САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ

ФАКУЛЬТЕТ ИНФОКОММУНИКАЦИОННЫХ ТЕХНОЛОГИЙ

Отчет по лабораторной работе №3

по курсу «Алгоритмы и структуры данных»

Тема: Graphs  
Вариант 18

Выполнил:

Лазарев Марк Олегович

К3241

Санкт-Петербург

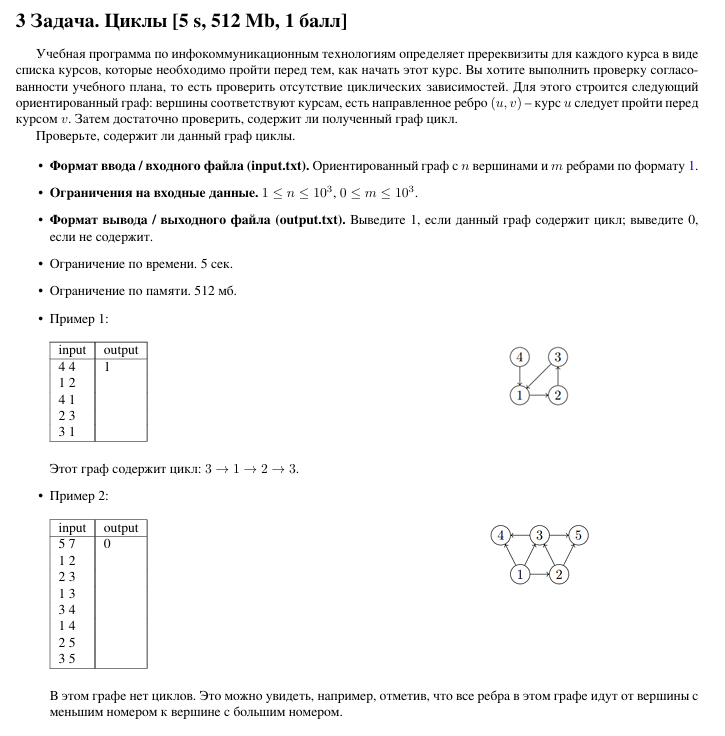
2025 г.

[**Задачи по варианту**](#_ipii2kspewkt)

[Задача №1.](#_i8z8rgqqhdu4) Циклы  
[Задача №2.](#_bncre75ba6xe) Аномалии курсов валют

[Задача №3.](#_i8z8rgqqhdu4) Грядки

# Задачи по варианту



**import time**

**import tracemalloc**

**def has\_cycle(graph, n):**

**visited = [0] \* n # 0 = не посещена, 1 = посещается, 2 = полностью обработана**

**def dfs(v):**

**visited[v] = 1**

**for neighbor in graph[v]:**

**if visited[neighbor] == 1:**

**return True # цикл обнаружен**

**if visited[neighbor] == 0 and dfs(neighbor):**

**return True**

**visited[v] = 2**

**return False**

**for i in range(n):**

**if visited[i] == 0:**

**if dfs(i):**

**return True**

**return False**

**def main():**

**# Старт замеров**

**tracemalloc.start()**

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

**# Чтение входных данных**

**with open("input3.txt") as f:**

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

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

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

**u, v = map(int, f.readline().split())**

**graph[u - 1].append(v - 1)**

**# Проверка наличия цикла**

**cycle = has\_cycle(graph, n)**

**# Запись результата**

**with open("output3.txt", "w") as f:**

**f.write("1" if cycle else "0")**

**# Завершение замеров**

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

**current, peak = tracemalloc.get\_traced\_memory()**

**tracemalloc.stop()**

**# Вывод в консоль**

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

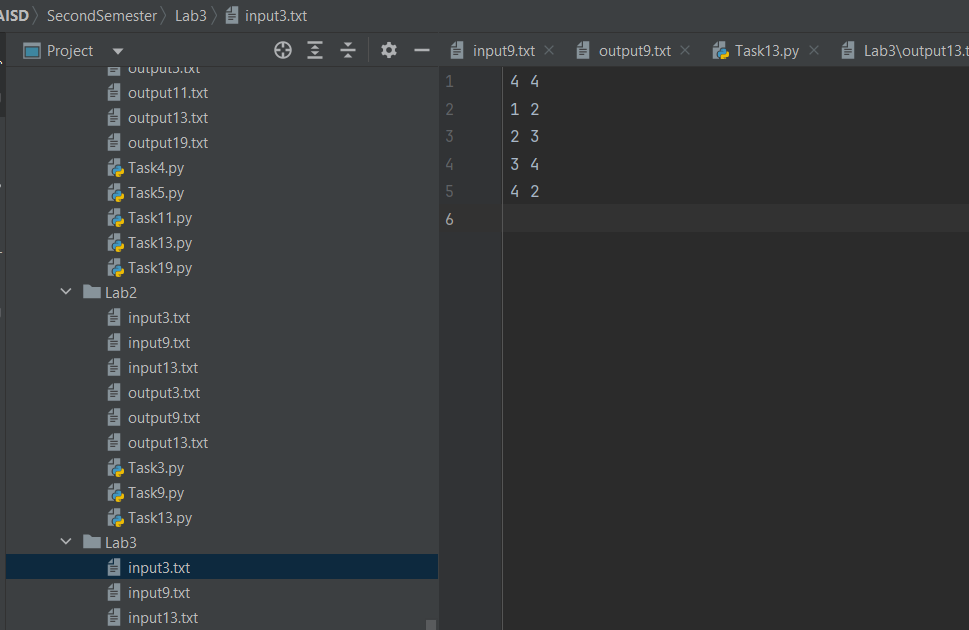
**print(f"Использовано памяти: {current / 1024:.2f} KB")**

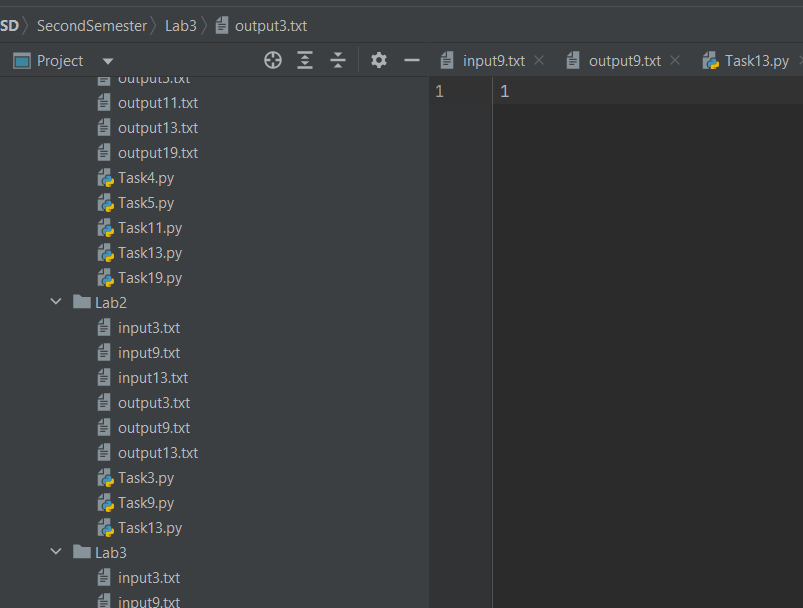
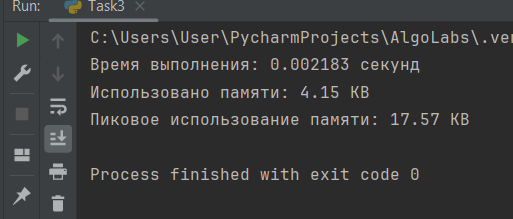
**print(f"Пиковое использование памяти: {peak / 1024:.2f} KB")**

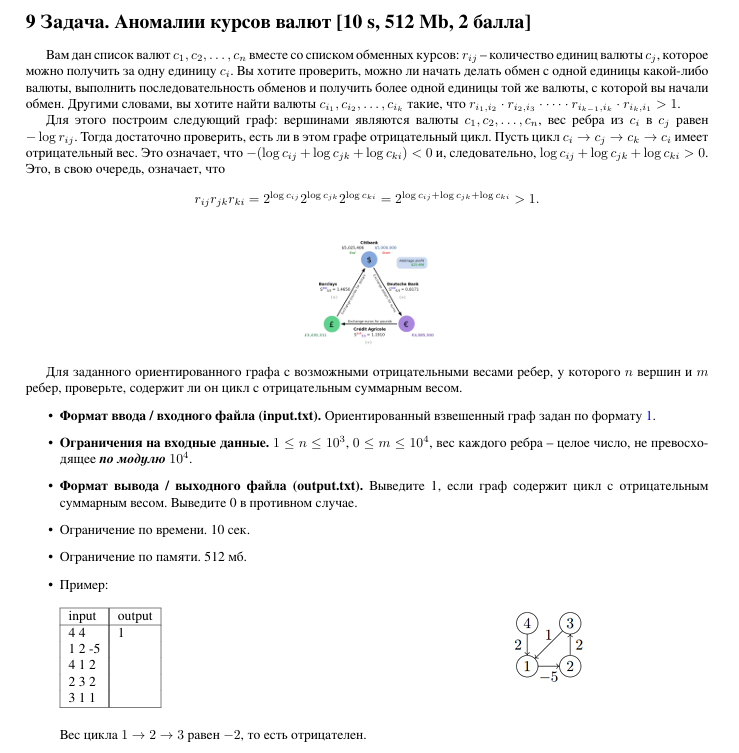
**if \_\_name\_\_ == "\_\_main\_\_":**

**main()**

Результат работы кода на примерах:





**Код программы:  
import time**

**import tracemalloc**

**def has\_negative\_cycle(n, edges):**

**# Запускаем алгоритм Беллмана-Форда из каждой вершины**

**for start in range(n):**

**dist = [float('inf')] \* n**

**dist[start] = 0**

**for \_ in range(n - 1):**

**updated = False**

**for u, v, weight in edges:**

**if dist[u] + weight < dist[v]:**

**dist[v] = dist[u] + weight**

**updated = True**

**if not updated:**

**break**

**for u, v, weight in edges:**

**if dist[u] + weight < dist[v]:**

**return True # Найден отрицательный цикл**

**return False**

**def main():**

**# Запуск замеров**

**tracemalloc.start()**

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

**# Чтение входных данных**

**with open("input9.txt", "r") as file:**

**n, m = map(int, file.readline().split())**

**edges = []**

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

**u, v, w = file.readline().split()**

**u = int(u) - 1**

**v = int(v) - 1**

**w = int(w)**

**edges.append((u, v, w))**

**# Проверка на наличие цикла отрицательного веса**

**result = has\_negative\_cycle(n, edges)**

**# Запись результата**

**with open("output9.txt", "w") as file:**

**file.write("1" if result else "0")**

**# Завершение замеров**

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

**current, peak = tracemalloc.get\_traced\_memory()**

**tracemalloc.stop()**

**# Вывод статистики**

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

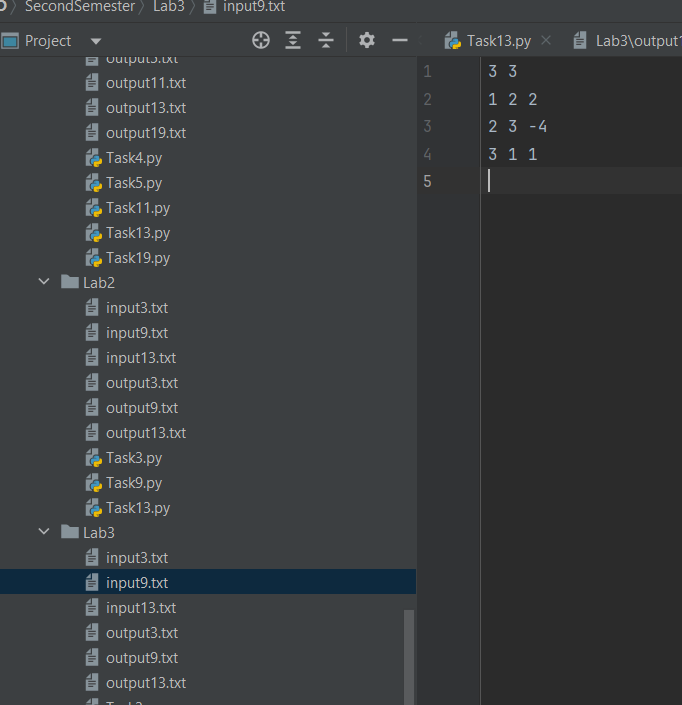
**print(f"Использовано памяти: {current / 1024:.2f} KB")**

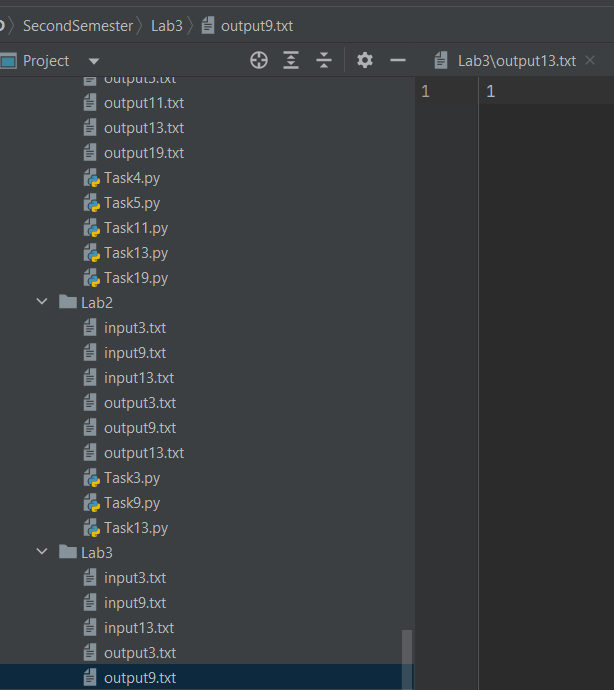
**print(f"Пиковое использование памяти: {peak / 1024:.2f} KB")**

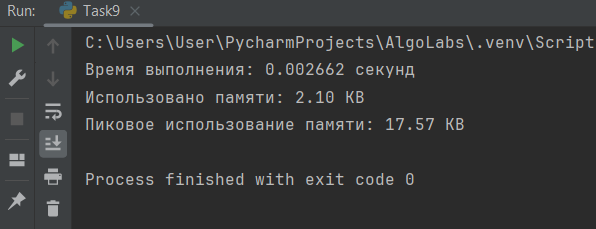
**if \_\_name\_\_ == "\_\_main\_\_":**

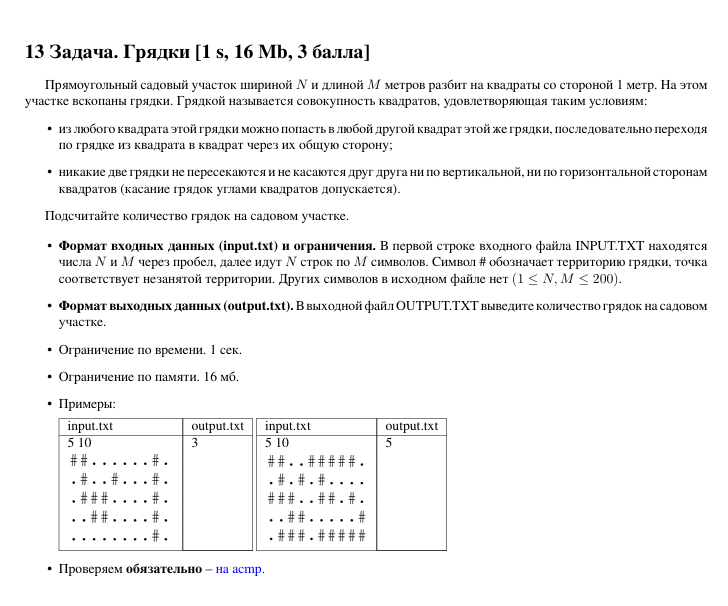
**main()**

Результат работы кода на примерах:

****

****

****



**Код программы:**

**import time**

**import tracemalloc**

**def count\_beds(garden, n, m):**

**visited = [[False] \* m for \_ in range(n)]**

**def dfs(x, y):**

**visited[x][y] = True**

**directions = [(-1, 0), (1, 0), (0, -1), (0, 1)]**

**for dx, dy in directions:**

**nx, ny = x + dx, y + dy**

**if 0 <= nx < n and 0 <= ny < m:**

**if garden[nx][ny] == '#' and not visited[nx][ny]:**

**dfs(nx, ny)**

**beds = 0**

**for i in range(n):**

**for j in range(m):**

**if garden[i][j] == '#' and not visited[i][j]:**

**dfs(i, j)**

**beds += 1**

**return beds**

**def main():**

**# Запуск замеров**

**tracemalloc.start()**

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

**# Чтение данных**

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

**n, m = map(int, file.readline().split())**

**garden = [list(file.readline().strip()) for \_ in range(n)]**

**# Подсчёт грядок**

**result = count\_beds(garden, n, m)**

**# Запись результата**

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

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

**# Завершение замеров**

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

**current, peak = tracemalloc.get\_traced\_memory()**

**tracemalloc.stop()**

**# Вывод в консоль**

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

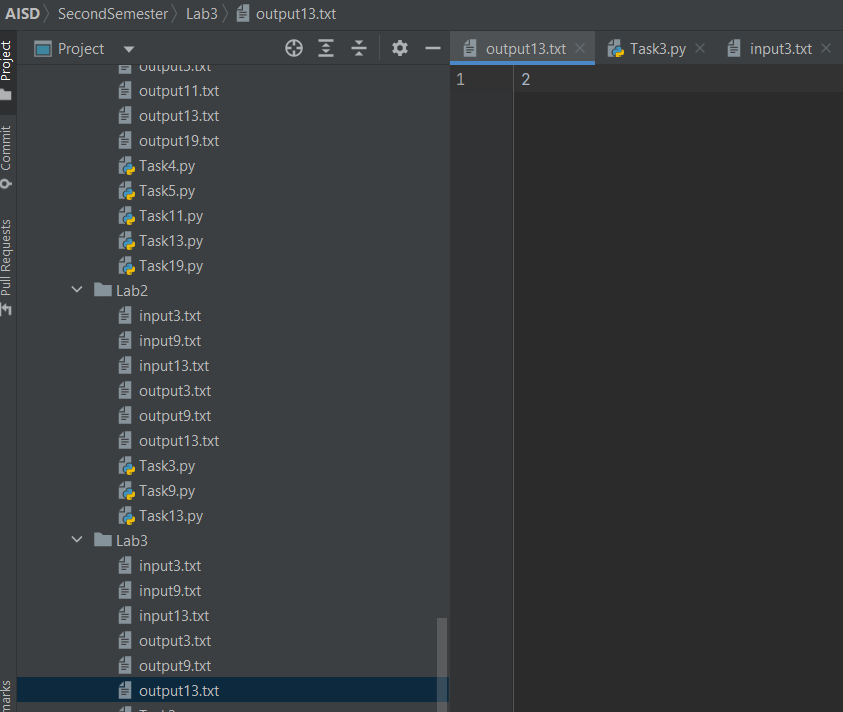
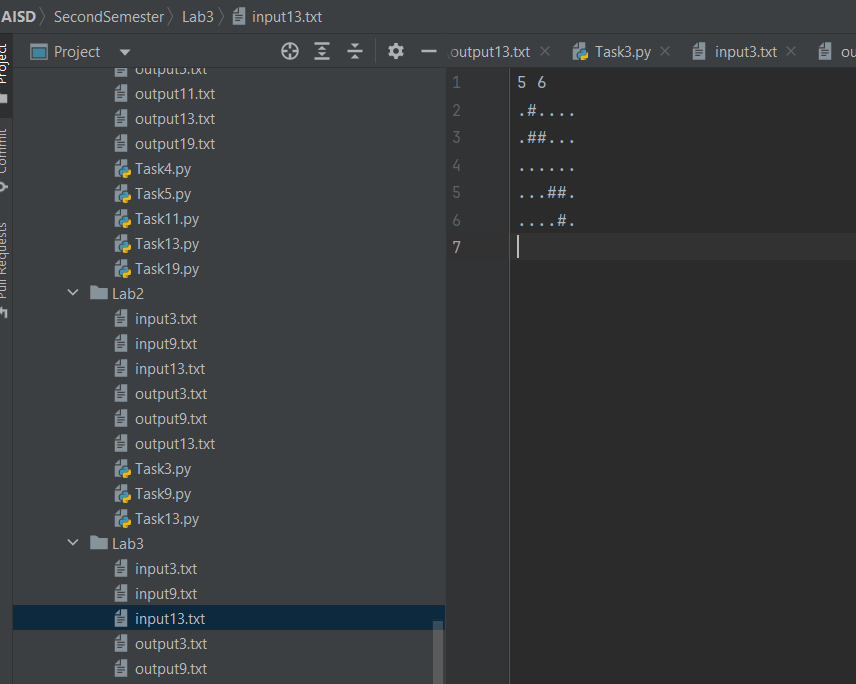
**print(f"Использовано памяти: {current / 1024:.2f} KB")**

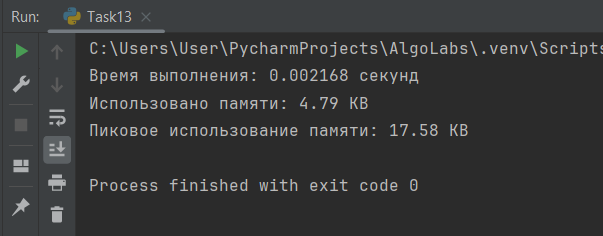
**print(f"Пиковое использование памяти: {peak / 1024:.2f} KB")**

**if \_\_name\_\_ == "\_\_main\_\_":**

**main()**

Результат работы

****

****