

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 14

Выполнил:
Савченко В. А.
К3139

Проверил:
Афанасьев А. В.

Санкт-Петербург
2024 г.

Содержание отчета

Содержание отчета	2
Задачи по варианту	3
Задача №1. Сортировка вставкой	3
Задача №3. Сортировка вставкой по убыванию	5
Задача №4. Линейный поиск	7
Задача №5. Сортировка выбором	9
Задача №6. Сортировка выбором	11
Вывод	13

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.

Код:

```
import time
t_start = time.perf_counter()

def insertion_sort(arr):
    n = len(arr)
    for i in range(1, n):
        key = arr[i]
        j = i - 1

        while j >= 0 and arr[j] > key:
            arr[j + 1] = arr[j]
            j -= 1
        arr[j + 1] = key

with open('input.txt', 'r') as file:
    n = int(file.readline().strip())
    arr = list(map(int, file.readline().strip().split()))

insertion_sort(arr)

with open('output.txt', 'w') as file:
    file.write(' '.join(map(str, arr)))

print(time.perf_counter() - t_start, 'секунд')
```

Текстовое объяснение решения:

В функции `insertion_sort()` цикл проходит по массиву и для каждого элемента (начиная со второго) находит правильную позицию для него в отсортированной части массива

Открываем файлы чтобы считать и записать в `output.txt` результат

Выводим время работы с помощью модуля `time`

Результат работы кода на примерах из текста задачи:

input.txt		output.txt	
1	6	1	26 31 41 41 58 59
2	31 41 59 26 41 58		

Вывод по задаче:

Для решения задачи мы работаем с массивами и циклами

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.

Код:

```
import time
t_start = time.perf_counter()

def swap(arr, i, j):
    arr[i], arr[j] = arr[j], arr[i]

def insertion_sort(arr):
    n = len(arr)
    for i in range(1, n):
        key = arr[i]
        j = i - 1

        while j >= 0 and arr[j] < key:
            swap(arr, j, j + 1)
            j -= 1
        arr[j + 1] = key

with open('input.txt', 'r') as file:
    n = int(file.readline().strip())
    arr = list(map(int, file.readline().strip().split()))

insertion_sort(arr)

with open('output.txt', 'w') as file:
    file.write(' '.join(map(str, arr)))

print(time.perf_counter() - t_start, 'секунд')
```

Текстовое объяснение решения:

В функции `insertion_sort()` цикл проходит по массиву и для каждого элемента (начиная со второго) находит правильную позицию для него в отсортированной части массива. В отличии от первой задачи, сортировка проходит по убыванию и используется функция `swap`

Открываем файлы чтобы считать и записать в output.txt результат

Выводим время работы с помощью модуля time

Результат работы кода на примерах из текста задачи:

input.txt		output.txt	
1	6	1	59 58 41 41 31 26
2	31 41 59 26 41 58		

Вывод по задаче:

Для решения задачи мы работаем с массивами и циклами

Задача №4. Линейный поиск

Рассмотрим задачу поиска. • Формат входного файла. Последовательность из n чисел $A = a_1, a_2, \dots, a_n$ 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.

Код:

```
import time
t_start = time.perf_counter()

def linear_search(arr, v):
    for i in range(len(arr)):
        if arr[i] == v:
            return i
    return -1

with open('input.txt', 'r') as file:
    arr = file.readline().strip().split()
    v = file.readline().strip()

    if arr[0].isdigit():
        arr = list(map(int, arr))
        v = int(v)

result = linear_search(arr, v)

with open('output.txt', 'w') as file:
    file.write(str(result))

print(time.perf_counter() - t_start, 'секунд')
```

Текстовое объяснение решения:

Функция `linear_search()` выполняет линейный поиск по массиву `arr`, сравнивая каждый элемент с искомым значением `v`.

Открываем файлы чтобы считать и записать в `output.txt` результат

Выводим время работы с помощью модуля `time`

Результат работы кода на примерах из текста задачи:

<div>≡ input.txt ×</div> <table><tr><td>1</td><td>1 5 3 -2 8 0</td></tr><tr><td>2</td><td>5</td></tr></table>	1	1 5 3 -2 8 0	2	5	<div>≡ output.txt ×</div> <table><tr><td>1</td><td>1</td></tr></table>	1	1
1	1 5 3 -2 8 0						
2	5						
1	1						
<div>≡ input.txt ×</div> <table><tr><td>1</td><td>apple banana orange</td></tr><tr><td>2</td><td>orange</td></tr></table>	1	apple banana orange	2	orange	<div>≡ output.txt ×</div> <table><tr><td>1</td><td>2</td></tr></table>	1	2
1	apple banana orange						
2	orange						
1	2						

Вывод по задаче:

Для решения задачи мы работаем с массивами и циклами

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.

Код:

```
import time
t_start = time.perf_counter()

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

with open('input.txt', 'r') as file:
    n = int(file.readline().strip())
    arr = list(map(int, file.readline().strip().split()))

selection_sort(arr)

with open('output.txt', 'w') as file:
    file.write(' '.join(map(str, arr)))

print(time.perf_counter() - t_start, 'секунд')
```

Текстовое объяснение решения:

Функция находит наименьший элемент из неотсортированной части массива и перемещая его в начало. Этот процесс повторяется для оставшейся части массива до тех пор, пока весь массив не будет отсортирован.

Открываем файлы чтобы считать и записать в output.txt результат

Выводим время работы с помощью модуля time

Результат работы кода на примерах из текста задачи:

input.txt		output.txt	
1	6	1	26 31 41 41 58 59
2	31 41 59 26 41 58		

Вывод по задаче:

Для решения задачи мы работаем с массивами и циклами

Задача №6. Сортировка выбором

Напишите код на Python и докажите корректность пузырьковой сортировки. Для доказательства корректности процедуры вам необходимо доказать, что она завершается и что $A'[1] \leq A'[2] \leq \dots \leq A'[n]$, 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.

Код:

```
import time
t_start = time.perf_counter()

def bubble_sort(arr):
    n = len(arr)
    for i in range(n - 1):
        for j in range(n - 1, i, -1):
            if arr[j] < arr[j - 1]:
                arr[j], arr[j - 1] = arr[j - 1], arr[j]

with open('input.txt', 'r') as file:
    n = int(file.readline().strip())
    arr = list(map(int, file.readline().strip().split()))

bubble_sort(arr)

with open('output.txt', 'w') as file:
    file.write(' '.join(map(str, arr)))

print(time.perf_counter() - t_start, 'секунд')
```

Текстовое объяснение решения:

Функция проходит по массиву несколько раз, сравнивая и переставляя соседние элементы, если они находятся в неправильном порядке.

Открываем файлы чтобы считать и записать в output.txt результат

Выводим время работы с помощью модуля time

Результат работы кода на примерах из текста задачи:

input.txt		output.txt	
1	6	1	26 31 41 41 58 59
2	31 41 59 26 41 58		

Вывод по задаче:

Для решения задачи мы работаем с массивами и циклами

Вывод

В данной лабораторной работе я познакомился с различными методами сортировки и реализовал их. Для этого мне потребовалось работать с массивами и циклами