

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

Задачи с платформы Stepik
по дисциплине
«Алгоритмы и структуры данных»

Выполнил: Студент группы Р3217

Сергачев Данила Дмитриевич

Преподаватели:

Романов Алексей Андреевич и

Волчек Дмитрий Геннадьевич

Санкт-Петербург

2019 г

Задание №1

Пример задачи на программирование

В данной задаче требуется вычислить сумму двух входных целых чисел, лежащих в отрезке от нуля до десяти. Никаких подвохов, это очевидная задача, предназначенная для того, чтобы познакомить вас с проверяющей системой. На следующем шаге приведены решения данной задачи на нескольких языках программирования (вы можете прямо сейчас перейти туда и скопировать решение оттуда). В этой задаче, как и во всех задачах на программирование, не нужно проверять, что входные данные удовлетворяют требованиям, заявленным в условии. Другими словами, во всех тестах, на которых будет проверяться ваша программа, на вход будут подаваться два целых числа от 0 до 10.

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

Дано целое число $1 \leq n \leq 40$, необходимо вычислить n -е число Фибоначчи (напомним, что $F_0 = 0$, $F_1 = 1$ и $F_n = F_{n-1} + F_{n-2}$ при $n \geq 2$).

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

Дано число $1 \leq n \leq 10^7$, необходимо найти последнюю цифру n -го числа Фибоначчи.

Как мы помним, числа Фибоначчи растут очень быстро, поэтому при их вычислении нужно быть аккуратным с переполнением. В данной задаче, впрочем, этой проблемы можно избежать, поскольку нас интересует только последняя цифра числа Фибоначчи: если $0 \leq a, b \leq 9$ — последние цифры чисел F_i и F_{i+1} соответственно, то $(a + b) \bmod 10$ — последняя цифра числа F_{i+2} .

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

Даны целые числа $1 \leq n \leq 10^{18}$ и $2 \leq m \leq 10^5$, необходимо найти остаток от деления n -го числа Фибоначчи на m .

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

По данным двум числам $1 \leq a, b \leq 2 \cdot 10^9$ найдите их 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

Первая строка содержит количество предметов $1 \leq n \leq 10^3$ и вместимость рюкзака $0 \leq W \leq 2 \cdot 10^6$. Каждая из следующих n строк задаёт стоимость $0 \leq c_i \leq 2 \cdot 10^6$ и объём $0 < w_i \leq 2 \cdot 10^6$ предмета (n, W, c_i, w_i — целые числа). Выведите максимальную стоимость частей предметов (от каждого предмета можно отделить любую часть, стоимость и объём при этом пропорционально уменьшатся), помещающихся в данный рюкзак, с точностью не менее трёх знаков после запятой.

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

```

Passed test #1. 180.0
Passed test #2. 500.0
Passed test #3. 166.66666666666666
Passed test #4. 0.0
Passed test #5. 0.0
Passed test #6. 7777.730984340044
Passed test #7. 66152.57202216066
Passed test #8. 239607.43790849674
Passed test #9. 200232.68059701493
Passed test #10. 1232251.0
Passed test #11. 1232251.0

```

Задание №7

различные слагаемые

По данному числу $1 \leq n \leq 10^9$ найдите максимальное число k , для которого n можно представить как сумму k различных натуральных слагаемых. Выведите в первой строке число k , во второй — k слагаемых.

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) {

```

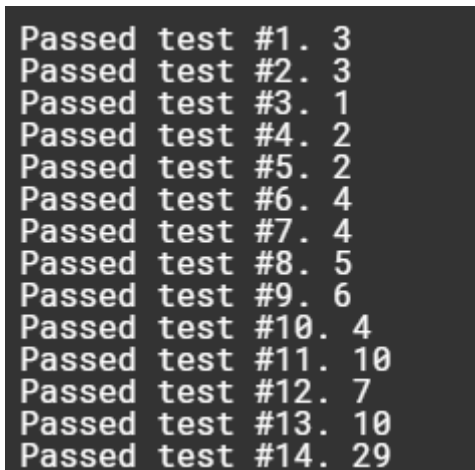
```

        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



```

Passed test #1. 3
Passed test #2. 3
Passed test #3. 1
Passed test #4. 2
Passed test #5. 2
Passed test #6. 4
Passed test #7. 4
Passed test #8. 5
Passed test #9. 6
Passed test #10. 4
Passed test #11. 10
Passed test #12. 7
Passed test #13. 10
Passed test #14. 29

```

Задание №8

декодирование Хаффмана

Восстановите строку по её коду и беспрефиксному коду символов.

В первой строке входного файла заданы два целых числа k и l через пробел — количество различных букв, встречающихся в строке, и размер получившейся закодированной строки, соответственно. В следующих k строках записаны коды букв в формате "letter: code". Ни один код не является префиксом другого. Буквы могут быть перечислены в любом порядке. В качестве букв могут встречаться лишь строчные буквы латинского алфавита; каждая из этих букв встречается в строке хотя бы один раз. Наконец, в последней строке записана закодированная строка. Исходная строка и коды всех букв непусты. Заданный код таков, что закодированная строка имеет минимальный возможный размер.

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

```

Passed test #1. a
Passed test #2. abacabad
Passed test #3. aa
Passed test #4. bbb
Passed test #5. zzzzzyyyyy
Passed test #6. zzzzztyyyyy
Passed test #7. bcbdbcb
Passed test #8. bcbdbcbcbcbdbcb
Passed test #9. accepted
Passed test #10. wronganswer
Passed test #11. abcdefghijklmnopqrstuvwxyz
Passed test #12. abcdefghijklmnopqrstuvwxyzzyxw
Passed test #13. abbcccddeeeefffffggggggghh
Passed test #14. kkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkk
Passed test #15. klkklkklkklkklkklkklkklkklkklkklkklk
Passed test #16. qpmsqwnupropwqxutpnvonxtvpvqxp

```


Задание №9

ДВОИЧНЫЙ ПОИСК

В первой строке даны целое число $1 \leq n \leq 10^5$ и массив $A[1 \dots n]$ из n различных натуральных чисел, не превышающих 10^9 , в порядке возрастания, во второй — целое число $1 \leq k \leq 10^5$ и k натуральных чисел b_1, \dots, b_k , не превышающих 10^9 . Для каждого i от 1 до k необходимо вывести индекс $1 \leq j \leq n$, для которого $A[j] = b_i$, или -1 , если такого j нет.

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

Первая строка содержит число $1 \leq n \leq 10^5$, вторая — массив $A[1 \dots n]$, содержащий натуральные числа, не превосходящие 10^9 . Необходимо посчитать число пар индексов $1 \leq i < j \leq n$, для которых $A[i] > A[j]$. (Такая пара элементов называется инверсией массива. Количество инверсий в массиве является в некотором смысле его мерой неупорядоченности: например, в упорядоченном по неубыванию массиве инверсий нет вообще, а в массиве, упорядоченном по убыванию, инверсию образуют каждые два элемента.)

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

Первая строка содержит число $1 \leq n \leq 10^4$, вторая — n натуральных чисел, не превышающих 10. Выведите упорядоченную по неубыванию последовательность этих чисел.

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

Дано целое число $1 \leq n \leq 10^3$ и массив $A[1 \dots n]$ натуральных чисел, не превосходящих $2 \cdot 10^9$. Выведите максимальное $1 \leq k \leq n$, для которого найдётся подпоследовательность $1 \leq i_1 < i_2 < \dots < i_k \leq n$ длины k , в которой каждый элемент делится на предыдущий (формально: для всех $1 \leq j < k$, $A[i_j] \mid A[i_{j+1}]$).

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);

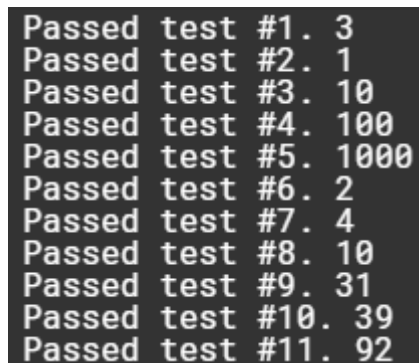
        int n = Integer.valueOf(br.readLine());
        int[] a = Arrays.stream(br.readLine().split(" ")).mapToInt(s -> Integer.parseInt(s)).toArray();
        int[] d = new int[n];

        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



```
Passed test #1. 3
Passed test #2. 1
Passed test #3. 10
Passed test #4. 100
Passed test #5. 1000
Passed test #6. 2
Passed test #7. 4
Passed test #8. 10
Passed test #9. 31
Passed test #10. 39
Passed test #11. 92
```

Задание №13

наибольшая невозрастающая подпоследовательность

Дано целое число $1 \leq n \leq 10^5$ и массив $A[1 \dots n]$, содержащий неотрицательные целые числа, не превосходящие 10^9 . Найдите наибольшую невозрастающую подпоследовательность в A . В первой строке выведите её длину k , во второй — её индексы $1 \leq i_1 < i_2 < \dots < i_k \leq n$ (таким образом, $A[i_1] \geq A[i_2] \geq \dots \geq A[i_k]$).

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

```
Passed test #1. 5
Passed test #2. 1
Passed test #3. 2
Passed test #4. 2
Passed test #5. 2
Passed test #6. 10
Passed test #7. 9
Passed test #8. 95
Passed test #9. 98
Passed test #10. 997
Passed test #11. 2996
```