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.

K3139

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 п ($1 \le n \le 10^3$) — число элементов в массиве. Во второй строке находятся п 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]
     i = i - 1
     while j \ge 0 and arr[j] > key:
        arr[i + 1] = arr[i]
        i -= 1
     arr[i + 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:



Вывод по задаче:

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.

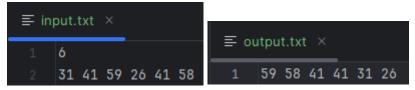
```
Код:
import time
t start = time.perf counter()
def swap(arr, i, j):
  arr[i], arr[j] = arr[j], arr[i]
def insertion_sort(arr):
  n = len(arr)
  for i in range(1, n):
     key = arr[i]
     i = i - 1
     while i \ge 0 and arr[i] < key:
        swap(arr, i, i + 1)
        i = 1
     arr[i + 1] = key
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:



Вывод по задаче:

Задача №4. Линейный поиск

Рассмотрим задачу поиска. • Формат входного файла. Последовательность изп чисел $A=a1,\,a2,\,\ldots$, ап в первой строке, числа разделены пробелом, и значение V во второй строке. Ограничения: $0 \le n \le 10^3 \,,\, -10^3 \le a_i \,,\, V \le 10^3 \, \bullet$ 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 і через запятую.

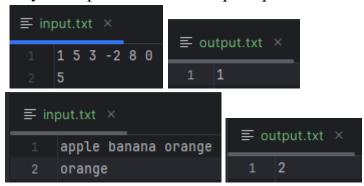
```
Код:
import time
t_start = time.perf_counter()
def linear_search(arr, v):
  for i in range(len(arr)):
     if arr[i] == v:
        return i
  return -1
with open('input.txt', 'r') as file:
  arr = file.readline().strip().split()
  v = file.readline().strip()
  if arr[0].isdigit():
     arr = list(map(int, 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:



Вывод по задаче:

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[i] < arr[min_index]:
           min index = i
     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:



Вывод по задаче:

Задача №6. Сортировка выбором

Напишите код на Python и докажите корректность пузырьковой сортировки. Для доказательства корректоности процедуры вам необходимо доказать, что она завершается и что $A'[1] \le A'[2] \le ... \le A'[n]$, 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')</pre>
```

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:



Вывод по задаче:

Вывод

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