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 1

Выполнил:

Бен Шамех Абделазиз

К3239

Проверил:

Ромакина Оксана Михайловна

Санкт-Петербург

2025 г.

Содержание отчета

Оглавление

Задачи по варианту

1 задача. 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 ≤ n ≤ 10**3) — число элементов в массиве. Во второй строке находятся n различных целых чисел, по модулю не превосходящих

109

• 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.

Листинг кода

```
def task1():
    f = open('input.txt')
    f2 = open('output.txt', 'w')
    n = int(f.readline())
    arr = [int(i) for i in f.readline().split()]

    def ins_sort(arr):
        for j in range(1, len(arr)):
            i = j-1
            k = arr[j]
            while arr[i] > k and i >= 0:
                arr[i+1] = arr[i]
                i = i-1
            arr[i+1] = k
        return arr

    print(*ins_sort(arr), file=f2)

    f.close()
```

```
f2.close()
```

Текстовое объяснение решения.

Перебирая каждый элемент массива, начиная с 1 индекса, ищем для него место переставляя все “ниже”, пока не передвинем элемент на своем месте в часть сортированной последовательности.

Результат работы кода на примере из задачи:

Результат работы кода на максимальных и минимальных значениях:

Время выполнения

Затраты памяти

Нижняя граница диапазона значений входных данных из текста задачи

0.0065 sec

14.77 Mb

Пример из задачи

0.00407 sec

14.80 Mb

Верхняя граница диапазона значений входных данных из текста задачи

0.02734 sec

15.13 Mb

Вывод по задаче: Увеличение значений вводимых переменных в пределах ограничений линейно увеличивает время работы программы и используемую память.

2 задача. Сортировка вставкой +

Измените процедуру Insertion-sort для сортировки таким образом, чтобы в выходном файле отображалось в первой строке n чисел, которые обозначают новый индекс элемента массива после обработки.

• Формат выходного файла (input.txt).В первой строке выходного файла выведите n чисел. При этом i-ое число равно индексу, на который, в момент обработки его сортировкой вставками, был перемещен i-ый элемент исходного массива. Индексы нумеруются, начиная с единицы. Между любыми двумя числами должен стоять ровно один пробел.

Листинг кода

```
def task2():
    f = open('input.txt')
    f2 = open('output.txt', 'w')
    n = int(f.readline())
    arr = [int(i) for i in f.readline().split()]


    def ins_sort(arr):
        ind_arr = [-1] * n
        ind_arr[0] = 1
        for j in range(1, len(arr)):
            i = j-1
            k = arr[j]
            while arr[i] > k and i >= 0:
                arr[i+1] = arr[i]
                i = i-1
            arr[i+1] = k
            ind_arr[j] = i+2
        return ind_arr, arr
```

```
arr1, arr2 = ins_sort(arr)

print(*arr1, file=f2)
print(*arr2, file=f2)

f.close()
f2.close()
```

Текстовое объяснение решения.

Перебирая каждый элемент массива, начиная с 1 индекса, ищем для него место переставляя все “ниже”, пока не передвинем элемент на своем месте в часть сортированной последовательности. А когда поставили на место сохраняем индекс в доп. массив.

Результат работы кода на примере из задачи:

Результат работы кода на максимальных и минимальных значениях:

Время выполнения

Затраты памяти

Нижняя граница диапазона значений входных данных из текста задачи

0.00409 sec

14.95 Mb

Пример из задачи

0.00518 sec

14.91 Mb

Верхняя граница диапазона значений входных данных из текста задачи

0.02212 sec

15.30 Mb

Вывод по задаче: Увеличение значений вводимых переменных в пределах ограничений линейно увеличивает время работы программы и используемую память.

3 задача. 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

Листинг кода

```
def task3():
    f = open('input.txt')
    f2 = open('output.txt', 'w')
    n = int(f.readline())
    arr = [int(i) for i in f.readline().split()]

    def reverse_ins_sort(arr):
        for i in range(len(arr)-1):
            while arr[i] < arr[i+1] and i >= 0:
                arr[i+1], arr[i] = arr[i], arr[i+1]
                i -= 1
        return arr

    print(*reverse_ins_sort(arr), file=f2)

    f.close()
    f2.close()
```

Текстовое объяснение решения.

Выполняем операции аналогичные обычной сортировке, но вместо сдвига меняем числа местами, а также знак операции противоположный.

Результат работы кода на примере из задачи:

Результат работы кода на максимальных и минимальных значениях:

Время выполнения

Затраты памяти

Нижняя граница диапазона значений входных данных из текста задачи

0.00411 sec

14.96 Mb

Пример из задачи

0.00407 sec

14.80 Mb

Верхняя граница диапазона значений входных данных из текста задачи

0.02943 sec

15.09 Mb

Вывод по задаче: Увеличение значений вводимых переменных в пределах ограничений линейно увеличивает время работы программы и используемую память.

4 задача. Линейный поиск

Рассмотрим задачу поиска.

• Формат входного файла. Последовательность изn чисел A = a1, a2, . . . , an в первой строке, числа разделены пробелом, и значение V во второй строке.

Ограничения: 0 ≤ n ≤ 10**3, −10**3 ≤ ai, V ≤ 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.

Листинг кода

```
def task4():
    f = open('input.txt')
    f2 = open('output.txt', 'w')
    arr = [int(i) for i in f.readline().split()]
    v = int(f.readline())

    finded = []

    for i in range(len(arr)):
        if arr[i] == v:
            finded.append(i)

    if len(finded) == 0:
        print(-1, file=f2)
    else:
        print(len(finded), file=f2)
        print(*finded, file=f2)

    f.close()
    f2.close()
```

Текстовое объяснение решения.

Проходимся по массиву и добавляем в новый массив индекс совпадения элементов, если массив пуст выводим -1.

Результат работы кода на максимальных и минимальных значениях:

Время выполнения

Затраты памяти

Нижняя граница диапазона значений входных данных из текста задачи

0.00385 sec

14.70 Mb

Верхняя граница диапазона значений входных данных из текста задачи

0.00619 sec

14.85 Mb

Вывод по задаче:

Увеличение значений вводимых переменных в пределах ограничений линейно увеличивает время работы программы и используемую память.

7 задача. Знакомство с жителями Сортлэнда

Владелец графства Сортлэнд, граф Бабблсортер, решил познакомиться со своими подданными. Число жителей в графстве нечетно и составляет n, где n может быть достаточно велико, поэтому граф решил ограничиться знакомством с тремя представителями народонаселения: с самым бедным жителем, с жителем, обладающим средним достатком, и с самым богатым жителем. Согласно традициям Сортлэнда, считается, что житель обладает средним достатком, если при сортировке жителей по сумме денежных сбережений он оказывается ровно посередине. Известно, что каждый житель графства имеет уникальный идентификационный номер, значение которого расположено в границах от единицы до n. Информация о размере денежных накоплений жителей хранится в массиве M таким образом, что сумма денежных накоплений жителя, обладающего идентификационным номером i, содержится в ячейке M[i]. Помогите секретарю графа мистеру Свопу вычислить идентификационные номера жителей, которые будут приглашены на встречу с графом.

• Формат входного файла (input.txt). Первая строка входного файла содержит число жителей n ($3 \le n \le 9999$, n нечетно). Вторая строка содержит

описание массива M, состоящее из положительных вещественных чисел,

разделенных пробелами. Гарантируется, что все элементы массива M различны, а их значения имеют точность не более двух знаков после запятой и не превышают 106

• Формат выходного файла (output.txt). В выходной файл выведите три целых положительных числа, разделенных пробелами — идентификационные номера беднейшего, среднего и самого богатого жителей Сортлэнда.

Листинг кода

```
def task7():
    f = open('input.txt')
    f2 = open('output.txt', 'w')
    n = int(f.readline())
    arr = [(float(item),ind+1) for ind,item in enumerate(f.readline().split())]

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

    print(arr[0][1], arr[n//2][1], arr[-1][1], file=f2)

    f.close()
    f2.close()
```

Текстовое объяснение решения.

Сохраняем изначальные данные с индексом, сортируем и выводим изначальные индексы полученных нулевого, последнего, среднего элементов.

Результат работы кода на примере из задачи:

Результат работы кода на максимальных и минимальных значениях:

Время выполнения

Затраты памяти

Нижняя граница диапазона значений входных данных из текста задачи

0.00518 sec

14.99 Mb

Пример из задачи

0.00503 sec

14.82 Mb

Верхняя граница диапазона значений входных данных из текста задачи

0.00896 sec

15.30 Mb

Вывод по задаче: Увеличение значений вводимых переменных в пределах ограничений линейно увеличивает время работы программы и используемую память.

10 задача⋆ . Палиндром

Палиндром - это строка, которая читается одинаково как справа налево, так и слева направо.

На вход программы поступает набор больших латинских букв (не обязательно различных). Разрешается переставлять буквы, а также удалять некоторые буквы.

Требуется из данных букв по указанным правилам составить палиндром наибольшей длины, а если таких палиндромов несколько, то выбрать первый из них в алфавитном порядке.

• Формат входного файла (input.txt). В первой строке входных данных содержится число n ($1 \leq n \leq 100000$). Во второй строке задается последовательность из n больших латинских букв (буквы записаны без пробелов).

• Формат выходного файла (output.txt). В единственной строке выходных

данных выдайте искомый палиндром.

Листинг кода

```
def task10():
    f = open('input.txt')
    f2 = open('output.txt', 'w')

    n = int(f.readline())
    s = f.readline().strip()

    cnt_let = [0] * 26
    offset = 65
    for i in s:
        cnt_let[ord(i) - offset] += 1

    mn_i = -1
    for i in range(len(cnt_let)):
        if cnt_let[i] % 2 == 1:
            mn_i = i
            break
```

```
new_s = ''

for i in range(len(cnt_let)):
    if cnt_let[i] != 0:
        new_s += chr(i + offset) * (cnt_let[i] // 2)

if mn_i != -1:
    new_s = new_s + chr(mn_i + offset) + new_s[::-1]
else:
    new_s += new_s[::-1]

print(new_s, file=f2)

f.close()
f2.close()
```

Текстовое объяснение решения:

Выполним сортировку подсчетом (посчитаем количество букв в массиве размером 26), найдем центр нашего палиндрома, это первая по алфавиту буква с нечетным количеством (все четные лучшего использовать в зеркальных частях палиндрома, также четное количество останется и после взятия из нечетного количества одного элемента), затем составим первую половину палиндрома прибавляя половину количества буквы с округлением вниз. Затем если мы до этого нашли центр соединим: 1 часть + центр + 1 часть зеркальная. В итоге мы получим максимальный палиндром, в том числе первый в алфавитном порядке.

Результат работы кода на примере из задачи:

Результат работы кода на максимальных и минимальных значениях:

Время выполнения

Затраты памяти

Нижняя граница диапазона значений входных данных из текста задачи

0.00546 sec

15.07 Mb

Пример из задачи

0.00376 sec

15.00 Mb

Верхняя граница диапазона значений входных данных из текста задачи

0.01597 sec

15.27 Mb

Вывод по задаче:

Увеличение значений вводимых переменных в пределах ограничений линейно увеличивает время работы программы и используемую память.

Вывод

- В этой лабораторной мы научились делать insertion sort и различные его вариации, а также находить максимальный палиндром
- Линейный алгоритм O(n) выполняется прямо пропорционально величине данных, затраты памяти также прямо пропорциональны
- С помощью методов *time.perf_counter()* и *psutil.Process().memory_info().rss* можно отслеживать ресурсозатратность алгоритмов.