    left := 0
    right := len(array) - 1
    stack := Stack[[2]int]{}
    stack.Push([2]int{left, right})
    for !stack.IsEmpty() {
        lr := stack.Pop()
        left, right = lr[0], lr[1]
        if right <= left {
            continue
        }
```

```
31          pivot := partition(array, left, right)
32          stack.Push([2]int{left, pivot - 1})
33          stack.Push([2]int{pivot + 1, right})
34      }
35  }
```

В таблице 4.1 приведены тесты для функций, реализующих алгоритмы сортировки. Все тесты пройдены успешно.

Входной массив	Результат	Ожидаемый результат
$[1, 2, 3, 4, 5]$	$[1, 2, 3, 4, 5]$	$[1, 2, 3, 4, 5]$
$[5, 4, 3, 2, 1]$	$[1, 2, 3, 4, 5]$	$[1, 2, 3, 4, 5]$
$[-1, -2, -3, -2, -5]$	$[-5, -3, -2, -2, -1]$	$[-5, -3, -2, -2, -1]$
$[-2, 7, 0, -1, 3]$	$[-2, -1, 0, 3, 7]$	$[-2, -1, 0, 3, 7]$
$[50]$	$[50]$	$[50]$
$[-40]$	$[-40]$	$[-40]$
Пустой массив	Пустой массив	Пустой массив

Таблица 4.1 – Тестирование функций

4.4 Вывод

В данном разделе были разработаны исходные коды трёх алгоритмов сортировки: пузырьком, вставками и быстрая сортировка.

5 Исследовательская часть

Ниже приведены технические характеристики устройства, на котором было проведено тестирование ПО:

- Операционная система: Windows 11 используя Windows Subsystem for Linux 2 (WSL2) [1] Имитирующей Arch Linux [2] 64-bit.
- Оперативная память: 16 GB.
- Процессор: 11th Gen Intel(R) Core(TM) i5-11320H @ 3.20GHz [3].

5.1 Время выполнения алгоритмов

Алгоритмы тестировались при помощи ”бенчмарков” предоставляемых встренными средствами языка Go [4]. Пример такого "бенчмарка приведен в листинге 5.1. Количество повторов регулируется тестирующей системой самостоятельно, хотя при желание оные можно задать. Точное количество повторов определяется в зависимости от того, был ли получен стабильный результат согласно системе.

Листинг 5.1 – Пример бенчмарка

```
func BenchmarkQuicksortSorted(b *testing.B) {
    for steps, amount := 0, 0; steps < STEPS; steps++ {
        amount += INC
        b.Run(fmt.Sprintf("size=%d", amount), func(b *testing.B) {
            sample := GenerateSortedArray(amount)
            sampleCopy := make([]int, amount)
            b.ResetTimer()
            for i := 0; i < b.N; i++ {
                //b.StopTimer()
                copy(sampleCopy, sample)
                //b.StartTimer()
                Quicksort(sampleCopy)
            }
        })
    }
}
```

Таблица времени выполнения сортировок на отсортированных данных (в наносекундах)

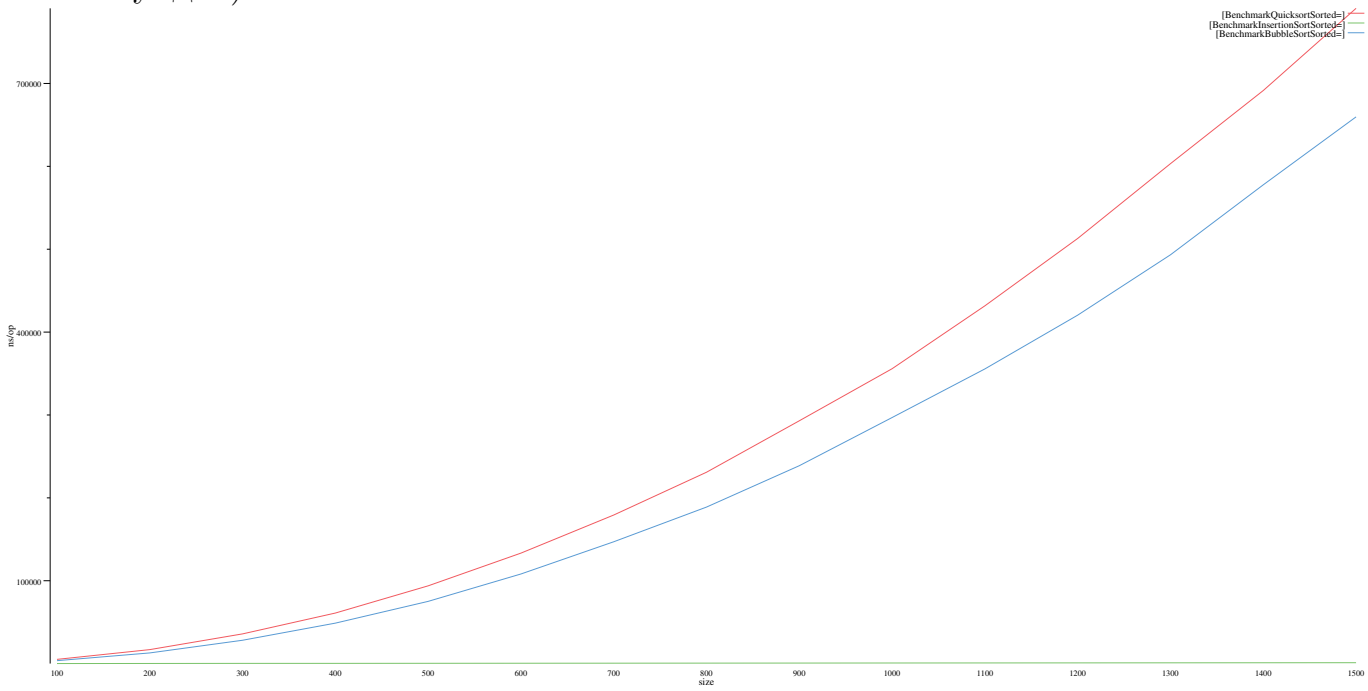


Таблица времени выполнения сортировок на отсортированных данных в обратном порядке (в наносекундах)

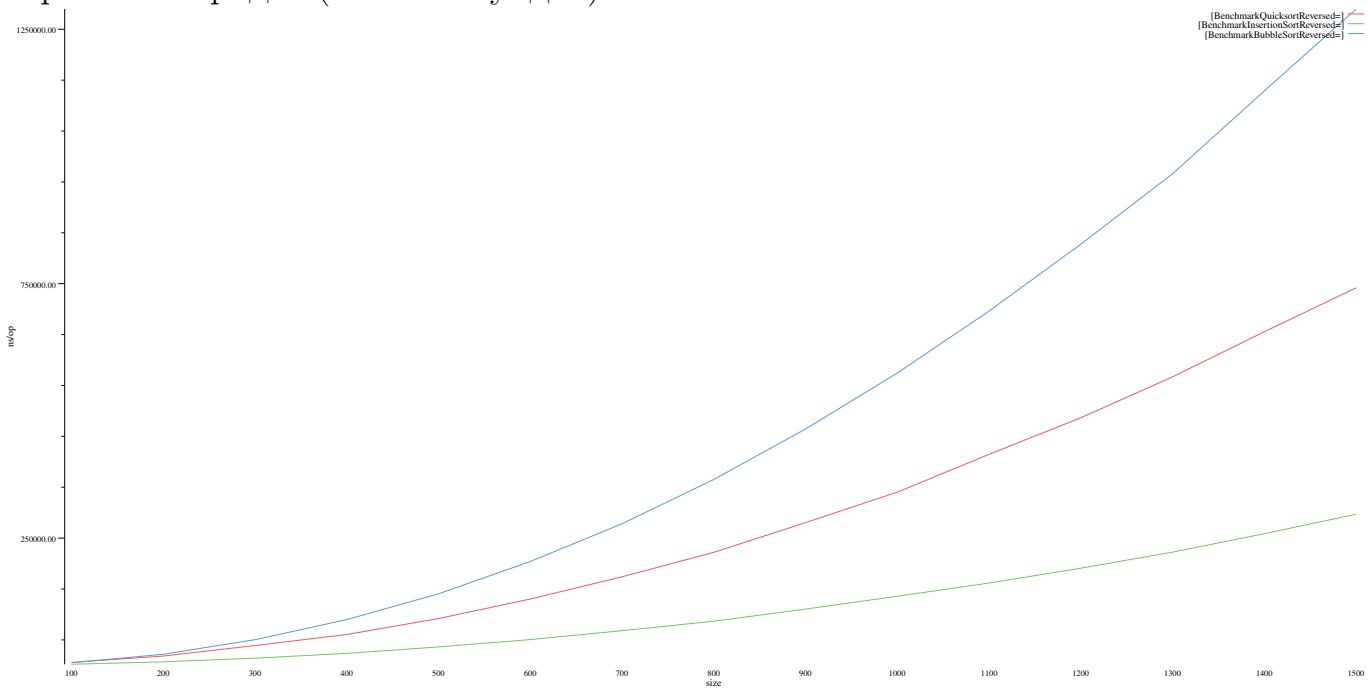
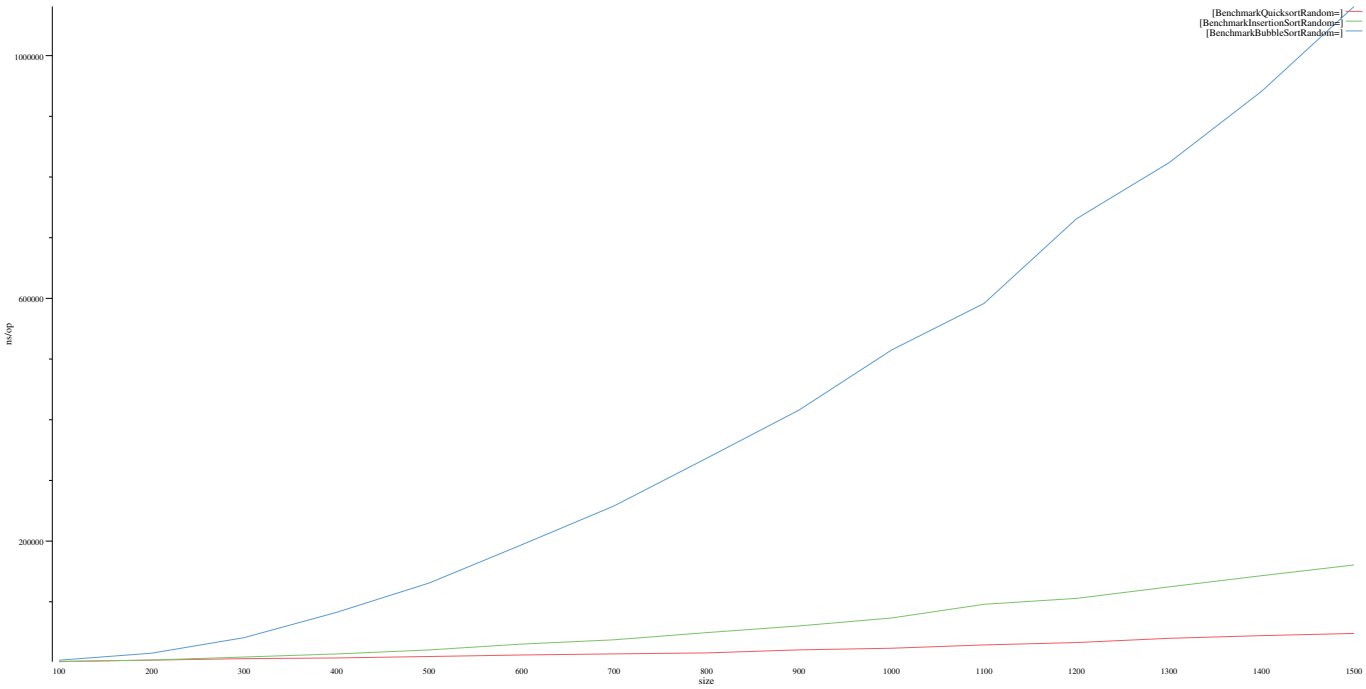


Таблица времени выполнения сортировок на случайных данных (в наносекундах)



5.2 Вывод

Как и ожидалось в результате оценки трудоемкости алгоритмов, сортировка вставками работает очень быстро на уже отсортированном массиве. Время сортировки вставками на всех трёх видах массивов примерно одинаково. При обратно отсортированном массиве, быстрая сортировка начинает значительно проигрывать по скорости.

6 ЗАКЛЮЧЕНИЕ

В ходе выполнения лабораторной работы были достигнуты следующие задачи:

- были изучены и реализованы 3 алгоритма сортировки: пузырёк, вставками, быстрая сортировка;
- были приведены аналитические данные о выше перечисленных алгоритмах;
- были приведены подробные блок-схемы, описывающие алгоритмы;
- был проведён сравнительный анализ алгоритмов на основе экспериментальных данных.

Экспериментальные данные показали различные сильные и слабые стороны каждого алгоритма. Так например:

– сортировка пузырьком работает крайне медленно независимо от входных данных;

– быстрая сортировка работает быстрее на случайных данных, поэтому лучше всего подходит как общее решение абстрактной задачи на сортировку данных;

– быстрая сортировка работает медленно на обратно отсортированном массиве.

– сортировка вставками работает быстрее на уже отсортированном массиве.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Windows Subsystem for Linux 2 [Электронный ресурс]. — Дата обращения: 19.09.2022. Режим доступа: `https://learn.microsoft.com/en-us/windows/wsl/about#what-is-wsl-2`.
2. Arch Linux [Электронный ресурс]. — Дата обращения: 19.09.2022. Режим доступа: `https://archlinux.org/`.
3. Процессор Intel® Core™ i5-11320H [Электронный ресурс]. — Дата обращения: 19.09.2022. Режим доступа: `https://ark.intel.com/content/www/ru/ru/ark/products/217183/intel-core-i511320h-processor-8m-cache-up-to-4-50-ghz-with-ipu.html`.
4. Go [Электронный ресурс]. — Дата обращения: 19.09.2022. Режим доступа: `https://go.dev/`.