

**МИНИСТЕРСТВО ТРАНСПОРТА РОССИЙСКОЙ ФЕДЕРАЦИИ**

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«РОССИЙСКИЙ УНИВЕРСИТЕТ ТРАНСПОРТА»**

**(МИИТ)**

|  |
| --- |
| **ИНСТИТУТ ТРАНСПОРТНОЙ ТЕХНИКИ И СИСТЕМ УПРАВЛЕНИЯ**  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

Кафедра «Управление и защита информации»

**Лабораторная работа №8**

**«Пирамидальная сортировка»**

**по дисциплине**

**«Методы программирования»**

**Выполнил:** студент группы ТКИ-311

Куминов В. П.

**Проверил:** к.т.н., доц. Логинова Л. Н.,

к.т.н., доц. Сафронов А. И.

**Москва – 2022 г.**

**1. Цель**

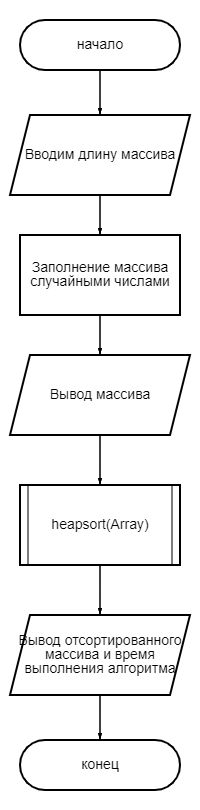
Изучить работу алгоритма пирамидальной сортировки.

**2. Постановка задачи**

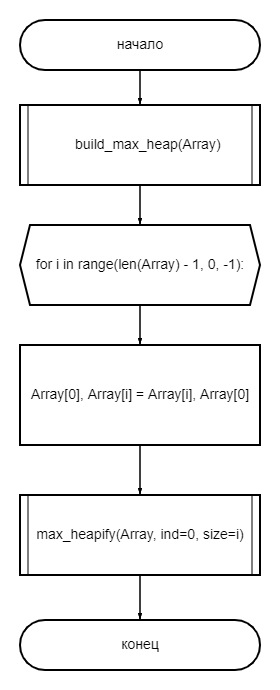
Реализовать на выбранном языке программирования алгоритм пирамидальной сортировки.

3. Блок-схема алгоритма пирамидальной сортировки

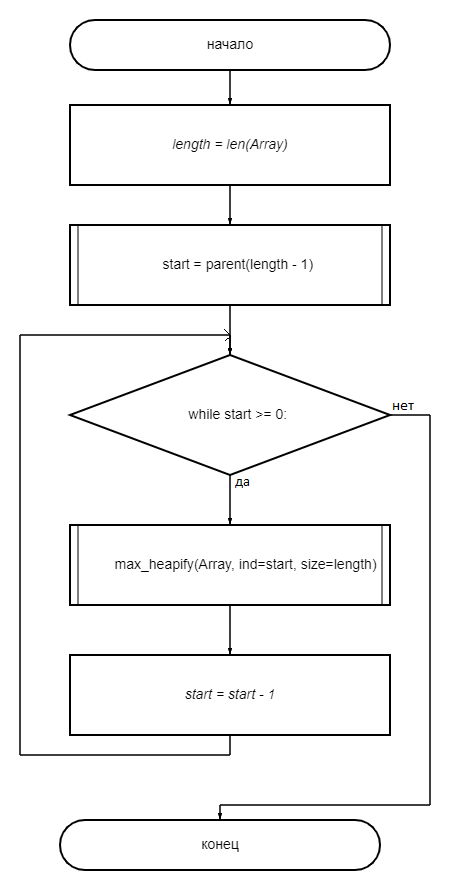
3.1. Блок-схема основной программы



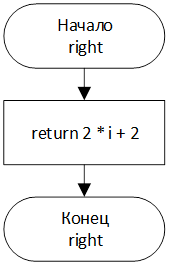
3.2. Блок-схема функции heapsort



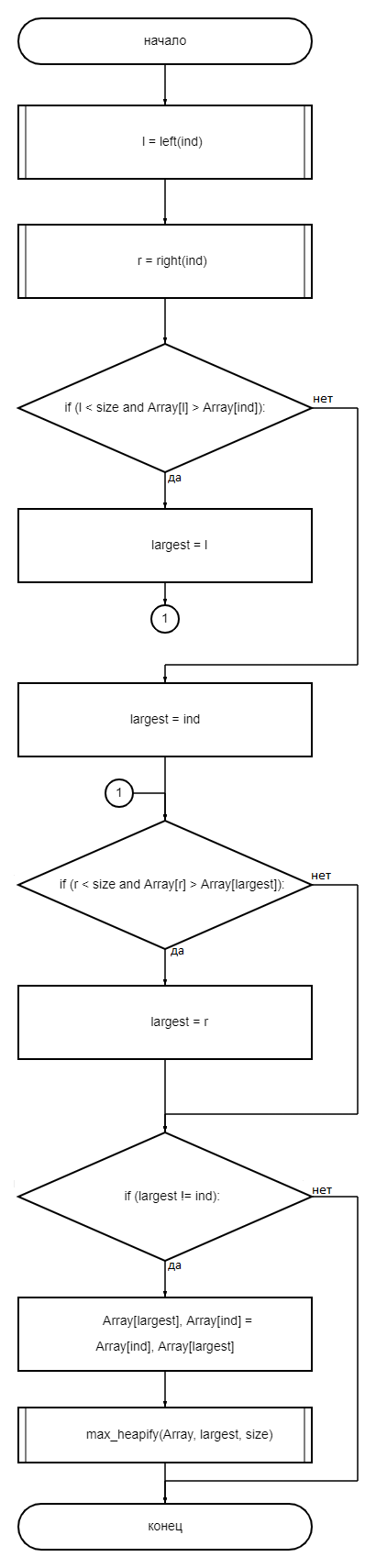
3.3. Блок-схема функции build\_max\_heap



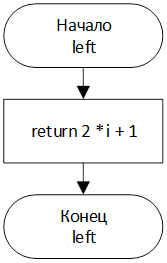
3.4. Блок-схема функции right



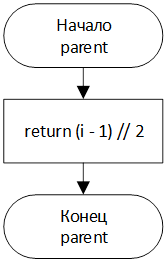
3.5. Блок-схема функции max\_heapify



3.5. Блок-схема функции left



3.5. Блок-схема функции parent



**4. Подбор тестовых примеров**

Входные данные: 4, 23, 10, 15, 3

4(0)

/ \

23(1) 10(2)

/ \

15(3) 3(4)

Числа в скобках - индексы в представлении данных в виде массива.

4 < 23 => меняем 0 и 1 элементы местами.

23(0)

/ \

4(1) 10(2)

/ \

15 (3) 3(4)

4 < 15 => меняем 1 и 3 элементы местами.

23(0)

/ \

15(1) 10(2)

/ \

4(3) 3(4)

Первый узел становится наибольшим элементом массива, а последний узел встаёт на место первого.

[23]

3(0)

/ \

15(1) 10(2)

/

4(3)

3 < 15 => меняем 0 и 1 элементы местами.

15(0)

/ \

3(1) 10(2)

/

4(3)

3 < 4 => меняем 1 и 3 элементы местами.

15(0)

/ \

4(1) 10(2)

/

3(3)

Первый узел становится самым левым элементом массива, а последний узел встаёт на место первого.

[15, 23]

10(0)

/ \

4(1) 3(2)

10 > 4 => обмена нет.

Первый узел становится самым левым элементом массива, а последний узел встаёт на место первого.

[10, 15, 23]

3(0)

/

4(1)

3 < 4 => меняем 0 и 1 элементы местами.

4(0)

/

3(1)

Первый Узел становится самым левым элементом массива, а последний узел встаёт на место первого.

[4, 10, 15, 23]

3(0)

Оставшийся узел становится самым левым элементом массива.

Результат сортировки: [3, 4, 10, 15, 23]

**5. Код программы**

import random

import time

def parent(i):

return (i - 1) // 2

def left(i):

return 2 \* i + 1

def right(i):

return 2 \* i + 2

def max\_heapify(Array, ind, size):

l = left(ind)

r = right(ind)

if (l < size and Array[l] > Array[ind]):

largest = l

else:

largest = ind

if (r < size and Array[r] > Array[largest]):

largest = r

if (largest != ind):

Array[largest], Array[ind] = Array[ind], Array[largest]

def build\_max\_heap(Array):

length = len(Array)

start = parent(length - 1)

while start >= 0:

max\_heapify(Array, ind=start, size=length)

start = start - 1

def heapsort(Array):

build\_max\_heap(Array)

for i in range(len(Array) - 1, 0, -1):

Array[0], Array[i] = Array[i], Array[0]

max\_heapify(Array, ind=0, size=i)

lengthArray = int(input('Введите длину массива: '))

Array = [random.randint(10, 99) for i in range(lengthArray)]

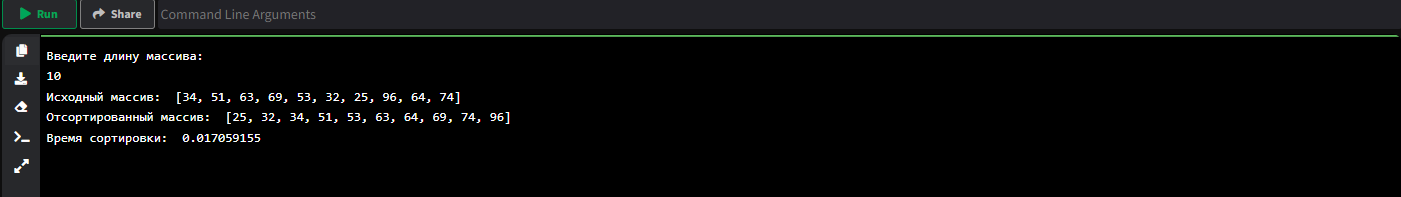
print('Исходный массив: ', Array)

heapsort(Array)

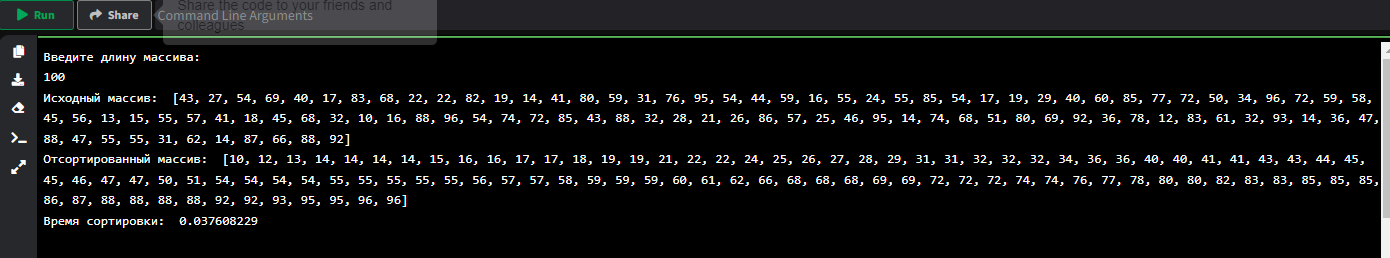
print('Отсортированный массив: ', Array)

print("Время сортировки: ", time.process\_time(),"\n")

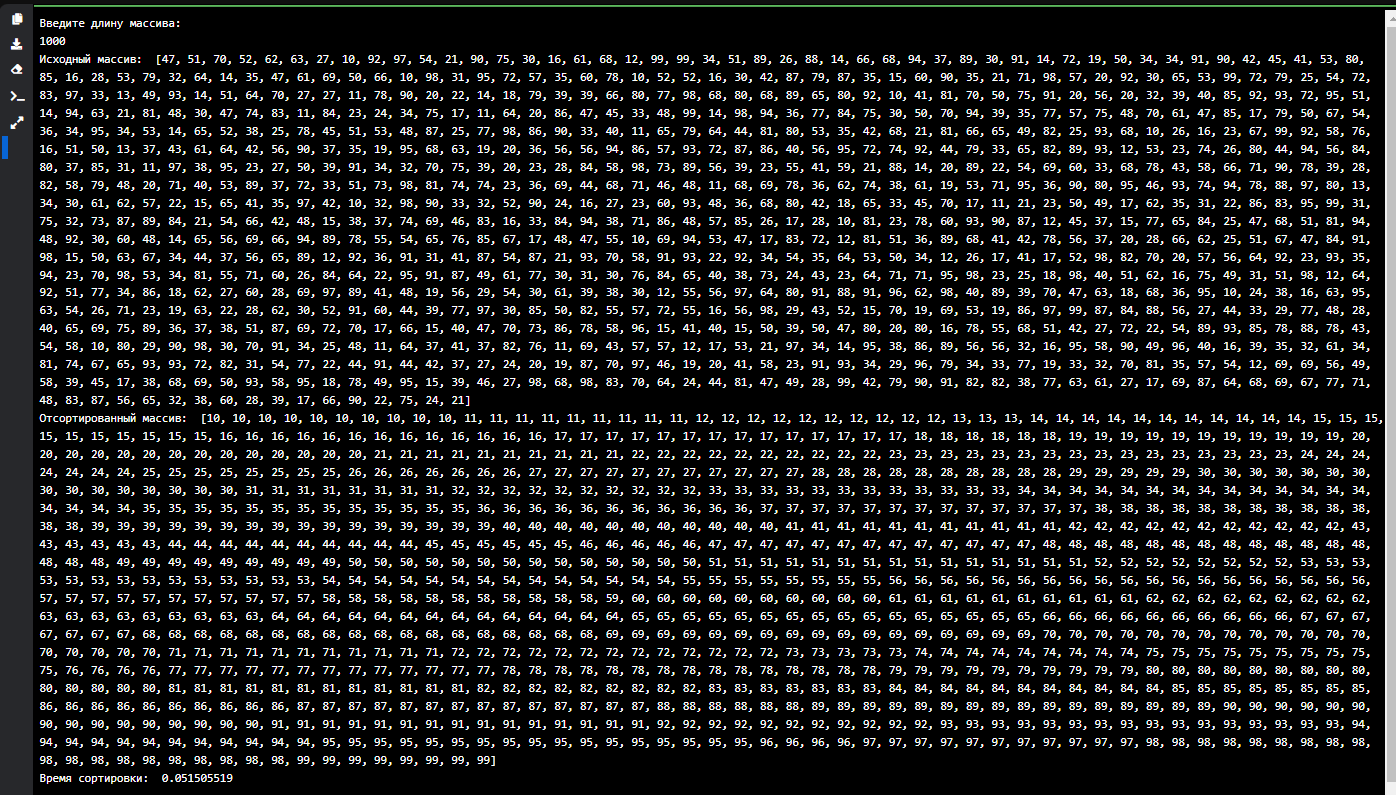
**6. Проверка тестовых примеров на ПК**



10 элементов



100 элементов



1000 элементов

Таблица 1 – Анализ всех реализованных сортировок

|  |  |  |  |
| --- | --- | --- | --- |
| Количество сортируемых элем. | Пирамидальная  Сортировка (t) мс | Сортировка вставкой (t) мс | Сортировка  Слиянием (t) мс |
| 10 | 0,017059155 | 0,0251427 | 0,02973772 |
| 100 | 0,037608229 | 0,0228232 | 0,02937872 |
| 1000 | 0,051505519 | 0,1173644 | 0,03523212 |

**7. Вывод**

В ходе выполнения работы изучил методы работы алгоритма пирамидальной сортировки. Алгоритм пирамидальной сортировки имеет не самый быстро действенный алгоритм, потому что исходя из таблицы 1, можно сделать вывод, что самый быстрый алгоритм сортировки – это алгоритм сортировки «слиянием». Данные о времени работы алгоритмов Сортировки «вставкой» и сортировки «слиянием» получены из предыдущих лабораторных работ.