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

Факультет Инфокоммуникационных Технологий

**Домашняя работа №4**

**Вариант № 4**

Выполнили:

Бакланова А.Г,

Крылов Д.С,

Улитина М.С

Проверил Мусаев А.А.

Санкт-Петербург, 2023

Оглавление

[Оглавление 2](#_Toc133244075)

[Введение. 3](#_Toc133244076)

[Алгоритмы выполнения заданий. 4](#_Toc133244077)

[Задание 1. 4](#_Toc133244078)

[Задание 2. 5](#_Toc133244079)

[Задание 3. 6](#_Toc133244080)

[Заключение. 8](#_Toc133244081)

# Введение.

Цель лабораторной работы – выполнение трех заданий с помощью Python по теме «Динамическое программирование». Вариант работы – 4.

# 

# Алгоритмы выполнения заданий.

## Задание 1.

В первом заданий нужно определить, как вору вынести наибольшую сумму, исходя из количества заходов и доступного количества килограмм за один заход, количества экспонатов.

На рисунке 1 приведен алгоритм работы программы. Пользователь вводит условия для вора: количество экспонатов, заходы, вместимость за один заход, цена экспоната и вес. Создается два массива с весом и ценностью экспонатов. Создадим двумерный массив dp размером (N+1)x(M+1), где dp[i][j] будет хранить максимальную сумму ценности экспонатов, которые можно унести. Далее заполняем массив dp по формуле: dp[i][j] = max(dp[i-1][j], dp[i-1][j-w[i]] + c[i]), где w[i] - вес i-го экспоната, c[i] - ценность. После заполнения массива dp, найдём максимальную сумму ценности, которую можем унести за M заходов, и восстановим последовательность экспонатов, которые нужно унести. Для этого будем двигаться по массиву dp, начиная с dp[N][M], и добавлять в список украденных экспонатов i-й экспонат, если значение dp[i][j] отличается от dp[i-1][j].

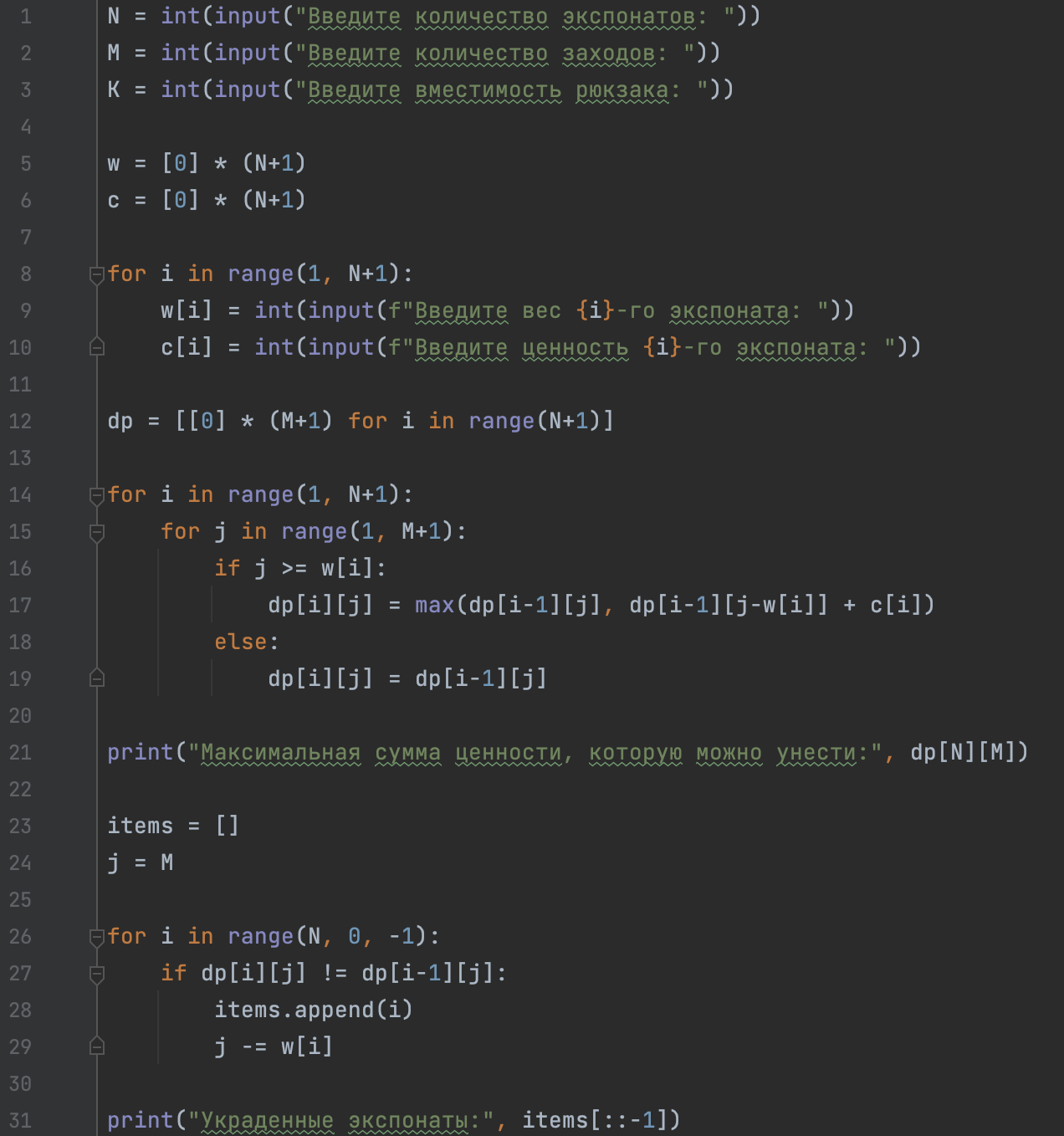


Рисунок 1 – Программа украденного.

На рисунке 2 результат работы программы.

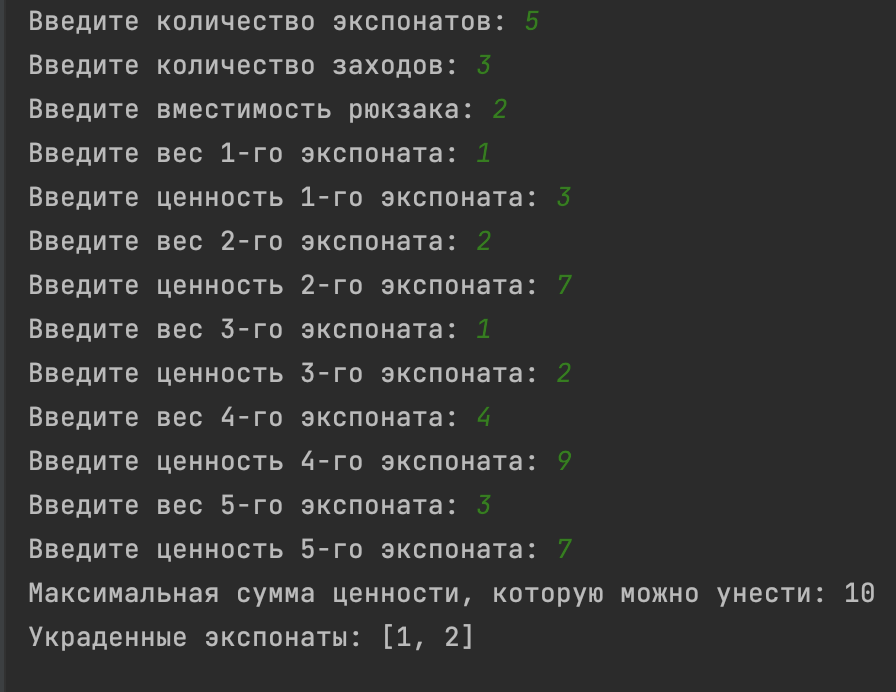


Рисунок 2 – результат работы программы.

## Задание 2.

Данная программа решает задачу нахождения минимального количества скалярных операций для перемножения заданной последовательности матриц A, B, C, ... , Z. Для этого используется динамическое программирование.

На рисунке 3 приведен алгоритм работы программы. Алгоритм решения задачи заключается в том, что мы рассматриваем все возможные способы разбиения последовательности матриц на две части и находим минимальное количество скалярных операций для перемножения каждой из этих частей.

Для хранения уже найденных значений используется двумерный массив K[a][b], где a и b - индексы первой и последней матрицы в подпоследовательности, для которой мы ищем минимальное количество скалярных операций. Если значение K[a][b] еще не найдено, то мы рекурсивно вызываем функцию min\_multy для нахождения этого значения.

В программе также используется глобальные переменные K и M, которые хранят значения массива K и последовательности матриц соответственно. Значения последовательности матриц M генерируются случайным образом с помощью функции randint.

В конце программы выводится минимальное количество скалярных операций для перемножения всей последовательности матриц.

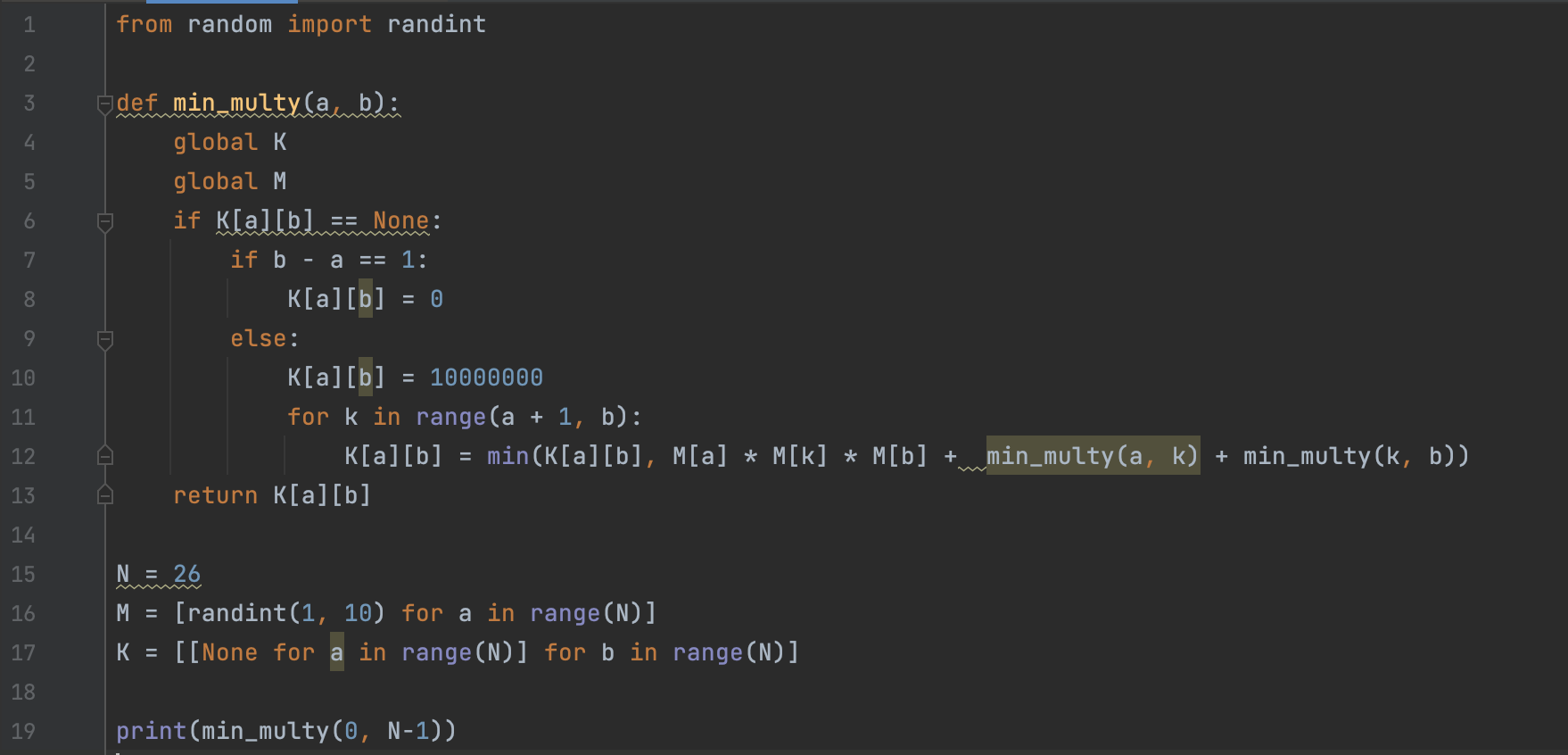


Рисунок 3 – Задача 2.

На рисунке 4 приведен результат работы программы.

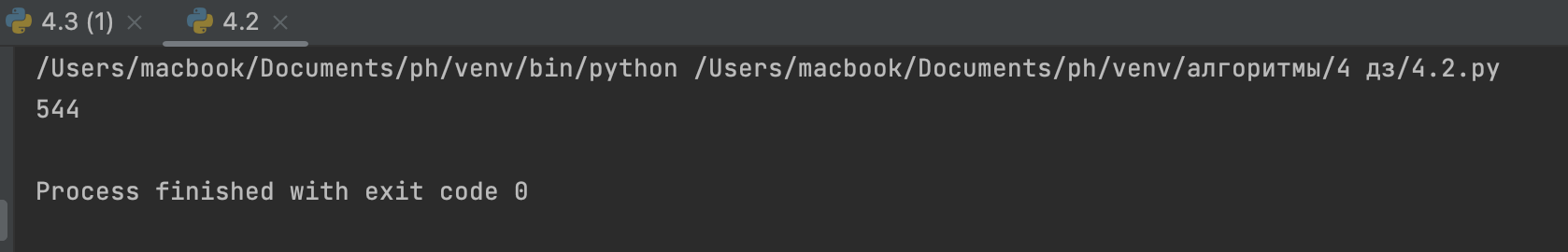


Рисунок 4 – Результат работы.

## Задание 3.

На рисунке 5 приведен алгоритм работы программы. Пользователь вводит размер массива, после чего создается размер массива с рандомными числами от -100 до 100.

Затем происходит поиск наибольшей непрерывной последовательности в этом массиве путем прохождения цикла, в котором происходит сравнение каждого элемента с предыдущим. Если текущий элемент больше предыдущего, тогда длина текущей последовательности (current\_len) увеличивается на 1. Иначе, и если длина текущей последовательности больше максимальной длины, то обновляются значения максимальной длины (они становятся равными текущей длине последовательности) и индекс начала последовательности. Если текущий элемент не больше предыдущего, то текущая последовательность сбрасывается до 1.В конце выводится найденная последовательность.

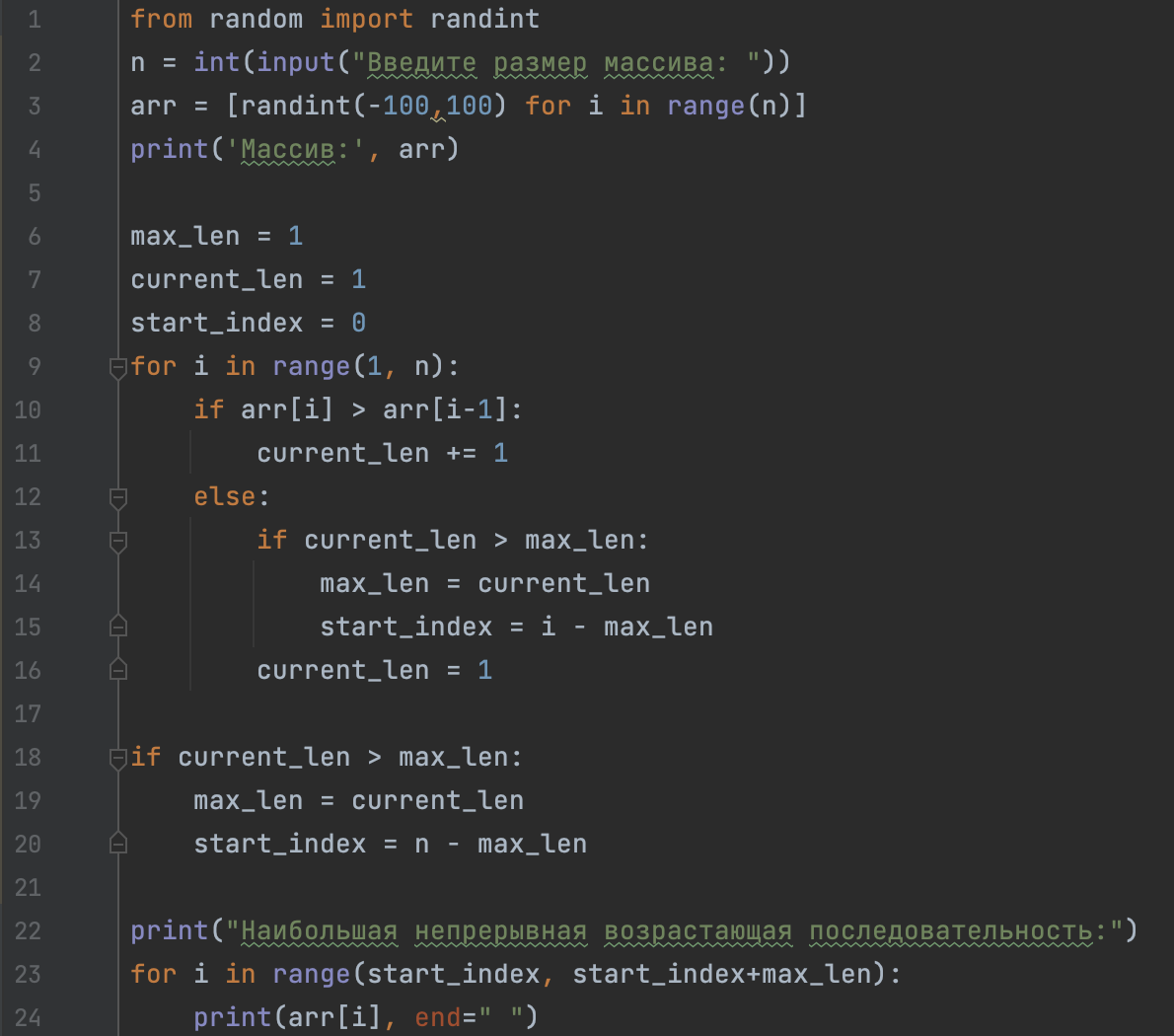


Рисунок 5 – Задача 3.

На рисунке 6 приведен результат работы программы.

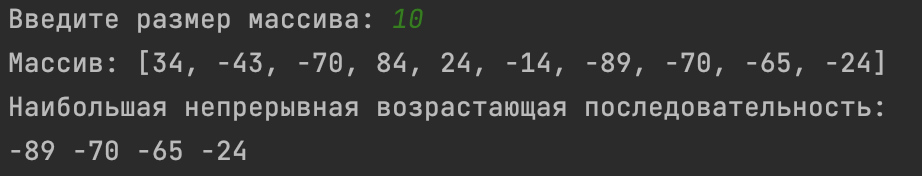


Рисунок 6 – Результат работы программы.

# Заключение.

Задания выполнены, таким образом, можно считать, что цель домашней работы выполнена. Изучено «динамическое программирование». Программы находятся по ссылке: <https://github.com/UlitiM2/hw4>