Санкт-Петербургский Национальный Исследовательский Университет Информационных Технологий, Механики и Оптики

Задачи с платформы Stepik

по дисциплине

«Алгоритмы и структуры данных»

Выполнил: Студент группы P3217

Сергачев Данила Дмитриевич

Преподаватели:

Романов Алексей Андреевич и

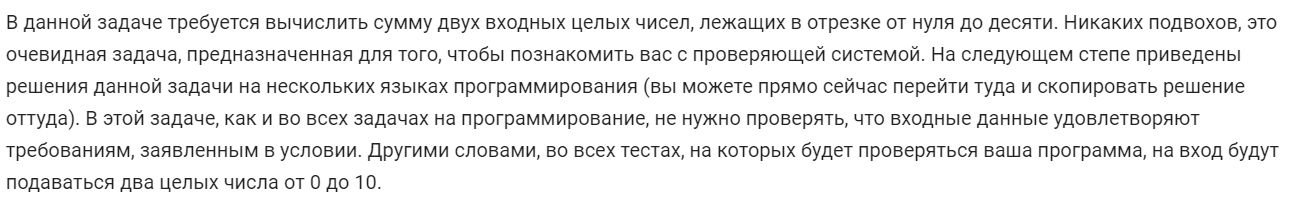
Волчек Дмитрий Геннадьевич

Санкт-Петербург

2019 г

**Задание №1**

**Пример задачи на программирование**



**Решение**

#include <iostream>

using namespace std;

int main()

{

int a,b;

cin >> a;

cin >> b;

int sum = a + b;

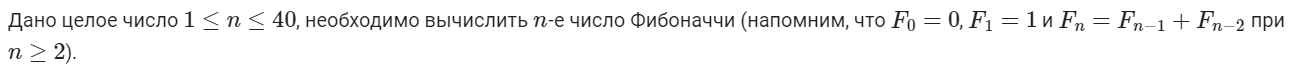
cout << sum;

return 0;

}

**Задание №2**

## небольшое число Фибоначчи



**Решение**

using System;

class Program

{

static void Main()

{

int n = int.Parse(Console.ReadLine());

Int64[] febonache = new Int64[n+1];

febonache[0] = 0; febonache[1] = 1;

for (int i = 2; i <= n; i++) febonache[i] = febonache[i - 1] + febonache[i - 2];

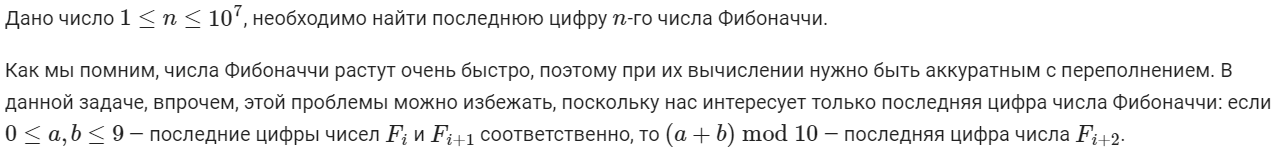
Console.WriteLine(febonache[n]);

}

}

**Задание №3**

## ****последняя цифра большого числа Фибоначчи****



**Решение**

import java.io.\*;

class Main {

private static final int MOD = (int)(10);

public static void main(String[] args) throws IOException {

BufferedReader br = new BufferedReader(new InputStreamReader(System.in), 8192);

int n = Integer.parseInt(br.readLine());

System.out.print(fibonacci(n));

}

private static int fibonacci(int n){

int a = 0;

int b = 1;

for(int i =0; i < n; i++){

int c = (a + b ) % MOD;

a = b;

b = c;

}

return a;

}

}

**Задание №4**

## ****огромное число Фибоначчи по модулю****

## 

**Решение**

package stepik.overview;

import java.io.\*;

import java.util.ArrayList;

public class Main {

public static void main(String[] args) throws IOException {

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

String[] line = br.readLine().split(" ");

long time = System.currentTimeMillis();

long n = Long.valueOf(line[0]);

int m = Integer.valueOf(line[1]);

OutputStream os = new BufferedOutputStream(System.out);

os.write(getFibonacciRest(n, m).toString().getBytes());

os.flush();

}

private static Long getFibonacciRest(long n, long m) {

ArrayList<Long> s = getSequencePeriod(m);

long period = s.size() - 2;

int val = (int) (n % period);

return s.get(val);

}

private static ArrayList<Long> getSequencePeriod(long m) {

ArrayList<Long> s = new ArrayList();

s.add(0l);

s.add(1l);

for (int i = 2; i < m \* 6; i++) {

s.add((s.get(i - 1) + s.get(i - 2)) % m);

if (s.get(i) == 1 && s.get(i - 1) == 0) {

break;

}

}

return s;

}

}

**Задание №5**

## наибольший общий делитель

## 

**Решение**

#include <iostream>

#include <cassert>

using

namespace std;

int EuclidGCD(int a, int b)

{

assert(a > 0 && b > 0);

while (a > 0 && b > 0)

{

if (a > b)

a %= b;

else

b %=a;

}

return a == 0 ? b : a;

}

int main(void)

{

int a,b;

cin >> a >> b;

cout << EuclidGCD(a,b) << endl;

return 0;

}

**Задание №6**

## непрерывный рюкзак

## 

**Решение**

package stepik.greedy\_algorithm;

import java.io.FileNotFoundException;

import java.util.Arrays;

import java.util.Scanner;

public class Task2 {

class Item implements Comparable<Item> {

int cost;

int weight;

public Item(int cost, int weight){

this.cost = cost;

this.weight = weight;

}

@Override

public int compareTo(Item o) {

double r1 = (double)cost / weight;

double r2 = (double)o.cost / o.weight;

return -Double.compare(r1, r2);

}

}

private void run() throws FileNotFoundException{

Scanner input = new Scanner(System.in);

int n = input.nextInt();

int W = input.nextInt();

Item[] items = new Item[n];

for(int i =0; i < n; i++){

items[i] = new Item(input.nextInt(),input.nextInt());

}

Arrays.sort(items);

double res = 0;

for(Item item : items){

if(item.weight <= W){

res += item.cost;

W -= item.weight;

} else{

res += (double)item.cost \* W / item.weight;

break;

}

}

System.out.println(res);

}

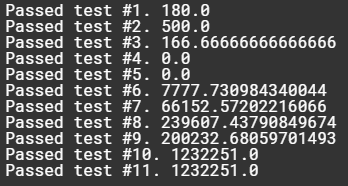
public static void main(String[] args) throws FileNotFoundException {

new Task2().run();

}

}

**Результат работы**



**Задание №7**

## различные слагаемые

## 

**Решение**

import java.io.\*;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.Map;

public class Main{

public static void main(String[] args) throws IOException {

try(BufferedReader br = new BufferedReader(new InputStreamReader(System.in), 8192);

OutputStream os = new BufferedOutputStream(System.out)){

int n = Integer.valueOf(br.readLine());

ArrayList<Integer> addends = new ArrayList<>(1024);

int i = 1;

while (true) {

if (i \* 2 < n) {

addends.add(i);

n = n - i;

} else {

addends.add(n);

break;

}

i++;

}

os.write((addends.size() + "\n").getBytes());

for (int k = 0; k < addends.size(); k++) {

os.write((addends.get(k) + " ").getBytes());

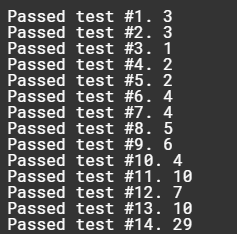
}

}

}

}

**Результат работы**



**Задание №8**

## декодирование Хаффмана

## 

**Решение**

import java.io.\*;

import java.util.HashMap;

import java.util.Map;

public class Main {

public static void main(String[] args) throws IOException {

BufferedReader br = new BufferedReader(new InputStreamReader(System.in), 8192);

OutputStream os = new BufferedOutputStream(System.out);

String[] metaData = br.readLine().split(" ");

int k = Integer.valueOf(metaData[0]);

int l = Integer.valueOf(metaData[1]);

Map<String, String> letterMap = new HashMap<>();

String[] line;

for (int i = 0; i < k; i++) {

line = br.readLine().split(": ");

letterMap.put(line[1], line[0]);

}

String encodedMessage = br.readLine();

StringBuilder decodedMessageBuilder = new StringBuilder(1024);

int i = 0;

while (i < l) {

int j = i + 1;

while (!letterMap.containsKey(encodedMessage.substring(i, j))) {

j++;

}

decodedMessageBuilder.append(letterMap.get(encodedMessage.substring(i, j)));

i = j;

}

os.write(decodedMessageBuilder.toString().getBytes());

os.flush();

os.close();

}

}

**Результат работы**



**Задание №9**

## ****двоичный поиск****

## 

**Решение**

import java.io.\*;

import java.util.Arrays;

public class Main {

public static void main(String[] args) throws IOException {

BufferedReader br = new BufferedReader(new InputStreamReader(System.in), 8192);

OutputStream os = new BufferedOutputStream(System.out);

int[] a = Arrays.stream(br.readLine().split(" ")).mapToInt(s -> Integer.parseInt(s)).toArray();

int[] k = Arrays.stream(br.readLine().split(" ")).mapToInt(s -> Integer.parseInt(s)).toArray();

for (int i = 1; i < k.length; i++) {

os.write((binarySearch(k[i], a) + " ").getBytes());

}

br.close();

os.flush();

os.close();

}

public static int binarySearch(int x, int[] arr) {

int ind = binarySearchLeftIterative(x, arr);

if (ind == arr.length || arr[ind] != x) {

return -1;

} else {

return ind;

}

}

public static int binarySearchLeftIterative(int x, int[] arr) {

int l = 1;

int r = arr.length;

while (l < r) {

int mid = l + (r - l) / 2;

if (arr[mid] < x) {

l = mid + 1;

} else {

r = mid;

}

}

return r;

}

}

**Задание №10**

## ****число инверсий****

## 

**Решение**

import java.io.\*;

import java.util.Scanner;

public class Main {

private static long inversionsNumber = 0;

public static void main(String[] args) throws IOException {

BufferedReader br = new BufferedReader(new InputStreamReader(System.in), 8192);

OutputStream os = new BufferedOutputStream(System.out);

Scanner sc = new Scanner(System.in);

int n = sc.nextInt();

int a[] = new int[n];

for (int i = 0; i < n; i++) {

a[i] = sc.nextInt();

}

mergeSort(a, 0, a.length);

os.write((inversionsNumber + "\n").getBytes());

br.close();

os.flush();

os.close();

}

// l1 < l2; r1 < r2

public static void merge(int[] array, int l1, int r1, int l2, int r2) throws IOException {

int i = 0;

int size = r2 - l1;

int destStart = l1;

int[] tmp = new int[size];

while (i < size) {

while (l1 < r1 && (l2 >= r2 || array[l1] <= array[l2])) { // stable

tmp[i++] = array[l1];

l1++;

}

while (l2 < r2 && (l1 >= r1 || array[l2] < array[l1])) {

tmp[i++] = array[l2];

inversionsNumber += r1 - l1;

l2++;

}

}

for (int k = 0; k < size; k++) {

array[destStart + k] = tmp[k];

}

}

public static void mergeSort(int[] array, int l, int r) throws IOException {

if (l == r - 1) {

return;

}

mergeSort(array, l, l + (r - l) / 2);

mergeSort(array, l + (r - l) / 2, r);

merge(array, l, l + (r - l) / 2, l + (r - l) / 2, r);

}

}

**Задание №11**

## ****сортировка подсчётом****

## 

**Решение**

import java.io.\*;

import java.util.Arrays;

import java.util.Scanner;

public class Main {

public static void main(String[] args) throws IOException {

BufferedReader br = new BufferedReader(new InputStreamReader(System.in), 8192);

OutputStream os = new BufferedOutputStream(System.out);

Scanner sc = new Scanner(System.in);

int n = sc.nextInt();

int[] count = new int[11];

for (int i = 0; i < n; i++) {

count[sc.nextInt()]++;

}

for (int i = 0; i < count.length; i++) {

for (int k = 0; k < count[i]; k++) {

os.write((i + " ").getBytes());

}

}

br.close();

os.flush();

os.close();

}

}

**Задание №12**

## наибольшая последовательнократная подпоследовательность

## 

**Решение**

import java.io.\*;

import java.util.Arrays;

public class Main {

public static void main(String[] args) throws IOException {

BufferedReader br = new BufferedReader(new InputStreamReader(System.in), 8192);

OutputStream os = new BufferedOutputStream(System.out);

int n = Integer.valueOf(br.readLine());

int[] a = Arrays.stream(br.readLine().split(" ")).mapToInt(s -> Integer.parseInt(s)).toArray();

int[] d = new int[n];

for (int i = 0; i < n; i++) {

d[i] = 1;

for (int j = 0; j < i; j++) {

if (a[i] % a[j] == 0) {

d[i] = Integer.max(d[i], 1 + d[j]);

}

}

}

os.write((String.valueOf(Arrays.stream(d).max().getAsInt()) + "\n").getBytes());

br.close();

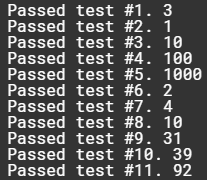
os.flush();

os.close();

}

}

**Результат работы**



**Задание №13**

## наибольшая невозрастающая подпоследовательность

## 

**Решение**

import java.io.\*;

import java.util.Arrays;

public class Main {

public static void main(String[] args) throws IOException {

BufferedReader br = new BufferedReader(new InputStreamReader(System.in), 8192);

OutputStream os = new BufferedOutputStream(System.out);

int n = Integer.valueOf(br.readLine());

int[] a = Arrays.stream(br.readLine().split(" ")).mapToInt(s -> Integer.parseInt(s)).toArray();

int[] tail = new int[n];

int[] previous = new int[n];

int sequenceLength = 0;

for (int i = 0; i < n; i++) {

int pos = binarySearchRight(a, tail, sequenceLength, a[i]);

if (pos == sequenceLength) {

sequenceLength++;

}

previous[i] = pos > 0 ? tail[pos - 1] : -1;

tail[pos] = i;

}

os.write((String.valueOf(sequenceLength) + "\n").getBytes());

int[] result = new int[sequenceLength];

for (int i = tail[sequenceLength - 1]; i >= 0; i = previous[i]) {

result[--sequenceLength] = i + 1;

}

for (int i = 0; i < result.length; i++) {

os.write(String.valueOf(result[i] + " ").getBytes());

}

br.close();

os.flush();

os.close();

}

static int binarySearchRight(int[] a, int[] tail, int sequenceLength, int key) {

int l = -1;

int r = sequenceLength;

while (l < r - 1) {

int mid = (l + r) >>> 1;

if (a[tail[mid]] >= key) {

l = mid;

} else {

r = mid;

}

}

return r;

}

}

**Результат работы**

