САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ
ФАКУЛЬТЕТ ИНФОКОММУНИКАЦИОННЫХ ТЕХНОЛОГИЙ

Отчет по лабораторной работе №3

По курсу «Алгоритмы и структуры данных»

Тема: Быстрая сортировка, сортировки за линейное время

Вариант №13

Выполнила:

Мельникова Настасья

К3141

Проверила:

Артамонова В.Е

Санкт-Петербург

2023

Содержание отчета

Задачи по варианту

Задача №1 Улучшение Quick sort

Цель задачи - переделать данную реализацию рандомизированного алгоритма быстрой сортировки, чтобы она работала быстро даже с последовательностями, содержащими много одинаковых элементов.

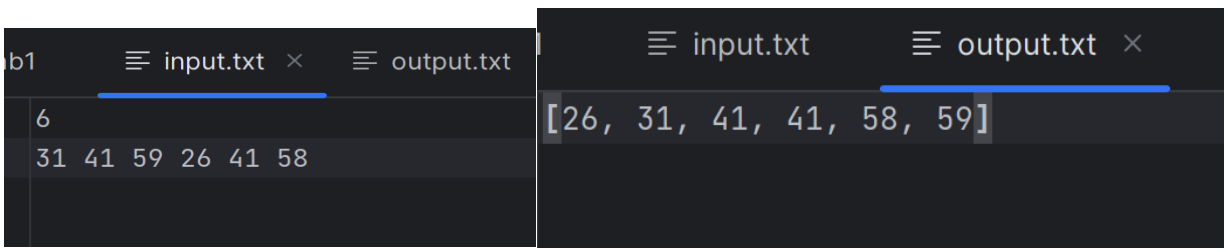
Входные данные(input.txt): В первой строке входного файла содержится число n (1 ≤ n ≤ 104) — число элементов в массиве. Во второй строке находятся n различных целых чисел, по модулю не превосходящих 10**9

Выходного данные (output.txt): Одна строка выходного файла с отсортированным массивом. Между любыми двумя числами должен стоять ровно один пробел.

```
f = open('input.txt')
n = int(f.readline())
s = list(map(int,f.readline().split()))
3 usages
def quicksort_triple_partition(s):
    if len(s) <= 1:
        return s

    pivot = s[len(s)//2]  # выбор опорного элемента
    less, equal, greater = [], [], []

    for element in s:
        if element < pivot:
            less.append(element)
        elif element > pivot:
            greater.append(element)
        else:
            equal.append(element)

    return quicksort_triple_partition(less) + equal + quicksort_triple_partition(greater)

sorted_numbers = quicksort_triple_partition(s)
d=open('output.txt','w')
d.write(str(sorted_numbers))
d.close()
```

Чтобы выполнить быструю сортировку я делю массив на три части, и сравнивая элементы записываю их в массив.

Пример:



Вывод по задаче: в этой задаче я научилась применять метод быстрой сортировки.

Задача №8 К ближайших точек к началу координат.

Цель. Заданы n точек на поверхности, найти K точек, которые находятся ближе к началу координат (0, 0), т.е. имеют наименьшее расстояние до начала координат.

Входные данные(input.txt): Первая строка содержит n - общее количество точек на плоскости и через пробел K - количество ближайший точек к началу координат, которые надо найти. Каждая следующая из n строк содержит 2 целых числа xi , yi , определяющие точку (xi , yi). Ограничения: 1 ≤ n ≤ 10**5 ; −10**9 ≤ xi , yi ≤ 10**9 - целые числа.

Выходного данные (output.txt): Выведите K ближайших точек к началу координат в строчку в квадратных скобках через запятую. Ответ вывести в порядке возрастания расстояния до начала координат. Если оно равно, порядок произвольный.

```
f = open('input.txt')
n = f.readline().split()
n1, kol = int(n[0]), int(n[1])
a = []
a_2 = []
while True:
    s = f.readline().split()
    if not s:
        break
    number_1, number_2 = int(s[0]), int(s[1])
    sqrt = int(math.sqrt(number_1**2 + number_2**2))
    a.append(sqrt)
    a_2.append(s)
```

```
def quicksort_triple_partition(a):
    if len(a) <= 1:
        return a
    pivot = a[len(a) // 2]  # выбор опорного элемента
    less, equal, greater = [], [], []
    for element in a:
        if element < pivot:
            less.append(element)
        elif element > pivot:
            greater.append(element)
        else:
            equal.append(element)
    return quicksort_triple_partition(less) + equal + quicksort_triple_partition(greater)

sorted_numbers = quicksort_triple_partition(a)
sorted_numbers = sorted_numbers[:kol]
a_1 = []
for i in sorted_numbers:
    for j in range(len(a)):
        for j in range(len(a_2)):
            if a[j] == i and a_2[j] not in a_1:
                a_1.append(a_2[j])

d=open('output.txt','w')
d.write(str(a_1))
d.close()
```

Чтобы выполнить эту задачу, я построчно искала длины от каждой точки до начала координат и потом, отсортировав массив с длинами, вывела самые ближайшие точки.

Пример:

```
5 2
2 4
3 6
-9 0
2 -4
```

```
[['2', '4'], ['2', '-4']]
```

Вывод по задаче: научилась работать с координатами и применять быструю сортировку.

Задача №9 Ближайшие точки

В этой задаче, ваша цель - найти пару ближайших точек среди данных n точек (между собой).

Входные данные(input.txt): Первая строка содержит n - количество точек. Каждая следующая из n строк содержит 2 целых числа xi , yi , определяющие точку (xi , yi). Ограничения: 1 ≤ n ≤ 10**5 ; −10**9 ≤ xi , yi ≤ 10**9 - целые числа.

Выходного данные (output.txt): Выведите минимальное расстояние. Абсолютная погрешность между вашим ответом и оптимальным решением должна быть не более 10**−3 . Чтобы это обеспечить, выведите ответ с 4 знаками после запятой.

```
f = open('input.txt')
n = f.readline().split()
a = []
a_1 = []
while True:
    s = f.readline().split()
    if not s:
        break
    a.append(s)
for i in range(0, len(a)):
    for j in range(i+1, len(a)):
        distance = math.sqrt((int(a[j][0]) - int(a[i][0])) ** 2 + (int(a[j][1]) - int(a[i][1])) ** 2)
        a_1.append(distance)
3 usages
def quicksort_triple_partition(a_1):
    if len(a_1) <= 1:
        return a_1
    pivot = a_1[len(a_1)//2]  # выбор опорного элемента
    less, equal, greater = [], [], []

    for element in a_1:
        if element < pivot:
            less.append(element)
        elif element > pivot:
            greater.append(element)
        else:
            equal.append(element)
    return quicksort_triple_partition(less) + equal + quicksort_triple_partition(greater)
sorted_numbers = quicksort_triple_partition(a_1)
min = min(sorted_numbers)
d = open('output.txt','w')
d.write(str("{0:.4f}".format(min)))
```

Для выполнения задачи я записываю пары координат в массив и проверяю их длины. Потом сортирую длины и беру минимальную.

Пример:

4 98 98 8 0 4 5 -3 9	1 6.4031

Вывод по задаче: научилась работать с координатами и применять быструю сортировку.

Доп задачи

Задача №3 Сортировка пугалом

«Сортировка пугалом» — это давно забытая народная потешка. Участнику под верхнюю одежду продевают деревянную палку, так что у него оказываются растопырены руки, как у огородного пугала. Перед ним ставятся n матрёшек в ряд. Из-за палки единственное, что он может сделать — это взять в руки две матрешки на расстоянии k друг от друга (то есть i-ую и i + k-ую), развернуться и поставить их обратно в ряд, таким образом поменяв их местами. Задача участника — расположить матрёшки по не убыванию размера.

Входные данные(input.txt): В первой строчке содержатся числа n и k (1 ≤ n, k ≤ 10**5) – число матрёшек и размах рук. Во второй строчке содержится n целых чисел, которые по модулю не превосходят 10**9 – размеры матрёшек.

Выходного данные (output.txt): Выведите «ДА», если возможно отсортировать матрёшки по не убыванию размера, и «НЕТ» в противном случае.

```
f = open('input.txt')
n, k = map(int, f.readline().split())
dolls = list(map(int, f.readline().strip().split()))
3 usages
def quick_sort(arr):
    if len(arr) <= 1:
        return arr
    p = arr[len(arr) // 2]
    left = [x for x in arr if x < p]
    middle = [x for x in arr if x == p]
    right = [x for x in arr if x > p]
    return quick_sort(left) + middle + quick_sort(right)

1 usage
def check(n, k, dolls):
    pugalo = quick_sort(dolls)
    for i in range(n - k):
        if pugalo[i] > pugalo[i + k]:
            return False
    return True
# Проверка сортировки и запись результата
if check(n, k, dolls):
    result = 'ДА'
else:
    result = 'НЕТ'
print(result)
d = open('output.txt', 'w')
d.write(result)
d.close()
```

Для выполнения задачи я сортирую массив с помощью быстрой сортировки и затем проверяю возможна ли она.

Вывод по задаче: научилась применять быструю сортировку, узнала о методе пугало.

Задача №5 Индекс Хирша

Для заданного массива целых чисел citations, где каждое из этих чисел - число цитирований i-ой статьи ученого-исследователя, посчитайте индекс Хирша этого ученого.

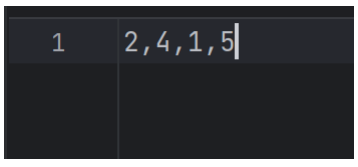
Входные данные(input.txt): Одна строка citations, содержащая n целых чисел, по количеству статей ученого (длина citations), разделенных пробелом или запятой.

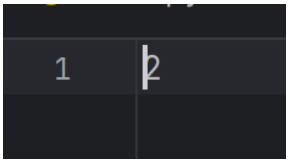
Выходного данные (output.txt): Одно число - индекс Хирша (h-индекс).

```python
f = open('input.txt')
citations = list(map(int,f.readline().strip().split(',')))
3 usages
def quicksort(arr):
    if len(arr) <= 1:
        return arr
    p = arr[len(arr) // 2]
    left = [x for x in arr if x < p]
    middle = [x for x in arr if x == p]
    right = [x for x in arr if x > p]
    return quicksort(left) + middle + quicksort(right)

# Функция для вычисления индекса Хирша
1 usage
def hindex(citations):
    citations = quicksort(citations)  # Сортировка массива цитирований
    n = len(citations)
    h = 0
    for i in range(n, 0, -1):
        if citations[n - i] >= i:
            h = i
            break
    return h

# Вычисляем индекс Хирша
h_index = hindex(citations)
d = open('output.txt', 'w')
d.write(str(h_index))
d.close()
```

Для выполнения этой задачи я использовала быструю сортировку и вычислила индекс Хирша с использованием отсортированного списка цитирований.

Пример:





Вывод по задаче: научилась применять быструю сортировку, узнала о шифре Хирша.

Тестирование алгоритмов и вывод

1	Время работы	Затраты памяти
Мин. значение	0.0014749298095703125	13.86328125 Мбайт
Макс. значение	2. 0165749298095703125	14.203125 Мбайт

8	Время работы	Затраты памяти
Мин. значение	0.00099945068359375	14.06640625 Мбайт
Макс. значение	2. 01325749298095705	16. 0078125 Мбайт

9	Время работы	Затраты памяти
Мин. значение	0.0004749298095703125	14.0078125 Мбайт
Макс. значение	2. 0465249298095703125	16. 0175125 Мбайт

3	Время работы	Затраты памяти
Мин. значение	0.0053749298095703125	13.890625 Мбайт
Макс. значение	3. 1465749298095703125	14.0078125 Мбайт

5	Время работы	Затраты памяти
Мин. значение	0.0004749298095703125	13.9609375 Мбайт
Макс. значение	4. 3265749298095703125	15.0078125 Мбайт

Вывод: в лабораторной работе я узнала о быстрой сортировке и научилась применять ее в задачах.