**ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ ИМ. ПРОФ. М.А. БОНЧ-БРУЕВИЧА»**

**(СПбГУТ)**

**Кафедра безопасности информационных систем**

**ОТЧЁТ**

по практической работе № 9 на тему:   
**«Нахождение кратчайшего маршрута»**

по дисциплине «Алгоритмы и структуры данных»

Выполнил: студент группы ИСТ-312, Серафимович Г.П

«14» Октября 2024 г. \_\_\_\_\_\_\_\_\_\_\_/ Г.П Серафимович /

Принял: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

«15» Октября 2024 г. \_\_\_\_\_\_\_\_\_\_\_/ И.А. Моисеев /

**Содержание**

**1. Титульный лист 1**

**2. Содержание 2**

**3. Основная часть 3**

3.1. Цель работы 3

3.2. Теоретическая часть 3

3.3. Практическая часть 4

3.4. Результаты выполнения программы 6

3.5. Выводы 7

**4. Приложение 9**

**Основная часть**

**Цель работы:**

Разработать алгоритм и программу для построения маршрута, позволяющего автомобилю доставить почту с почтамта до каждого отделения связи и вернуться обратно с минимальными затратами на расстояние. Для решения задачи будет построен неориентированный граф, и применен алгоритм Дейкстры для нахождения кратчайшего пути.

**Практическая часть:**

Создает неориентированный граф с 5 вершинами и определенными ребрами.

• Находит кратчайший маршрут, который проходит через все указанные вершины (1, 2, 3, 4) и возвращается в начальную точку (0).

• Выводит все маршруты с их расстояниями и наименьшее расстояние среди всех возможных маршрутов.

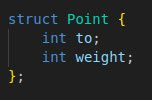


Рисунок 1.1

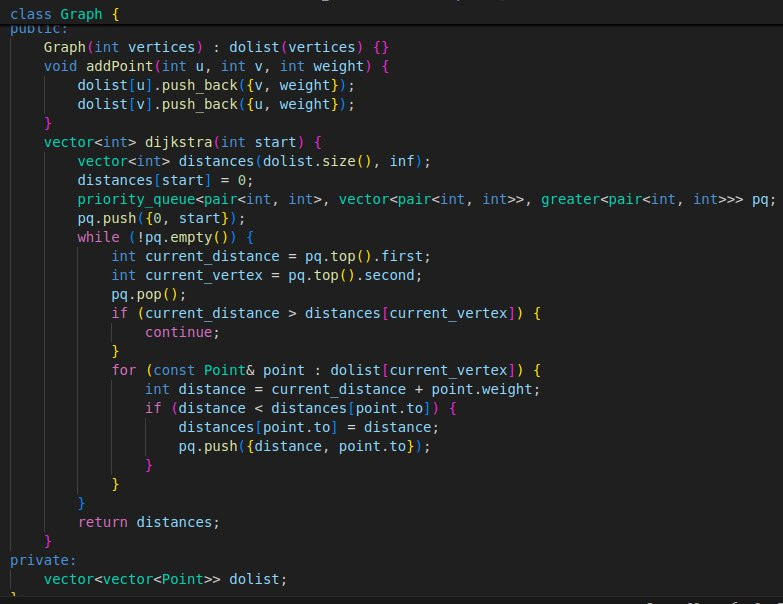


Рисунок 1.2

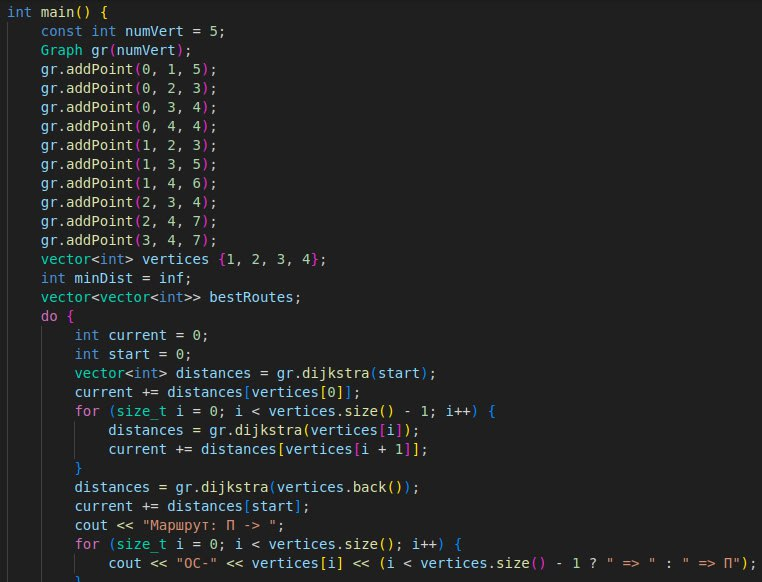


Рисунок 1.3

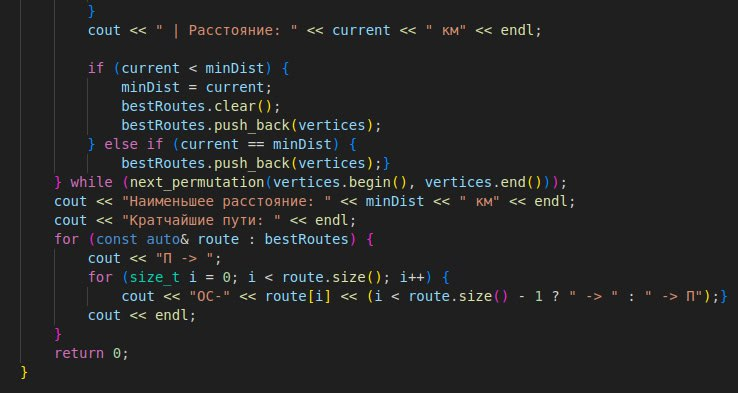


Рисунок 1.4

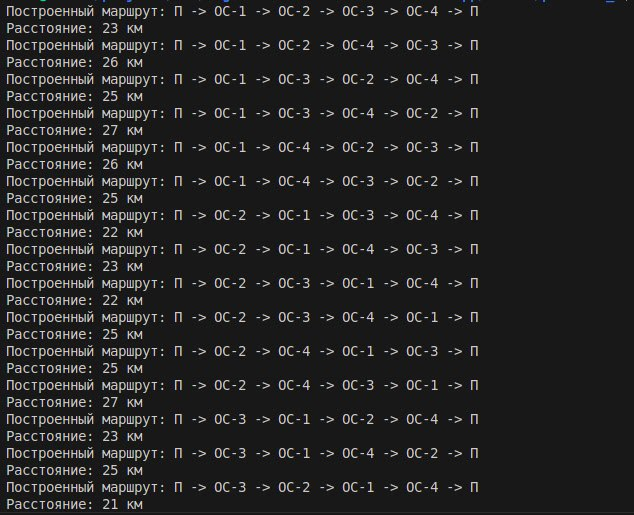


Рисунок 1.5 Результаты выполнения программы.

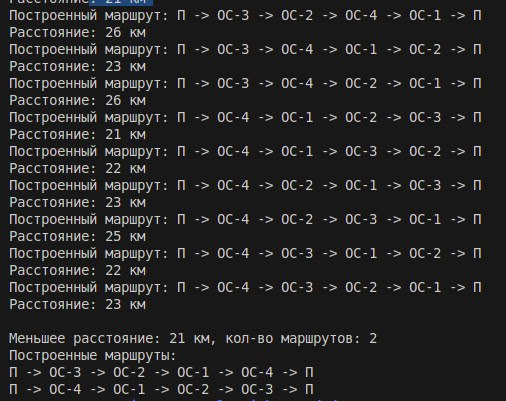


Рисунок 1.5 Результаты выполнения программы.

Для каждого значения m программа корректно находит последнего оставшегося участника.

**Выводы:**

В результате выполнения работы удалось разработать оптимальный маршрут для автомобиля, доставляющего почту между почтамтом и районными отделениями связи. При помощи алгоритма Дейкстры был построен кратчайший путь, позволяющий минимизировать затраты на транспортировку, что положительно сказывается на экономической эффективности логистических операций. Применение графовой модели и алгоритма Дейкстры обеспечило точное и быстрое вычисление опти мального маршрута, учитывая все расстояния между точками доставки.

**Приложение**

Листинг программы:

#include <iostream>

#include <vector>

#include <limits>

#include <algorithm>

#include <queue>

using namespace std;

const int inf = numeric\_limits<int>::max();

struct Point {

int to;

int weight;

};

class Graph {

public:

Graph(int vertices) : dolist(vertices) {}

void addPoint(int u, int v, int weight) {

dolist[u].push\_back({v, weight});

dolist[v].push\_back({u, weight});

}

vector<int> dijkstra(int start) {

vector<int> distances(dolist.size(), inf);

distances[start] = 0;

priority\_queue<pair<int, int>, vector<pair<int, int>>, greater<pair<int, int>>> pq;

pq.push({0, start});

while (!pq.empty()) {

int current\_distance = pq.top().first;

int current\_vertex = pq.top().second;

pq.pop();

if (current\_distance > distances[current\_vertex]) {

continue;

}

for (const Point& point : dolist[current\_vertex]) {

int distance = current\_distance + point.weight;

if (distance < distances[point.to]) {

distances[point.to] = distance;

pq.push({distance, point.to});

}

}

}

return distances;

}

private:

vector<vector<Point>> dolist;

};

int main() {

setlocale(LC\_ALL, "Russian");

const int numVert = 5;

Graph gr(numVert);

gr.addPoint(0, 1, 5);

gr.addPoint(0, 2, 3);

gr.addPoint(0, 3, 4);

gr.addPoint(0, 4, 4);

gr.addPoint(1, 2, 3);

gr.addPoint(1, 3, 5);

gr.addPoint(1, 4, 6);

gr.addPoint(2, 3, 4);

gr.addPoint(2, 4, 7);

gr.addPoint(3, 4, 7);

vector<int> vertices {1, 2, 3, 4};

int minDist = inf;

vector<vector<int>> bestRoutes;

do {

int current = 0;

int start = 0;

vector<int> distances = gr.dijkstra(start);

current += distances[vertices[0]];

for (size\_t i = 0; i < vertices.size() - 1; i++) {

distances = gr.dijkstra(vertices[i]);

current += distances[vertices[i + 1]];

}

distances = gr.dijkstra(vertices.back());

current += distances[start];

cout << "Маршрут: П -> ";

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

cout << "ОС-" << vertices[i] << (i < vertices.size() - 1 ? " => " : " => П");

}

cout << " | Расстояние: " << current << " км" << endl;

if (current < minDist) {

minDist = current;

bestRoutes.clear();

bestRoutes.push\_back(vertices);

} else if (current == minDist) {

bestRoutes.push\_back(vertices);}

} while (next\_permutation(vertices.begin(), vertices.end()));

cout << "Наименьшее расстояние: " << minDist << " км" << endl;

cout << "Кратчайшие пути: " << endl;

for (const auto& route : bestRoutes) {

cout << "П -> ";

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

cout << "ОС-" << route[i] << (i < route.size() - 1 ? " -> " : " -> П");}

cout << endl;

}

return 0;

}