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
Вариант 7

Выполнил:
Барецкий. М. С.
К3141

Проверил:
Афанасьев А. В.

Санкт-Петербург
2024 г.

Содержание отчета

Задачи по варианту

Задача №1. Сортировка вставкой

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

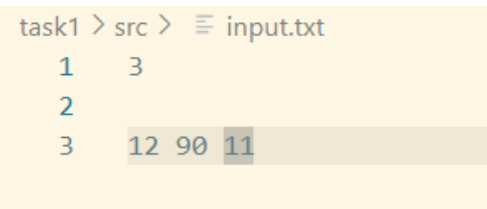
- **Формат входного файла (input.txt).** В первой строке входного файла содержится число n ($1 \leq n \leq 10^3$) — число элементов в массиве. Во второй строке находятся n различных целых чисел, по модулю не превосходящих 10^9.
- **Формат выходного файла (output.txt).** Одна строка выходного файла с отсортированным массивом. Между любыми двумя числами должен стоять ровно один пробел.
- Ограничение по времени. 2сек.
- Ограничение по памяти. 256 мб.

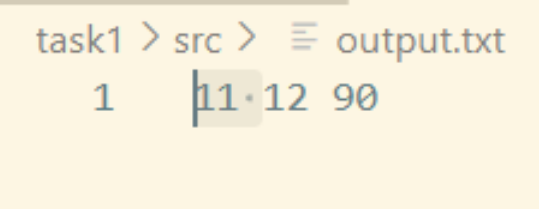
Выберите любой набор данных, подходящих по формату, и протестируйте алгоритм.

```
import time
import tracemalloc

tracemalloc.start()
t_start = time.perf_counter()

input = open('task1/src/input.txt')
n = int(input.readline().split()[0])
nums = list(map(int, input.readlines()[1].split()))
new_nums = [num for num in nums if abs(num) <= abs(109)][:n]

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

    return arr
open('task1/src/soutput.txt',        'w').write('        '.join(map(str,
insertion_sort(new_nums))))
```

```
print("Time: %s second " % (time.perf_counter() - t_start))
print("Memory used:", tracemalloc.get_traced_memory()[1] / (1024 ** 2),
"MB" )
tracemalloc.stop()
```

1. Открываем файл input.txt и заносим первую строку в переменную
2. Затем в nums записываем вторую строку, а в new_nums создаем список, ограничивая его n
3. В функции insertion_sort реализовал сортировку вставками
4. Выводим в output.txt результат работы функции
5. Выводим в консоль затраты по памяти и времени





	Время выполнения	Затраты памяти
Пример из задачи	0.0007108339996193536 секунд	0.013239860534667969 МБ

```
Time: 0.0016918999999688822 second
Memory used: 0.01851177215576172 MB
```

Задача №3. Сортировка вставкой по убыванию

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

Формат входного и выходного файла и ограничения - как в задаче 1.

Подумайте, можно ли переписать алгоритм сортировки вставкой с использованием рекурсии?

```
import time
import tracemalloc

tracemalloc.start()
t_start = time.perf_counter()
```

```
input = open('task3/src/input.txt')
n = int(input.readline().split()[0])
nums = list(map(int, input.readlines()[1].split()))
new_nums = [num for num in nums if abs(num) <= abs(109)][:n]


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

open('task3/src/output.txt',         'w').write('         '.join(map(str,
insertion_sort(new_nums))))

print("Time: %s second " % (time.perf_counter() - t_start))
print("Memory used:", tracemalloc.get_traced_memory()[1] / (1024 ** 2),
"MB" )
tracemalloc.stop()
```

1. Аналогично с обычным, но реализовал swap, который меняет местами

	Время выполнения	Затраты памяти
Пример из задачи	0.000710833999619353 6 секунд	0.013239860534667969 МБ

```
Time: 0.0011244999996051774 second
Memory used: 0.018564224243164062 MB
```

Задача №4. Линейный поиск

Рассмотрим задачу поиска.

- **Формат входного файла**. Последовательность из n чисел $A = a_1, a_2, \ldots, a_n$ в первой строке, числа разделены пробелом, и значение V во второй строке. Ограничения: $0 \leq n \leq 10^3$, $-10^3 \leq a_i, V \leq 10^3$
- **Формат выходного файла**. Одно число - индекс i, такой, что $V = A[i]$, или значение -1, если V в отсутствует.
- 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.
- Дополнительно: попробуйте найти свинью, как в лекции. Используйте во входном файле последовательность слов из лекции, и найдите соответствующий индекс.

```
import time
import tracemalloc

tracemalloc.start()
t_start = time.perf_counter()

input = open('task4/input.txt')
V = int(input.readlines()[1])
numbers = list(map(int,open('task4/input.txt').readline().split()))
if -10**3<=V<=10**3 and 0 <= numbers[-1] <=10*3:
    def linear_search(arr, target):
        count = 0
        indexes = []
        for i, e in enumerate(arr):
            if e == target:
                count+=1
                indexes.append(i)
        return indexes, count
     output_indexes = ' '.join(str(i) for i in linear_search(numbers,
V)[0])
    output_count = str(linear_search(numbers, V)[1])
      open('task4/src/output.txt', 'w').write(f'The number {V} occurs
{output_count} on the indexes {output_indexes} in list
{open('task4/input.txt').readline()}')
else:
```

```
    print('write other numbers')

print("Time: %s second " % (time.perf_counter() - t_start))
print("Memory used:", tracemalloc.get_traced_memory()[1] / (1024 ** 2),
"MB" )
tracemalloc.stop()
```

1. Открываем файл input.txt и заносим первую строку в переменную
2. Затем в v записываем строку, которая содержит кол-во чисел, а numbers сами числа
3. В функции linear_search реализовал линейный поиск
4. Выводим в output.txt результат работы функции
5. Выводим в консоль затраты по памяти и времени

	Время выполнения	Затраты памяти
Пример из задачи	0.0.00115720000030705 7 секунд	0.03540515899658203 МБ

```
y
Time: 0.00115720000030705 7 second
Memory used: 0.03540515899658203 MB
```

Дополнительные задачи

Задача №5 . Сортировка выбором.

Рассмотрим сортировку элементов массива , которая выполняется следующим образом. Сначала определяется наименьший элемент массива , который ставится на место элемента $A[1]$. Затем производится поиск второго наименьшего элемента массива A, который ставится на место элемента $A[2]$. Этот процесс продолжается для первых $n - 1$ элементов массива A.

Напишите код этого алгоритма, также известного как сортировка выбором (selection sort). Определите время сортировки выбором в наихудшем случае и в среднем случае и сравните его со временем сортировки вставкой.

Формат входного и выходного файла и ограничения - как в задаче 1.

```
import time
import tracemalloc

tracemalloc.start()
t_start = time.perf_counter()

input = open('task5/input.txt')
```

```
n = int(input.readline().split()[0])
nums = list(map(int, input.readlines()[1].split()))
new_nums = [num for num in nums if abs(num) <= abs(109)][:n]

def selection_sort(arr):
    for i in range(len(arr)):
        min_id = i
        for j in range(i+1, len(arr)):
            if arr[j] < arr[min_id]:
                min_id = j
        arr[i], arr[min_id] = arr[min_id], arr[i]
    return arr

open('task5/src/output.txt',         'w').write('         '.join(map(str,
selection_sort(new_nums))))

print("Time: %s second " % (time.perf_counter() - t_start))
print("Memory used:", tracemalloc.get_traced_memory()[1] / (1024 ** 2),
"MB" )
tracemalloc.stop()
```

1. Открываем файл input.txt и заносим первую строку в переменную
2. Затем в nums записываем вторую строку, а в new_nums создаем список, ограничивая его n
3. В функции selection_sort реализовал сортировку выбором
4. Выводим в output.txt результат работы функции
5. Выводим в консоль затраты по памяти и времени

```
Time: 0.0006751999999323743 second
Memory used: 0.018556594848632812 MB
```

	Время выполнения	Затраты памяти
Пример из задачи	0.0006751999999323743 секунд	0.018556594848632812 МБ

Задача №6. Пузырьковая сортировка

Пузырьковая сортировка представляет собой популярный, но не очень эффективный алгоритм сортировки. В его основе лежит многократная перестановка соседних элементов, нарушающих порядок сортировки. Вот псевдокод этой сортировки:

```
Bubble_Sort(A):
  for i = 1 to A.length - 1
    for j = A.length downto i+1
      if A[j] < A[j-1]
        поменять A[j] и A[j-1] местами
```

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.

Определите время пузырьковой сортировки в наихудшем случае и в среднем случае и сравните его со временем сортировки вставкой.

Формат входного и выходного файла и ограничения - как в задаче 1.

```python
import time
import tracemalloc

tracemalloc.start()
t_start = time.perf_counter()

input = open('task6/src/input.txt')
n = int(input.readline().split()[0])
nums = list(map(int, input.readlines()[1].split()))
new_nums = [num for num in nums if abs(num) <= abs(109)][:n]

def bubble_sort(arr):
    for i in range(len(arr)):
        swapped = False
        for j in range(0, n-i-1):
            if arr[j] > arr[j+1]:
                arr[j], arr[j+1] = arr[j+1], arr[j]
                swapped = True
        if not swapped:
            break
    return arr

open('task6/src/output.txt',        'w').write('        '.join(map(str,
bubble_sort(new_nums))))
```

```
print("Time: %s second " % (time.perf_counter() - t_start))
print("Memory used:", tracemalloc.get_traced_memory()[1] / (1024 ** 2),
"MB" )
tracemalloc.stop()
```

1. Открываем файл input.txt и заносим первую строку в переменную
2. Затем в nums записываем вторую строку, а в new_nums создаем список, ограничивая его n
3. В функции bubble_sort реализовал сортировку пузырьком
4. Выводим в output.txt результат работы функции
5. Выводим в консоль затраты по памяти и времени

	Время выполнения	Затраты памяти
Пример из задачи	0.0008049999996728729 секунд	0.018556594848632812 МБ

```
Time: 0.0008049999996728729 second
Memory used: 0.018556594848632812 MB
```

Задача №9. Сложение двоичный чисел

Расмотрим задачу сложения двух n-битовых двоичных целых чисел, хранящихся в n-элементных массивах A и B. Сумму этих двух чисел необходимо занести в двоичной форме в $(n+1)$-элементный массив C. Напишите скрипт для сложения этих двух чисел.

- **Формат входного файла (input.txt).** В одной строке содержится два n-битовых двоичных числа, записанные через пробел ($1 \leq n \leq 10^3$)
- **Формат выходного файла (output.txt).** Одна строка - двоичное число, которое является суммой двух чисел из входного файла.
- Оцените асимптотическое время выполнение вашего алгоритма.

```
input = open('task9/src/input.txt').readline().strip().split()
first_number_from_list = list(map(int, input[0]))
second_number_from_list = list(map(int, input[1]))

def add_two_binary(first_number, second_number):
    n = max(len(first_number), len(second_number))

    first_number = [0] * (n - len(first_number)) + first_number
```

```
    second_number = [0] * (n - len(second_number)) + second_number

    result = [0] * (n+1)
    carry = 0
    for i in range(n-1, -1, -1):
        temp_sum = first_number[i] + second_number[i] + carry
        result[i+1] = temp_sum % 2
        carry = temp_sum // 2
    result[0] = carry
    return result

open('task9/src/output.txt',                'w').write(''.join(map(str,
add_two_binary(first_number_from_list,
second_number_from_list))).lstrip('0') or '0')

print("Time: %s second " % (time.perf_counter() - t_start))
print("Memory used:", tracemalloc.get_traced_memory()[1] / (1024 ** 2),
"MB" )
tracemalloc.stop()
```

	Время выполнения	Затраты памяти
Пример из задачи	0.0011068000003433554 секунд	0.017080307006835938 МБ

```
Time: 0.0011068000003433554 second
Memory used: 0.017080307006835938 MB
```

Вывод

В результате выполнения данной лабораторной работы было изучен алгоритм сортировки вставкой, пузырьковой сортировки, линейного поиска и сложения двоичных чисел