САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ
ФАКУЛЬТЕТ ИНФОКОММУНИКАЦИОННЫХ ТЕХНОЛОГИЙ

Отчет по лабораторной работе №1
по курсу «Алгоритмы и структуры данных»
Тема: Сортировка вставками, выбором, пузырьковая

Выполнил:
Авдиенко Данила Андреевич
Группа К3140

Проверил:
Афанасьев А. В.

Санкт-Петербург
2024 г.

Содержание отчета

Оглавление

Оглавление

Задачи по варианту

Задание № 1. Сортировка вставкой

Текст задачи.

Используя код процедуры Insertion-sort, напишите программу и проверьте сортировку массива $A = \{31, 41, 59, 26, 41, 58\}$.

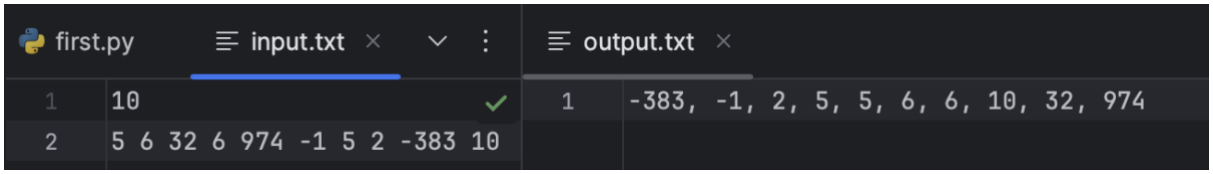
Код:

```
import sys
import time
import resource

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


if __name__ == "__main__":
    time_start = time.perf_counter()
    with open("input.txt", "r") as inp:
        list_length = int(inp.readline())
        arr = list(map(int, inp.readline().split()))
        inp.close()
    answ = ', '.join(str(i) for i in insertion_sort(list_length, arr))
    with open("output.txt", "w+") as out:
        out.write(answ)
        out.close()
    time_elapsed = (time.perf_counter() - time_start)
    mmry = resource.getrusage(resource.RUSAGE_SELF).ru_maxrss / 1024.0 /
1024.0
    print("Время:", time_elapsed)
    print("Память:%5.1f МБ" % (mmry))
```

Текстовое объяснение функции insertion_sort.

При помощи цикла for программа проходит весь массив, начиная с первого индекса, заканчивая последним. На каждой итерации берем новый элемент и вставляем его в нужное место в уже отсортированной части

Примеры работы кода:



Тесты:

```
import unittest
from ..task1.first import insertion_sort
```

```
class TestFirst(unittest.TestCase):

    def test_insertion_sort0(self):
        arr = [234, 325, 5, 1, -6, 0]
        list_length = len(arr)
        result = insertion_sort(list_length, arr)
        self.assertEqual(result, [-6, 0, 1, 5, 234, 325])

    def test_insertion_sort1(self):
        arr = [-1, -5, 5, 5, 3, -3]
        list_length = len(arr)
        result = insertion_sort(list_length, arr)
        self.assertEqual(result, [-5, -3, -1, 3, 5, 5])
```

Вывод: для решения задачи была реализована функция insertion_sort, функция была протестирована при помощи unittest.

Задание №3. Обратная сортировка вставкой

Текст задачи.

Перепишите процедуру Insertion-sort для сортировки в невозрастающем порядке вместо неубывающего с использованием процедуры Swap.

Используя код процедуры Insertion-sort, напишите программу и проверьте сортировку массива $A = \{31, 41, 59, 26, 41, 58\}$.

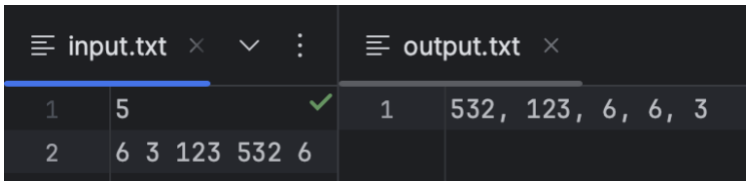
Код:

```
import sys
import time
import resource


def reversed_insertion_sort(list_length, arr):
    for i in range(1, list_length):
        key = arr[i]
        j = i - 1
        while j >= 0 and key > arr[j]:
            arr[j + 1] = arr[j]
            j -= 1
        arr[j + 1] = key
    return arr


if __name__ == "__main__":
    time_start = time.perf_counter()
    with open("input.txt", "r") as inp:
        list_length = int(inp.readline())
        arr = list(map(int, inp.readline().split()))
        inp.close()
    answ = ', '.join(str(i) for i in reversed_insertion_sort(list_length,
arr))
    with open("output.txt", "w+") as out:
        out.write(answ)
        out.close()
    time_elapsed = (time.perf_counter() - time_start)
    mmry = resource.getrusage(resource.RUSAGE_SELF).ru_maxrss / 1024.0 /
1024.0
    print("Время:", time_elapsed)
    print("Память:%5.1f МБ" % (mmry))
```

Текстовое объяснение функции insertion_sort.

При помощи цикла for программа проходит весь массив, начиная с первого индекса, заканчивая последним. На каждой итерации берем новый элемент и вставляем его в нужное место в уже отсортированной части

Примеры работы кода:



Тесты:

```python
import unittest
from ..task3.third import reversed_insertion_sort


class TestTask3(unittest.TestCase):

    def test_reverse_insertion_sort0(self):
        arr = [4, 1, 24, 512, 5, 3]
        list_length = len(arr)
        result = reversed_insertion_sort(list_length, arr)
        self.assertEquals(result, [512, 24, 5, 4, 3, 1])

    def test_reverse_insertion_sort1(self):
        arr = [-5, -10, -12034, -431, -4]
        list_length = len(arr)
        result = reversed_insertion_sort(list_length, arr)
        self.assertEquals(result, [-4, -5, -10, -431, -12034])
```

Вывод: для решения задачи была реализована функция reverse_insertion_sort, функция была протестирована при помощи unittest.

Задание №4. Линейный поиск

Текст задачи:

- Напишите код линейного поиска, при работе которого выполняется сканирование последовательности в поисках значения V.
- Если число встречается несколько раз, то выведите, сколько раз встречается число и все индексы i через запятую.

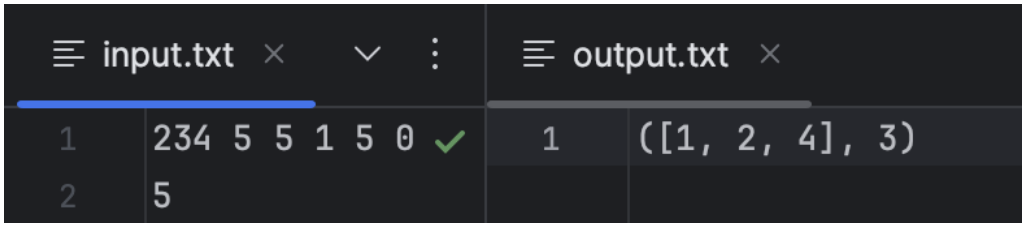
Код:

```
import sys
import time
import resource


def linear_search(v: int, arr: list):
    cnt = 0
    idx = []
    for i, char in enumerate(arr):
        if char == v:
            cnt += 1
            idx.append(i)
    if cnt == 1:
        return idx[0]
    elif cnt == 0:
        return -1
    else:
        return idx, cnt


if __name__ == "__main__":
    time_start = time.perf_counter()
    with open("input.txt", "r") as inp:
        arr = list(map(int, inp.readline().split()))
        v = int(inp.readline())
        inp.close()

    with open("output.txt", "w") as out:
        answ = str(linear_search(v, arr))
        out.write(answ)
    time_elapsed = (time.perf_counter() - time_start)
    mmry = resource.getrusage(resource.RUSAGE_SELF).ru_maxrss / 1024.0 /
1024.0
    print("Время:", time_elapsed)
    print("Память:%5.1f МБ" % (mmry))
```

Текстовое объяснение решения.

Для выполнения задания была реализована функция linear_search, обработаны случаи, когда в списке только один подходящий элемент, несколько или такового нет вовсе. При помощи цикла for, каждый элемент списка сравнивается с искомым и в случае нахождения заменяется на другой, чтобы обрабатывать индексы при нескольких подходящих элементах.

Пример работы кода:



Тесты:

```
import unittest
from ..task4.four import linear_search


class TestFirst(unittest.TestCase):

    def test_linear_search0(self):
        arr = [234, 325, 5, 1, -6, 0]
        v = 5
        result = linear_search(v, arr)
        self.assertEqual(result, 2)

    def test_linear_search1(self):
        arr = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16]
        v = 7
        result = linear_search(v, arr)
        self.assertEqual(result, 6)

    def test_linear_search(self):
        arr = [1, 2, 3, 4, 5, 6, 7, 7, 7, 10, 11, 12, 13, 14, 15, 16]
        v = 7
        result = linear_search(v, arr)
        self.assertEqual(result, ([6, 7, 8], 3))
```

Вывод по задаче:

Для решения задачи была реализована функция linear_search, функция была протестирована при помощи unittest.

Задание №6. Пузырьковая сортировка

Текст задачи.

Напишите код на Python и докажите корректность пузырьковой сортировки. Для доказательства корректоности процедуры вам необходимо доказать, что она завершается и что $A'[1] \leq A'[2] \leq ... \leq A'[n]$, где A' - выход процедуры Bubble_Sort, а n - длина массива A.

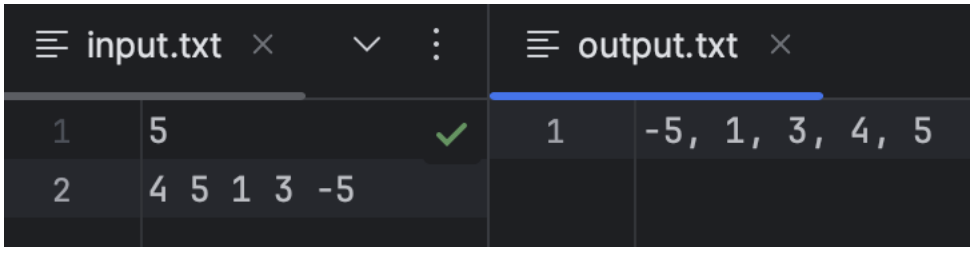
Код:

```
import sys
import time
import resource


def bubble_sort(list_length, arr):
    for i in range(list_length):
        for j in range(list_length - 1):
            if arr[j] > arr[j + 1]:
                arr[j], arr[j + 1] = arr[j + 1], arr[j]
    return arr


if __name__ == "__main__":
    time_start = time.perf_counter()
    with open("input.txt", "r") as inp:
        list_length = int(inp.readline())
        arr = list(map(int, inp.readline().split()))
        inp.close()
    answ = ', '.join(str(i) for i in bubble_sort(list_length, arr))
    with open("output.txt", "w+") as out:
        out.write(answ)
        out.close()
    time_elapsed = (time.perf_counter() - time_start)
    mmry = resource.getrusage(resource.RUSAGE_SELF).ru_maxrss / 1024.0 /
1024.0
    print("Время:", time_elapsed)
    print("Память:%5.1f МБ" % (mmry))
```

Текстовое объяснение решения.

Для выполнения задания была реализована функция bubble_sort, в ней, при помощи двух циклов сравниваются соседние элементы в сортируемом массиве и выстраиваются в нужном порядке.

Пример работы кода:



Тесты:

```
import unittest
from ..taks6.six import bubble_sort


class TestFirst(unittest.TestCase):

    def test_bubble_sort0(self):
```

```
        arr = [234, 325, 5, 1, -6, 0]
        list_length = len(arr)
        result = bubble_sort(list_length, arr)
        self.assertEqual(result, [-6, 0, 1, 5, 234, 325])

    def test_bubble_sort1(self):
        arr = [-1, -5, 5, 5, 3, -3]
        list_length = len(arr)
        result = bubble_sort(list_length, arr)
        self.assertEqual(result, [-5, -3, -1, 3, 5, 5])
```

Вывод по задаче:

Для решения задачи была реализована функция bubble_sort, функция была протестирована при помощи unittest, доказательство корректности прослеживается в тестах кода.

Задание №8. Секретарь Своп.

Текст задачи:

Дан массив, состоящий из n целых чисел. Вам необходимо его отсортировать по неубыванию. Но делать это нужно так же, как это делает мистер Своп — то есть, каждое действие должно быть взаимной перестановкой пары элементов. Вам также придется записать все, что Вы делали, в файл, чтобы мистер Своп смог проверить Вашу работу.

Код:

```
import sys
import time
import resource


def secretar_swap(list_length: int, arr: list, out):
    for j in range(list_length):
        for i in range(list_length - 1):
            if arr[i] > arr[i + 1]:
                arr[i], arr[i + 1] = arr[i + 1], arr[i]
                out.write(f"Swap elements at indices {i + 1} and {i +
2}.\n")
    out.write("No more swaps needed.\n")
    return arr


if __name__ == "__main__":
    time_start = time.perf_counter()
    with open("input.txt", "r") as inp:
        list_length = int(inp.readline())
        arr = list(map(int, inp.readline().split()))
        inp.close()

    with open("output.txt", "w") as out:
        answ = ', '.join(str(i) for i in secretar_swap(list_length, arr,
out))
        out.write(answ)
    time_elapsed = (time.perf_counter() - time_start)
    mmry = resource.getrusage(resource.RUSAGE_SELF).ru_maxrss / 1024.0 /
1024.0
    print("Время:", time_elapsed)
    print("Память:%5.1f МБ" % (mmry))
```

Текстовое объяснение решения.

Для выполнения задания была реализована функция secretar_swap, которая при помощи двух циклов проходится по массиву и сортирует элементы взаимными перестановками, записывая свои шаги в выходной файл. Тестов для этого кода нет.

Пример работы кода:

input.txt

```
5
3 4 4 1 7
```

output.txt

```
Swap elements at indices 3 and 4.
Swap elements at indices 2 and 3.
Swap elements at indices 1 and 2.
No more swaps needed.
1, 3, 4, 4, 7
```

Вывод:

В лабораторной работе я использовал алгоритмы сортировки вставкой, пузырьковой сортировки и сортировки выбором, код каждой функции, кроме 8-ой был протестирован при помощи unittest.