

**Московский государственный технический
университет им. Н.Э. Баумана.**

Факультет «Информатика и системы управления»

Кафедра ИУ5. Курс «Парадигмы и конструкции языков программирования»

Отчет по домашнему заданию
«Поиск оптимального маршрута в метро»

Выполнил:

Каженец Д. Н.
ИУ5-31Б

Подпись и дата:

Проверил:

Нардид А. Н.

Подпись и дата:

Постановка задачи

Тема домашнего задания: создание консольного приложения на Java, которое поможет посчитать затраченное время в пути в метро в некотором вымышленном городе. Дополнительной задачей домашнего задания является разработка простого сайта с помощью Python, JS и HTML для поиска затраченного времени в метро в том же вымышленном городе.

Программа на языке Java, которая находит наиболее оптимальный путь от одной станции до другой и выводит время, необходимое для него.

```
import java.util.*;

public class UMetroGraph {
    private final Map<String, Station> stations;

    public UMetroGraph() {
        stations = new HashMap<>();
    }

    public void addStation(String name) {
        if (!stations.containsKey(name)) {
            Station station = new Station(name);
            stations.put(name, station);
        }
    }

    public void addConnection(String station1, String station2, int
travelTime) {
        if (stations.containsKey(station1) && stations.containsKey(station2))
        {
            stations.get(station1).connections.add(new Connection(station2,
travelTime));
        } else {
            System.out.println("Ошибка: одна из станций '" + station1 + "'
или '" + station2 + "' не найдена.");
        }
    }

    public void findShortestPath(String startStationName, String
endStationName) {
        Station startStation = stations.get(startStationName);
        Station endStation = stations.get(endStationName);

        if (startStation == null || endStation == null) {
            System.out.println("Ошибка: одна из станций не найдена.");
            return;
        }

        // Алгоритм Дейкстры
        PriorityQueue<Station> pq = new
PriorityQueue<>(Comparator.comparingInt(station ->
station.travelTimeFromStart));
        Map<Station, Integer> travelTimes = new HashMap<>();
        Map<Station, Station> previousStations = new HashMap<>();

        for (Station station : stations.values()) {
            travelTimes.put(station, Integer.MAX_VALUE); // Изначально все
```

```

        время в бесконечность
        previousStations.put(station, null);
    }

    travelTimes.put(startStation, 0);
    startStation.travelTimeFromStart = 0;
    pq.add(startStation);

    while (!pq.isEmpty()) {
        Station currentStation = pq.poll();

        if (currentStation.equals(endStation)) {
            printPath(previousStations, startStation, endStation);
            System.out.println("Общее время в пути: " +
travelTimes.get(endStation) + " мин.");
            return;
        }

        for (Connection connection : currentStation.connections) {
            Station neighborStation = stations.get(connection.name);
            int newTravelTime = travelTimes.get(currentStation) +
connection.travelTime;

            if (newTravelTime < travelTimes.get(neighborStation)) {
                travelTimes.put(neighborStation, newTravelTime);
                previousStations.put(neighborStation, currentStation);
                pq.add(neighborStation);
                neighborStation.travelTimeFromStart = newTravelTime; //
обновляем время в пути
            }
        }
    }

    // Если кратчайший путь не найден
    System.out.println("Нет доступного пути между " + startStationName +
" и " + endStationName + ".");
}

private static final Map<String, String> metroStations = new HashMap<>();

static {
    // Заполнение словаря названиями станций
    metroStations.put("● Аэропорт", "rAeroport");
    metroStations.put("● Есенинская", "rEseninskaya");
    metroStations.put("● Маяковская", "rMayakovskaya");
    metroStations.put("● Снегирёвская", "rSnegiryovskaya");
    metroStations.put("● Менделеевская", "rMendeleevskaya");
    metroStations.put("● Пантеон", "rPanteon");
    metroStations.put("● Театральная", "rTeatralnaya");
    metroStations.put("● Дворец Культуры", "rDvoretsKultury");
    metroStations.put("● Старый Город", "rStaryiGorod");
    metroStations.put("● Автозаводская", "rAvtozavodskaya");
    metroStations.put("● Метрогородок", "rMetrogorodok");

    metroStations.put("● Восточный Порт", "yVostochnyiPort");
    metroStations.put("● Маяковская", "yMayakovskaya");
    metroStations.put("● Лермонтовская", "yLermontovskaya");
    metroStations.put("● Пушкинская", "yPushkinskaya");
    metroStations.put("● Выставочная", "yVystavochnaya");
    metroStations.put("● Финансовая", "yFinansovaya");
}

```

```

metroStations.put("🚶 Дом Советов", "yDomSovetov");
metroStations.put("🚶 Студенческая", "yStudencheskaya");
metroStations.put("🚶 Чистые пруды", "yChistyePrudy");
metroStations.put("🚶 Внуковская", "yVnukovskaya");
metroStations.put("🚶 Автозаводская", "yAvtozavodskaya");
metroStations.put("🚶 Бабушкинская", "yBabushkinskaya");
metroStations.put("🚶 Юго-Западная", "yYugoZapadnaya");
metroStations.put("🚶 Электrozаводская", "yElektrouzavodskaya");

metroStations.put("🚶 Набережная", "bNaberezhnaya");
metroStations.put("🚶 Туристическая", "bTuristicheskaya");
metroStations.put("🚶 Янтарная", "bYantarnaya");
metroStations.put("🚶 Крылатское", "bKrylatskoe");
metroStations.put("🚶 Народное Ополчение", "bNarodnoyeOpolchenie");
metroStations.put("🚶 Студенческая", "bStudencheskaya");
metroStations.put("🚶 Физтех", "bFiztekh");
metroStations.put("🚶 Пантеон", "bPanteon");
metroStations.put("🚶 Фрунзенская", "bFrunzenskaya");
metroStations.put("🚶 Университет", "bUniversitet");
metroStations.put("🚶 Парк Победы", "bParkPobedy");
metroStations.put("🚶 Рабочая", "bRabochaya");
metroStations.put("🚶 Промышленная", "bPromyshlennaya");
metroStations.put("🚶 Юго-Восточная", "bYugoVostochnaya");
metroStations.put("🚶 ТЭЦ", "bТЕС");

metroStations.put("🚶 Западный Порт", "gZapadnyiPort");
metroStations.put("🚶 Теплый Стан", "gTyoplyStan");
metroStations.put("🚶 Янтарная", "gYantarnaya");
metroStations.put("🚶 Туристическая", "gTuristicheskaya");
metroStations.put("🚶 Выставочная", "gVystavochnaya");
metroStations.put("🚶 Министерство Юстиции",
"gMinisterstvoYustitsii");
metroStations.put("🚶 Менделеевская", "gMendeleevskaya");
metroStations.put("🚶 Рабочая", "gRabochaya");
metroStations.put("🚶 Яшьлек", "gYashlek");
metroStations.put("🚶 Окская", "gOkskaya");
metroStations.put("🚶 Улица Радио", "gUlitsaRadio");

metroStations.put("🏠 Улица Радио", "mUlitsaRadio");
metroStations.put("🏠 Снегирёвская", "mSnegiryovskaya");
metroStations.put("🏠 Внуковская", "mVnukovskaya");
}

public static String getKeyByValue(Map<String, String> map, String value)
{
    for (Map.Entry<String, String> entry : map.entrySet()) {
        if (entry.getValue().equals(value)) {
            return entry.getKey(); // Возвращаем ключ, если значение
совпадает
        }
    }
    return null; // Возвращаем null, если значение не найдено
}

private void printPath(Map<Station, Station> previousStations, Station

```

```

startStation, Station endStation) {
    List<Station> path = new ArrayList<>();
    for (Station at = endStation; at != null; at =
previousStations.get(at)) {
        path.add(at);
    }
    Collections.reverse(path); // Разворачиваем путь для правильного
порядка
    System.out.print("Кратчайший путь: " + '\n');
    for (Station station : path) {
        // Используем метки из словаря для вывода
        System.out.print(getStationName(station.name) + '\n');
    }
    System.out.println();
}

private String getStationName(String key) {
    // Получаем название станции по соответствующему ключу
    return metroStations.keySet().stream()
        .filter(station -> metroStations.get(station).equals(key))
        .findFirst()
        .orElse("Неизвестная станция");
}

public static void main(String[] args) {
    UMetroGraph metroGraph = new UMetroGraph();

    // Добавление станций Красной ветки
    metroGraph.addStation("rAeroport");
    metroGraph.addStation("rEseninskaya");
    metroGraph.addStation("rMayakovskaya");
    metroGraph.addStation("rSnegiryovskaya");
    metroGraph.addStation("rMendeleevskaya");
    metroGraph.addStation("rPanteon");
    metroGraph.addStation("rTeatralnaya");
    metroGraph.addStation("rDvoretsKultury");
    metroGraph.addStation("rStaryiGorod");
    metroGraph.addStation("rAvtozavodskaya");
    metroGraph.addStation("rMetrogorodok");

    // Добавление станций Желтой ветки
    metroGraph.addStation("yVostochnyiPort");
    metroGraph.addStation("yMayakovskaya");
    metroGraph.addStation("yLermontovskaya");
    metroGraph.addStation("yPushkinskaya");
    metroGraph.addStation("yVystavochnaya");
    metroGraph.addStation("yFinansovaya");
    metroGraph.addStation("yDomSovetov");
    metroGraph.addStation("yStudencheskaya");
    metroGraph.addStation("yChistyePrudy");
    metroGraph.addStation("yVnukovskaya");
    metroGraph.addStation("yAvtozavodskaya");
    metroGraph.addStation("yBabushkinskaya");
    metroGraph.addStation("yYugoZapadnaya");
    metroGraph.addStation("yElektrozavodskaya");

    // Добавление станций Синей ветки
    metroGraph.addStation("bNaberezhnaya");
    metroGraph.addStation("bTuristicheskaya");
    metroGraph.addStation("bYantarnaya");
    metroGraph.addStation("bKrylatskoe");
    metroGraph.addStation("bNarodnoyeOpolchenie");
    metroGraph.addStation("bStudencheskaya");
    metroGraph.addStation("bFiztekhn");
}

```

```

metroGraph.addStation("bPanteon");
metroGraph.addStation("bFrunzenskaya");
metroGraph.addStation("bUniversitet");
metroGraph.addStation("bParkPobedy");
metroGraph.addStation("bRabochaya");
metroGraph.addStation("bPromyshlennaya");
metroGraph.addStation("bYugoVostochnaya");
metroGraph.addStation("bTEC");

// Добавление станций Зеленой ветки
metroGraph.addStation("gZapadnyiPort");
metroGraph.addStation("gTyoplyStan");
metroGraph.addStation("gYantarnaya");
metroGraph.addStation("gTuristicheskaya");
metroGraph.addStation("gVystavochnaya");
metroGraph.addStation("gMinisterstvoYustitsii");
metroGraph.addStation("gMendeleevskaya");
metroGraph.addStation("gRabochaya");
metroGraph.addStation("gYashlek");
metroGraph.addStation("gOkskaya");
metroGraph.addStation("gUlitsaRadio");

// Добавление станций МЦД
metroGraph.addStation("mUlitsaRadio");
metroGraph.addStation("mSnegiryovskaya");
metroGraph.addStation("mVnukovskaya");

// Загружаем связи между станциями
metroGraph.addConnection("mUlitsaRadio", "mSnegiryovskaya", 3);
metroGraph.addConnection("mUlitsaRadio", "mVnukovskaya", 3);
metroGraph.addConnection("mSnegiryovskaya", "mUlitsaRadio", 3);
metroGraph.addConnection("mSnegiryovskaya", "mVnukovskaya", 3);
metroGraph.addConnection("mVnukovskaya", "mUlitsaRadio", 3);
metroGraph.addConnection("mVnukovskaya", "mSnegiryovskaya", 3);

metroGraph.addConnection("mUlitsaRadio", "gUlitsaRadio", 2);
metroGraph.addConnection("mSnegiryovskaya", "rSnegiryovskaya", 2);
metroGraph.addConnection("mVnukovskaya", "yVnukovskaya", 2);
metroGraph.addConnection("gUlitsaRadio", "mUlitsaRadio", 2);
metroGraph.addConnection("rSnegiryovskaya", "mSnegiryovskaya", 2);
metroGraph.addConnection("yVnukovskaya", "mVnukovskaya", 2);

// Добавляем связи для всех веток
addBranchConnections(metroGraph);

// Ввод данных пользователем для поиска кратчайшего пути
Scanner scanner = new Scanner(System.in);

System.out.println("Выберите ветку:");
System.out.println("1 - ● Красная");
System.out.println("2 - ○ Желтая");
System.out.println("3 - ● Синяя");
System.out.println("4 - ● Зелёная");
System.out.println("5 - 🚊 МЦД");

int branchChoice = scanner.nextInt();
String startStation = "";
String endStation = "";

switch (branchChoice) {
    case 1:
        System.out.println("Выберите станцию:\n" +
            "1 - ● Аэропорт\n" +

```

```


        "2 - ☒ Есенинская\n" +
        "3 - ☒ Маяковская\n" +
        "4 - ☒ Снегирёвская\n" +
        "5 - ☒ Менделеевская\n" +
        "6 - ☒ Пантеон\n" +
        "7 - ☒ Театральная\n" +
        "8 - ☒ Дворец Культуры\n" +
        "9 - ☒ Старый Город\n" +
        "10 - ☒ Автозаводская\n" +
        "11 - ☒ Метрогородок");
    startStation = selectStation(scanner, "Красная");
    break;
case 2:
    System.out.println("Выберите станцию:\n" +
        "1 - ☐ Восточный Порт\n" +
        "2 - ☐ Маяковская\n" +
        "3 - ☐ Лермонтовская\n" +
        "4 - ☐ Пушкинская\n" +
        "5 - ☐ Выставочная\n" +
        "6 - ☐ Финансовая\n" +
        "7 - ☐ Дом Советов\n" +
        "8 - ☐ Студенческая\n" +
        "9 - ☐ Чистые пруды\n" +
        "10 - ☐ Внуковская\n" +
        "11 - ☐ Автозаводская\n" +
        "12 - ☐ Бабушкинская\n" +
        "13 - ☐ Юго-Западная\n" +
        "14 - ☐ Электрозаводская");
    startStation = selectStation(scanner, "Желтая");
    break;
case 3:
    System.out.println("Выберите станцию:\n" +
        "1 - ☐ Набережная\n" +
        "2 - ☐ Туристическая\n" +
        "3 - ☐ Янтарная\n" +
        "4 - ☐ Крылатское\n" +
        "5 - ☐ Народное Ополчение\n" +
        "6 - ☐ Студенческая\n" +
        "7 - ☐ Физтех\n" +
        "8 - ☐ Пантеон\n" +
        "9 - ☐ Фрунзенская\n" +
        "10 - ☐ Университет\n" +
        "11 - ☐ Парк Победы\n" +
        "12 - ☐ Рабочая\n" +
        "13 - ☐ Промышленная\n" +
        "14 - ☐ Юго-Восточная\n" +
        "15 - ☐ ТЭЦ");
    startStation = selectStation(scanner, "Синяя");
    break;
case 4:
    System.out.println("Выберите станцию:\n" +
        "1 - ☐ Западный Порт\n" +
        "2 - ☐ Теплый Стан\n" +
        "3 - ☐ Янтарная\n" +

```



```

        "4 -  Туристическая\n" +
        "5 -  Выставочная\n" +
        "6 -  Министерство Юстиции\n" +
        "7 -  Менделеевская\n" +
        "8 -  Рабочая\n" +
        "9 -  Яшьлек\n" +
        "10 -  Окская\n" +
        "11 -  Улица Радио");
        startStation = selectStation(scanner, "Зеленая");
        break;
    case 5:
        System.out.println("Выберите станцию:\n" +
            "1 -  Улица Радио\n" +
            "2 -  Снегирёвская\n" +
            "3 -  Внуковская");
        startStation = selectStation(scanner, "МЦД");
        break;
    default:
        System.out.println("Некорректный выбор ветки!");
        return;
}

System.out.println("Выбранная станция: " +
    getKeyByValue(metroStations, startStation));

// Повторяем выбор для конечной станции с аналогичными опциями
System.out.println("Выберите ветку:");
System.out.println("1 -  Красная");
System.out.println("2 -  Желтая");
System.out.println("3 -  Синяя");
System.out.println("4 -  Зелёная");
System.out.println("5 -  МЦД");

branchChoice = scanner.nextInt();

switch (branchChoice) {
    case 1:
        System.out.println("Выберите станцию:\n" +
            "1 -  Аэропорт\n" +
            "2 -  Есенинская\n" +
            "3 -  Маяковская\n" +
            "4 -  Снегирёвская\n" +
            "5 -  Менделеевская\n" +
            "6 -  Пантеон\n" +
            "7 -  Театральная\n" +
            "8 -  Дворец Культуры\n" +
            "9 -  Старый Город\n" +
            "10 -  Автозаводская\n" +
            "11 -  Метрогородок");
        endStation = selectStation(scanner, "Красная");
        break;
    case 2:
        System.out.println("Выберите станцию:\n" +
            "1 -  Восточный Порт\n" +
            "2 -  Маяковская\n" +
            "3 -  Лермонтовская\n" +
            "4 -  Пушкинская\n" +

```



```

        "5 - ☐ Выставочная\n" +
        "6 - ☐ Финансовая\n" +
        "7 - ☐ Дом Советов\n" +
        "8 - ☐ Студенческая\n" +
        "9 - ☐ Чистые пруды\n" +
        "10 - ☐ Внуковская\n" +
        "11 - ☒ Автозаводская\n" +
        "12 - ☐ Бабушкинская\n" +
        "13 - ☐ Юго-Западная\n" +
        "14 - ☐ Электrozаводская");
    endStation = selectStation(scanner, "Желтая");
    break;
case 3:
    System.out.println("Выберите станцию:\n" +
        "1 - ☐ Набережная\n" +
        "2 - ☒ Туристическая\n" +
        "3 - ☐ Янтарная\n" +
        "4 - ☐ Крылатское\n" +
        "5 - ☐ Народное Ополчение\n" +
        "6 - ☒ Студенческая\n" +
        "7 - ☐ Физтех\n" +
        "8 - ☐ Пантеон\n" +
        "9 - ☒ Фрунзенская\n" +
        "10 - ☐ Университет\n" +
        "11 - ☐ Парк Победы\n" +
        "12 - ☒ Рабочая\n" +
        "13 - ☒ Промышленная\n" +
        "14 - ☒ Юго-Восточная\n" +
        "15 - ☐ ТЭЦ");
    endStation = selectStation(scanner, "Синяя");
    break;
case 4:
    System.out.println("Выберите станцию:\n" +
        "1 - ☐ Западный Порт\n" +
        "2 - ☒ Теплый Стан\n" +
        "3 - ☒ Янтарная\n" +
        "4 - ☐ Туристическая\n" +
        "5 - ☐ Выставочная\n" +
        "6 - ☒ Министерство Юстиции\n" +
        "7 - ☐ Менделеевская\n" +
        "8 - ☐ Рабочая\n" +
        "9 - ☐ Яшьлек\n" +
        "10 - ☒ Окская\n" +
        "11 - ☐ Улица Радио");
    endStation = selectStation(scanner, "Зеленая");
    break;
case 5:
    System.out.println("Выберите станцию:\n" +
        "1 -  Улица Радио\n" +
        "2 -  Снегирёвская\n" +
        "3 -  Внуковская");
    endStation = selectStation(scanner, "МЦД");
    break;
default:
    System.out.println("Некорректный выбор ветки!");

```

```

        return;
    }

    System.out.println("Конечная станция: " +
        getKeyByValue(metroStations, endStation));

    // Поиск кратчайшего пути
    metroGraph.findShortestPath(startStation, endStation);
}

private static void addBranchConnections(UMetroGraph metroGraph) {
    // Добавление связей по всем станциям и их веткам
    // Красная ветка
    String[] redStations = {"rAeroport", "rEseninskaya", "rMayakovskaya",
        "rSnegiryovskaya",
        "rMendeleevskaya", "rPanteon", "rTeatralnaya",
        "rDvoretsKultury",
        "rStaryiGorod", "rAvtozavodskaya", "rMetrogorodok"};

    for (int i = 0; i < redStations.length - 1; i++) {
        metroGraph.addConnection(redStations[i], redStations[i + 1], 1);
        metroGraph.addConnection(redStations[i + 1], redStations[i], 1);
    }

    metroGraph.addConnection("rMayakovskaya", "yMayakovskaya", 2);
    metroGraph.addConnection("rPanteon", "bPanteon", 2);
    metroGraph.addConnection("rMendeleevskaya", "gMendeleevskaya", 2);
    metroGraph.addConnection("rAvtozavodskaya", "yAvtozavodskaya", 2);

    // Желтая ветка
    String[] yellowStations = {"yVostochnyiPort", "yMayakovskaya",
        "yLermontovskaya",
        "yPushkinskaya", "yVystavochnaya", "yFinansovaya",
        "yDomSovetov",
        "yStudencheskaya", "yChistyePrudy", "yVnukovskaya",
        "yAvtozavodskaya", "yBabushkinskaya", "yYugoZapadnaya",
        "yElektrouzavodskaya"};

    for (int i = 0; i < yellowStations.length - 1; i++) {
        metroGraph.addConnection(yellowStations[i], yellowStations[i +
1], 1);
        metroGraph.addConnection(yellowStations[i + 1],
yellowStations[i], 1);
    }

    metroGraph.addConnection("yMayakovskaya", "rMayakovskaya", 2);
    metroGraph.addConnection("yStudencheskaya", "bStudencheskaya", 2);
    metroGraph.addConnection("yVystavochnaya", "gVystavochnaya", 2);
    metroGraph.addConnection("yAvtozavodskaya", "rAvtozavodskaya", 2);

    // Синяя ветка
    String[] blueStations = {"bNaberezhnaya", "bTuristicheskaya",
        "bYantarnaya",
        "bKrylatskoe", "bNarodnoyeOpolchenie", "bStudencheskaya",
        "bFiztekhn", "bPanteon", "bFrunzenskaya", "bUniversitet",
        "bParkPobedy", "bRabochaya", "bPromyshlennaya",
        "bYugoVostochnaya", "bTEC"};

    for (int i = 0; i < blueStations.length - 1; i++) {
        metroGraph.addConnection(blueStations[i], blueStations[i + 1],
1);
        metroGraph.addConnection(blueStations[i + 1], blueStations[i],
1);
    }
}

```

```

metroGraph.addConnection("bStudencheskaya", "yStudencheskaya", 2);
metroGraph.addConnection("bPanteon", "rPanteon", 2);
metroGraph.addConnection("bRabochaya", "gRabochaya", 2);
metroGraph.addConnection("bYantarnaya", "gYantarnaya", 2);
metroGraph.addConnection("bTuristicheskaya", "gTuristicheskaya", 2);

// Зеленая ветка
String[] greenStations = {"gZapadnyiPort", "gTyoplyStan",
    "gYantarnaya",
        "gTuristicheskaya", "gVystavochnaya",
    "gMinisterstvoYustitsii",
        "gMendeleevskaya", "gRabochaya", "gYashlek", "gOkskaya",
        "gUlitsaRadio"};

for (int i = 0; i < greenStations.length - 1; i++) {
    metroGraph.addConnection(greenStations[i], greenStations[i + 1],
1);
    metroGraph.addConnection(greenStations[i + 1], greenStations[i],
1);
}

metroGraph.addConnection("gRabochaya", "bRabochaya", 2);
metroGraph.addConnection("gYantarnaya", "bYantarnaya", 2);
metroGraph.addConnection("gTuristicheskaya", "bTuristicheskaya", 2);
metroGraph.addConnection("gVystavochnaya", "yVystavochnaya", 2);
metroGraph.addConnection("gMendeleevskaya", "rMendeleevskaya", 2);
}

private static String selectStation(Scanner scanner, String branch) {
    // Список станций по веткам
    Map<String, String[]> branches = new HashMap<>();
    branches.put("Красная", new String[]{"rAeroport", "rEseninskaya",
    "rMayakovskaya",
        "rSnegiryovskaya", "rMendeleevskaya", "rPanteon",
    "rTeatralnaya",
        "rDvoretsKultury", "rStaryiGorod", "rAvtozavodskaya",
    "rMetrogorodok"});

    branches.put("Желтая", new String[]{"yVostochnyiPort",
    "yMayakovskaya", "yLermontovskaya",
        "yPushkinskaya", "yVystavochnaya", "yFinansovaya",
    "yDomSovetov",
        "yStudencheskaya", "yChistyePrudy", "yVnukovskaya",
        "yAvtozavodskaya", "yBabushkinskaya", "yYugoZapadnaya",
    "yElektrouzavodskaya"});

    branches.put("Синяя", new String[]{"bNaberezhnaya",
    "bTuristicheskaya", "bYantarnaya",
        "bKrylatskoe", "bNarodnoyeOpolchenie", "bStudencheskaya",
        "bFiztekhn", "bPanteon", "bFrunzenskaya", "bUniversitet",
        "bParkPobedy", "bRabochaya", "bPromyshlennaya",
        "bYugoVostochnaya", "bTEC"});

    branches.put("Зеленая", new String[]{"gZapadnyiPort", "gTyoplyStan",
    "gYantarnaya",
        "gTuristicheskaya", "gVystavochnaya",
    "gMinisterstvoYustitsii",
        "gMendeleevskaya", "gRabochaya", "gYashlek", "gOkskaya",
        "gUlitsaRadio"});

    branches.put("МЦД", new String[]{"mUlitsaRadio", "mSnegiryovskaya",
    "mVnukovskaya"});
}

```

```

String[] stations = branches.get(branch);

int stationChoice = scanner.nextInt();
if (stationChoice < 1 || stationChoice > stations.length) {
    System.out.println("Некорректный выбор станции!");
    return "";
}
return stations[stationChoice - 1];
}

private static class Station {
    String name;
    List<Connection> connections;
    int travelTimeFromStart; // добавляем это поле для хранения времени
из начала

    Station(String name) {
        this.name = name;
        this.connections = new ArrayList<>();
        this.travelTimeFromStart = Integer.MAX_VALUE; // изначально
установим максимально возможное значение
    }
}

private static class Connection {
    String name;
    int travelTime;

    Connection(String name, int travelTime) {
        this.name = name;
        this.travelTime = travelTime;
    }
}
}

```

Выберите ветку:

- 1 - ● Красная
- 2 - ● Желтая
- 3 - ● Синяя
- 4 - ● Зелёная
- 5 - 🚊 МЦД

2

Выберите станцию:

- 1 - ● Восточный Порт
- 2 - ● Маяковская
- 3 - ● Лермонтовская
- 4 - ● Пушкинская
- 5 - ● Выставочная
- 6 - ● Финансовая
- 7 - ● Дом Советов
- 8 - ● Студенческая
- 9 - ● Чистые пруды
- 10 - ● Внуковская
- 11 - ● Автозаводская
- 12 - ● Бабушкинская
- 13 - ● Юго-Западная
- 14 - ● Электрозаводская

10

Выбранная станция: ● Внуковская

Выберите ветку:

- 1 - ● Красная
- 2 - ● Желтая
- 3 - ● Синяя
- 4 - ● Зелёная
- 5 - 🚊 МЦД

4

Выберите станцию:

- 1 - ● Западный Порт
- 2 - ● Теплый Стан
- 3 - ● Янтарная
- 4 - ● Туристическая
- 5 - ● Выставочная
- 6 - ● Министерство Юстиции
- 7 - ● Менделеевская
- 8 - ● Рабочая
- 9 - ● Яшьлек
- 10 - ● Окская
- 11 - ● Улица Радио

7

Конечная станция: ● Менделеевская

Кратчайший путь:

- Внуковская
- Чистые пруды
- Студенческая
- Дом Советов
- Финансовая
- Выставочная
- Выставочная
- Министерство Юстиции
- Менделеевская

Общее время в пути: 9 мин.

Пользователь выбирает начальную и конечную станции, после чего выводится наиболее оптимальный путь и кратчайшее время.

Логика работы программы на языке Python

```
import heapq
from flask import Flask, render_template, request, jsonify

app = Flask(__name__)

class Station:
    def __init__(self, name):
        self.name = name
        self.connections = []

class MetroGraph:
    def __init__(self):
        self.stations = {}

    def add_station(self, name):
        if name not in self.stations: # Проверяем, что станция еще не
добавлена
            station = Station(name)
            self.stations[name] = station

    def add_connection(self, station1, station2, travel_time):
        if station1 in self.stations and station2 in self.stations:
            self.stations[station1].connections.append((station2,
travel_time))
        else:
            print(f"Ошибка: одна из станций '{station1}' или '{station2}' не
найдена.")

    def display_connections(self):
        if not self.stations:
            print("Нет доступных станций.")
            return

        print("Связи между станциями:")
        for station_name, station in self.stations.items():
            connections_output = ', '.join([f"{conn[0]} (время: {conn[1]}
мин)" for conn in station.connections])
            print(f"Станция {station_name}: {connections_output if
connections_output else 'нет соединений'}")

metro_graph = MetroGraph()

# Добавление станций Красной ветки
metro_graph.add_station("rAeroport")
metro_graph.add_station("rEseninskaya")
metro_graph.add_station("rMayakovskaya")
metro_graph.add_station("rSnegiryovskaya")
metro_graph.add_station("rMendeleevskaya")
metro_graph.add_station("rPanteon")
metro_graph.add_station("rTeatralnaya")
metro_graph.add_station("rDvoretsKultury")
metro_graph.add_station("rStaryiGorod")
metro_graph.add_station("rAvtozavodskaya")
metro_graph.add_station("rMetrogorodok")

# Добавление станций Желтой ветки
metro_graph.add_station("yVostochnyiPort")
metro_graph.add_station("yMayakovskaya")
metro_graph.add_station("yLermontovskaya")
metro_graph.add_station("yPushkinskaya")
```

```

metro_graph.add_station("yVystavochnaya")
metro_graph.add_station("yFinansovaya")
metro_graph.add_station("yDomSovetov")
metro_graph.add_station("yStudencheskaya")
metro_graph.add_station("yChistyePrudy")
metro_graph.add_station("yVnukovskaya")
metro_graph.add_station("yAvtozavodskaya")
metro_graph.add_station("yBabushkinskaya")
metro_graph.add_station("yYugoZapadnaya")
metro_graph.add_station("yElektrouzavodskaya")

# Добавление станций Синей ветки
metro_graph.add_station("bNaberezhnaya")
metro_graph.add_station("bTuristicheskaya")
metro_graph.add_station("bYantarnaya")
metro_graph.add_station("bKrylatskoe")
metro_graph.add_station("bNarodnoyeOpolchenie")
metro_graph.add_station("bStudencheskaya")
metro_graph.add_station("bFiztekh")
metro_graph.add_station("bPanteon")
metro_graph.add_station("bFrunzenskaya")
metro_graph.add_station("bUniversitet")
metro_graph.add_station("bParkPobedy")
metro_graph.add_station("bRabochaya")
metro_graph.add_station("bPromyshlennaya")
metro_graph.add_station("bYugoVostochnaya")
metro_graph.add_station("bTEC")

# Добавление станций Зеленой ветки
metro_graph.add_station("gZapadnyiPort")
metro_graph.add_station("gTyoplyStan")
metro_graph.add_station("gYantarnaya")
metro_graph.add_station("gTuristicheskaya")
metro_graph.add_station("gVystavochnaya")
metro_graph.add_station("gMinisterstvoYustitsii")
metro_graph.add_station("gMendeleevskaya")
metro_graph.add_station("gRabochaya")
metro_graph.add_station("gYashlek")
metro_graph.add_station("gOkskaya")
metro_graph.add_station("gUlitsaRadio")

# добавление станций МЦД
metro_graph.add_station("mUlitsaRadio")
metro_graph.add_station("mSnegiryovskaya")
metro_graph.add_station("mVnukovskaya")

metro_stations = {
    "🟢 Аэропорт": "rAeroport",
    "🟢 Есенинская": "rEseninskaya",
    "🟢 Маяковская": "rMayakovskaya",
    "🟢 Снегирёвская": "rSnegiryovskaya",
    "🟢 Менделеевская": "rMendeleevskaya",
    "🟢 Пантеон": "rPanteon",
    "🟢 Театральная": "rTeatralnaya",
    "🟢 Дворец Культуры": "rDvoretsKultury",
    "🟢 Старый Город": "rStaryiGorod",
    "🟢 Автозаводская": "rAvtozavodskaya",
    "🟢 Метрогородок": "rMetrogorodok",

    "🟡 Восточный Порт": "yVostochnyiPort",
    "🟡 Маяковская": "yMayakovskaya",

```

```

"🕒 Лермонтовская": "yLermontovskaya",
"🕒 Пушкинская": "yPushkinskaya",
"🕒 Выставочная": "yVystavochnaya",
"🕒 Финансовая": "yFinansovaya",
"🕒 Дом Советов": "yDomSovetov",
"🕒 Студенческая": "yStudencheskaya",
"🕒 Чистые пруды": "yChistyePrudy",
"🕒 Внуковская": "yVnukovskaya",
"🕒 Автозаводская": "yAvtozavodskaya",
"🕒 Бабушкинская": "yBabushkinskaya",
"🕒 Юго-Западная": "yYugoZapadnaya",
"🕒 Электrozаводская": "yElektrouzavodskaya",

"🕒 Набережная": "bNaberezhnaya",
"🕒 Туристическая": "bTuristicheskaya",
"🕒 Янтарная": "bYantarnaya",
"🕒 Крылатское": "bKrylatskoe",
"🕒 Народное Ополчение": "bNarodnoyeOpolchenie",
"🕒 Студенческая": "bStudencheskaya",
"🕒 Физтех": "bFiztek",
"🕒 Пантеон": "bPanteon",
"🕒 Фрунзенская": "bFrunzenskaya",
"🕒 Университет": "bUniversitet",
"🕒 Парк Победы": "bParkPobedy",
"🕒 Рабочая": "bRabochaya",
"🕒 Промышленная": "bPromyshlennaya",
"🕒 Юго-Восточная": "bYugoVostochnaya",
"🕒 ТЭЦ": "bТЕС",

"🕒 Западный Порт": "gZapadnyiPort",
"🕒 Теплый Стан": "gTyoplyStan",
"🕒 Янтарная": "gYantarnaya",
"🕒 Туристическая": "gTuristicheskaya",
"🕒 Выставочная": "gVystavochnaya",
"🕒 Министерство Юстиции": "gMinisterstvoYustitsii",
"🕒 Менделеевская": "gMendeleevskaya",
"🕒 Рабочая": "gRabochaya",
"🕒 Яшьлек": "gYashlek",
"🕒 Окская": "gOkskaya",
"🕒 Улица Радио": "gUlitsaRadio",

"🚶 Улица Радио": "mUlitsaRadio",
"🚶 Снегирёвская": "mSnegiryovskaya",
"🚶 Внуковская": "mVnukovskaya"
}

metro_graph.add_connection("mUlitsaRadio", "mSnegiryovskaya", 3)
metro_graph.add_connection("mUlitsaRadio", "mVnukovskaya", 3)
metro_graph.add_connection("mSnegiryovskaya", "mUlitsaRadio", 3)
metro_graph.add_connection("mSnegiryovskaya", "mVnukovskaya", 3)
metro_graph.add_connection("mVnukovskaya", "mUlitsaRadio", 3)
metro_graph.add_connection("mVnukovskaya", "mSnegiryovskaya", 3)

metro_graph.add_connection("mUlitsaRadio", "gUlitsaRadio", 2)
metro_graph.add_connection("mSnegiryovskaya", "rSnegiryovskaya", 2)

```



```

metro_graph.add_connection("mVnukovskaya", "yVnukovskaya", 2)
metro_graph.add_connection("gUlitsaRadio", "mUlitsaRadio", 2)
metro_graph.add_connection("rSnegiryovskaya", "mSnegiryovskaya", 2)
metro_graph.add_connection("yVnukovskaya", "mVnukovskaya", 2)

# Добавляем связи: имя станции, с которой идет связь, имя станции, куда идет
# связь, время в пути
# Красная ветка
red_stations = ["rAeroport", "rEseninskaya", "rMayakovskaya",
                "rSnegiryovskaya",
                "rMendeleevskaya", "rPanteon", "rTeatralnaya",
                "rDvoretsKultury",
                "rStaryiGorod", "rAvtozavodskaya", "rMetrogorodok"]

# Связи на Красной ветке (1 минута между соседними станциями)
for i in range(len(red_stations) - 1):
    metro_graph.add_connection(red_stations[i], red_stations[i + 1], 1)
    metro_graph.add_connection(red_stations[i + 1], red_stations[i], 1)

# Переходы на Красной ветке
metro_graph.add_connection("rMayakovskaya", "yMayakovskaya", 2)
metro_graph.add_connection("rPanteon", "bPanteon", 2)
metro_graph.add_connection("rMendeleevskaya", "gMendeleevskaya", 2)
metro_graph.add_connection("rAvtozavodskaya", "yAvtozavodskaya", 2)

# Желтая ветка
yellow_stations = ["yVostochnyiPort", "yMayakovskaya", "yLermontovskaya",
                  "yPushkinskaya", "yVystavochnaya", "yFinansovaya",
                  "yDomSovetov", "yStudencheskaya", "yChistyePrudy",
                  "yVnukovskaya", "yAvtozavodskaya", "yBabushkinskaya",
                  "yYugoZapadnaya", "yElektrouzavodskaya"]

# Связи на Желтой ветке (1 минута между соседними станциями)
for i in range(len(yellow_stations) - 1):
    metro_graph.add_connection(yellow_stations[i], yellow_stations[i + 1], 1)
    metro_graph.add_connection(yellow_stations[i + 1], yellow_stations[i], 1)

# Переходы на Желтой ветке
metro_graph.add_connection("yMayakovskaya", "rMayakovskaya", 2)
metro_graph.add_connection("yStudencheskaya", "bStudencheskaya", 2)
metro_graph.add_connection("yVystavochnaya", "gVystavochnaya", 2)
metro_graph.add_connection("yAvtozavodskaya", "rAvtozavodskaya", 2)

# Синяя ветка
blue_stations = ["bNaberezhnaya", "bTuristicheskaya", "bYantarnaya",
                 "bKrylatskoe",
                 "bNarodnoyeOpolchenie", "bStudencheskaya", "bFiztekhn",
                 "bPanteon", "bFrunzenskaya", "bUniversitet",
                 "bParkPobedy", "bRabochaya", "bPromyshlennaya",
                 "bYugoVostochnaya", "bTEC"]

# Связи на Синей ветке (1 минута между соседними станциями)
for i in range(len(blue_stations) - 1):
    metro_graph.add_connection(blue_stations[i], blue_stations[i + 1], 1)
    metro_graph.add_connection(blue_stations[i + 1], blue_stations[i], 1)

# Переходы на Синей ветке
metro_graph.add_connection("bStudencheskaya", "yStudencheskaya", 2)
metro_graph.add_connection("bPanteon", "rPanteon", 2)
metro_graph.add_connection("bRabochaya", "gRabochaya", 2)
metro_graph.add_connection("bYantarnaya", "gYantarnaya", 2)
metro_graph.add_connection("bTuristicheskaya", "gTuristicheskaya", 2)

# Зеленая ветка

```

```

green_stations = ["gZapadnyiPort", "gTyoplyStan", "gYantarnaya",
                  "gTuristicheskaya",
                  "gVystavochnaya", "gMinisterstvoYustitsii",
                  "gMendeleevskaya",
                  "gRabochaya", "gYashlek", "gOkskaya", "gUlitsaRadio"]

# Связи на Зеленой ветке (1 минута между соседними станциями)
for i in range(len(green_stations) - 1):
    metro_graph.add_connection(green_stations[i], green_stations[i + 1], 1)
    metro_graph.add_connection(green_stations[i + 1], green_stations[i], 1)

# Переходы на Зеленой ветке
metro_graph.add_connection("gRabochaya", "bRabochaya", 2)
metro_graph.add_connection("gYantarnaya", "bYantarnaya", 2)
metro_graph.add_connection("gTuristicheskaya", "bTuristicheskaya", 2)
metro_graph.add_connection("gVystavochnaya", "yVystavochnaya", 2)
metro_graph.add_connection("gMendeleevskaya", "rMendeleevskaya", 2)

def calculate_travel_time(graph, start_station, end_station):
    # Устанавливаем начальные значения
    distances = {station: float('inf') for station in graph.stations}
    distances[start_station] = 0
    priority_queue = [(0, start_station)]
    previous_stations = {station: None for station in graph.stations}

    while priority_queue:
        current_distance, current_station = heapq.heappop(priority_queue)

        if current_distance > distances[current_station]:
            continue

        for neighbor, travel_time in
graph.stations[current_station].connections:
            distance = current_distance + travel_time

            if distance < distances[neighbor]:
                distances[neighbor] = distance
                previous_stations[neighbor] = current_station
                heapq.heappush(priority_queue, (distance, neighbor))

    # Восстанавливаем путь
    path = []
    while end_station is not None:
        path.append(end_station)
        end_station = previous_stations[end_station]

    path.reverse()
    reversed_metro_stations = {v: k for k, v in metro_stations.items()}
    translated_path = []
    for station in path:
        translated_station = reversed_metro_stations.get(station) #
Используем .get() для безопасного получения
        if translated_station:
            translated_path.append(translated_station)
    p = [path[i][0] for i in range(len(path))]
    stations = "\n".join(translated_path)
    #return stations
    return f"Ваш путь:\n{stations}\nВремя в пути - {distances[path[-1]]}
минут,\nКоличество пересадок - {len(set(p)) - 1}"

@app.route('/')
def index():

```

```

    return render_template('index.html')

@app.route('/calculate', methods=['POST'])
def calculate():
    data = request.json
    from_station = data.get('from')
    to_station = data.get('to')

    # Вызываем вашу функцию вычисления времени
    travel_time = calculate_travel_time(metro_graph, from_station,
to_station)

    # Возвращаем результат в формате JSON
    return jsonify({'time': travel_time})

if __name__ == "__main__":
    app.run(debug=True)

# start = input()
# end = input()
# path, travel_time = dijkstra(metro_graph, metro_stations[start],
metro_stations[end])
# reversed_metro_stations = {v: k for k, v in metro_stations.items()}
# if path:
#     for i in range(len(path)):
#         path[i] = reversed_metro_stations[path[i]]
#     stations = " -> ".join(path)
#     print("Ваш путь:")
#     print(stations)
#     print(f"Общее время в пути: {travel_time} минут")
#     p = [path[i][0] for i in range(len(path))]
#     print(f"Количество пересадок - {len(set(p))-1}.")
# else:
#     print("К сожалению, путь не найден.")

```

Серверное приложение на Node.js с использованием фреймворка Express, которое моделирует метрополитен и предоставляет API для вычисления времени в пути между станциями

```

const express = require('express');
const bodyParser = require('body-parser');

const app = express();
app.use(bodyParser.json());

class Station {
    constructor(name) {
        this.name = name;
        this.connections = [];
    }
}

class MetroGraph {
    constructor() {
        this.stations = {};
    }
}

```

```

    addStation(name) {
        if (!this.stations[name]) { // Проверяем, что станция еще не
            добавлена
            const station = new Station(name);
            this.stations[name] = station;
        }
    }

    addConnection(station1, station2, travelTime) {
        if (this.stations[station1] && this.stations[station2]) {
            this.stations[station1].connections.push([station2, travelTime]);
        } else {
            console.error(`Ошибка: одна из станций '${station1}' или
                '${station2}' не найдена.`);
        }
    }

    displayConnections() {
        if (Object.keys(this.stations).length === 0) {
            console.log("Нет доступных станций.");
            return;
        }

        console.log("Связи между станциями:");
        for (const stationName in this.stations) {
            const connectionsOutput = this.stations[stationName].connections
                .map(conn => `${conn[0]} (время: ${conn[1]} мин)`)
                .join(', ');
            console.log(`Станция ${stationName}: ${connectionsOutput} || 'нет
                соединений'`);
        }
    }
}

const metroGraph = new MetroGraph();

// Добавление станций Красной, Желтой, Синей и Зеленой веток
const redStations = ["rAeroport", "rEseninskaya", "rMayakovskaya",
    "rSnegiryovskaya", "rMendeleevskaya",
    "rPanteon", "rTeatralnaya", "rDvoretzKultury", "rStaryiGorod",
    "rAvtozavodskaya", "rMetrogorodok"];
const yellowStations = ["yVostochnyiPort", "yMayakovskaya",
    "yLermontovskaya", "yPushkinskaya",
    "yVystavochnaya", "yFinansovaya", "yDomSovetov", "yStudencheskaya",
    "yChistyePrudy",
    "yVnukovskaya", "yAvtozavodskaya", "yBabushkinskaya", "yYugoZapadnaya",
    "yElektrozavodskaya"];
const blueStations = ["bNaberezhnaya", "bTuristicheskaya", "bYantarnaya",
    "bKrylatskoe",
    "bNarodnoyeOpolchenie", "bStudencheskaya", "bFiztekhn", "bPanteon",
    "bFrunzenskaya",
    "bUniversitet", "bParkPobedy", "bRabochaya", "bPromyshlennaya",
    "bYugoVostochnaya", "bTEC"];
const greenStations = ["gZapadnyiPort", "gTyoplyStan", "gYantarnaya",
    "gTuristicheskaya",
    "gVystavochnaya", "gMinisterstvoYustitsii", "gMendeleevskaya",
    "gRabochaya", "gYashlek",
    "gOkskaya", "gUlitsaRadio"];

// Функция для добавления станций и соединений
function addStationsAndConnections(stations) {
    stations.forEach(station => metroGraph.addStation(station));
    for (let i = 0; i < stations.length - 1; i++) {
        metroGraph.addConnection(stations[i], stations[i + 1], 1);
    }
}

```

```

        metroGraph.addConnection(stations[i + 1], stations[i], 1);
    }
}

// Добавляем станции и соединения
addStationsAndConnections(redStations);
addStationsAndConnections(yellowStations);
addStationsAndConnections(blueStations);
addStationsAndConnections(greenStations);

// Пример соединений между станциями
const connections = [
    ["mUlitsaRadio", "mSnegiryovskaya", 3],
    ["mUlitsaRadio", "mVnukovskaya", 3],
    ["mSnegiryovskaya", "mVnukovskaya", 3],
    ["mUlitsaRadio", "gUlitsaRadio", 2],
    ["gUlitsaRadio", "mUlitsaRadio", 2],
    ["rSnegiryovskaya", "mSnegiryovskaya", 2],
    ["yVnukovskaya", "mVnukovskaya", 2]
];

// Добавление соединений
connections.forEach(([station1, station2, time]) => {
    metroGraph.addConnection(station1, station2, time);
});

// Функция для вычисления времени в пути
function calculateTravelTime(graph, startStation, endStation) {
    // Устанавливаем начальные значения
    const distances = {};
    const previousStations = {};
    const priorityQueue = [];

    for (const station in graph.stations) {
        distances[station] = Infinity;
        previousStations[station] = null;
    }
    distances[startStation] = 0;
    priorityQueue.push([0, startStation]);

    while (priorityQueue.length > 0) {
        const [currentDistance, currentStation] = priorityQueue.shift();

        if (currentDistance > distances[currentStation]) {
            continue;
        }

        for (const [neighbor, travelTime] of
graph.stations[currentStation].connections) {
            const distance = currentDistance + travelTime;

            if (distance < distances[neighbor]) {
                distances[neighbor] = distance;
                previousStations[neighbor] = currentStation;
                priorityQueue.push([distance, neighbor]);
            }
        }
    }

    // Восстанавливаем путь
    const path = [];
    let current = endStation;
    while (current !== null) {
        path.push(current);
    }
}

```

```

        current = previousStations[current];
    }
    path.reverse();

    // Формируем строку для вывода
    const stationsNames = path.join('\n');
    return `Ваш путь:\n${stationsNames}\nВремя в пути -
    ${distances[path[path.length - 1]]} минут`;
}

// REST API
app.get('/', (req, res) => {
    res.send('Метро API'); // Здесь можно вернуть HTML или что-то другое
});

app.post('/calculate', (req, res) => {
    const { from, to } = req.body;

    // Вызываем функцию для вычисления времени
    const travelTime = calculateTravelTime(metroGraph, from, to);

    // Возвращаем результат в формате JSON
    res.json({ time: travelTime });
});

// Запуск сервера
const PORT = process.env.PORT || 3000;
app.listen(PORT, () => {
    console.log(`Сервер запущен на порту ${PORT}`);
});

```

Функциональность для вычисления времени поездки между двумя станциями: отправление запроса на сервер и отображение результата на веб-странице.

```

document.getElementById('calculateButton').addEventListener('click',
function() {
    const fromStation = document.getElementById('fromStation').value;
    const toStation = document.getElementById('toStation').value;

    fetch('/calculate', {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json'
        },
        body: JSON.stringify({ from: fromStation, to: toStation })
    })
    .then(response => response.json())
    .then(data => {
        document.getElementById('result').textContent = `${data.time}`;
    })
    .catch(error => {
        console.error('Ошибка:', error);
    });
});

```

CSS-стили, которые применяются к HTML-документу для оформления веб-страницы.

```
body {
  font-family: Arial, sans-serif;
  background-color: #f4f4f4;
  display: flex;
  justify-content: center;
  align-items: center;
  height: 100vh;
  margin: 0;
}

.wrapper {
  display: flex;
  justify-content: center;
  align-items: center;
  height: 100vh; /* Высота на весь экран */
}

.container {
  background: white;
  padding: 20px;
  border-radius: 5px;
  box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
  width: 200px; /* Установка ширины контейнера */
  margin-right: 20px; /* Отступ вправо для контейнера */
}

.right-image {
  max-width: 900px; /* Установка максимальной ширины изображения */
  height: auto; /* Автоматическая высота для поддержания пропорций */
  border-radius: 5px; /* Закругление углов изображения */
}

h1 {
  font-size: 24px;
  margin-bottom: 20px;
}

label {
  font-weight: bold;
}

select {
  width: 100%;
  padding: 10px;
  margin: 10px 0;
}

button {
  padding: 10px 15px;
  background-color: #007bff;
  color: white;
  border: none;
  border-radius: 5px;
  cursor: pointer;
}

button:hover {
  background-color: #0056b3;
}
```

```

/* Уменьшение стилей для итогового результата */
h2#result {
    margin-top: 20px;
    font-size: 18px; /* Уменьшение размера шрифта */
    padding: 10px; /* Уменьшение отступов вокруг текста */
    background-color: #e9ecef; /* Светлый фон для выделения результата */
    border-radius: 5px; /* Закругления углов */
    text-align: center; /* Центрирование текста */
}

```

HTML-документ, который создает веб-страницу для вычисления времени в пути на метро.

```

<!DOCTYPE html>
<html lang="ru">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Время в пути на метро</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='styles.css')
}}">
</head>
<body>
    <div class="container">
        <h1>Вычислить время в пути на метро</h1>
        <label for="fromStation">Выберите станцию откуда:</label>
        <select id="fromStation">
            <option value="rAeroport">🚏 Аэропорт</option>
            <option value="rEseninskaya">🚏 Есенинская</option>
            <option value="rMayakovskaya">🚏 Маяковская</option>
            <option value="rSnegiryovskaya">🚏 Снегирёвская</option>
            <option value="rMendeleevskaya">🚏 Менделеевская</option>
            <option value="rPanteon">🚏 Пантеон</option>
            <option value="rTeatralnaya">🚏 Театральная</option>
            <option value="rDvoretsKultury">🚏 Дворец Культуры</option>
            <option value="rStaryiGorod">🚏 Старый Город</option>
            <option value="rAvtozavodskaya">🚏 Автозаводская</option>
            <option value="rMetrogorodok">🚏 Метрогородок</option>
            <option value="yVostochnyiPort">🚏 Восточный Порт</option>
            <option value="yMayakovskaya">🚏 Маяковская</option>
            <option value="yLermontovskaya">🚏 Лермонтовская</option>
            <option value="yPushkinskaya">🚏 Пушкинская</option>
            <option value="yVystavochnaya">🚏 Выставочная</option>
            <option value="yFinansovaya">🚏 Финансовая</option>
            <option value="yDomSovetov">🚏 Дом Советов</option>
            <option value="yStudencheskaya">🚏 Студенческая</option>
            <option value="yChistyePrudy">🚏 Чистые пруды</option>
            <option value="yVnukovskaya">🚏 Внуковская</option>
            <option value="yAvtozavodskaya">🚏 Автозаводская</option>
            <option value="yBabushkinskaya">🚏 Бабушкинская</option>
            <option value="yYugoZapadnaya">🚏 Юго-Западная</option>
            <option value="yElektrozavodskaya">🚏 Электрозаводская</option>
            <option value="bNaberezhnaya">🚏 Набережная</option>

```



```

        <option value="bTuristicheskaya">● Туристическая</option>
        <option value="bYantarnaya">● Янтарная</option>
        <option value="bKrylatskoe">● Крылатское</option>
        <option value="bNarodnoyeOpolchenie">● Народное
Ополчение</option>
        <option value="bStudencheskaya">● Студенческая</option>
        <option value="bFiztekhn">● Физтех</option>
        <option value="bPanteon">● Пантеон</option>
        <option value="bFrunzenskaya">● Фрунзенская</option>
        <option value="bUniversitet">● Университет</option>
        <option value="bParkPobedy">● Парк Победы</option>
        <option value="bRabochaya">● Рабочая</option>
        <option value="bPromyshlennaya">● Промышленная</option>
        <option value="bYugoVostochnaya">● Юго-Восточная</option>
        <option value="bTEC">● ТЭЦ</option>
        <option value="gZapadnyiPort">● Западный Порт</option>
        <option value="gTyoplyStan">● Тёплый Стан</option>
        <option value="gYantarnaya">● Янтарная</option>
        <option value="gTuristicheskaya">● Туристическая</option>
        <option value="gVystavochnaya">● Выставочная</option>
        <option value="gMinisterstvoYustitsii">● Министерство
Юстиции</option>
        <option value="gMendeleevskaya">● Менделеевская</option>
        <option value="gRabochaya">● Рабочая</option>
        <option value="gYashlek">● Яшьлек</option>
        <option value="gOkskaya">● Окская</option>
        <option value="gUlitsaRadio">● Улица Радио</option>
        <option value="mUlitsaRadio">🏠 Улица Радио</option>
        <option value="mVnukovskaya">🏠 Внуковская</option>
        <option value="mSnegiryovskaya">🏠 Снегирёвская</option>
    </select>

    <label for="toStation">Выберите станцию куда:</label>
    <select id="toStation">
        <option value="rAeroport">● Аэропорт</option>
        <option value="rEseninskaya">● Есенинская</option>
        <option value="rMayakovskaya">● Маяковская</option>
        <option value="rSnegiryovskaya">● Снегирёвская</option>
        <option value="rMendeleevskaya">● Менделеевская</option>
        <option value="rPanteon">● Пантеон</option>
        <option value="rTeatralnaya">● Театральная</option>
        <option value="rDvoretsKultury">● Дворец Культуры</option>
        <option value="rStaryiGorod">● Старый Город</option>
        <option value="rAvtozavodskaya">● Автозаводская</option>
        <option value="rMetrogorodok">● Метрогородок</option>
        <option value="yVostochnyiPort">● Восточный Порт</option>
        <option value="yMayakovskaya">● Маяковская</option>
        <option value="yLermontovskaya">● Лермонтовская</option>
        <option value="yPushkinskaya">● Пушкинская</option>
        <option value="yVystavochnaya">● Выставочная</option>
        <option value="yFinansovaya">● Финансовая</option>
        <option value="yDomSovetov">● Дом Советов</option>
        <option value="yStudencheskaya">● Студенческая</option>

```

```

<option value="yChistyePrudy">● Чистые пруды</option>
<option value="yVnukovskaya">● Внуковская</option>
<option value="yAvtozavodskaya">● Автозаводская</option>
<option value="yBabushkinskaya">● Бабушкинская</option>
<option value="yYugoZapadnaya">● Юго-Западная</option>
<option value="yElektrouzavodskaya">● Электроузовская</option>
<option value="bNaberezhnaya">● Набережная</option>
<option value="bTuristicheskaya">● Туристическая</option>
<option value="bYantarnaya">● Янтарная</option>
<option value="bKrylatskoe">● Крылатское</option>
<option value="bNarodnoyeOpolchenie">● Народное
Ополчение</option>
<option value="bStudencheskaya">● Студенческая</option>
<option value="bFiztekhn">● Физтех</option>
<option value="bPanteon">● Пантеон</option>
<option value="bFrunzenskaya">● Фрунзенская</option>
<option value="bUniversitet">● Университет</option>
<option value="bParkPobedy">● Парк Победы</option>
<option value="bRabochaya">● Рабочая</option>
<option value="bPromyshlennaya">● Промышленная</option>
<option value="bYugoVostochnaya">● Юго-Восточная</option>
<option value="bТЕС">● ТЭЦ</option>
<option value="gZapadnyiPort">● Западный Порт</option>
<option value="gTyoplyStan">● Тёплый Стан</option>
<option value="gYantarnaya">● Янтарная</option>
<option value="gTuristicheskaya">● Туристическая</option>
<option value="gVystavochnaya">● Выставочная</option>
<option value="gMinisterstvoYustitsii">● Министерство
Юстиции</option>
<option value="gMendeleevskaya">● Менделеевская</option>
<option value="gRabochaya">● Рабочая</option>
<option value="gYashlek">● Яшьлек</option>
<option value="gOkskaya">● Окская</option>
<option value="gUlitsaRadio">● Улица Радио</option>
<option value="mUlitsaRadio">🏠 Улица Радио</option>
<option value="mVnukovskaya">🏠 Внуковская</option>
<option value="mSnegiryovskaya">🏠 Снегирёвская</option>
</select>

<button id="calculateButton">Вычислить время</button>
<h2 id="result"></h2>
</div>

</div>
<script src="{{ url_for('static', filename='script.js') }}"></script>
</body>
</html>

```

Получившаяся веб-страница

Вычислить время в пути на метро

Выберите станцию откуда:

● Аэропорт

▼

Выберите станцию куда:

● Лермонтовская

▼

Вычислить время

Ваш путь:

● Аэропорт

● Есенинская

● Маяковская

● Лермонтовская

Время в пути - 5 минут, Количество пересадок - 1