САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ

ФАКУЛЬТЕТ ИНФОКОММУНИКАЦИОННЫХ ТЕХНОЛОГИЙ

Отчет по лабораторной работе №1

по курсу «Алгоритмы и структуры данных»

Тема: Greedy and DP

Выполнил:

Лазарев Марк Олегович

К3241

Санкт-Петербург

2025 г.

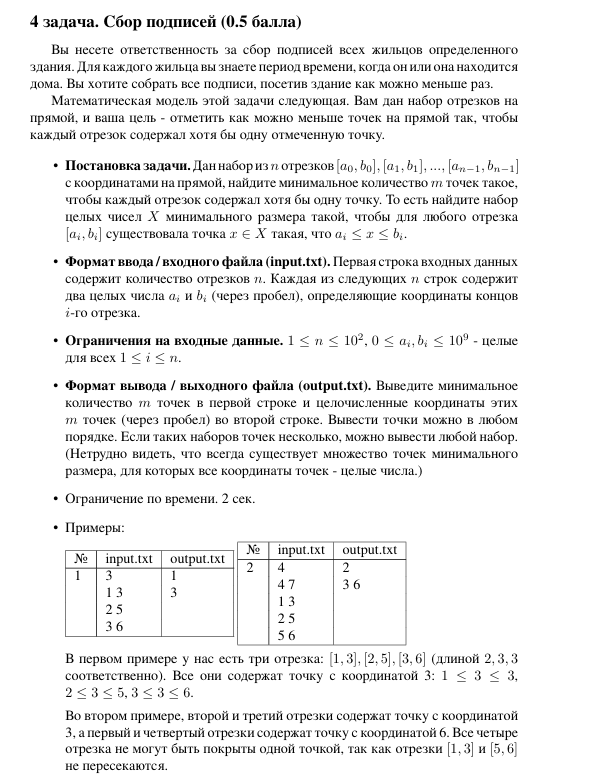
[**Задачи по варианту**](#_e0pij87brwfh)

[Задача №1.](#_y9xrr9qdf7ef) Сбор подписей  
[Задача №2.](#_g3d552ii7rgc) Максимальное количество призов

[Задача №3.](#_y9xrr9qdf7ef) Максимальное количество золота   
[Задача №4.](#_g3d552ii7rgc) Сувениры

[Задача №](#_y9xrr9qdf7ef)5. Произведение матриц

# Задачи по варианту



**Код программы:**

**def collect\_signatures(segments):**

**# Сортируем по правому краю**

**segments.sort(key=lambda x: x[1])**

**points = []**

**last\_point = -1**

**for seg in segments:**

**if last\_point < seg[0]:**

**last\_point = seg[1]**

**points.append(last\_point)**

**return points**

**with open("input4.txt", "r") as f:**

**n = int(f.readline())**

**segments = [tuple(map(int, f.readline().split())) for \_ in range(n)]**

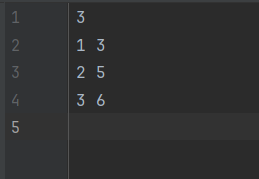
**points = collect\_signatures(segments)**

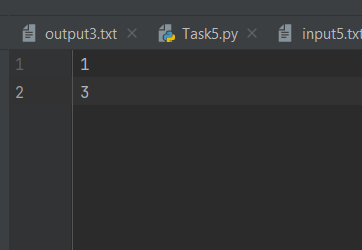
**with open("output4.txt", "w") as f:**

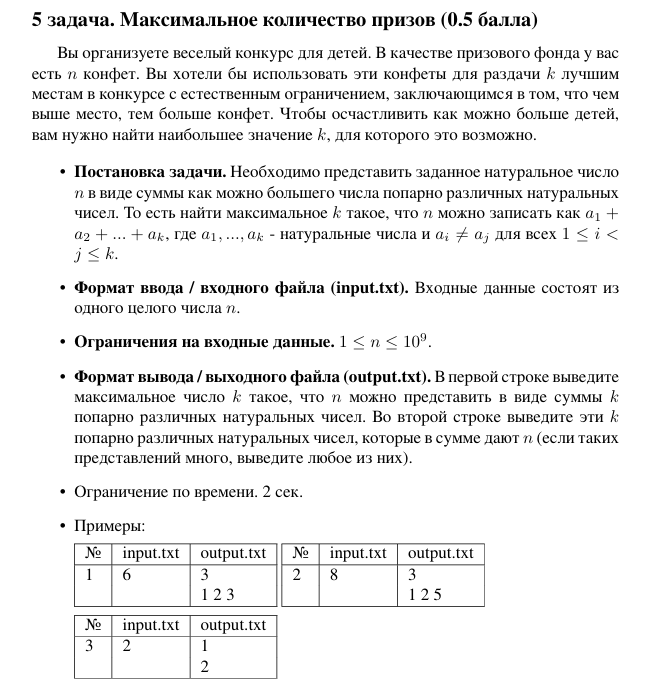
**f.write(f"{len(points)}\n")**

**f.write(" ".join(map(str, points)))**

Результат работы кода на примерах:







**Код программы:**

**def max\_unique\_sum(n):**

**result = []**

**current = 1**

**while n > 0:**

**if n - current > current:**

**result.append(current)**

**n -= current**

**current += 1**

**else:**

**result.append(n)**

**break**

**return result**

**with open("input5.txt", "r") as f:**

**n = int(f.readline())**

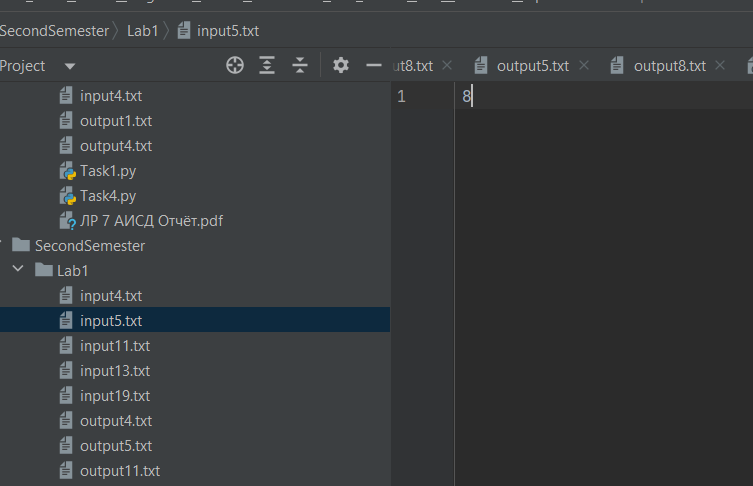
**result = max\_unique\_sum(n)**

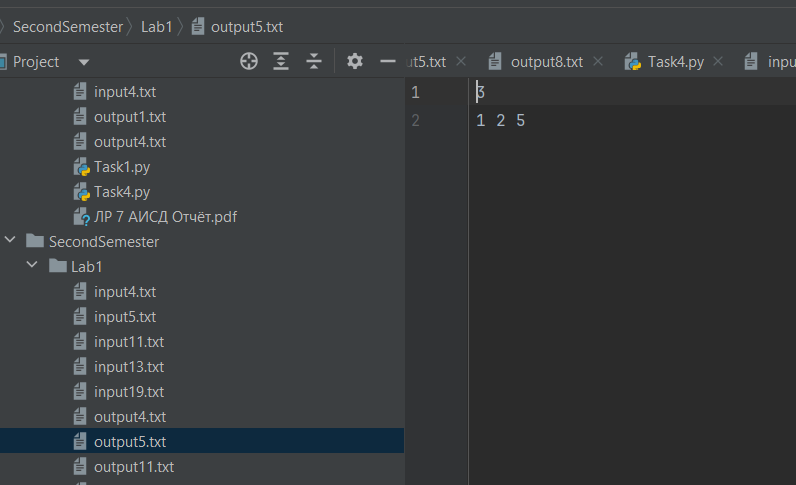
**with open("output5.txt", "w") as f:**

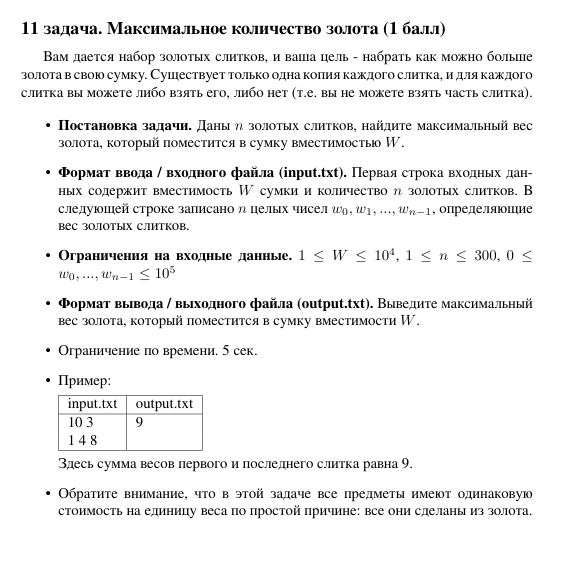
**f.write(f"{len(result)}\n")**

**f.write(" ".join(map(str, result)))**

Результат работы кода на примерах:

****

****

****

**Код программы:**

**def max\_gold(W, weights):**

**dp = [0] \* (W + 1)**

**for weight in weights:**

**for w in range(W, weight - 1, -1):**

**dp[w] = max(dp[w], dp[w - weight] + weight)**

**return dp[W]**

**# Чтение из input.txt**

**with open("input11.txt", "r") as f:**

**W, n = map(int, f.readline().split())**

**weights = list(map(int, f.readline().split()))**

**# Вычисляем результат**

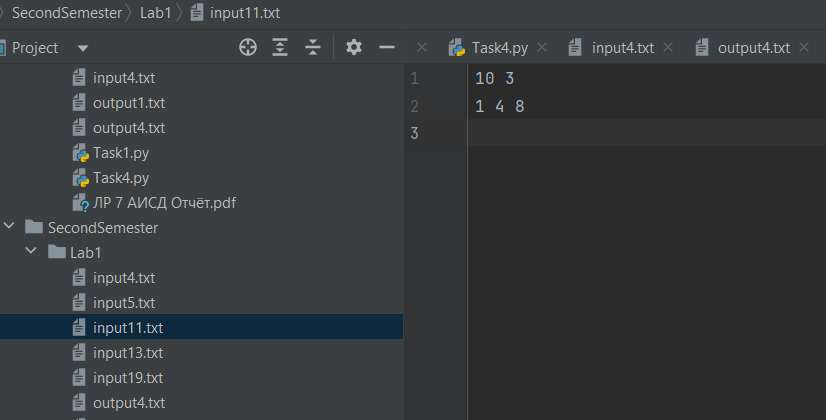
**result = max\_gold(W, weights)**

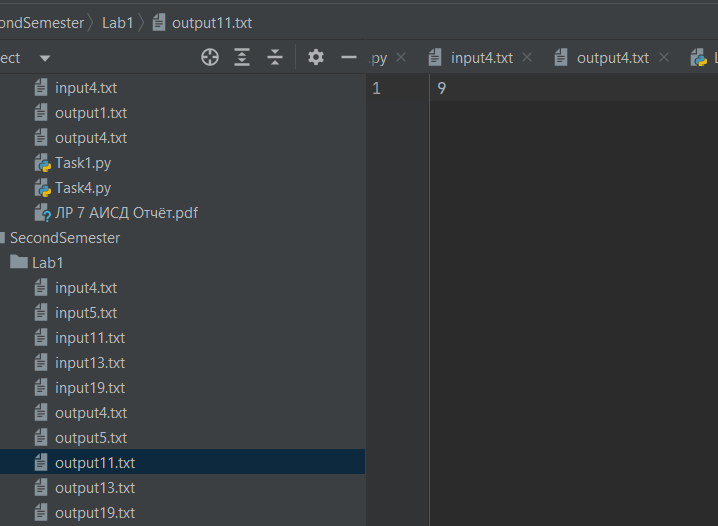
**# Запись в output.txt**

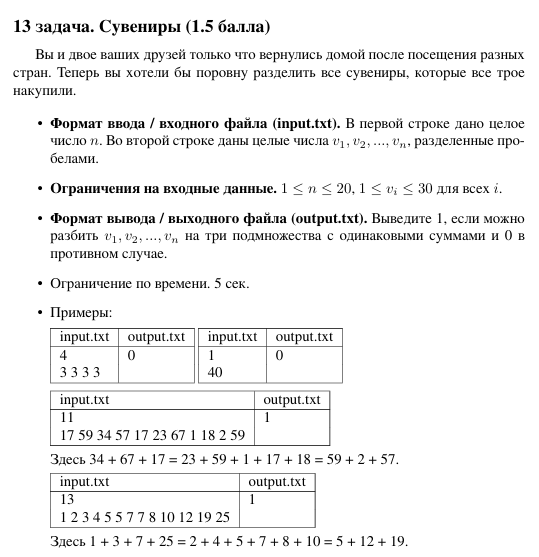
**with open("output11.txt", "w") as f:**

**f.write(str(result))**

Результат работы



****

****

**Код программы:**

**import time**

**import os**

**def can\_partition\_into\_three\_subsets(nums):**

**total\_sum = sum(nums)**

**# Если сумма не делится на 3, то невозможно разделить на три подмножества с одинаковыми суммами**

**if total\_sum % 3 != 0:**

**return 0**

**target = total\_sum // 3**

**n = len(nums)**

**# dp[i][j] будет True, если можно получить сумму j, используя первые i элементов**

**dp = [[False] \* (target + 1) for \_ in range(n + 1)]**

**dp[0][0] = True**

**for i in range(1, n + 1):**

**for j in range(target, -1, -1):**

**if j >= nums[i - 1]:**

**dp[i][j] = dp[i][j] or dp[i - 1][j - nums[i - 1]]**

**dp[i][j] = dp[i][j] or dp[i - 1][j]**

**return 1 if dp[n][target] else 0**

**start\_time = time.perf\_counter()**

**with open('input13.txt', 'r') as file:**

**lines = file.readlines()**

**# Проверка на наличие хотя бы двух строк в файле**

**if len(lines) < 2:**

**raise ValueError("Файл input13.txt должен содержать как минимум две строки: количество чисел и сами числа.")**

**# Чтение количества чисел**

**n = int(lines[0].strip())**

**# Чтение чисел и преобразование их в список целых чисел**

**nums = list(map(int, lines[1].strip().split()))**

**# Проверка возможности разделить на три подмножества с одинаковыми суммами**

**result = can\_partition\_into\_three\_subsets(nums)**

**with open('output13.txt', 'w') as file:**

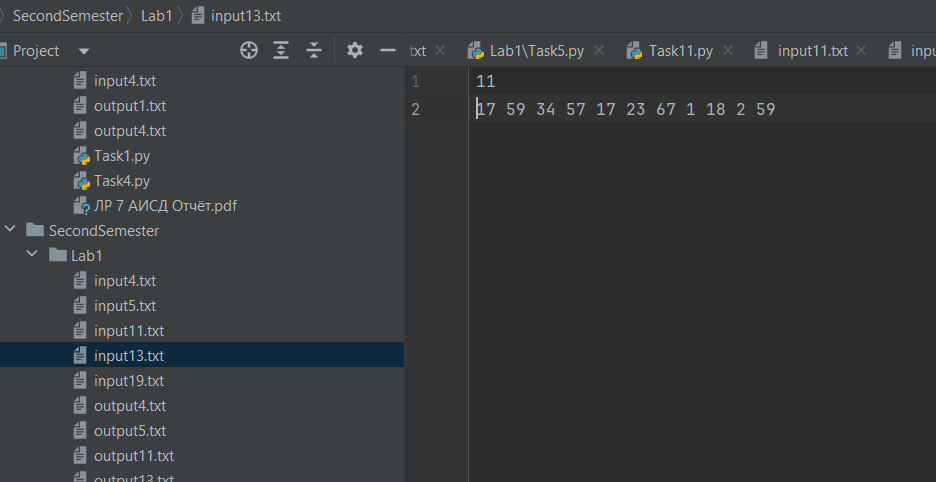
**file.write(str(result))**

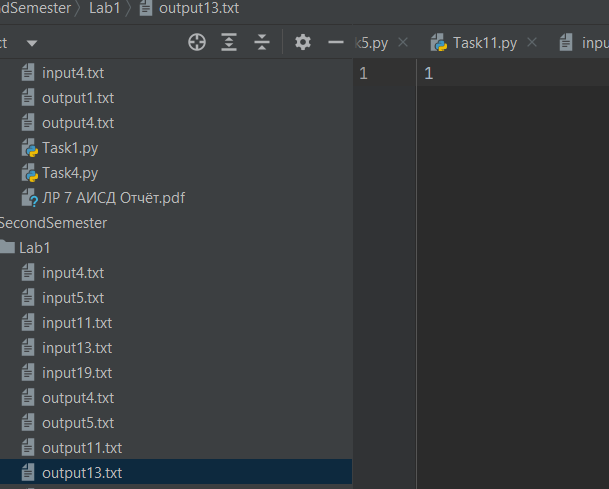
**end\_time = time.perf\_counter()**

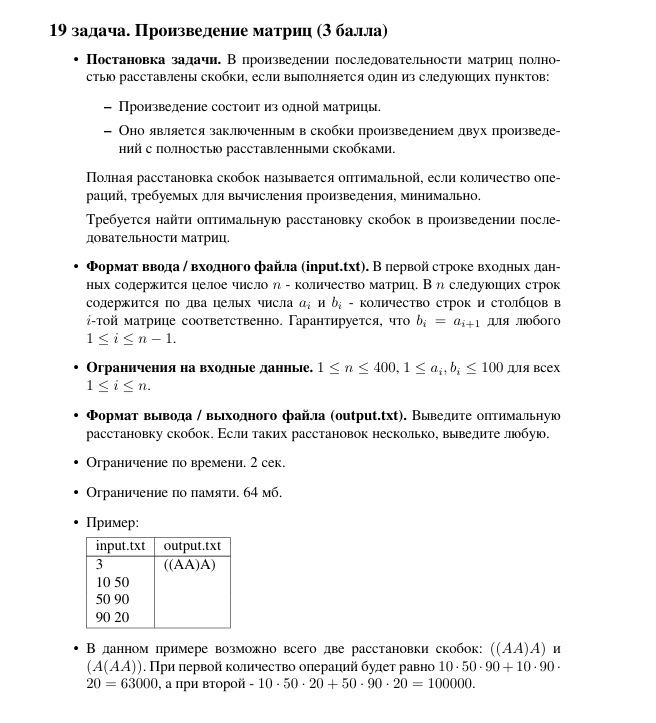
**execution\_time = end\_time - start\_time**

**print(f"Время выполнения: {execution\_time:.6f} секунд")**

Результат работы кода на примерах:

****

****

****

**Код программы:**

**def matrix\_chain\_order(p):**

**n = len(p) - 1 # Кол-во матриц**

**dp = [[0]\*n for \_ in range(n)]**

**s = [[0]\*n for \_ in range(n)]**

**for l in range(2, n+1): # длина цепочки**

**for i in range(n - l + 1):**

**j = i + l - 1**

**dp[i][j] = float('inf')**

**for k in range(i, j):**

**cost = dp[i][k] + dp[k+1][j] + p[i]\*p[k+1]\*p[j+1]**

**if cost < dp[i][j]:**

**dp[i][j] = cost**

**s[i][j] = k**

**return dp, s**

**def build\_optimal\_parenthesization(s, i, j):**

**if i == j:**

**return f"A{i+1}"**

**else:**

**return "(" + build\_optimal\_parenthesization(s, i, s[i][j]) + \**

**build\_optimal\_parenthesization(s, s[i][j]+1, j) + ")"**

**with open("input19.txt", "r") as f:**

**n = int(f.readline())**

**dims = []**

**for \_ in range(n):**

**a, b = map(int, f.readline().split())**

**dims.append(a)**

**dims.append(b) # добавляем последний столбец**

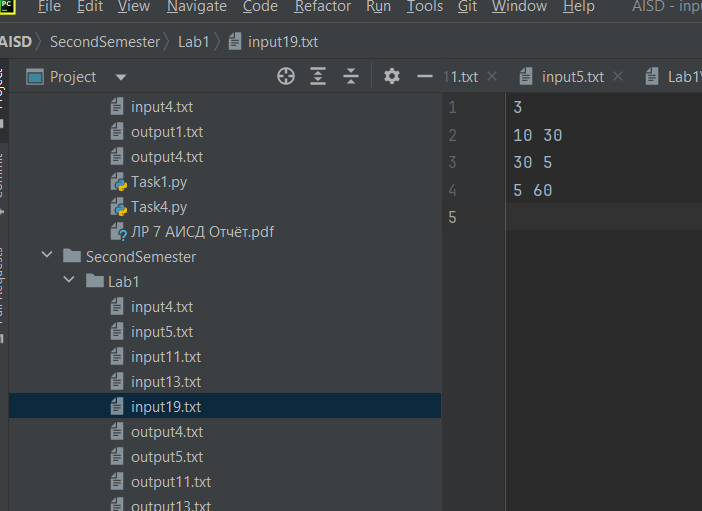
**dp, s = matrix\_chain\_order(dims)**

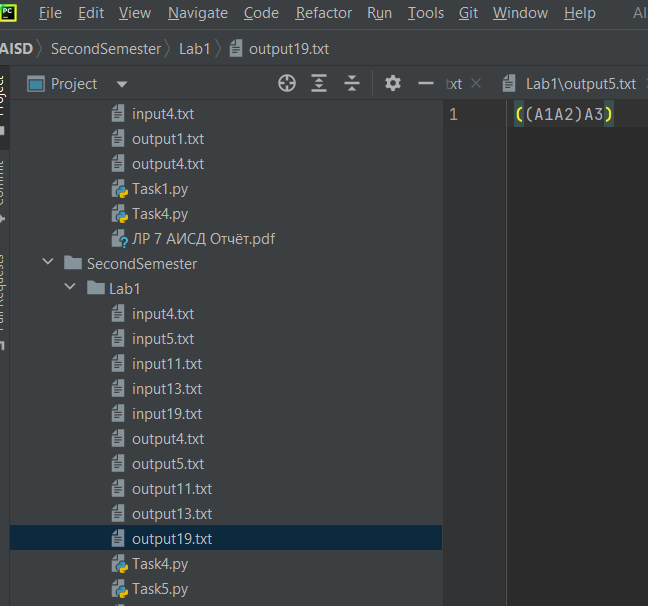
**result = build\_optimal\_parenthesization(s, 0, n-1)**

**with open("output19.txt", "w") as f:**

**f.write(result)**

Результат работы кода на примерах:

****

****