

**МИНИСТЕРСТВО ТРАНСПОРТА РОССИЙСКОЙ ФЕДЕРАЦИИ**

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«РОССИЙСКИЙ УНИВЕРСИТЕТ ТРАНСПОРТА»**

**(РУТ (МИИТ)**

|  |
| --- |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

Кафедра «Управление и защита информации»

**Отчет по заданию**

«Пирамидальная сортировка»

**по дисциплине**

«Методы программирования»

**Выполнил:** студент

группы ТКИ-311

Станчук П.Н.

**Проверил:** доценты кафедры УиЗИ, к.т.н. Логинова Л. Н. и

Сафронов А.И.

**Москва 2022 г.**

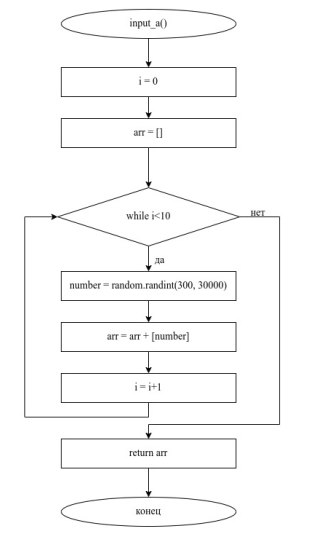
**1. Цель работы**

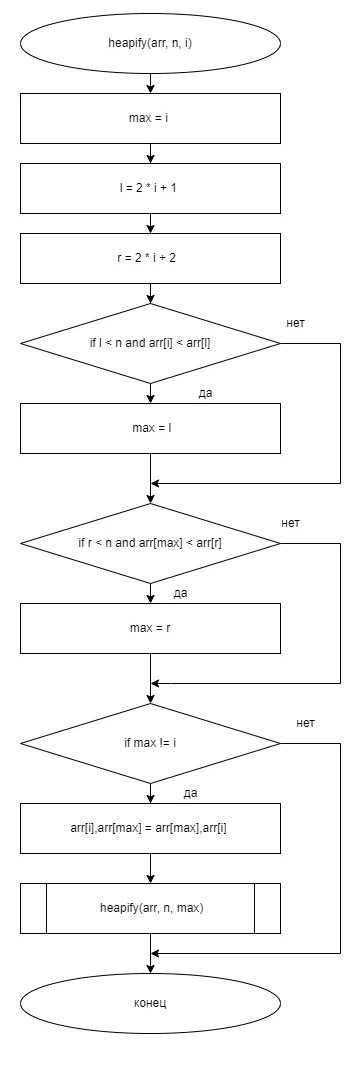
Изучение алгоритма пирамидальной сортировки.

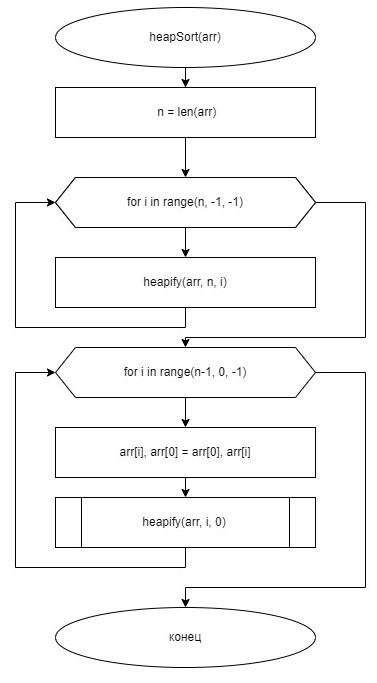
**2. Формулировка задачи**

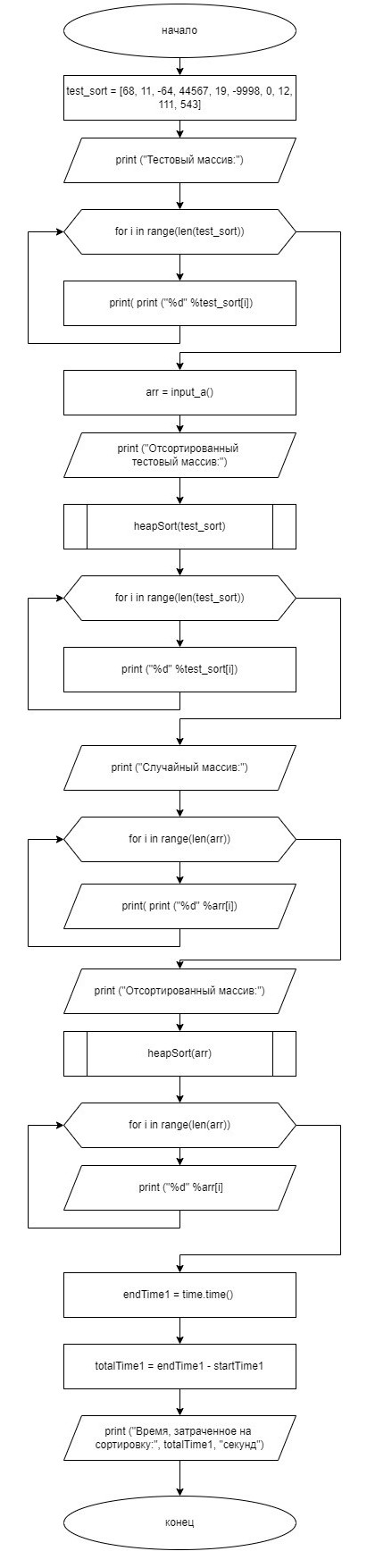
Реализовать программу, выполняющую алгоритм пирамидальной сортировки.

**3. Блок-схема алгоритма**

****





****



**4. Подбор тестовых примеров**

Ввод: массив из 10 элементов: 68, 11, -64, 44567, 19, -9998, 0, 12, 111, 543, -9.

Вывод: -9998, -64, 0, 11, 12, 19, 68, 111, 543, 44567.

**5. Код программы:**

import time

startTime1 = time.time()

def heapify(arr, n, i):

max = i

l = 2 \* i + 1

r = 2 \* i + 2

if l < n and arr[i] < arr[l]:

max = l

if r < n and arr[max] < arr[r]:

max = r

if max != i:

arr[i],arr[max] = arr[max],arr[i]

heapify(arr, n, max)

def heapSort(arr):

n = len(arr)

for i in range(n, -1, -1):

heapify(arr, n, i)

for i in range(n-1, 0, -1):

arr[i], arr[0] = arr[0], arr[i]

heapify(arr, i, 0)

def input\_a():

import random

i = 0

arr = []

while i<10:

number = random.randint(300, 30000)

arr = arr + [number]

i = i+1

return arr

test\_sort = [68, 11, -64, 44567, 19, -9998, 0, 12, 111, 543]

print ("Тестовый массив:")

for i in range(len(test\_sort)):

print(test\_sort[i])

arr = input\_a()

print ("Отсортированный тестовый массив:")

heapSort(test\_sort)

for i in range(len(test\_sort)):

print ("%d" %test\_sort[i])

print ("Случайный массив:")

for i in range(len(arr)):

print(arr[i])

print ("Отсортированный массив:")

heapSort(arr)

for i in range(len(arr)):

print ("%d" %arr[i])

print(heapSort(arr))

endTime1 = time.time() #время конца замера

totalTime1 = endTime1 - startTime1 #вычисляем затраченное время

print ("Время, затраченное на сортировку:", totalTime1, "секунд")**6. Результат расчёта тестовых примеров на ПК**

Проверим работоспособность алгоритма

Изображение выглядит как текст

Автоматически созданное описание

Тестовый массив отсортирован верно.

Сравним время работы алгоритма. Для этого закомментируем обработку тестового массива.

Изображение выглядит как текст

Автоматически созданное описание

На сортировку массива из 10 элементов затрачено 0.0011925697326660156

секунд.

На сортировку массива из 100 элементов затрачено 0.001949310302734375

секунд, то есть при увеличении количества элементов в 10 раз время увеличилось приблизительно в 1,6 раз.

**7. Вывод**

В данной работе я реализовала алгоритм пирамидальной сортировки на Python и протестировала его, а также узнала, что при увеличении количества элементов массива в 10 раз время сортировки слиянием увеличилось приблизительно в 1,6 раз.