Практическая работа №2

Тема: «Алгоритмы сортировки».

Цель работы: изучить алгоритмы сортировки.

Ход работы:

Сортировка перемешивание, или двунаправленная (англ. Cocktail sort) — разновидность пузырьковой сортировки. Анализируя метод пузырьковой сортировки, можно отметить два обстоятельства. Во-первых, если при движении по части массива перестановки не происходят, то эта часть массива уже отсортирована и, следовательно, её можно исключить из рассмотрения. Во-вторых, при движении от конца массива к началу минимальный элемент «всплывает» на первую позицию, а максимальный элемент сдвигается только на одну позицию вправо. Эти две идеи приводят к следующим модификациям в методе пузырьковой сортировки. Границы рабочей части массива (то есть части массива, где происходит движение) устанавливаются в месте последнего обмена на каждой итерации. Массив просматривается поочередно справа налево и слева направо.

Листинг 1.Сортировка перемешивания

```
from random2 import randint

N = 10
arr = []
for i in range(N):
    arr.append(randint(1, 99))
print(arr)

left = 0
right = len(arr) - 1

while left <= right:
    for i in range(left, right, +1):
        if arr[i] > arr[i + 1]:
            arr[i], arr[i + 1] = arr[i + 1], arr[i]
    right -= 1

    for i in range(right, left, -1):
        if arr[i - 1] > arr[i]:
            arr[i], arr[i - 1] = arr[i - 1], arr[i]
    left += 1

print(arr)
```

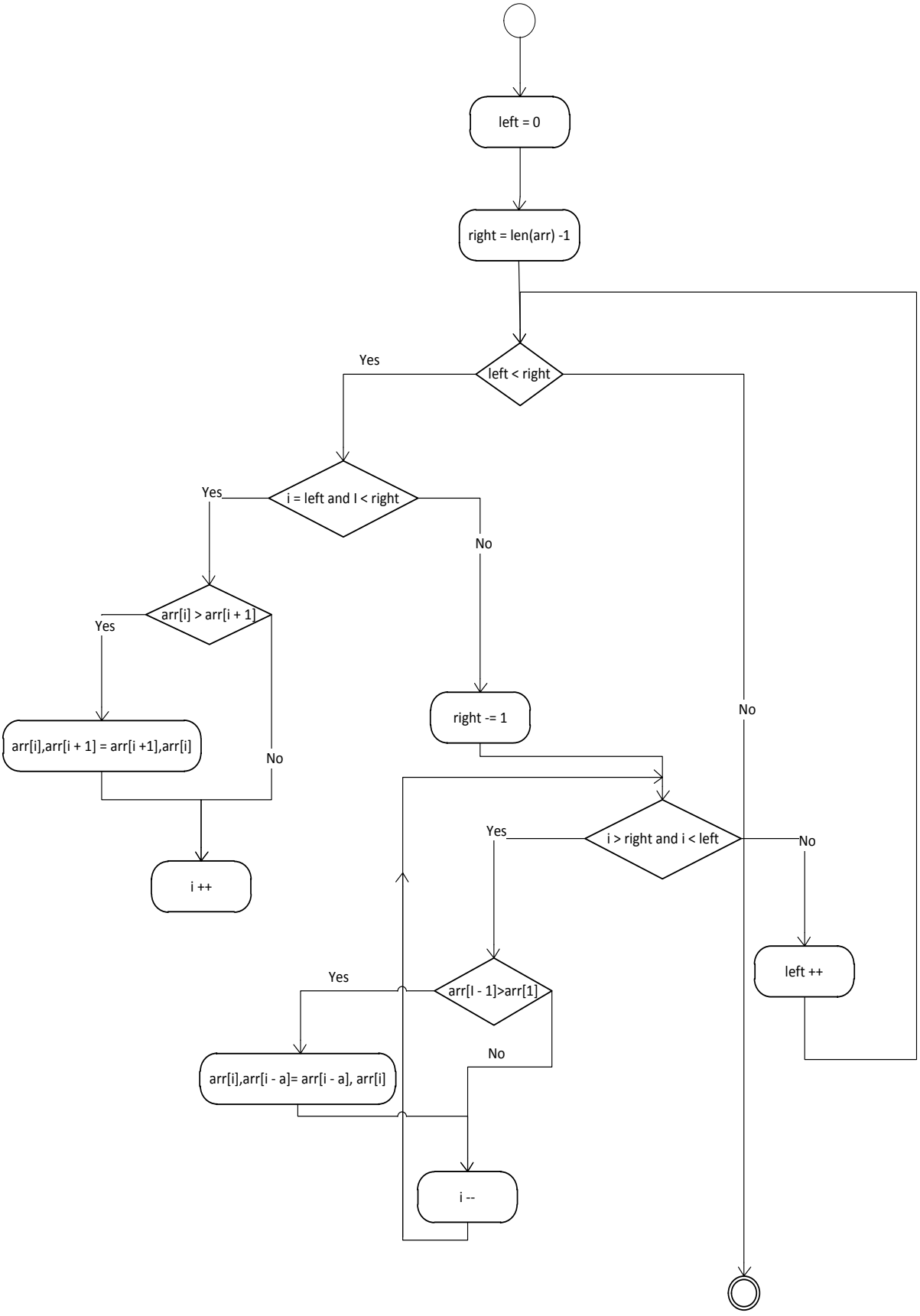

Рис. 1 Диаграмма деятельности сортировки перемешивания

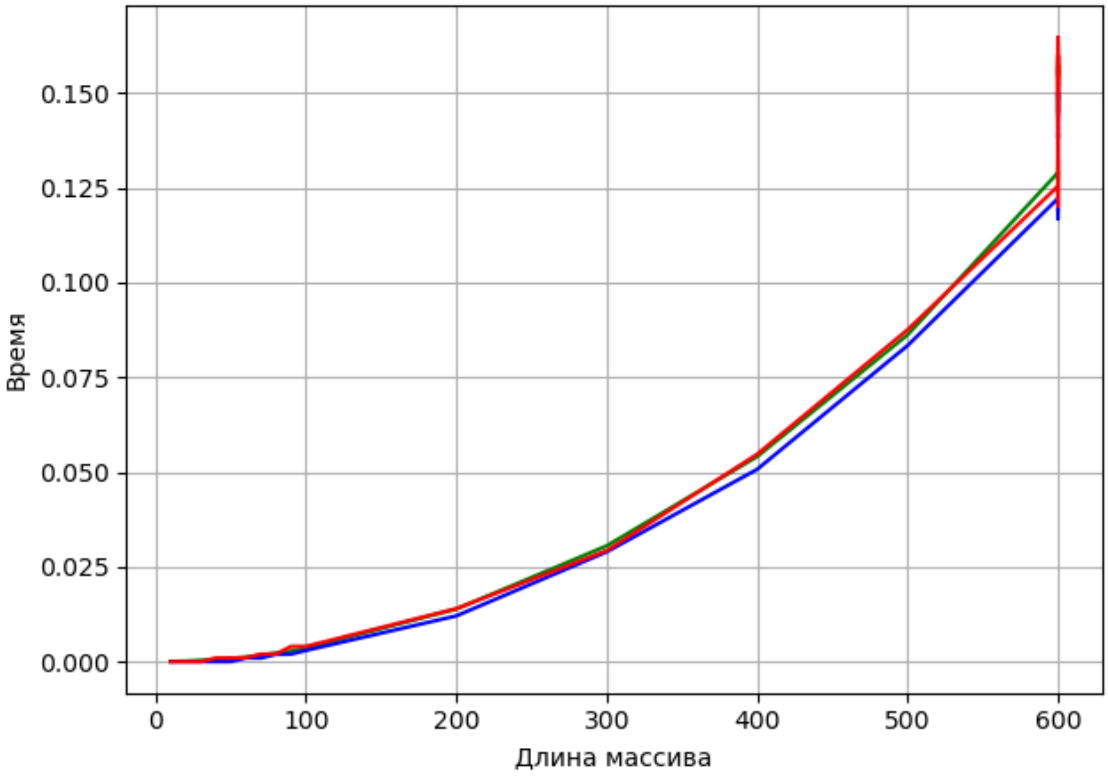

Рис. 2 График сортировки перемешивания

Сортировка пузырьком (англ. bubble sort) — простой алгоритм сортировки. Для понимания и реализации этот алгоритм — простейший, но эффективен он лишь для небольших массивов. Сложность алгоритма: O(n2). Алгоритм считается учебным и практически не применяется вне учебной литературы, вместо него на практике применяются более эффективные алгоритмы сортировки. В то же время метод сортировки обменами лежит в основе некоторых более совершенных алгоритмов, таких как шейкерная сортировка, пирамидальная сортировка и быстрая сортировка. Алгоритм состоит из повторяющихся проходов по сортируемому массиву. За каждый проход элементы последовательно сравниваются попарно и, если порядок в паре неверный, выполняется обмен элементов. Проходы по массиву повторяются N-1 раз или до тех пор, пока на очередном проходе не окажется, что обмены больше не нужны, что означает — массив отсортирован. При каждом проходе алгоритма по внутреннему циклу, очередной наибольший элемент масс

сива ставится на своё место в конце массива рядом с предыдущим «наибольшим элементом», а наименьший элемент перемещается на одну позицию к началу массива («всплывает» до нужной позиции, как пузырёк в воде — отсюда и название алгоритма). Сложность: O(n2).

Листинг 2. Пузырьковая сортировка

```
from random2 import randint

N = 10
a = []
for i in range(N):
    a.append(randint(1, 99))
print(a)

i = 0
while i < N - 1:
    j = 0
    while j < N - 1 - i:
        if a[j] > a[j+1]:
            a[j], a[j+1] = a[j+1], a[j]
        j += 1
    i += 1

print(a)
```

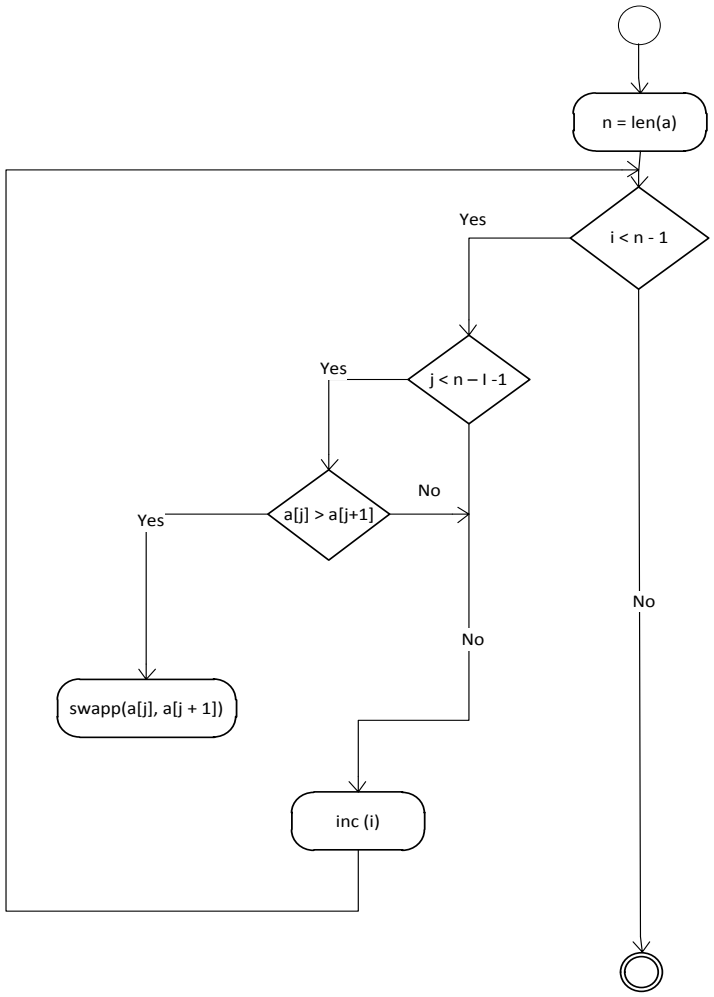

Рис.3 Диаграмма деятельности пузырьковой сортировки

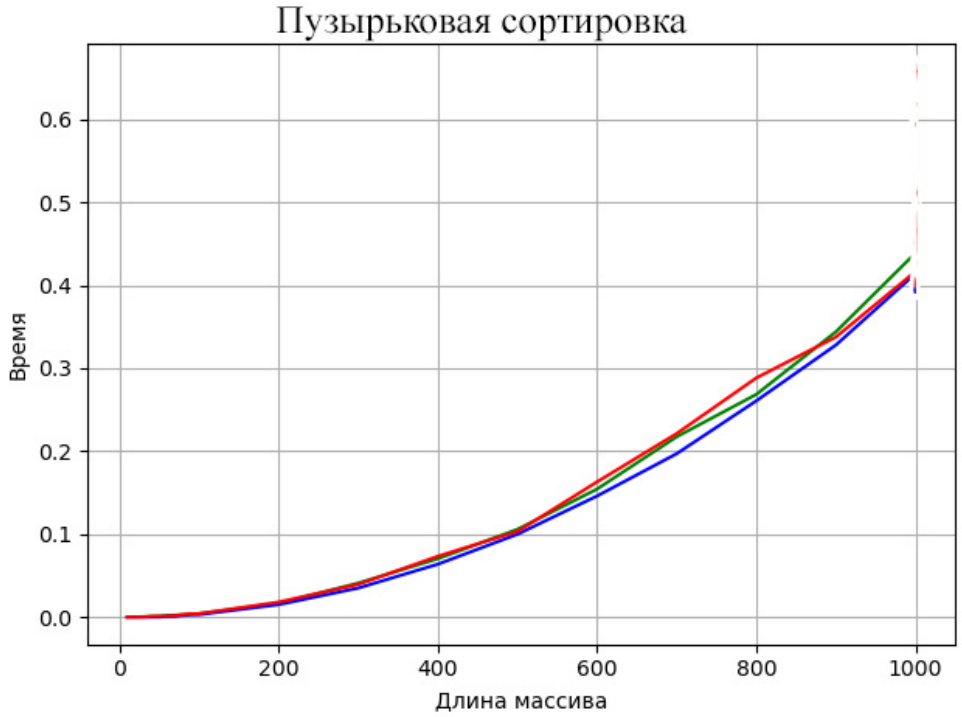

Рис.4 График пузырьковой сортировки

Сортировка вставками (англ. Insertion sort) — алгоритм сортировки, в котором элементы входной последовательности просматриваются по одному, и каждый новый поступивший элемент размещается в подходящее место среди ранее упорядоченных элементов. Вычислительная сложность — 0(n2). Время выполнения алгоритма зависит от входных данных: чем большее множество нужно отсортировать, тем большее время потребуется для выполнения сортировки. Также на время выполнения влияет исходная упорядоченность массива. Время работы алгоритма для различных входных данных одинакового размера зависит от элементарных операций, или шагов, которые потребуется выполнить. Временная сложность алгоритма — 0(n2). Однако, из-за константных множителей и членов более низкого порядка алгоритм с более высоким порядком роста может выполняться для небольших входных данных быстрее, чем алгоритм с более низким порядком роста.

Листинг 3. Сортировка вставками

```
from random2 import randint

def insertion_sort(arr):
    for i in range(1, len(arr)):
        key = arr[i]
        j = i-1
        while j >=0 and key < arr[j] :
            arr[j+1] = arr[j]
            j -= 1
        arr[j+1] = key

N = 10
arr = []
for i in range(N):
    arr.append(randint(1, 99))

print (arr)

insertion_sort(arr)
print ("Sorted array is:")
for i in range(len(arr)):
    print (arr[i])

input()
```

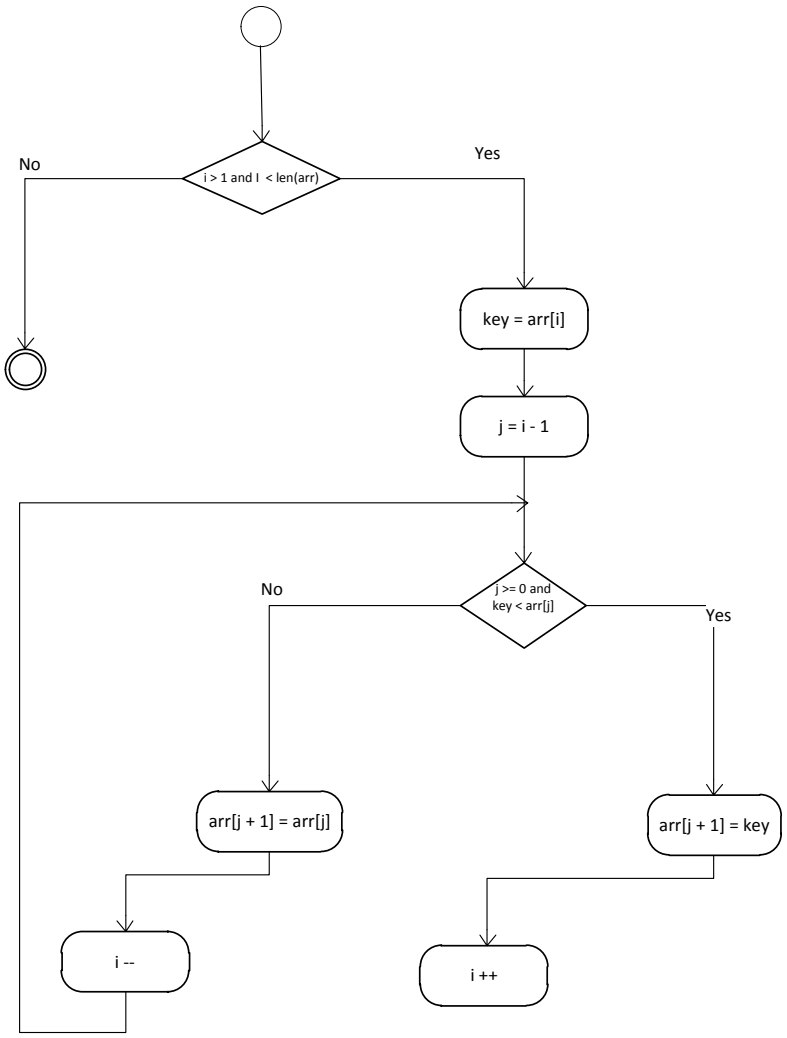

Рис.5 Диаграмма деятельности сортировки вставками

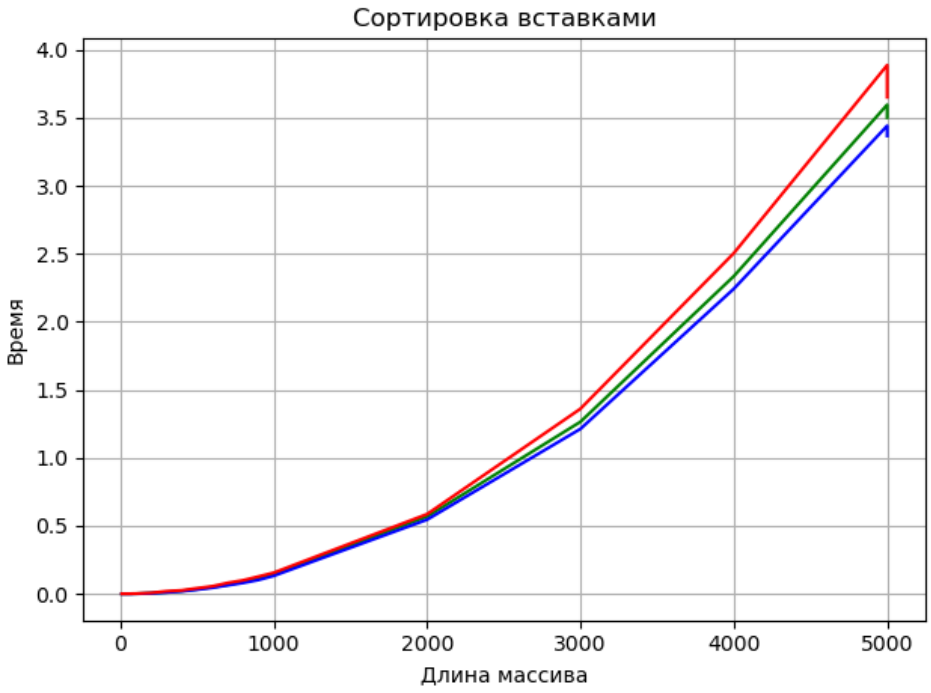

Рис.6 График сортировки вставками

Сортировка выбором (Selection sort) — алгоритм сортировки. Может быть как устойчивый, так и неустойчивый. На массиве из n элементов имеет время выполнения в худшем, среднем и лучшем случае Θ(n2), предполагая что сравнения делаются за постоянное время. Наихудший случай:

Число сравнений в теле цикла равно (N-1)*N/2.

Число сравнений в заголовках циклов (N-1)*N/2.

Число сравнений перед операцией обмена N-1.

Суммарное число сравнений N2−1.

Число обменов N-1.

Наилучший случай:

Время сортировки 10000 коротких целых чисел на одном и том же программно-аппаратном комплексе сортировкой выбором составило ≈40сек., а ещё более улучшенной сортировкой пузырьком ≈30сек.

Пирамидальная сортировка сильно улучшает базовый алгоритм, используя структуру данных «куча» для ускорения нахождения и удаления минимального элемента.

Существует также двунаправленный вариант сортировки методом выбора, в котором на каждом проходе отыскиваются и устанавливаются на свои места и минимальное, и максимальное значения.

Листинг 4.Сортировка выбором

```
from random2 import randint
N = 10
arr = []
for i in range(N):
    arr.append(randint(1, 99))
print(arr)

    i = 0
    m = i
    j = i + 1

    while j < N:
        if arr[j] < arr[m]:
            m = j
            j += 1
    arr[i], arr[m] = arr[m], arr[i]

    i += 1

print(arr)
```

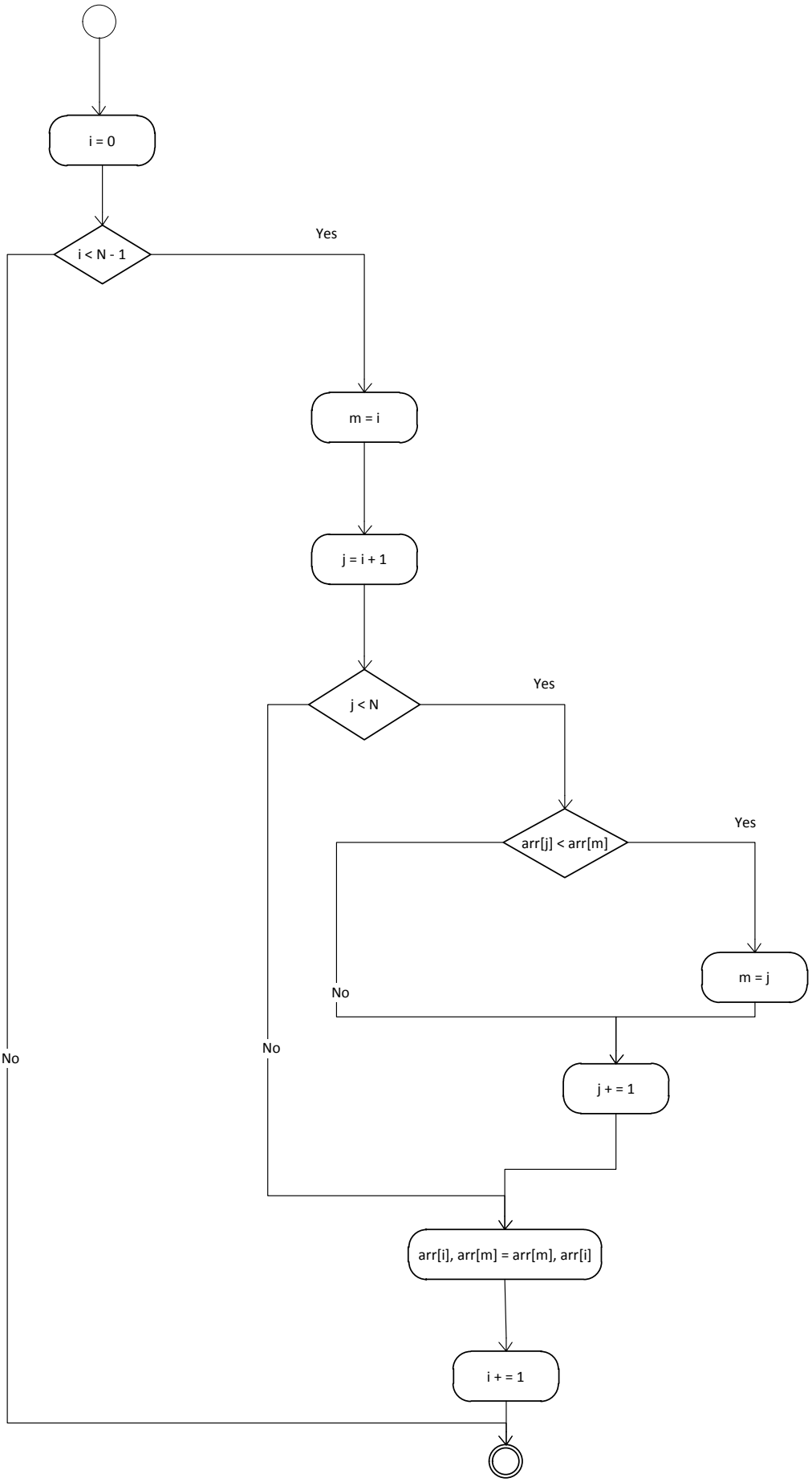

Рис.7 Диаграмма деятельности сортировки выбором

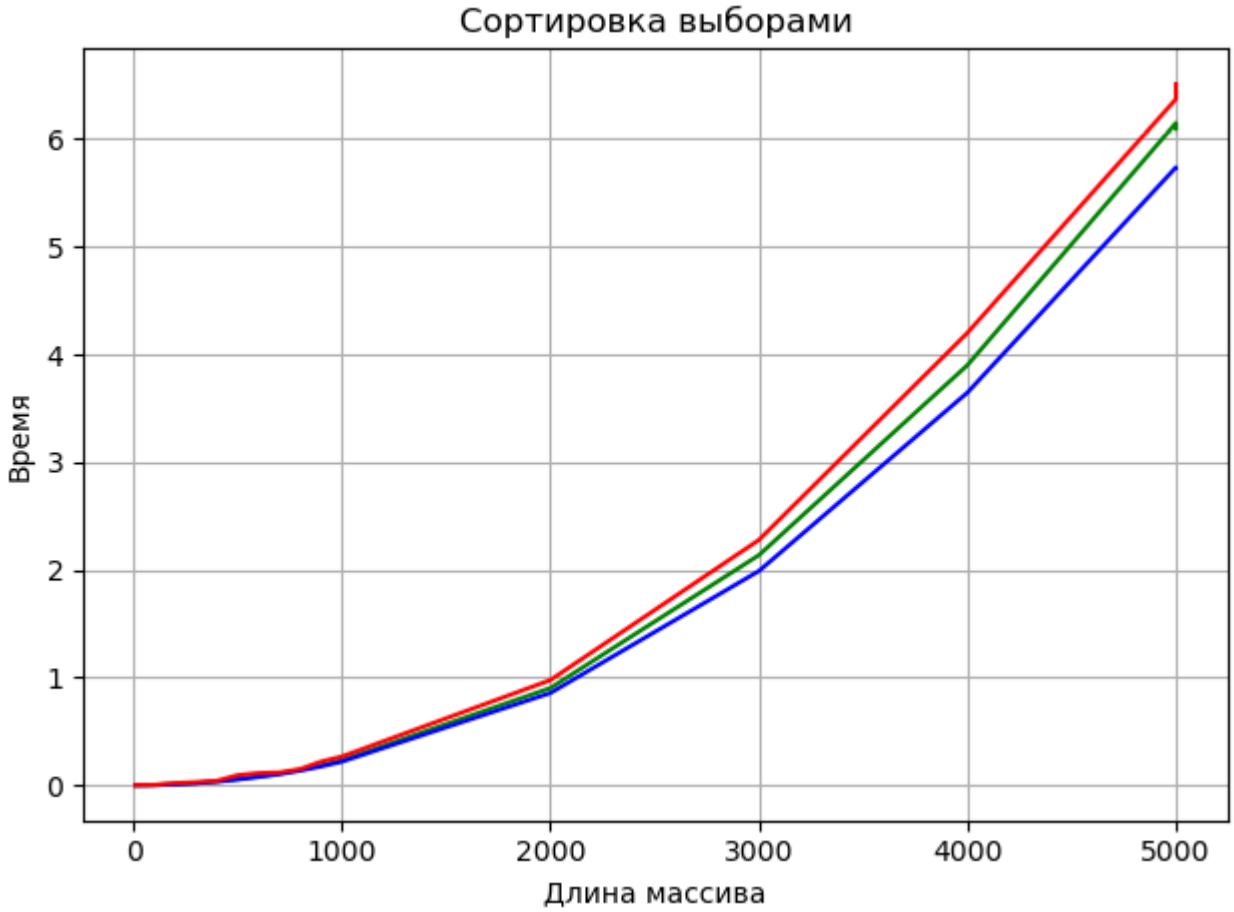

Рис.8 График сортировки выбором

Вывод: в ходе выполнения работы были изучены алгоритмы сортировки: сортировка перемешивания, пузырьковая, выбором и сортировка вставками , а так же построены диаграммы деятельности каждой из сортировок.