|  |  |
| --- | --- |
| Город | Расстояние из первого |
| 1 | 0 |
| 2 | 125 |
| 3 | 105 |
| 4 | 185 |
| 5 | 245 |

Рассмотрим код для каждого из методов:

**Алгоритм Дейкстры:**

import heapq

def dijkstra(graph, start):

    distances = {vertex: float('infinity') for vertex in graph}

    distances[start] = 0

    priority\_queue = [(0, start)]

    while priority\_queue:

        current\_distance, current\_vertex = heapq.heappop(priority\_queue)

        if current\_distance > distances[current\_vertex]:

            continue

        for neighbor, weight in graph[current\_vertex].items():

            distance = current\_distance + weight

            if distance < distances[neighbor]:

                distances[neighbor] = distance

                heapq.heappush(priority\_queue, (distance, neighbor))

    return distances

# Ваш граф

graph = {

    1: {2: 125, 3: 105},

    2: {1: 125, 3: 135, 4: 75},

    3: {1: 105, 2: 135, 4: 80, 5: 140},

    4: {2: 75, 3: 80, 5: 110},

    5: {3: 140, 4: 110}

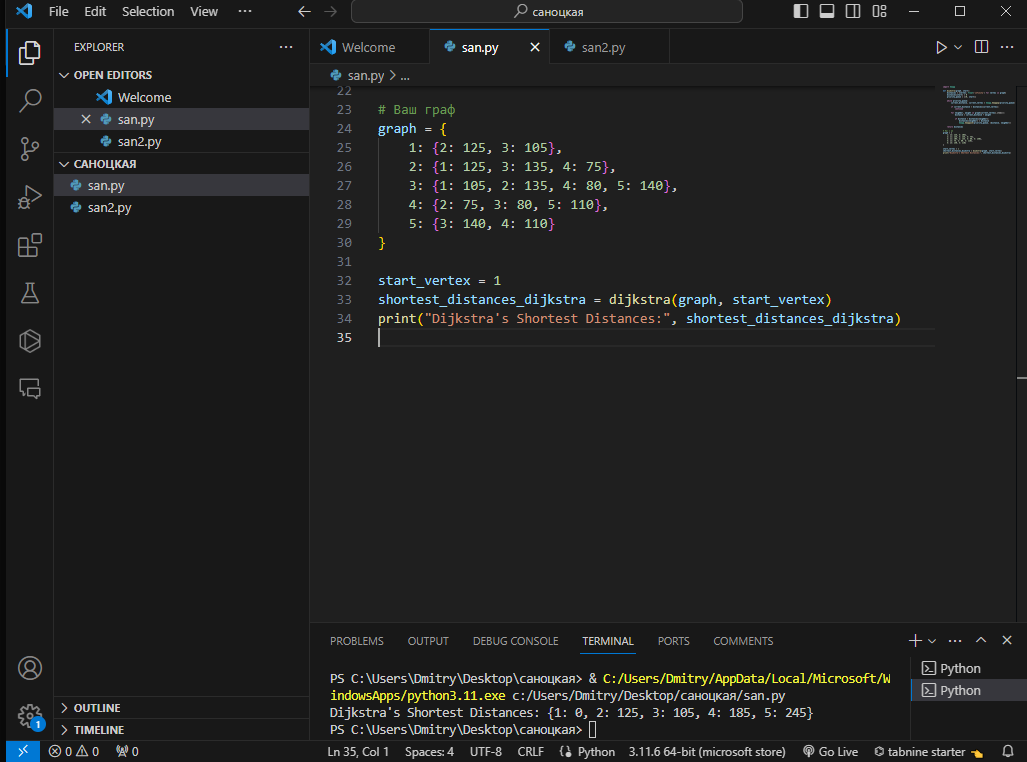
}

start\_vertex = 1

shortest\_distances\_dijkstra = dijkstra(graph, start\_vertex)

print("Dijkstra's Shortest Distances:", shortest\_distances\_dijkstra)

**Результат:**

****

**Алгоритм Флойда-Уоршелла:**

def floyd\_warshall(graph):

    vertices = list(graph.keys())

    num\_vertices = len(vertices)

    # Инициализация матрицы расстояний

    distances = [[float('infinity')] \* num\_vertices for \_ in range(num\_vertices)]

    for i in range(num\_vertices):

        distances[i][i] = 0

        for neighbor, weight in graph[i + 1].items():

            distances[i][neighbor - 1] = weight

    # Алгоритм Флойда-Уоршелла

    for k in range(num\_vertices):

        for i in range(num\_vertices):

            for j in range(num\_vertices):

                if distances[i][k] + distances[k][j] < distances[i][j]:

                    distances[i][j] = distances[i][k] + distances[k][j]

    return distances

# Ваш граф

graph = {

    1: {2: 125, 3: 105},

    2: {1: 125, 3: 135, 4: 75},

    3: {1: 105, 2: 135, 4: 80, 5: 140},

    4: {2: 75, 3: 80, 5: 110},

    5: {3: 140, 4: 110}

}

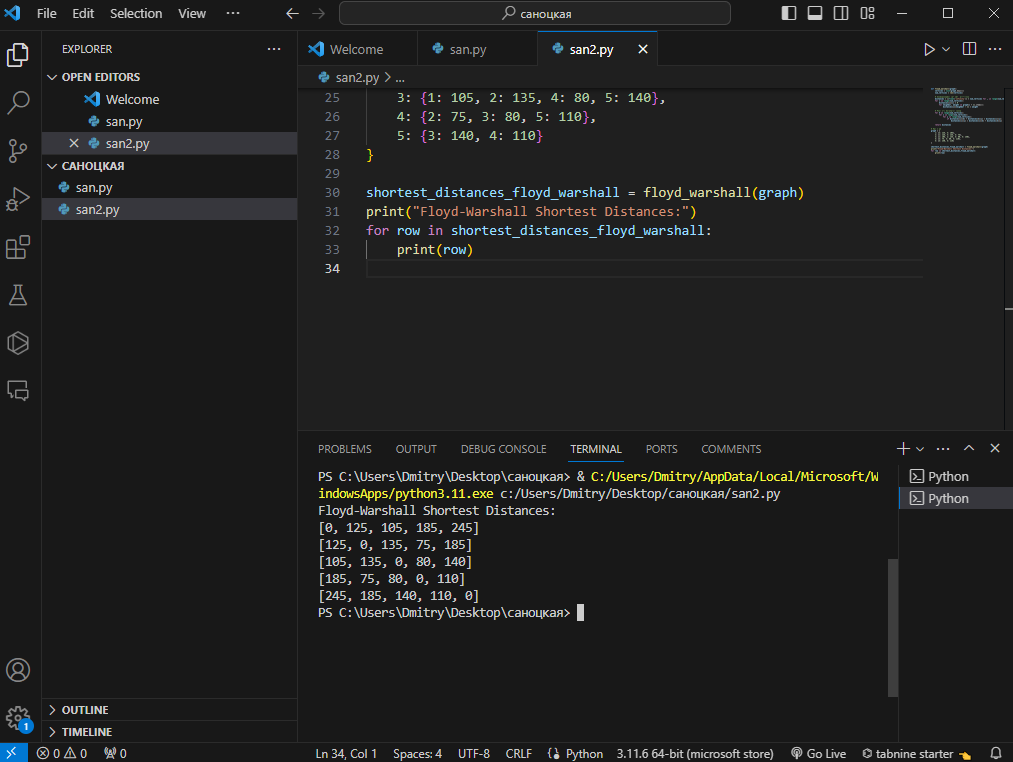
shortest\_distances\_floyd\_warshall = floyd\_warshall(graph)

print("Floyd-Warshall Shortest Distances:")

for row in shortest\_distances\_floyd\_warshall:

    print(row)

**Результат:**

****

Оба эти скрипта выводят кратчайшие расстояния от города 1 до остальных городов с использованием соответствующих алгоритмов.