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 24

Выполнил:
Янин Е. В.
К3141

Проверил:
Афанасьев А. В.

Санкт-Петербург
2024 г.

Содержание отчета

Оглавление

Задачи по варианту

Задача №1. Сортировка вставкой

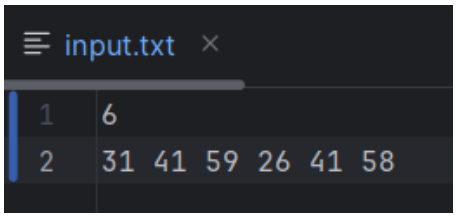
В данной задаче требуется написать алгоритм сортировки вставками и проверить его на массиве A = {31, 41, 59, 26, 41, 58}

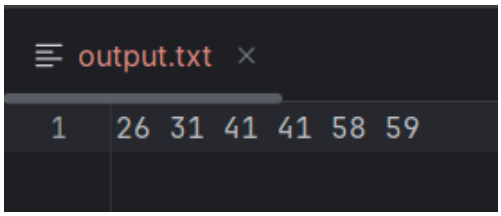
```
import time
import psutil


def insertion(lst):
    for j in range(1, len(lst)):
        key = lst[j]
        i = j - 1
        while i >= 0 and lst[i] > key:
            lst[i + 1] = lst[i]
            i = i - 1
        lst[i + 1] = key
    return lst


start = time.perf_counter()
with open('../tests/input.txt') as fin, open('../tests/output.txt', 'w') as
fout:
    n = int(fin.readline())
    A = [int(x) for x in fin.readline().split()]
    print('Затрачено времени: ', time.perf_counter() - start, 'сек.')
    print(f"Память: {psutil.Process().memory_info().rss / 1024 ** 2:.2f}
МБ")
    fout.write(' '.join([str(x) for x in insertion(A)]))
```

Объяснение решения:

1) Открываются два файла, входной и выходной
2) Элементы массива сортируются путем сравнения и замены элементов
3) В файл output.txt выводится отсортированный массив

Результат работы кода:





Вывод: я реализовал алгоритм сортировки вставки, работающий за квадратичное время.

Задача №6. Пузырьковая сортировка

В данной задаче требуется написать алгоритм пузырьковой сортировки и доказать, что он выводит отсортированный массив.

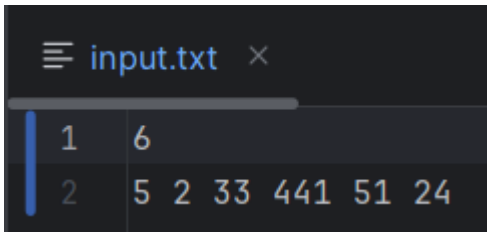
```
import time
import psutil
```

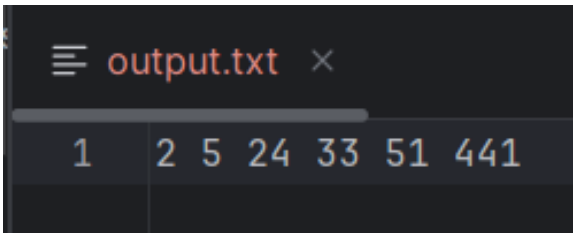
```
def bubble_sort(lst):
    for i in range(len(lst) - 1):
        for j in range(len(lst) - 1, i, -1):
            if lst[j] < lst[j - 1]:
                lst[j], lst[j - 1] = lst[j - 1], lst[j]
    return lst


start = time.perf_counter()
with open('../tests/input.txt') as fin, open('../tests/output.txt', 'w') as
fout:
    n = int(fin.readline())
    A = [int(x) for x in fin.readline().split()]
    sorted_A = bubble_sort(A)
    print('Затрачено времени: ', time.perf_counter() - start, 'сек.')
    print(f"Память: {psutil.Process().memory_info().rss / 1024 ** 2:.2f}
МБ")
    fout.write(' '.join([str(x) for x in sorted_A]))
    flag = True
    for i in range(len(sorted_A) - 1):
        if sorted_A[i] > sorted_A[i + 1]:
            flag = False
    if flag:
        print('Список отсортирован')
    else:
        print('Список не отсортирован')
```

Объяснение решения:

1) Открываются два файла, входной и выходной
2) Элементы массива сортируются путем сравнения и замены элементов местами
3) В файл output.txt выводится отсортированный массив.

Пример работы кода:





Вывод: я реализовал алгоритм пузырьковой сортировки и убедился в его работоспособности.

Задача №9. Сложение двоичных чисел

В данной задаче требуется написать алгоритм сложения двух n-битовых чисел, хранящихся в n-элементных массивах A и B. Найденную сумму нужно вывести в файл output.txt

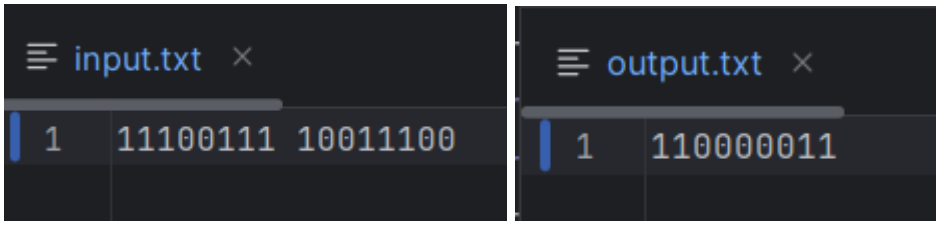
```
import time
import psutil


def applying(n1, n2):
    result = [0] * (len(n1) + 1)
    n1 = n1[::-1]
    n2 = n2[::-1]
```

```
    for i in range(len(n1)):
        isum = n1[i] + n2[i] + result[i]
        if isum == 0:
            result[i] = 0
        elif isum == 1:
            result[i] = 1
        elif isum == 2:
            result[i] = 0
            result[i + 1] = 1
        elif isum == 3:
            result[i] = 1
            result[i + 1] = 1
    return result[::-1]


start = time.perf_counter()
with open('../tests/input.txt') as fin, open('../tests/output.txt', 'w') as
fout:
    a, b = fin.readline().split()
    lst1 = [int(x) for x in a]
    lst2 = [int(x) for x in b]
    ans = applying(lst1, lst2)
    print('Затрачено времени: ', time.perf_counter() - start, 'сек.')
    print(f"Память: {psutil.Process().memory_info().rss / 1024 ** 2:.2f}
МБ")
    fout.write(''.join([str(x) for x in ans]))
```

Объяснение решения:

1) Открываются два файла, входной и выходной
2) Считываются два двоичных числа
3) Числа складываются «столбиком»
4) В файл output.txt выводится их сумма.

Пример работы кода:



Вывод: алгоритм справляется со сложением двоичных чисел.

Вывод

В данной лабораторной работе я реализовал несколько алгоритмов сортировки, работающих за квадратичное время.