Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский институт ИТМО»

Факультет МР и П

Алгоритмы и структуры данных

Лабораторная работа №1.

«Сортировка вставками, выбором, пузырьковая.»

Вариант «9»

Выполнил студент:

Розметов Джалолиддин

Группа № D3210

Преподаватель: Артамонова Валерия Евгеньевна

г. Санкт-Петербург

2024

Оглавление

Задание 1

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 ($1 \le n \le 10^3$) — 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.

Код

```
def insertion_sort(arr):
    for i in range(1, 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

with open('input.txt', 'r') as file:
    n = int(file.readline().strip())
    if n < 1 or n > 10 ** 3:
        raise ValueError("n должно быть в диапазоне от 1 до 10^3.")
    array = list(map(int, file.readline().split()))

    if len(array) != n:
        raise ValueError("Количество элементов массива не соответствует
значению n.")
    if any(abs(x) > 10 ** 9 for x in array):  # Проверяем ограничения для
абсолютного значения элементов массива
```

```
        raise ValueError("Абсолютное значение элементов массива не должно
превышать 10^9.")
insertion_sort(array)

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

Вводные данные:
6
9 5 3 3 1 34
Вывод кода:
1 3 3 5 9 34
Описание кода:

1. Функция insertion_sort(arr): Реализует алгоритм сортировки вставками для сортировки массива.
2. Чтение данных из input.txt:

- Считывает значение n (количество элементов массива).
- Проверяет, что n находится в диапазоне от 1 до 1000.
- Считывает массив чисел и проверяет, что количество элементов соответствует n.
- Проверяет, что все элементы массива находятся в диапазоне от -10^9 до 10^9.

3. Сортировка массива: Сортирует массив с помощью функции insertion_sort.
4. Запись результата в output.txt: Записывает отсортированный массив в файл.

Описание проведенных тестов:
Тестирование включало проверку правильности считывания и сортировки массива, а также обработку ошибок, связанных с пустым файлом, некорректными значениями n, несоответствием длины массива, элементами вне допустимого диапазона и работой сортировки для массивов различных размеров.

Вывод по работе кода:
Код считывает массив чисел из файла, проверяет корректность входных данных, сортирует массив с использованием алгоритма вставок и записывает результат в файл, обеспечивая обработку ошибок, связанных с некорректными значениями и форматом данных.

Задание 2

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

• Формат входного файла (input.txt). В первой строке входного файла содержится число n ($1 \le n \le 10^3$) — 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.

Код

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

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

with open('input.txt', 'r') as file:
    n = int(file.readline().strip())
    if n < 1 or n > 10**3:
        raise ValueError("n должно быть в диапазоне от 1 до 10^3.")
    array = list(map(int, file.readline().split()))

    if len(array) != n:
        raise ValueError("Количество элементов массива не соответствует
значению n.")
    if any(abs(x) > 10**9 for x in array):
        raise ValueError("Абсолютное значение элементов массива не должно
превышать 10^9.")

insertion_sort_descending(array)

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

Вводные данные:

6

12 32 3 23 43 31

Вывод кода:

43 32 31 23 12 3

Описание кода:

1. Функция swap(arr, i, j): Обменивает местами два элемента массива.
2. Функция insertion_sort_descending(arr): Реализует алгоритм сортировки вставками для сортировки массива по убыванию.
3. Чтение данных из файла input.txt:

4. Сортировка массива: Сортирует массив с помощью функции insertion_sort_descending.
5. Запись результата в файл output.txt: Записывает отсортированный массив в файл.

Описание проведенных тестов:
Тестирование включало проверку правильности считывания и сортировки массива по убыванию, а также обработку ошибок, связанных с пустым файлом, некорректными значениями n, несоответствием длины массива и элементами вне допустимого диапазона. Были протестированы массивы различных размеров для проверки корректности работы сортировки.

Вывод по работе кода:
Код считывает массив чисел из файла, проверяет корректность входных данных, сортирует массив по убыванию с использованием алгоритма вставок и записывает результат в файл. Обрабатываются ошибки, связанные с некорректными значениями и форматом данных, обеспечивая надежную работу программы.

Задание 3

Напишите код на Python и докажите корректность пузырьковой сортировки. Для доказательства корректоности процедуры вам необходимо доказать, что она завершается и что A'[1] ≤ A'[2] ≤ ... ≤ 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 файла (input.txt). В первой строке входного файла содержится число n ($1 \le n \le 10^3$) — 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.

Код

```
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]

arr = [64, 34, 25, 12, 22, 11, 90]
bubble_sort(arr)
print("Отсортированный массив:", arr)
```

Доказательство завершаемости:
Каждый проход внешнего цикла (от i=1 до n-1) "выталкивает" наименьший элемент в правильное положение в конец массива.
После n-1 таких проходов все элементы будут расположены в правильном порядке.
Таким образом, алгоритм завершается.

Доказательство корректности:
На каждом шаге внутреннего цикла происходит сравнение соседних элементов и, если они находятся в неправильном порядке, они меняются местами.
Это означает, что после каждого прохода внутреннего цикла наибольший из оставшихся элементов будет перемещен в конец массива.
После n-1 проходов внешнего цикла массив будет отсортирован.

Сложность времени:
В наихудшем случае время работы пузырьковой сортировки составляет O(n^2), когда массив уже отсортирован в обратном порядке.
В среднем случае также O(n^2). Это связано с тем,
что независимо от начального порядка элементов, алгоритм всегда выполняет одинаковое количество сравнений и обменов.

Описание кода:
Данный код реализует алгоритм сортировки пузырьком для сортировки массива чисел. Он проходит по массиву несколько раз, сравнивая соседние элементы и меняя их местами, если они находятся в неправильном порядке. Этот процесс повторяется до тех пор, пока весь массив не будет отсортирован.

Вывод:
Код успешно реализует алгоритм сортировки пузырьком и правильно сортирует массив чисел. Доказательства завершаемости и корректности алгоритма обосновывают его правильность и корректность работы. Однако, следует отметить, что пузырьковая сортировка имеет квадратичную сложность времени, что делает ее неэффективной для больших массивов данных.

Вывод

Работа по сортировке вставками, выбором и пузырьком позволила изучить основные алгоритмы сортировки и их применение на практике. Каждый из алгоритмов имеет свои особенности и эффективность в зависимости от типа данных и размера массива. Изучение этих алгоритмов позволяет лучше понять принципы сортировки данных и их применение в реальных приложениях.