**Министерство образования и науки Российской Федерации**

**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ,   
МЕХАНИКИ И ОПТИКИ**

Факультет программной инженерии и компьютерной техники  
Направление подготовки 09.03.04 Программная инженерия

Дисциплина «Aлгоритмы и структуры данных»

**ОТЧЁТ**

по лабораторной работе Динамическое программирование (Stepic)

Молодецкий Арсений

группа P3217

Санкт-Петербург

2019 г.

Содержание

[Задача 1: наибольшая последовательнократная подпоследовательность 3](#_Toc8428347)

[Исходный код к задаче 1 3](#_Toc8428348)

[Задача 2: наибольшая невозрастающая подпоследовательность 3](#_Toc8428349)

[Исходный код к задаче 2 4](#_Toc8428350)

[Задача 3: расстояние редактирования 4](#_Toc8428351)

[Исходный код к задаче 3 5](#_Toc8428352)

[Задача 4: рюкзак 5](#_Toc8428353)

[Исходный код к задаче 4 6](#_Toc8428354)

[Задача 5: лестница 6](#_Toc8428355)

[Исходный код к задаче 5 7](#_Toc8428356)

[Задача 6: калькулятор 7](#_Toc8428357)

[Исходный код к задаче 6 8](#_Toc8428358)

# Задача 1: наибольшая последовательнократная подпоследовательность

Дано целое число 1≤*n*≤103 и массив *A*[1…*n*] натуральных чисел, не превосходящих 2⋅109. Выведите максимальное 1≤*k*≤*n*, для которого найдётся подпоследовательность 1≤*i*1<*i*2<…<*ik*≤*n* длины *k*, в которой каждый элемент делится на предыдущий (формально: для  всех 1≤*j*<*k*, *A*[*ij*]|*A*[*ij*+1]).

**Sample Input:**

4

3 6 7 12

**Sample Output:**

3

# Исходный код к задаче 1

#include <iostream>

#include <vector>

#include <algorithm>

int main()

{

int num = 0;

std::cin >> num;

std::vector<int> arr;

while (--num >= 0) {

int val = 0;

std::cin >> val;

arr.push\_back(val);

}

auto size = arr.size();

std::vector<int> path\_len(size);

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

path\_len[i] = 1;

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

if ((arr[i] % arr[j] == 0) && (path\_len[j] + 1 > path\_len[i])) {

path\_len[i] = path\_len[j] + 1;

}

}

}

auto res = std::max\_element(path\_len.begin(), path\_len.end(), [](int e1, int e2) {return e1 < e2;});

std::cout << \*res << std::endl;

return 0;

}

# Задача 2: наибольшая невозрастающая подпоследовательность

Дано целое число 1≤*n*≤105и массив *A*[1…*n*], содержащий неотрицательные целые числа, не превосходящие 109. Найдите наибольшую невозрастающую подпоследовательность в *A*. В первой строке выведите её длину *k*, во второй — её индексы 1≤*i*1<*i*2<…<*ik*≤*n* (таким образом, *A*[*i*1]≥*A*[*i*2]≥…≥*A*[*in*]

).

**Sample Input:**

5

5 3 4 4 2

**Sample Output:**

4

1 3 4 5

# Исходный код к задаче 2

#include <iostream>

#include <vector>

#include <algorithm>

#include <sstream>

int main()

{

int num = 0;

std::cin >> num;

std::vector<int> arr;

while (--num >= 0) {

int val = 0;

std::cin >> val;

arr.push\_back(val);

}

auto size = arr.size();

std::vector<int> path\_len(size + 1);

std::vector<int> pos(size + 1);

for (size\_t i = 0; i < size + 1; ++i) pos[i] = -1;

path\_len[0] = std::numeric\_limits<int>::max();

for (size\_t i = 1; i < size + 1; ++i) path\_len[i] = std::numeric\_limits<int>::min();

for (size\_t i = 0; i < size; ++i) {

size\_t j = static\_cast<size\_t>(std::upper\_bound(path\_len.begin(), path\_len.end(), arr[i],

[](int e1, int e2) { return e1 > e2;}) - path\_len.begin());

if ((path\_len[j] < arr[i]) && (path\_len[j - 1] >= arr[i])) {

path\_len[j] = arr[i];

pos[j] = i;

}

else if (j == 0) {

path\_len[1] = arr[0];

pos[1] = 0;

}

}

auto res = std::count\_if(path\_len.begin(), path\_len.end(), [](int e1) {return e1 > std::numeric\_limits<int>::min() && e1 < std::numeric\_limits<int>::max();});

std::cout << res << std::endl;

std::ostringstream oss;

for (int i = 0; i < size + 1; ++i) {

if (pos[i] >= 0) {

oss << pos[i] + 1 << " ";

}

}

std::cout << oss.str() << std::endl;

return 0;

}

# Задача 3: расстояние редактирования

Вычислите расстояние редактирования двух данных непустых строк длины не более 102, содержащих строчные буквы латинского алфавита.

**Sample Input 1:**

ab

ab

**Sample Output 1:**

0

**Sample Input 2:**

short

ports

**Sample Output 2:**

3

# Исходный код к задаче 3

#include <iostream>

#include <vector>

#include <string>

#include <algorithm>

int main()

{

// Считываем первую строку

std::string strA;

std::cin >> strA;

// Считываем вторую строку

std::string strB;

std::cin >> strB;

// Определяем размер строк

auto sizeA = strA.size();

auto sizeB = strB.size();

// Создаем массивы для двух строк

std::vector<std::vector<int>> tab;

// Инициализируем данные

for (auto i = 0; i <= sizeB; ++i) tab.push\_back(std::vector<int>(sizeA + 1));

for (auto i = 0; i <= sizeA; ++i) tab[0][i] = i;

for (auto i = 1; i <= sizeB; ++i) tab[i][0] = i;

// Считаем таблицу

for (auto j = 1; j <= sizeB; ++j) {

for (auto i = 1; i <= sizeA; ++i) {

auto c = (strA.at(i - 1) == strB.at(j - 1)) ? 0 : 1;

auto min\_val = std::min({ tab[j][i - 1] + 1, tab[j - 1][i] + 1, tab[j - 1][i - 1] + c });

tab[j][i] = min\_val;

}

}

std::cout << tab[sizeB][sizeA] << std::endl;

return 0;

}

# Задача 4: рюкзак

Первая строка входа содержит целые числа 1≤*W*≤104 и 1≤*n*≤300 — вместимость рюкзака и число золотых слитков. Следующая строка содержит *n* целых чисел 0≤*w*1,…,*wn*≤105, задающих веса слитков. Найдите максимальный вес золота, который можно унести в рюкзаке.

**Sample Input:**

10 3

1 4 8

**Sample Output:**

9

# Исходный код к задаче 4

#include <iostream>

#include <vector>

#include <algorithm>

int main()

{

//Считываем вместимость рюкзака

size\_t W = 0;

std::cin >> W;

// Считываем количество вещей

size\_t n = 0;

std::cin >> n;

// Считываем веса вещей

std::vector<int> v;

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

auto value = 0;

std::cin >> value;

v.push\_back(value);

}

// Инициализируем таблицу значений

std::vector<std::vector<int>> tab(W + 1, std::vector<int>(n + 1, 0));

// Рассчитываем значения

for (size\_t j = 1; j <= n; ++j) {

for (size\_t w = 1; w <= W; ++w) {

if (v[j - 1] > w) tab[w][j] = tab[w][j - 1];

else tab[w][j] = std::max({ tab[w][j - 1], tab[w - v[j - 1]][j - 1] + v[j - 1] });

}

}

std::cout << tab[W][n] << std::endl;

return 0;

}