GatherRound

Отчет 07.12.2024

Цель проекта: создать приложение, которое упростит компаниям из нескольких друзей поиск культурного события (концерта, фестиваля, выставки) для встречи.

Место выбирается так, чтобы время, потраченное на дорогу каждым участником было оптимальным.

В качестве транспорта рассматривается метрополитен как самый доступный.

Задачи, которые были выполнены ранее:

- Настроить инструмент для измерения покрытия кода на Kotlin тестами.
- Подать заявку и получить API-ключ портала открытых данных правительства Москвы.
- Реализовать класс для представления взвешенного графа.
- Реализовать и покрыть тестами алгоритм Дейкстры поиска и восстановления кратчайших путей в взвешенном графе.
- Реализовать класс для хранения информации о станциях метро.
- Получить необходимые данные о станциях московского метро (внутри кольцевой линии) и времени в пути между ними. Распарсить и сохранить данные для использования программой.

Выполненная задача:

Реализовать и протестировать функцию построения оптимального маршрута между станциями метро.

Описание:

Нахождение оптимального (с минимальным временем поездки) пути между станциями основано на алгоритме Дейкстры.

Для проверки корректности функции поиска и восстановления оптимального пути между станциями написаны тесты двух видов:

1). Базовая проверка: алгоритм Дейкстры запускается от произвольно выбранной станции. Проверяется, что алгоритм дошел до всех станций, т.е. в массиве расстояний нет бесконечностей и у каждой станции кроме стартовой найден предок (помогло обнаружить ошибку в компараторе упорядоченного множества – пары (расстояние, вершина) сравнивались только по первому элементу (расстоянию) и считались равными в случае равенства расстояния, из-за чего некоторые вершины не были добавлены в множество).

2). Вручную выбраны фиксированные 3 пары станций. Результат работы функции вручную сравнивается с результатами mosmetro.ru. Результат считается корректным, если совпадают последовательности станций в оптимальных маршрутах.

Также проведено сравнение времени в поездке – результат работы моей программы отличается от результатов с сайта выше не более чем на 3% (или не более чем на 2 минуты). Значит, используемая моей программой информация о времени проезда между соседними станциями (датирующаяся 2018 г.) до сих пор достаточно точна.

Выполненная задача:

Реализовать класс для хранения информации о мероприятиях.

Описание:

Класс Place имеет поля, соответствующие следующим характеристикам места проведения мероприятий:
- уникальный id,
- название
- адрес,
- координаты (широта, долгота),

В дальнейшем по мере необходимости будут добавлены и другие характеристики в соответствии с документацией API KudaGo.ru:
- расписание работы,
- статус (открыто или закрыто) на момент обращения
- Контактный телефон
- категории места
- описание
- фотографии

и другие.

```kotlin
@Serializable
data class Place(
    val id: Int,
    val title: String,
    val address: String,
    val coords: Coordinates? = null,
    @Serializable(with = SubwayDeserializer::class)
    val subway: List<String> = emptyList(),
) {
    ivanz-thinkpad *
    @Serializable
    data class Coordinates(
        val lat: Double?,
        val lon: Double?
    )
```

Задача:

Реализовать функцию поиска станции метро, максимальной расстояние от которой до множества выбранных станций минимально.

Алгоритм:

Рассмотрим целочисленную функцию f(x). f(x) = 1, если существует такая станция, что до нее можно доехать от любой станции из множества менее чем за x секунд. Иначе f(x) = 0. Заметим, что функция f монотонна: для всех значений x < X она возвращает 0, а для x >= X она возвращает 1 (X – некоторое фиксированное число). Следовательно, можно применить бинарный поиск.

Пусть мы зафиксировали время в пути t. Теперь надо эффективно проверить, существует ли станция, достижимая от любой из станций множества менее чем за t секунд. Для этого выполним следующий предподсчет: для каждой из k станций из множества при помощи алгоритма Дейкстры найдем расстояния до всех остальных станций и отсортируем все станции по возрастанию расстояния (предподсчет выполняется до запуска бинарного поиска). Вернемся к бинпоиску. Поскольку для каждой из k станций из множества у нас есть список станций, отсортированных по времени в пути о них, задача сводится к следующей. Даны k отсортированных массивов. Проверить, есть ли в них общее число.

Решить полученную задачу можно следующим образом. Найдем массив, в котором минимальное число максимально. Очевидно, что все меньшие числа нам точно не подходят, поскольку они не содержатся в рассматриваемом массиве. Выкинем все меньшие числа из остальных массивов. Проверим, равны ли минимальные числа во всех массивах. Если это так, то искомое число найдено. Иначе повторяем описанные выше шаги пока искомое число не найдется либо массивы не опустеют. Описанный алгоритм реализуется методом указателей.

Время работы:

Пусть k – число стартовых станций в множестве, n – общее число станций в метро.

Тогда предподсчет работает за O(k nlogn) (т.к. для каждой станции вызывается сначала алгоритм Дейкстры, а затем сортировка списка всех вершин)

Проверка в бинпоиске работает за O(k n), поскольку указатель в каждом массиве движется только вперед. То есть бинпоиск работает за O(k n log C), где C – максимальное время поездки в метро в секундах

Итоговая сложность алгоритма: O(k n log C)

Применение:

API KudaGo.ru предоставляет данные о ближайших к местам проведения мероприятий станциях метро.

Пользуясь этими данными и описанным выше алгоритмом, будет реализована функция поиска событий, максимальное расстояние до которых от станций из списка минимально.