Министерство цифрового развития, связи и массовых коммуникаций

Ордена Трудового Красного Знамени

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Московский технический университет связи и информатики»



Отчет по курсовой работе

по дисциплине «Структура и алгоритмы обработки данных»

Выполнил: студент группы

БВТ1902

Подпоркин В.С.

Москва

2021 г

**Оглавление**

[Цель работы 3](#__RefHeading___Toc216_4292971786)

[Задание 1 «Треугольник с максимальным периметром» 3](#__RefHeading___Toc218_4292971786)

[Код программы 3](#__RefHeading___Toc220_4292971786)

[Задание 2 «Максимальное число» 3](#__RefHeading___Toc222_4292971786)

[Код программы 3](#__RefHeading___Toc224_4292971786)

[Задание 3 «Сортировка диагоналей в матрице» 3](#__RefHeading___Toc226_4292971786)

[Код программы 3](#__RefHeading___Toc228_4292971786)

[Задание 4 «Шарики и стрелы» 3](#__RefHeading___Toc230_4292971786)

[Код программы 3](#__RefHeading___Toc232_4292971786)

[Задание 5 «Стопки монет» 3](#__RefHeading___Toc234_4292971786)

[Код программы 3](#__RefHeading___Toc236_4292971786)

[Задание 6 «Победная строка» 3](#__RefHeading___Toc238_4292971786)

[Код программы 3](#__RefHeading___Toc240_4292971786)

[Задание 7 «Палиндром» 3](#__RefHeading___Toc242_4292971786)

[Код программы 4](#__RefHeading___Toc244_4292971786)

[Задание 8 4](#__RefHeading___Toc246_4292971786)

[Код программы 4](#__RefHeading___Toc248_4292971786)

[Снимки экрана работы программ 4](#__RefHeading___Toc250_4292971786)

[Вывод 5](#__RefHeading___Toc252_4292971786)

## Цель работы

Написать программу для решения ниже поставленных задач.

## Задание 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.

## Код программы

fn tri\_exists(a: i32, b: i32, c: i32) -> bool {  
 let max = a.max(b).max(c);  
 let min = a.min(b).min(c);  
 let mid = a + b + c - max - min;  
 max < (min + mid)  
}  
  
fn max\_p(arr: &mut [i32]) -> Option<(i32, i32, i32)> {  
  
 arr.sort();  
  
 for i in (2..arr.len()).rev() {  
 let a = arr[i];  
 let b = arr[i - 1];  
 let c = arr[i - 2];  
 if tri\_exists(a, b, c) {  
 return *Some*((a, b, c));  
 }  
 }  
 *None*  
}

## Задание 2 «Максимальное число»

Дан массив неотрицательных целых чисел nums. Расположите их в таком порядке, чтобы вместе они образовали максимально возможное число.

## Код программы

pub fn make\_max\_mum(nums: &mut [i32]) -> String {  
 let mut str = String::*new*();  
 nums.sort\_by(|a, b| Ord::cmp(&concat(\*b, \*a), &concat(\*a, \*b)));  
 for x in nums {  
 *// В случае, если к нулю добавляется число, очищаем строку (убираем незначащий ноль)*  
if str == "0" { str.truncate(0); }  
 write!(&mut str, "{}", x).unwrap(); *// добавление числа к не нулевой или пустой строке*  
}  
 str  
}  
  
*// Конкатинирует 2 числа как если бы это были строки*  
fn concat(n1: i32, n2: i32) -> i64 {  
 let (n1, n2) = (n1 as i64, n2 as i64);  
 let offset = pow10ceil(n2);*// показывает насколько нужно умножить первое число, чтобы все они заменились значением из второго*  
 n1\*offset + n2 *// return n1\*offset + n2*  
}  
  
*// Округляет по степени 10 вверх (0 -> 10; 2 -> 10; 10 -> 100; 24 -> 100)*  
fn pow10ceil(mut n: i64) -> i64 {  
 let mut pow = 10;  
 while n >= 10 {  
 n /= 10;  
 pow \*= 10;  
 }  
 pow *// return pow*  
}

## Задание 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.

## Код программы

fn sort\_matrix(mat: &mut Array2<i32>) {  
 let m = mat.shape()[0];  
 let n = mat.shape()[1];  
  
 for d\_i in 0..(m + n - 1) {  
 let (start\_x, start\_y) = if d\_i < n { (0, n - d\_i - 1) } else { (d\_i - n, 0) };  
  
 let mut arr = Vec::*new*();  
  
 for offset in 0.. {  
 let (x, y) = (start\_x + offset, start\_y + offset);  
 if x >= m || y >= n { break }  
 arr.push(mat[(x, y)]);  
 }  
  
 arr.sort\_by(|a, b| b.cmp(a));  
  
 for offset in 0.. {  
 let (x, y) = (start\_x + offset, start\_y + offset);  
 if x >= m || y >= n { break }  
 mat[(x, y)] = arr.pop().unwrap();  
 }  
 }  
}

## Задание 4 «Шарики и стрелы»

Дан массив points, где points[i] = [xstart, xend]. Напишите функцию, возвращающую минимальное количество стрел, которые нужно выпустить, чтобы уничтожить все шарики.

## Код программы

fn merge\_ranges(r1: [i32; 2], r2: [i32; 2]) -> Option<[i32; 2]> {  
 let left = r1[0].max(r2[0]);  
 let right = r1[1].min(r2[1]);  
 if left <= right {  
 *Some*([left, right])  
 } else {  
 *None*  
}  
}  
  
fn balls(r: &[[i32; 2]]) -> usize {  
 let mut source = Vec::*from*(r);  
 let mut changed = true;  
 *'root*:  
 while changed {  
 changed = false;  
 for i in 0..(source.len() - 1) {  
 let mut has\_pair = false;  
 for j in (i + 1)..source.len() {  
 let r1 = source[i];  
 let r2 = source[j];  
 match merge\_ranges(r1, r2) {  
 *Some*(r) => {  
 source.remove(j);  
 source.remove(i);  
 source.push(r);  
 changed = true;  
 has\_pair = true;  
 continue *'root*;  
 }  
 *None* => (),  
 }  
 }  
 }  
 }  
 source.len()  
}

## Задание 5 «Стопки монет»

Дан массив целых положительных чисел piles. Напишите функцию, возвращающую максимальное число монет, которое вы можете получить.

## Код программы

fn bob\_alice(s: &[i32]) -> i32 {  
 let mut sum = 0;  
 let mut sorted = Vec::*from*(s);  
 sorted.sort();  
  
 let for\_me\_and\_alice = &sorted[(sorted.len()/3)..];  
 for i in 0..(for\_me\_and\_alice.len()/2) {  
 sum += for\_me\_and\_alice[i\*2];  
 }  
 sum  
}

## Задание 6 «Победная строка»

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.

## Код программы

fn str2\_winner(str1: &str, str2: &str) -> bool {  
 let mut s1: Vec<\_> = str1.chars().collect();  
 let mut s2: Vec<\_> = str2.chars().collect();  
 s1.sort();  
 s2.sort();  
 Iterator::zip(s1.iter(), s2.iter()).all(|(a, b)| \*a >= \*b)  
 || Iterator::zip(s1.iter(), s2.iter()).all(|(a, b)| \*a <= \*b)  
}

## Задание 7 «Палиндром»

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

## Код программы

fn is\_pol(s: &str) -> bool {  
 let rev = s.chars().rev().collect::<String>();  
 s == rev  
}  
  
fn str2\_longer\_pol(s: &str) -> &str {  
 let mut longer = &s[0..1];  
 for i in 0..(s.len()-1) {  
 for j in (i+1)..s.len() {  
 let c = &s[i..j];  
 if c.len() > longer.len() && is\_pol(c) {  
 longer = c;  
 }  
 }  
 }  
 longer  
}

## Задание 8

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).

## Код программы

fn str2\_concat\_c(s: &str) -> usize {  
 let mut count = 0;  
 for i in 0..(s.len()-1) {  
 for j in (i+1)..s.len() {  
 let c = &s[i..j];  
 if c.len() % 2 == 0 {  
 if c[0..(c.len()/2)] == c[(c.len()/2)..c.len()] {  
 count += 1;  
 }  
 }  
 }  
 }  
 count  
}

## Снимки экрана работы программ

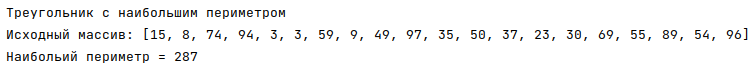
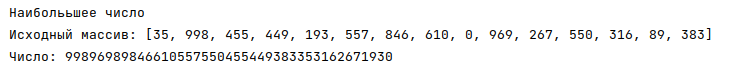
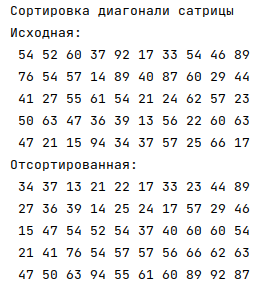
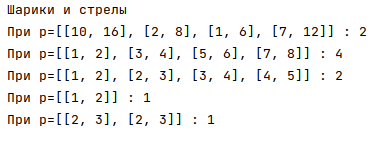
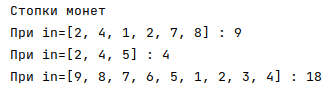


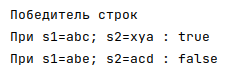
Рисунок 1 – Результат выполнения 1

Рисунок 2 – Результат выполнения 2

Рисунок 3 – Результат выполнения 3

Рисунок 4 – Результат выполнения 4

Рисунок 5 – Результат выполнения 5

Рисунок 6 – Результат выполнения 6

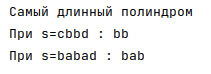
Рисунок 7 – Результат выполнения 7

Рисунок 8 – Результат выполнения 8

## Вывод

Я реализовал алгоритмы для решения выше поставленных задач.