Министерство образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования «Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики»

Мегафакультет трансляционных информационных технологий

Факультет инфокоммуникационных технологий

Дисциплина: Алгоритмы и структуры данных

**Отчет по Лабораторной работе №2**

Выполнила: Микулина Алиса Романовна

Группа: K3143, 1 курс

Преподаватель: Харьковская Татьяна Александровна

Санкт-Петербург

10.09.2021

**Описание задания**

**Задание 1.**

Написать код для сортировки массива вставкой.

**Задание 2.**

Модифицировать код сортировки массива вставкой, чтобы в выходном файле в первой строке отображалось n чисел, которые обозначают новый индекс элемента массива после обработки.

**Задание 3.**

Переписать код для сортировки массива вставкой так, чтобы сортировка шла в убывающем порядке.

**Задание 4.**

Написать код для осуществления линейного поиска элемента.

**Задание 5.**

Написать код для сортировки массива выбором.

**Задание 6.**

Отредактировать и привести в чувство код для пузырьковой сортировки.

**Задание 10.**

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.

**Описание решения и исходный код**

**Задача 1.**

В данном коде я осуществила стандартную сортировку вставкой. Происходит поиск места, подходящего для вставки рассматриваемого элемента. Каждый элемент по очереди сравнивается со стоящими слева от него, и, пока он не больше, чем символ слева, он остается на месте. Затем, по нахождении элемента слева, равного элементу или меньшего ему, элемент встает на указанное место, при этом он удаляется со своего изначального места.

def intuitive\_sorting(massive, length):  
 for i in range(length):  
 counter = i  
 while massive[i] < massive[counter-1]:  
 counter = counter - 1  
 massive.insert(max(counter,0), massive[i])  
 massive.pop(i + 1)  
 return massive  
  
f = open("input.txt", "r")  
length = int(f.readline())  
massive = list(map(int, f.readline().split(" ")))  
f.close()  
  
d = open("output.txt", "w")  
d.write(' '.join(list(map(str,intuitive\_sorting(massive, length)))))  
d.close()

**Задача 2.**

Принцип работы данного кода аналогичен коду в первой задаче. Единственное отличие — введение массива **indexes**, в который при каждом вписывании элемента на определенное место с индексом вписывается этот индекс.

def intuitive\_sorting(massive, length, indexes):  
 for i in range(length):  
 counter = i  
 while massive[i] < massive[counter-1]:  
 counter = counter - 1  
 massive.insert(max(counter,0), massive[i])  
 indexes.append(max(counter,0) + 1)  
 massive.pop(i + 1)  
 return [indexes, massive]  
  
f = open("input.txt", "r")  
length = int(f.readline())  
massive = list(map(int, f.readline().split(" ")))  
indexes = []  
f.close()  
  
  
d = open("output.txt", "w")  
ans = intuitive\_sorting(massive, length, indexes)  
d.write(" ".join(map(str, ans[0])) + '\n' + " ".join(map(str, ans[1])))  
d.close()

**Задача 3.**

Данный код аналогично основан на принципе работы сортировки вставкой из первого задания. Для того, чтобы выходной массив был отсортирован в порядке убывания, достаточно просто перевернуть его, написав massive = massive[::-1]. Данное действие я вынесла в отдельную функцию, так как оно не является частью основного кода.

def intuitive\_sorting(massive, length):  
 for i in range(length):  
 counter = i  
 while massive[i] < massive[counter - 1]:  
 counter = counter - 1  
 massive.insert(max(counter,0), massive[i])  
 massive.pop(i + 1)  
 return swap(massive)  
  
def swap(massive):  
 massive = massive[::-1]  
 return massive  
  
f = open("input.txt", "r")  
length = int(f.readline())  
massive = list(map(int, f.readline().split(" ")))  
f.close()  
  
d = open("output.txt", "w")  
d.write(' '.join(list(map(str,intuitive\_sorting(massive, length)))))  
d.close()

**Задача 4.**

Я написала код стандартного линейного поиска. Мы пробегаем по элементам массива, по очереди сравнивая каждый элемент с искомым. Счетчик **countiterations** считает кол-во элементов массива, совпадающих с искомым значением. В случае, если **countiterations == 0**, будет выведен **-1**.

f = open("input.txt", "r")  
massive = [int(a) for a in f.readline().split(" ")]  
element = int(f.readline())  
f.close()  
  
  
countiterations = [0]  
iterations = []  
for i in range(len(massive)):  
 if massive[i] == element:  
 countiterations[0] += 1  
 iterations.append(i+1)  
if countiterations[0] == 0:  
 ans = -1  
elif countiterations[0] == 1:  
 ans = iterations[0]  
else:  
 ans = [countiterations, iterations]  
  
  
  
d = open("output.txt", "w")  
if countiterations[0] > 1:  
 d.write(" ".join(map(str, ans[0])) + ', ' + (", ".join(map(str, ans[1]))))  
else:  
 d.write(str(ans))  
d.close()

**Задача 5.**

Сортировка выбором основана на постоянном поиске минимального элемента массива. Сначала определяется наименьший элемент массива, который ставится на место элемента massive[0]. Затем производится поиск второго наименьшего элемента массива, который ставится на место элемента massive[1]. Этот процесс продолжается до конца сортировки.

def choice\_sorting(massive, length):  
 minimum = massive[0]  
 for j in range(length):  
 minimum = massive[j]  
 for i in range(j, length):  
 if minimum > massive[i]:  
 minimum, massive[i] = massive[i], minimum  
 massive[j] = minimum  
 return massive  
  
f = open("input.txt", "r")  
length = int(f.readline())  
massive = list(map(int, f.readline().split(" ")))  
f.close()  
  
d = open("output.txt", "w")  
d.write(' '.join(list(map(str,choice\_sorting(massive, length)))))  
d.close()

**Задача 6.**

Пузырьковая сортировка заключается в том, что происходит поочередное сравнение соседних элементов, и если правый элемент меньше левого, то эти два элемента меняются местами. Цикл for идет с конца сортируемого списка. Таким образом, в начале образуется неизменяемый “хвостик”, в который цикл уже не заходит. Тем самым, длина части массива, по которой проходит цикл, сокращается.

def bubble\_sorting(massive, length):  
 for i in range(length):  
 for j in range(length - 1, i, -1):  
 if massive[j] < massive[j - 1]:  
 massive[j], massive[j - 1] = massive[j - 1], massive[j]  
 return massive  
  
f = open("input.txt", "r")  
length = int(f.readline())  
massive = list(map(int, f.readline().split(" ")))  
f.close()  
  
d = open("output.txt", "w")  
d.write(' '.join(list(map(str,bubble\_sorting(massive, length)))))  
d.close()

**Задача 10.**

Задача решена с помощью использования такого объекта, как dictionary. В качестве ключей берутся буквы, встречающиеся в строке, а сами элементы – количество вхождений этих букв в строку. Все действия производятся с помощью простейших циклов **for**.

Если в строке встречаются буквы, число которых нечетно, нечетное количество букв оставляется только у первой из них по алфавиту. У остальных букв, число вхождений которых нечетно, удаляется по одному вхождению, чтобы количество было четным.

def make\_palindrome(line):  
 unique = {}  
 odd = []  
 palindrome = ""  
 for letter in line:  
 if letter not in unique:  
 unique.setdefault(letter, 1)  
 else:  
 unique[letter] += 1  
 unique\_sorted = {i: unique[i] for i in sorted(unique)}  
  
 for letter, amount in unique\_sorted.items():  
 if amount % 2 != 0:  
 odd.append(letter)  
  
 for letter in odd:  
 unique\_sorted[letter] -= 1  
  
 unique\_sorted\_reversed = {i: unique\_sorted[i] for i in sorted(unique\_sorted, reverse=True)}  
  
 for letter, amount in unique\_sorted.items():  
 palindrome += letter \* (amount // 2)  
  
 palindrome += odd[0]  
  
 for letter, amount in unique\_sorted\_reversed.items():  
 palindrome += letter \* (amount // 2)  
  
 return palindrome  
  
f = open("input.txt", "r")  
length = int(f.readline())  
line = f.readline()  
f.close()  
  
d = open("output.txt", "w")  
d.write(make\_palindrome(line))  
d.close()

**Описание проведенных тестов.**

**1**

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Время выполнения** |
| 19  2 2 2 5 6 7 8 3 4 6 8 7 4 2 5 5 12 12 14 | 2 2 2 3 4 5 6 6 7 8 8 7 4 2 5 5 12 12 14 | 0.0020676000000000028 секунд |
| 6  31 41 59 26 41 58 | 26 31 41 41 58 59 | 0.0008539999999999937 секунд |
| 22  656 234 89 12334 45 88900 123 654 1 23 6 876 2 12 34 8 9 87 6 3 23456 87 | 1 2 3 6 6 8 9 12 23 34 45 87 87 89 123 234 654 656 876 12334 23456 88900 | 0.0007656000000000052 секунд |

**2**

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Время выполнения** |
| 19  2 2 2 5 6 7 8 3 4 6 8 7 4 2 5 5 12 12 14 | 1 2 3 4 5 6 7 4 5 8 11 10 6 4 9 10 17 18 19  2 2 2 2 3 4 4 5 5 5 6 6 7 7 8 8 12 12 14 | 0.0011467000000000005 секунд |
| 10  1 8 4 2 3 7 5 6 9 0 | 1 2 2 2 3 5 5 6 9 1  0 1 2 3 4 5 6 7 8 9 | 0.0008920000000000039 секунд |
| 22  656 234 89 12334 45 88900 123 654 1 23 6 876 2 12 34 8 9 87 6 3 23456 87 | 1 1 1 4 1 6 3 5 1 2 2 10 2 4 6 4 5 10 4 3 20 13  1 2 3 6 6 8 9 12 23 34 45 87 87 89 123 234 654 656 876 12334 23456 88900 | 0.0008974999999999955 секунд |

**3**

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Время выполнения** |
| 19  2 2 2 5 6 7 8 3 4 6 8 7 4 2 5 5 12 12 14 | 14 12 12 8 8 7 7 6 6 5 5 5 4 4 3 2 2 2 2 | 0.0008100000000000052 секунд |
| 6  31 41 59 26 41 58 | 59 58 41 41 31 26 | 0.0009368999999999975 секунд |
| 22  656 234 89 12334 45 88900 123 654 1 23 6 876 2 12 34 8 9 87 6 3 23456 87 | 88900 23456 12334 876 656 654 234 123 89 87 87 45 34 23 12 9 8 6 6 3 2 1 | 0.0007499000000000047 секунд |

**4**

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Время выполнения** |
| 2 2 2 5 6 7 8 3 4 6 8 7 4 2 5 5 12 12 14  2 | 4, 1, 2, 3, 14 | 0.0009106000000000114 секунд |
| 31 41 59 26 41 58  59 | 3 | 0.0007014999999999938 секунд |
| 656 234 89 12334 45 88900 123 654 1 23 6 876 2 12 34 8 9 87 6 3 23456 87  156 | -1 | 0.0008038999999999963 секунд |

**5**

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Время выполнения** |
| 19  2 2 2 5 6 7 8 3 4 6 8 7 4 2 5 5 12 12 14 | 2 2 2 2 3 4 4 5 5 5 6 6 7 7 8 8 12 12 14 | 0.0006202000000000013 секунд |
| 6  31 41 59 26 41 58 | 26 31 41 41 58 59 | 0.0007033000000000039 секунд |
| 22  656 234 89 12334 45 88900 123 654 1 23 6 876 2 12 34 8 9 87 6 3 23456 87 | 1 2 3 6 6 8 9 12 23 34 45 87 87 89 123 234 654 656 876 12334 23456 88900 | 0.0008396999999999988 секунд |

**6**

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Время выполнения** |
| 19  2 2 2 5 6 7 8 3 4 6 8 7 4 2 5 5 12 12 14 | 2 2 2 2 3 4 4 5 5 5 6 6 7 7 8 8 12 12 14 | 0.0007116000000000067 секунд |
| 6  31 41 59 26 41 58 | 26 31 41 41 58 59 | 0.0006516000000000022 секунд |
| 22  656 234 89 12334 45 88900 123 654 1 23 6 876 2 12 34 8 9 87 6 3 23456 87 | 1 2 3 6 6 8 9 12 23 34 45 87 87 89 123 234 654 656 876 12334 23456 88900 | 0.0007912999999999948 секунд |

**10**

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Время выполнения** |
| DHJKLLKJTREWQWFGHJKLMVCDFMSDFGHJKLKJHREWQNMJUYTRDCVBNYTRQWER | CDDEFGHHJJJKKLLMNQRRTVWWYBYWWVTRRQNMLLKKJJJHHGFEDDC | 0.0006465000000000082 секунд |
| AAB | ABA | 0.0006647000000000042 секунд |
| QAZQAZA | AQZAZQA | 0.0006667999999999952 секунд |

**Выводы по проделанной работе.**

При выполнении данной лабораторной работы я научилась всевозможным способам сортировки в Python, таким как сортировка вставками, выбором, пузырьковая и подсчетом.