**Федеральное агентство связи**

**Ордена трудового красного знамени**

**Федеральное государственное образовательное**

**Бюджетное**

**Учреждение высшего профессионального образования**

**Московский Технический Университет связи и информатики**

**Кафедра информатики**

# Курсовая работа

по дисциплине «СиАОД»

Выполнил: студ. гр. БСТ1902

Потрываев А.Г

Вариант №15

Проверил: Кутейников И.А.

Москва 2021

## Содержание

## Контрольные задания №1

### Задача 1. «Треугольник с максимальным периметром»

Массив A состоит из целых положительных чисел длин отрезков. Составьте из трех отрезков такой треугольник, чтобы его периметр был максимально возможным. Если невозможно составить треугольник с положительной площадью функция возвращает 0.

```
arr = []
arraySize = input("Введите кол-во элементов массива: ")
for i in range(0, int(arraySize)):
    nextNum = input("Введите следующие число: ")
    arr.append(int(nextNum))


def heapify(arr, n, i):
    smallest = i
    l = 2 * i + 1
    r = 2 * i + 2

    if l < n and arr[i] > arr[l]:
        smallest = l

    if r < n and arr[smallest] > arr[r]:
        smallest = r

    if smallest != i:
```

```
        arr[i],arr[smallest] = arr[smallest],arr[i]

        heapify(arr, n, smallest)


def HeapSort(array):
    arr=array.copy()
    n = len(arr)

    for j in range(n, -1, -1):
        heapify(arr, n, j)

    for j in range(n-1, 0, -1):
        arr[j], arr[0] = arr[0], arr[j]
        heapify(arr, j, 0)
    return arr


import math

def isInt(value):
    try:
        int(value)
        return True
    except ValueError:
        return False


def Trinagle(a,b,c):
    p= (a+b+c)/2
    if((p*(p-a)*(p-b)*(p-c))<=0):
        return 0
    if(isInt((p*(p-a)*(p-b)*(p-c))**(-2))):
        return (a+b+c)
    else:
        return 0

def FindTrinagle(arr):
    for i in range(0,len(arr)):
        for j in range(i+1,len(arr)):
            for k in range(j+1,len(arr)):
                if(Trinagle(arr[i],arr[j],arr[k])>0):
                    return Trinagle(arr[i],arr[j],arr[k])
    return 0
```

### Задача 2. «Максимальное число»

Дан массив неотрицательных целых чисел nums. Расположите их в таком порядке, чтобы вместе они образовали максимально возможное число. Замечание: Результат может быть очень большим числом, поэтому представьте его как string, а не integer.

```
def isBiger(first_num, second_num):
    return str(first_num)+str(second_num)>str(second_num)+str(first_num)

def heapify_2(arr, n, i):
    largest = i
    l = 2 * i + 1
    r = 2 * i + 2

    if l < n and isBiger(arr[i], arr[l]):
        largest = l

    if r < n and isBiger(arr[largest], arr[r]):
        largest = r

    if largest != i:
        arr[i],arr[largest] = arr[largest],arr[i]

        heapify_2(arr, n, largest)


def HeapSort_2(array):
    arr=array.copy()
    n = len(arr)
    for j in range(n, -1, -1):
        heapify_2(arr, n, j)

    for j in range(n-1, 0, -1):
        arr[j], arr[0] = arr[0], arr[j]
        heapify_2(arr, j, 0)
    return arr


array = []
arraySize = input("Введите кол-во элементов массива: ")
for i in range(0, int(arraySize)):
    nextNum = input("Введите следующие число: ")
    array.append(int(nextNum))
print(array)
array = HeapSort_2(array)
string = ""
```

```
for i in range (0, len(array)):
    string += str(array[i])
print (string)
```

### Задача 3. «Сортировка диагоналей в матрице»

Дана матрица mat размером m * n, значения целочисленные. Напишите функцию, сортирующую каждую диагональ матрицы по возрастанию и возвращающую получившуюся матрицу.

```
m=5
n=5
min_limit=-250
max_limit=1015

strM=input("\n\nВведите m: ")
if len(strM)>0: m=int(strM)

strN=input("Введите n: ")
if len(strN)>0: n=int(strN)

while True:
    strMin_limit=input("Введите минимальное значение: ")
    if len(strMin_limit)>0: min_limit=int(strMin_limit)

    strMax_limit=input("Введите максимальное значение: ")
    if len(strMax_limit)>0: max_limit=int(strMax_limit)

    if (min_limit<=max_limit): break
    else: print("\nМинимальный элемент не может быть больше максимального, повтор
ите попытку\n")


import random
def randomize(min_limit = -250, max_limit = 1013, n = 5, m = 5):

    matrix = [[random.randrange(min_limit, max_limit) for y in range(m)] for x in
 range(n)]
    return matrix


def MatrixSort(maxtrix):
    for x in range(0, n):
        iterator = 0
        while(x+iterator<n and iterator<m):
            j = iterator
            while(x+j<n and j<m):
                if(matrix[j+x][j]>matrix[x+iterator][iterator]):
```

```
                    matrix[j+x][j], matrix[x+iterator][iterator] = matrix[x+itera
tor][iterator], matrix[j+x][j]
                j += 1
            iterator += 1


    for x in range(m, 0, -1):
        iterator = 0
        while(iterator<n and x+iterator<m):
            j=iterator
            while(x+j<m and j<n):
                if(matrix[j][x+j]>matrix[iterator][x+iterator]):
                    matrix[j][x+j], matrix[iterator][x+iterator] = matrix[iterato
r][x+iterator], matrix[j][x+j]
                j += 1
            iterator += 1



def see(matrix):
    for i in range (0, n):
        print("\n")
        s = ""
        for j in range (0, m):
            s += str(matrix[i][j])+ "\t"
        print(s)


matrix = randomize(min_limit, max_limit, n, m)
see(matrix)
MatrixSort(matrix)
print("\n")
see(matrix)
```

**Контрольные задачи на строки**

**Задача 1.**

Даны две строки: s1 и s2 с одинаковым размером, проверьте, может ли некоторая перестановка строки s1 “победить” некоторую перестановку строки s2 или наоборот. Строка x может “победить” строку y (обе имеют размер n), если x[i]> = y [i] (в алфавитном порядке) для всех i от 0 до n-1.

fun longestPalindrome(s: String): String? {

```
        if (s.isEmpty()) return ""
        var start = 0
        var end = 0
        for (i in 0 until s.length) {
            val len1 = expandAroundCenter(s, i, i)
            val len2 = expandAroundCenter(s, i, i + 1)
            val len = Math.max(len1, len2)
            if (len > end - start) {
                start = i - (len - 1) / 2
                end = i + len / 2
            }
        }
        return s.substring(start, end + 1)
    }
```

**Задача 2.**

Дана строка s, вернуть самую длинную полиндромную подстроку в s.

```
fun longestPalindrome(s: String): String? {
    if (s.isEmpty()) return ""
    var start = 0
    var end = 0
    for (i in 0 until s.length) {
        val len1 = expandAroundCenter(s, i, i)
        val len2 = expandAroundCenter(s, i, i + 1)
        val len = Math.max(len1, len2)
        if (len > end - start) {
            start = i - (len - 1) / 2
            end = i + len / 2
        }
```

```
    }
    return s.substring(start, end + 1)
}

fun expandAroundCenter(s: String, left: Int, right: Int): Int {
    var L = left
    var R = right
    while (L >= 0 && R < s.length && s[L] == s[R]) {
        L--
        R++
    }
    return R - L - 1
}
```

**Задача 3.**

Вернуть количество отдельных непустых подстрок текста, которые могут быть записаны как конкатенация некоторой строки с самой собой (т.е. она может быть записана, как a + a, где a - некоторая строка).

```
fun concetCout(s: String): Int {
    var j: Int
    var cout = 0
    for (i in s.indices) {
        j = 1
        while (i + 2 * j < s.length) {
            if (s.substring(i, i + j) == s.substring(i + j, i + 2 * j))
                cout++
            j++
        }
```

```
    }
    return cout
}
```

**Контрольные задачи №2**

**Задача 1. «Объединение отрезков»**

Дан массив отрезков intervals, в котором intervals[i] = [start i , end i ], некоторые отрезки могут пересекаться. Напишите функцию, которая объединяет все пересекающиеся отрезки в один и возвращает новый массив непересекающихся отрезков.

```
fun problem_intervals(intervals: Array<IntArray>): Array<IntArray>? {
    Arrays.sort(intervals) { a: IntArray, b: IntArray -> a[0].compareTo(b[0]) }
    val merged = LinkedList<IntArray>()
    for (interval in intervals) {
        if (merged.isEmpty() || merged.last[1] < interval[0]) {
            merged.add(interval)
        } else {
            merged.last[1] = Math.max(merged.last[1], interval[1])
        }
    }
    return merged.toTypedArray()
}
```

**Задача 2. «Стопки монет»**

На столе стоят 3n стопок монет. Вы и ваши друзья Алиса и Боб забираете стопки монет по следующему алгоритму: 1. Вы выбираете 3 стопки монет из оставшихся на столе. 2. Алиса забирает себе стопку с максимальным количеством монет. 3. Вы забираете одну из двух оставшихся стопок. 4. Боб забирает последнюю стопку. 5. Если еще остались стопки, то действия повторяются с первого шага. Дан массив целых положительных чисел piles.

Напишите функцию, возвращающую максимальное число монет, которое вы можете получить.

```
fun problem_coins(piles: IntArray): Int {
    Arrays.sort(piles)
    var i = piles.size - 2
    var j = 0
    var count = 0
    while (j++ < piles.size / 3) {
        count += piles[i]
        i -= 2
    }
    return count
}
```

### Задача 3. «Шарики и стрелы»

Некоторые сферические шарики распределены по двухмерному пространству. Для каждого шарика даны хкоординаты начала и конца его горизонтального диаметра. Так как пространство двумерно, то укоординаты не имеют значения в данной задаче. Координата xstart всегда меньше xend. Стрелу можно выстрелить строго вертикально (вдоль уоси) из разных точек хоси. Шарик с координатами xstart и xend уничтожается стрелой, если она была выпущена из такой позиции x, что xstart ⩽ x ⩽ xend. Когда стрела выпущена, она летит в пространстве бесконечное время (уничтожая все шарики на пути). Дан массив points, где points[i] = [xstart, xend]. Напишите функцию, возвращающую минимальное количество стрел, которые нужно выпустить, чтобы уничтожить все шарики.

```
fun probmes_baloons(points: Array<IntArray>): Int {
    Arrays.sort(points) { a: IntArray, b: IntArray -> a[1].compareTo(b[1]) }
    var end = points[0][1]
```

```
    var arrow = 1
    for (i in points.indices) {
        if (points[i][0] > end) {
            arrow++
            end = points[i][1]
        }
    }
    return arrow
}
```

Вывод: В результате курсовой работы были решены 9 задач