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 23

Выполнила:
Федорова М. В.
К3139

Проверил:
Афанасьев А. В.

Санкт-Петербург
2024 г.

Содержание отчета

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 ($1 \le n \le 10^3$) — 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.

Выберите любой набор данных, подходящих по формату, и протестируйте алгоритм.

```python
import psutil
import time
t_start = time.perf_counter()
with open ("input.txt","r") as file:
    n=int(file.readline())
    arr=[int(i) for i in file.readline().split()]
def insertionSort(n,arr):
    if n <= 1:
        return
    for i in range(1, n):
        key = arr[i]
        j = i-1
        while j >= 0 and key < arr[j]:
            arr[j+1] = arr[j]
            j -= 1
        arr[j+1] = key
with open("output.txt","w") as file:
    if 0<=n<=10**3 and min(arr)>=-10**9 and max(arr)<10**9:
        insertionSort(n,arr)
        file.write(" ".join(str(a) for a in arr))
    else:
        print("Число в массиве по модулю превосходит 10^9 или количсетво элементов не соответс
print ("Время работы: %s секунд " % (time.perf_counter () - t_start))
print(f"Память: {psutil.Process().memory_info().rss / 1024 ** 2:.2f} МБ")
```

1. Считываем значения *n* и *arr* из файла *input.txt*
2. Создаём функцию Insertion-sort принимая в ней переменные из пункта 1
3. Insertion-sort:

   - Эта функция принимает два параметра: количество элементов (n) и сам массив (arr). Если размер массива меньше или равен единице — ничего не делаем.

   - Основной цикл начинается со второго элемента (индекс от одного). Для каждого элемента определяется его позиция относительно уже отсортированной части массива слева от него путем перемещения больших значений вправо до нахождения подходящей позиции для текущего значения (key).

   - Далее открываем файл *output.txt* проверяет условия на допустимые размеры входных данных, если они соответствуют заданным границам, то вызывается функция сортировки. После успешной сортировки результат записывается в выходной файл через пробелы между числами.

4. После завершения всех операций программа выводит затраченное на выполнение время с момента старта и объем используемой оперативной памяти процессом.
5. Результат:

Результат работы кода на максимальных и минимальных значениях:

```
Время работы: 0.017355541999999998 секунд
Память: 11.92 МБ
```

```
Время работы: 0.0001976669999999986 секунд
Память: 11.77 МБ
```

Вывод по задаче: В ходе выполнения задания была разработана программа для сортировки массива чисел с использованием алгоритма вставок. Также реализована функция для измерения времени работы программы и контроля за потреблением оперативной памяти, что позволяет оценить эффективность реализации.

Задача №6. Пузырьковая сортировка

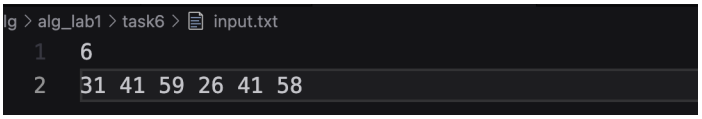
Напишите код на Python и докажите корректность пузырьковой сортировки. Для доказательства корректности процедуры вам необходимо доказать, что она завершается и что A'(1] ≤ A'(2] ≤ ... ≤ A'[n], где A' - выход процедуры *Bubble_Sort*, а *n* - длина массива A.
Определите время пузырьковой сортировки в наихудшем случае и в среднем случае и сравните его со временем сортировки вставкой.
Формат входного и выходного файла и ограничения - как в задаче 1.

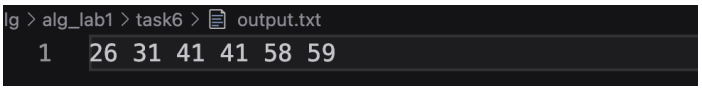
alg_lab1 > task6 > index6.py > ...

```python
import psutil
import time
import random

t_start = time.perf_counter()
with open ("input.txt","r") as file:
    n=int(file.readline())
    arr=[int(i) for i in file.readline().split()]

def bubble_sort(n,arr):
    for i in range(n):
        for j in range(0, n - i - 1):
            if arr[j] > arr[j + 1]:
                arr[j], arr[j + 1] = arr[j + 1], arr[j]
    return arr
with open("output.txt","w") as file:
    if 0<=n<=10**3 and min(arr)>=-10**9 and max(arr)<10**9:
        #testMaxArr = [random.randint(-10**9, 10**9) for i in range(10**5)]
        #bubble_sort(n,testMaxArr)
        bubble_sort(n,arr)
        file.write(" ".join(str(a) for a in arr))
    else:
        print("Число в массиве по модулю превосходит 10^9 или количсетво элементов не соответс
print ("Время работы: %s секунд " % (time.perf_counter () - t_start))
print(f"Память: {psutil.Process().memory_info().rss / 1024 ** 2:.2f} МБ")
```

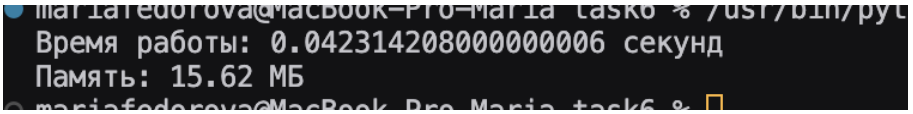
1. Считываем значения *n* и *arr* из файла *input.txt*
2. Создаём функцию *Bubble_Sort*
3. *Bubble_Sort*:
   - Эта функция принимает два параметра: количество элементов (n) и сам массив (arr). Сортировка выполняется за счет двух вложенных циклов.
   - Внешний цикл проходит по всем элементам массива от первого до последнего.
   - Внутренний цикл сравнивает соседние элементы и меняет их местами при необходимости — если текущий элемент больше следующего элемента. Таким образом самые большие элементы «всплывают» вверх по массиву после каждой итерации внешнего цикла.
   - Далее открываем файл *output.txt* проверяет условия на допустимые размеры входных данных, если они соответствуют заданным границам, то вызывается функция сортировки. После успешной сортировки результат записывается в выходной файл через пробелы между числами.

4. Результат:





Результат работы кода на максимальных и минимальных значениях:



	Время выполнения, ms	Затраты памяти, MB
Нижняя граница диапазона значений входных данных из текста задачи	0.244	11.78

Пример из задачи	0.62	11.84
Верхняя граница диапазона значений входных данных из текста задачи	42.314	15.62

Вывод по задаче: В ходе выполнения задания была разработана программа для сортировки массива чисел с использованием пузырьковой сортировки.

Задача №8. Секретарь Своп

Дан массив, состоящий из n целых чисел. Вам необходимо его отсортировать по неубыванию. Но делать это нужно так же, как это делает мистер Своп — то есть, каждое действие должно быть взаимной перестановкой пары элементов. Вам также придется записать все, что Вы делали, в файл, чтобы мистер Своп смог проверить Вашу работу.

- **Формат входного файла (input.txt).** В первой строке входного файла содержится число n ($3 \leq n \leq 5000$) — число элементов в массиве. Во второй строке находятся n целых чисел, по модулю не превосходящих 10^9. Числа могут совпадать друг с другом.
- **Формат выходного файла (output.txt).** В первых нескольких строках выведите осуществленные Вами операции перестановки элементов. Каждая строка должна иметь следующий формат:

  ```
  Swap elements at indices X and Y.
  ```

  Здесь X и Y — различные индексы массива, элементы на которых нужно переставить ($1 \leq X, Y \leq n$). Мистер Своп любит порядок, поэтому сделайте так, чтобы $X < Y$.

  После того, как все нужные перестановки выведены, выведите следующую фразу:

  ```
  No more swaps needed.
  ```

- Пример:

input.txt	output.txt
5 3 1 4 2 2	Swap elements at indices 1 and 2. Swap elements at indices 2 and 4. Swap elements at indices 3 and 5. No more swaps needed.

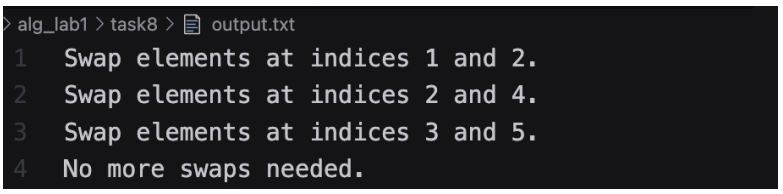
```python
import psutil
import time
import random

t_start = time.perf_counter()
with open ("input.txt","r") as file:
    n= int(file.readline())
    arr=list(map(int, file.readline().split()))

def sort_with_swaps(n,arr):
    swaps=[]
    for i in range(n - 1):
        k = i + 1
        minArr = arr[k]
        for j in range(i + 1, n):
            if arr[j] < minArr:
                minArr = arr[j]
                k = j
        if minArr < arr[i]:
            arr[i], arr[k] = arr[k], arr[i]
            swaps.append([i+1,k+1])
    return swaps
#testMaxArr = [random.randint(-10**9, 10**9) for i in range(10**5)]
#swap_index=sort_with_swaps(n,testMaxArr)
swap_index = sort_with_swaps(n,arr)
with open('output.txt', 'w') as f:
    if 3<=n<=5000 and min(arr)>=-10**9 and max(arr)<10**9:
       for [x, y] in swap_index:
            f.write(f"Swap elements at indices {x} and {y}.\n")
       f.write("No more swaps needed.")
    else:
        print("Число в массиве по модулю превосходит 10^9 или количсетво элементов не соответствует
print ("Время работы: %s секунд " % (time.perf_counter () - t_start))
print(f"Память: {psutil.Process().memory_info().rss / 1024 ** 2:.2f} МБ")
```

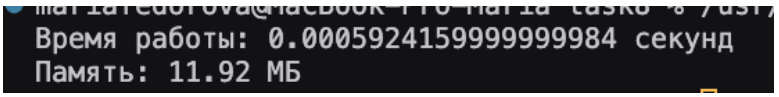
1. Считываем значения *n* и *arr* из файла *input.txt*
2. Создаём функцию *sort_with_swaps*
3. *sort_with_swaps*:

- Создается пустой список *swaps*, который будет хранить индексы элементов массива, которые были обменены.
- Внешний цикл проходит по всем элементам до предпоследнего (n-1). Для каждого элемента ищется минимальный элемент среди оставшихся неотсортированных значений.
- Во внутреннем цикле происходит сравнение текущего элемента с остальными; если находится меньший элемент (minArr), обновляются индекс этого элемента (k) и само значение минимального элемента.

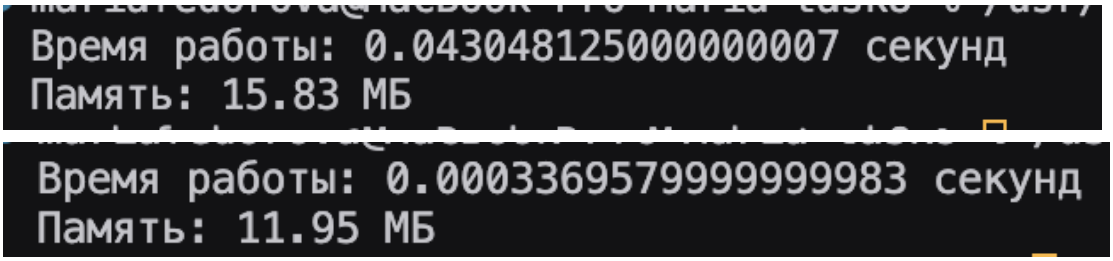
- Если найденный минимум меньше текущего элемента на позиции (i), то они меняются местами (обмен).
- Индексы этих двух элементов сохраняются в списке swap как индекс первого + 1 , индекс второго + 1.

4. Далее каждая пара индексов замены записывается в выходной файл *output.txt*. После всех записей добавляется сообщение о том что больше замен не требуется.
5. Результат:





Результат работы кода на максимальных и минимальных значениях:



	Время выполнения, ms	Затраты памяти, MB
Нижняя граница диапазона значений входных данных из текста задачи	0.336	11.95
Пример из задачи	0.592	11.92
Верхняя граница диапазона значений входных данных из текста задачи	43.048	15.83

Вывод по задаче: Применила алгоритм сортировки выбором на практике, убедилась в правильности его работы и протестировала его на минимальных и максимальных значениях.

Дополнительные задачи

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.

Подумайте, можно ли переписать алгоритм сортировки вставкой с использованием рекурсии?

alg_lab1 > task3 > index3.py > ...

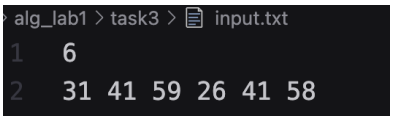
```python
import psutil
import time

t_start = time.perf_counter()
def insertion_sort(n,arr):
    for i in range(1, n):
        key = arr[i]
        j = i - 1
        while j >= 0 and key > arr[j]:
            arr[j + 1] = arr[j]
            j -= 1
        arr[j + 1] = key
    return arr
with open ("input.txt","r") as file:
    n=int(file.readline())
    arr=[int(i) for i in file.readline().split()]
with open("output.txt","w") as file:
    if 0<=n<=10**3 and min(arr)>=-10**9 and max(arr)<10**9:
        insertion_sort(n,arr)
        file.write(" ".join(str(a) for a in arr))
    else:
        print("Число в массиве по модулю превосходит 10^9 или количсетво элементов не соответс
print ("Время работы: %s секунд " % (time.perf_counter () - t_start))
print(f"Память: {psutil.Process().memory_info().rss / 1024 ** 2:.2f} МБ")
```

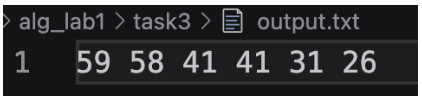
1. Считываем значения *n* и *arr* из файла *input.txt*
2. Создаём функцию *insertion_sort*
3. *insertion_sort*:

- Эта функция принимает два аргумента: количество элементов массива (n) и сам массив (arr). Сортировка происходит следующим образом:

- Проходим по каждому элементу массива начиная со второго.
- Для каждого элемента сравниваем его с предыдущими и находим правильное место для него.
- Если текущий элемент больше предыдущего — смещаем элементы вправо до тех пор, пока не найдём подходящее место для вставки.

4. Далее открываем файл *output.txt* проверяет условия на допустимые размеры входных данных, если они соответствуют заданным границам, то вызывается функция сортировки. После успешной сортировки результат записывается в выходной файл через пробелы между числами.
5. Результат:





Вывод по задаче: в ходе работы над задаче я смогла разработать алгоритм сортировки вставкой по убыванию,убедилась в правильности его работы и протестировала его на минимальных и максимальных значениях.

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

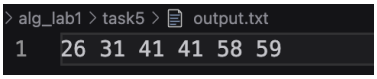
наихудшем случае и в среднем случае и сравните его со временем сортировки вставкой.
Формат входного и выходного файла и ограничения - как в задаче 1.

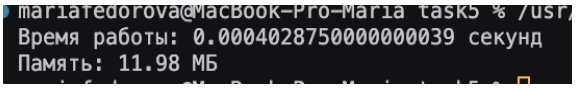
```python
import psutil
import time
import random
t_start = time.perf_counter()
def insertion_sort(n, arr):
    for i in range(1, n):
        key = arr[i]
        j = i - 1
        while j >= 0 and key < arr[j]:
            arr[j + 1] = arr[j]
            j -= 1
        arr[j + 1] = key
    return arr


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

with open ("input.txt","r") as file:
    n=int(file.readline())
    arr=[int(i) for i in file.readline().split()]
with open("output.txt","w") as file:
    if 0<=n<=10**3 and min(arr)>=-10**9 and max(arr)<10**9:
        selection_sort(n,arr)
        #testMaxArr = [random.randint(-10**9, 10**9) for i in range(10**5)]
        #selection_sort(n,testMaxArr)
        file.write(" ".join(str(a) for a in arr))
    else:
        print("Число в массиве по модулю превосходит 10^9 или количсетво элементов не соответствует ограничниям")
print ("Время работы: %s секунд " % (time.perf_counter () - t_start))
print(f"Память: {psutil.Process().memory_info().rss / 1024 ** 2:.2f} МБ")
```

1. Считываем значения *n* и *arr* из файла *input.txt*
2. Создаём функцию *selection_sort*
3. *selection_sort:*

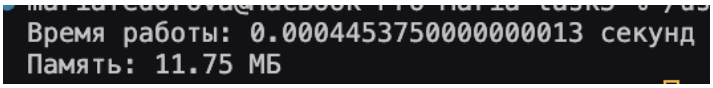
- Эта функция принимает два аргумента: количество элементов массива (n) и сам массив (arr). Сортировка происходит следующим образом: она ищет минимальный элемент среди оставшихся элементов и меняет его местами с первым из них до конца списка.

4. Далее открываем файл *output.txt* проверяет условия на допустимые размеры входных данных, если они соответствуют заданным границам, то вызывается функция сортировки. После успешной сортировки результат записывается в выходной файл через пробелы
5. Результат:





Результат работы кода на максимальных и минимальных значениях:



Вывод по задаче: в ходе работы над задачей я смогла разработать алгоритм сортировки выбором, убедилась в правильности его работы и протестировала его на минимальных и максимальных значениях.

Задача №10. 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.
- Пример:

input.txt	output.txt
5 AAB	ABA
6 QAZQAZ	AQZZQA
6 ABCDEF	A

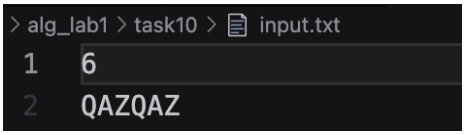
- Ограничение по времени. 1сек.
- Ограничение по памяти. 64 мб.

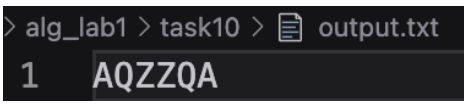
```
alg_lab1 > task10 > index10.py > ...
import psutil
import time

t_start = time.perf_counter()
with open ("input.txt","r") as file:
    n= int(file.readline())
    s=[i for i in file.readline()]
def palindrome(n, s):
    dict = {}
    for letter in s:
        if letter in dict:
            dict[letter] += 1
        else:
            dict[letter] = 1
    half_palindrome = []
    center_char = ''
    for char in sorted(dict.keys()):
        count = dict[char]
        half_palindrome.append(char * (count // 2))
        if count % 2 == 1 and (center_char == '' or char < center_char):
            center_char = char
    left_half = ''.join(half_palindrome)
    palindrome_result = left_half + center_char + left_half[::-1]
    return palindrome_result
with open("output.txt","w") as file:
        str_palindrome=palindrome(n,s)
        file.write(str_palindrome)
print ("Время работы: %s секунд " % (time.perf_counter () - t_start))
print(f"Память: {psutil.Process().memory_info().rss / 1024 ** 2:.2f} МБ")
```

1. Открываем файл "input.txt" для чтения. Первая строка содержит число (n). Вторая строка читается как список символов и сохраняется в переменной s.
2. Создаём функцию *palindrome*
3. *palindrome:*

- Внутри функции создаём словарь (dict) для подсчета количества каждого символа во входной строке: Для каждого буквы (символа) проверяем — есть ли она уже в словаре? Если да – увеличиваем её количество на один. Если нет – добавляем эту букву со значением равным одному.
- Создаем пустой список для хранения половины палиндрома. Перебираем отсортированные уникальные буквы из словаря. Для каждой буквы добавляем ее половину во временный массив (*half_palindrome*), используя целочисленное деление на два.
- Также проверяем наличие центрального элемента — если число букв нечетно, выбирается минимальный по алфавиту знак как центр будущего палиндрома.
- Создается левая часть результата путем объединения всех частей массива. Затем формируется финальная строка-палиндром за счет соединения левой части с центровым элементом и перевернутой левой частью
- Открываем выходной файл "output.txt" для записи результата после вызова функции поиска палиндрома. Результат записывается обратно как строковое представление найденного слова-палиндром.

4. Результат:





Вывод по задаче: в ходе работы над задачей я смогла разработать алгоритм по нахождению палиндрома в строке, убедилась в правильности его работы.

Вывод

В ходе работы были изучены алгоритмы пузырьковой сортировки,а также вставками и выбором. Протестированы на максимальных и минимальных вводных данных. Также был разработан алгоритм по нахождению палиндрома максимальной длинны в строке. Было проанализировано время работы каждого алгоритма и объем используемой памяти.