

Федеральное государственное автономное образовательное учреждение высшего  
образования

САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ  
УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ  
ФАКУЛЬТЕТ ПРОГРАММНОЙ ИНЖЕНЕРИИ И КОМПЬЮТЕРНОЙ ТЕХНИКИ

## ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №4

«Алгоритмы и структуры данных:  
Алгоритмы сортировки.»

Проверил:  
Сентерев Ю.А. \_\_\_\_\_  
« \_\_\_\_\_ » \_\_\_\_\_ 201\_ г.

Выполнил:  
Студент группы Р3255  
Кабардинов Д. В. \_\_\_\_\_

Оценка \_\_\_\_\_

Санкт-Петербург

2019

## Задание

С использованием любого из улучшенных методов сортировки решить задачу согласно своему варианту. 5. Составить программу для быстрой перестройки данного массива А, в котором элементы расположены в порядке возрастания, так, чтобы после перестройки эти же элементы оказались расположенными в порядке убывания.

### Цель работы:

- исследовать и изучить улучшенные методы сортировки на примерах метода Шелла и быстрой сортировки;
- овладеть навыками написания программ с использованием улучшенных методов сортировки на языке программирования Python.

### Ход выполнения работы

#### 1. Сортировка Шелла

Идея метода заключается в сравнение разделенных на группы элементов последовательности, находящихся друг от друга на некотором расстоянии. Изначально это расстояние равно  $d$  или  $N/2$ , где  $N$  — общее число элементов. На первом шаге каждая группа включает в себя два элемента расположенных друг от друга на расстоянии  $N/2$ ; они сравниваются между собой, и, в случае необходимости, меняются местами. На последующих шагах также происходят проверка и обмен, но расстояние  $d$  сокращается на  $d/2$ , и количество групп, соответственно, уменьшается. Постепенно расстояние между элементами уменьшается, и на  $d=1$  проход по массиву происходит в последний раз. Реализация алгоритма сортировки Шелла на языке Python:

```
def shellSort(array):
    gapSizes = [7, 3, 1]
    for gap in gapSizes:
        for startPosition in range(gap):
            gapInsertionSort(array, startPosition, gap)

def gapInsertionSort(array, low, gap):
    for i in range(low + gap, len(array), gap):
        currentvalue = array[i]
        position = i

        while position >= gap and array[position - gap] < currentvalue:
            array[position] = array[position - gap]
            position = position - gap

        array[position] = currentvalue

sortedList = [1, 2, 5, 10, 15, 43, 55, 67, 87, 90, 99, 100]
shellSort(sortedList)
print(sortedList)
```

Результат работы программы:

```
dmitrii@kdv:~/projects/TeX$ python3 algorithms\ and\ data\ structures\lab4\ShellSort.py
[100, 99, 90, 87, 67, 55, 43, 15, 10, 5, 2, 1]
```

## 2. Быстрая сортировка

В данном примере в списке *left* собираются элементы, меньшие  $q$ , в списке *right* — большие  $q$ , а в списке *middle* — равные  $q$ . Разделение на три списка, а не на два используется для того, чтобы алгоритм не заикливался, например, в случае, когда в списке остались только равные элементы. Барьерный элемент  $q$  выбирается случайным образом из списка при помощи функции `choice` из модуля `random`.

```
import random

def quickSort(nums):
    if len(nums) <= 1:
        return nums
    else:
        q = random.choice(nums)
        left = [elem for elem in nums if elem > q]

        middle = [q] * nums.count(q)
        right = [elem for elem in nums if elem < q]
        return quickSort(left) + middle + quickSort(right)

sorted = [1, 2, 4, 6, 7, 8, 10, 13, 46, 56, 68, 98]
reversed = quickSort(sorted)
print(reversed)
```

Результат работы программы:

```
dmitrii@kdv:~/projects/TeX$ python3 algorithms\ and\ data\ structures/lab4/quickSort.py
[98, 68, 56, 46, 13, 10, 8, 7, 6, 4, 2, 1]
```

### Выводы

Мне удалось выполнить поставленную задачу двумя 2 способами:

1. Сортировка Шелла
2. Быстрая сортировка

Таким образом, цель работы достигнута.

### Список используемой литературы:

1. <https://docs.python.org/3/tutorial/introduction.html#lists>  
- Lists
2. <https://docs.python.org/3.5/tutorial/index.html> - The Python Tutorial
3. Стивен Скиена - Алгоритмы. Руководство по разработке
4. Никлаус Вирт - Алгоритмы и структуры данных
5. Томас Кормен - Алгоритмы. Построение и анализ
6. <https://www.pythoncentral.io/singly-linked-list-insert-node/> - Python Data Structures Tutorial