Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»
(Университет ИТМО)

Факультет **Инфокоммуникационных технологий**

Образовательная программа **Интеллектуальные системы в гуманитарной сфере**

Направление подготовки **45.03.04 Интеллектуальные системы в гуманитарной сфере**

О Т Ч Е Т

лабораторной работе 3

на тему: “Быстрая сортировка, сортировки за линейное время”

Обучающийся (или несколько) ФИО, № группы
Королева Екатерина
К3143

Работа выполнена с оценкой ____

Преподаватель:

_____
(подпись)

Дата 08.10.2021

Санкт-Петербург, 2021

1. Улучшение Quick-Sort*

```
import random#импортировать модуль random


def partition(array, start, stop):#объявить функцию partition и
дать на вход список array и переменные start и stop - начало и конец
списка
   b_array, c_start, d_start = array[start], start,
start#переменным присвоить соответственно начальный элемент списка и
начальный индекс start
   for i in range(start + 1, stop):#объявить цикл
       if array[i] < b_array:#если итый элемент списка меньше
начального, то
           c_start += 1#начальный индекс увеличить на 1
           array[c_start], array[i] = array[i],
array[c_start]#споменять местами значения начального и итого
элементов списка
           d_start += 1#начальный индекс увеличить на 1
           if c_start != d_start:#если перменные не равны, то
               array[d_start], array[i] = array[i],
array[d_start]#поменять местами значения элемента с индексом d_start
и итого
       elif array[i] == b_array:#иначе если первое условие не
выполнено, то
           d_start += 1#переменную увеличить на 1
           array[d_start], array[i] = array[i],
array[d_start]#поменять местами значения элемента с индексом d_start
и итого
   array[start], array[c_start] = array[c_start],
array[start]#поменять местами значения начального и итого элементов
списка(поставить опорный элемент на свое законное место)
   return c_start, d_start#вернуть значения переменных


def randomized_quicksort(array, start, stop):#объявить функцию и
дать на вход список и переменные start и stop - начало и конец
списка
   if start < stop:#если переменная начала списка меньше переменной
конца, то
       key = random.randint(start, stop - 1)#рандомно найти индекс
опорного элемента списка
       array[start], array[key] = array[key], array[start]#поменять
местами начальный и опроный элементы списка
       pivot1, pivot2 = partition(array, start, stop)#вызвать
функцию partition и найти границы деления
       randomized_quicksort(array, start, pivot1)#отсортировать
разделенные согласно границам массивы с помощью рекурсивного вызова
```

```
функции
        randomized_quicksort(array, pivot2 + 1, stop)#отсортировать
разделенные согласно границам массивы с помощью рекурсивного вызова
функции
        return array#вернуть массив


list = []#создать пустой список
for i in range(10):#объявить цикл
   array = random.randint(1, 100)#переменной присвоить рандомное
значение от 1 до 100
   list.append(array)#добавить переменную в список
print('Неотсортированный массив:', *list)#напечатать
неотсортированный массив
randomized_quicksort(list, 0, len(list))#вызвать функцию и на вход
дать неотсортированный массив, 0 - начальный индекс - и номер длины
массива - конечный индекс
print('  Отсортированный массив:', *list)#напечатать отсоритрованный
массив
```

Сложность сортировки по времени:

1. Сбалансированное деление - O(nlogn)
   в наилучшем случае: две подзадачи размером не больше n/2 (размер одной из них -⌊n/2⌋, второй - ⌈n/2⌉ − 1): T(n) = 2T(n/2) + Θ(n) = Θ(n log n)
2. Нет - O(n^2)
3. в наихудшем случае: одна подзадача с n − 1 элементами, вторая - пустая: T(n) = T(n − 1) + T(0) + Θ(n) = T(n − 1) + Θ(n)
   T(n) = n + (n − 1) + (n − 2) + ... = Θ(n^2) − арифметическая прогрессия

5. Индекс Хирша*

```
import random#импортировать модуль random


def partition(array, start, stop):#объявить функцию partition и
дать на вход список array и переменные start и stop - начало и конец
списка
   b_array, c_start, d_start = array[start], start,
start#переменным присвоить соответсвенно начальный элемент списка и
начальный индекс start
   for i in range(start + 1, stop):#объявить цикл
       if array[i] < b_array:#если итый элемент списка меньше
начального, то
           c_start += 1#начальный индекс увеличить на 1
           array[c_start], array[i] = array[i],
array[c_start]#поменять местами значения начального и итого
элементов списка
           d_start += 1#начальный индекс увеличить на 1
           if c_start != d_start:#если перменные не равны, то
               array[d_start], array[i] = array[i],
array[d_start]#поменять местами значения элемента с индексом d_start
и итого
       elif array[i] == b_array:#иначе если первое условие не
выполнено, то
           d_start += 1#переменную увеличить на 1
           array[d_start], array[i] = array[i],
array[d_start]#поменять местами значения элемента с индексом d_start
и итого
   array[start], array[c_start] = array[c_start],
array[start]#споменять местами значения начального и итого элементов
списка(поставить опорный элемент на свое законное место)
   return c_start, d_start#вернуть значения переменных


def randomized_quicksort(array, start, stop):#объявить функцию и
дать на вход список и переменные start и stop - начало и конец
списка
   if start < stop:#если переменная начала списка меньше переменной
конца, то
       key = random.randint(start, stop - 1)#рандомно найти индекс
опорного элемента списка
       array[start], array[key] = array[key], array[start]#поменять
местами начальный и опроный элементы списка
       pivot1, pivot2 = partition(array, start, stop)#вызвать
функцию partition и найти границы деления
```

```
        randomized_quicksort(array, start, pivot1)#отсортировать разделенные согласно границам массивы с помощью рекурсивного вызова функции
        randomized_quicksort(array, pivot2 + 1, stop)#отсортировать разделенные согласно границам массивы с помощью рекурсивного вызова функции
        return array#вернуть массив

list = []#создать пустой список
for i in range(10):#объявить цикл
   array = random.randint(1, 10)  #переменной присвоить рандомное значение от 1 до 10
   list.append(array)  #добавить переменную в список
randomized_quicksort(list, 0, len(list))#вызвать функцию и на вход дать неотсортированный массив, 0 - начальный индекс - и номер длины массива - конечный индекс
index_list = []#создать пустой список
for i in range(len(list)):#объявить цикл
   key = 0#переменной присвоить 0
   if list[i] != 0:#если итый элемент списка не равен 0, то
       for c in range(len(list[i:])):#объявить цикл
           if list[c] >= list[i] and len(list[i:]) >= list[i]:#если элемент списка с больше лиибо = итому элементу и длина итого элемента больше либо равна итому элементу, то
               key = 1#переменной переприсвоить значение с 0 на 1
           else:#иначе
               pass#оператор pass - ничего не выполнять
   if key == 1:#если переменная = 1, то
       index_list.append(list[i])#в список добавить итый элемент списка list
   else:#иначе
       pass#оператор pass - ничего не выполнять
index = index_list[len(index_list) - 1]#перменной присвоить значение списка равное длине списка-1
print('Индекс Хирша =', index)#напечать переменную
```

8. К ближайших точек к началу координат*

```
import random#импортировать модуль random


def partition(array, start, stop):#объявить функцию partition и
дать на вход список array и переменные start и stop - начало и конец
списка
   b_array, c_start = array[start], start#переменным присвоить
соответсвенно начальный элемент списка и начальный индекс start
   for i in range(start + 1, stop + 1):#объявить цикл
       if array[i] <= b_array:#если итый элемент списка меньше либо
= начальному, то
           c_start += 1#начальный индекс увеличить на 1
           array[c_start], array[i] = array[i],
array[c_start]#поменять местами значения начального и итого
элементов списка
   array[start], array[c_start] = array[c_start],
array[start]#поменять местами значения начального и итого элементов
списка(поставить опорный элемент на свое законное место)
   return c_start#вернуть значение переменной


def randomized_quicksort(array, start, stop):#объявить функцию и
дать на вход список и переменные start и stop - начало и конец
списка
   if start < stop:#если переменная начала списка меньше переменной
конца, то
       key = random.randint(start, stop)#рандомно найти индекс
опорного элемента списка
       array[start], array[key] = array[key], array[start]#поменять
местами начальный и опроный элементы списка
       pivot = partition(array, start, stop)#вызвать функцию
partition и найти границу деления
       randomized_quicksort(array, start, pivot - 1)#отсортировать
разделенные согласно границам массивы с помощью рекурсивного вызова
функции
       randomized_quicksort(array, pivot + 1, stop)#отсортировать
разделенные согласно границам массивы с помощью рекурсивного вызова
функции
   return array#вернуть массив


with open('input.txt', 'r') as f:#открыть файл input
   n, k = map(int, f.readline().split())#переменным присвоить
значения из первой строки файла типа integer
   x = [[]] * n#расширить словарь
   for i in range(n):#объявить цикл
```

```
        x[i] = [int(j) for j in f.readline().split()]#присвоить
итому элементу словаря следующее значение

a = {}#объявить словарь

for i in range(n):#объявить цикл
   a[float(((x[i][0] ** 2) + (x[i][1] ** 2)) ** 0.5)] = x[i]#по
формуле итый элемент присвоить элементу в словаре

list = [float(key) for key in a]#объявить список
randomized_quicksort(list, 0, len(list) - 1)#вызвать функцию

with open('output.txt', 'w') as f:#открыть файл output
   for i in range(k-1):#объявить цикл
       f.write(str(a[list[i]]))#записать в файл элемент словаря
типа string
       f.write(',')#записать в файл запятушечку
   f.write(str(a[list[k - 1]]))#записать в файл элемент словаря
типа string
```

3. Сортировка пугалом

```
import random  # импортировать модуль random


def randomized_quicksort(array, start,
                         stop):  # объявить функцию и дать на вход
список и переменные start и stop - начало и конец списка
   if start < stop:  # если переменная начала списка меньше
переменной конца, то
       key = random.randint(start, stop)  # рандомно найти индекс
опорного элемента списка
       array[start], array[key] = array[key], array[start]  #
поменять местами начальный и опроный элементы списка
       pivot = partition(array, start, stop)  # вызвать функцию
partition и найти границу деления
       randomized_quicksort(array, start,
                            pivot - 1)  # отсортировать разделенные
согласно границам массивы с помощью рекурсивного вызова функции
       randomized_quicksort(array, pivot + 1,
                            stop)  # отсортировать разделенные
согласно границам массивы с помощью рекурсивного вызова функции
   return array  # вернуть массив


def partition(array, start,
              stop):  # объявить функцию partition и дать на вход
список array и переменные start и stop - начало и конец списка
   b_array, c_start = array[start][
                           0], start  # переменным присвоить
соответсвенно начальный элемент списка и начальный индекс start
   for i in range(start + 1, stop + 1):  # объявить цикл
       if array[i][0] <= b_array:  # если срез массива меньше либо =
начальному, то
           c_start += 1  # начальный индекс увеличить на 1
           array[c_start], array[i] = array[i], array[
               c_start]  # поменять местами значения начального и
итого элементов списка
   array[start], array[c_start] = array[c_start], array[
       start]  # поменять местами значения начального и итого
элементов списка(поставить опорный элемент на свое законное место)
   return c_start  # вернуть значение переменной


def game(array):  # объявить функцию и дать на вход список
   if array == 1:  # если список равен 1, то
```

```
            return 'YES'  # вернуть YES
        for i in range(n):  # объявить цикл
            b = 0  # переменной присвоить 0
            c = 0  # переменной присвоить 0
            while c < len(a[list[i][0]]):  # пока переменная меньше длины
словаря,
                if abs(i - a[list[i][0]][c]) % array == 0:  # вернуть
абсолютное значение
                    b += 1  # переменную увеличить на 1
                    a[list[i][0]].pop(c)  #
                c += 1  # переменную увеличить на 1
            if (b == 0):  # если переменная равно 0, то
                return 'NO'  # вернуть NO
        return 'YES'  # вернуть YES


with open('input.txt') as f:  # открыть файл input
    n, k = map(int, f.readline().split())  # переменным присвоить
значения из первой строки файла типа integer
    list = [int(x) for x in f.readline().split()]  #объявить список

a = {}  # объявить словарь
for i in range(n):  # объявить цикл
    list[i] = [int(list[i]), i]  #итый элемент поменять на тип
integer
    a[list[i][0]] = a.get(list[i][0], [])  #вернуть значение для
указанного ключа
    a[list[i][0]].append(list[i][1])  #добавить в словарь итый
элемент списка

randomized_quicksort(list, 0, len(list) - 1)  # вызвать функцию

with open('output.txt', 'w') as f:  # открыть файл output
    f.write(game(k))  # в файл записать результат работы функции game
```

4. Точки и отрезки

```
import random#импортировать модуль random


def partition(array, start, stop):#объявить функцию partition и
дать на вход список array и переменные start и stop - начало и конец
списка
   b_array, c_start = array[start], start#переменным присвоить
соответсвенно начальный элемент списка и начальный индекс start
   for i in range(start + 1, stop + 1):#объявить цикл
       if array[i] <= b_array:#если итый элемент списка меньше либо
= начальному, то
           c_start += 1#начальный индекс увеличить на 1
           array[c_start], array[i] = array[i],
array[c_start]#поменять местами значения начального и итого
элементов списка
   array[start], array[c_start] = array[c_start],
array[start]#поменять местами значения начального и итого элементов
списка(поставить опорный элемент на свое законное место)
   return c_start#вернуть значение переменной


def randomized_quicksort(array, start, stop):#объявить функцию и
дать на вход список и переменные start и stop - начало и конец
списка
   if start < stop:#если переменная начала списка меньше переменной
конца, то
       key = random.randint(start, stop)#рандомно найти индекс
опорного элемента списка
       array[start], array[key] = array[key], array[start]#поменять
местами начальный и опроный элементы списка
       pivot = partition(array, start, stop)#вызвать функцию
partition и найти границу деления
       randomized_quicksort(array, start, pivot - 1)#отсортировать
разделенные согласно границам массивы с помощью рекурсивного вызова
функции
       randomized_quicksort(array, pivot + 1, stop)#отсортировать
разделенные согласно границам массивы с помощью рекурсивного вызова
функции
   return array#вернуть массив


with open('input.txt') as f:#открыть файл input
   s, p = map(int, f.readline().split())#переменным присвоить
значения из первой строки файла типа integer
   a_start, a_end = [], []#создать пустые списки
   for i in range(s):#объявить цикл
```

```
        ai, bi = map(int, f.readline().split())#переменным присвоить
значения из строки файла типа integer
        a_start.append(ai)#добавить переменную в список
        a_end.append(bi)#добавить переменную в список
    points = list(map(int, f.readline().split()))#переменной
присвоить значение

randomized_quicksort(a_start, 0, len(a_start) - 1)#вызвать функцию
randomized_quicksort(a_end, 0, len(a_end) - 1)#вызвать функцию
answer = ''#переменной присвоить "пробел"

for point in points:#объявить цикл
    count = 0#переменной присвоить 0
    for i in range(s):#объявить цикл
        if point >= a_start[i]:#если переменная больше или = итому
элементу, то
            count += 1#переменную увеличить на 1
        if point > a_end[i]:#если переменная больше итого элемента,
то
            count -= 1#переменную уменьшить на 1
    answer += str(count)#переменную увеличить на стринговое значение
переменной count
    answer += ' '#переменную увеличить на " "

with open('output.txt', 'w') as f:#открыть файл output
    f.write(answer)#записать в файл ответ
```