Старинная задачка о паровозах

Исходная постановка

Задана железнодорожная сеть. Сеть состоит из конечного количества станций, соединенных (однонаправленными) железнодорожными перегонами. Для каждого перегона задана длина (положительное целое число).
Задано некоторое количество «паровозов». Паровоз — точечный объект. Для каждого паровоза задан маршрут в виде списка станций, через которые он проходит. По прибытию на последнюю станцию паровоз исчезает. Паровозы стартуют по маршруту одновременно со скоростью 1.

Программа должна считывать из файла начальные данные (структуру железнодорожной сети и маршруты) и отвечать на вопрос: столкнуться ли два каких-либо паровоза? (дополнительно можно реализовать ответ на вопрос какие и где).

Требования к формату файлов не предъявляются. Корректность данных можно не проверять.

Очевидные следствия

Из условий:

- Однонаправленность всех перегонов
- Одинаковая скорость всех паровозов

следует, что на перегонах между станциями паровозы столкнуться никак не могут. Столкновения происходят только на станциях. Иными словами, приходим к определению:

Столкновение (crash) — это одновременное прибытие двух (или более) паровозов на одну станцию.

В этом смысле, если начальные станции двух (или более) паровозов совпадают, то этот факт также является столкновением.

Алгоритм

Алгоритм суперпростой. Функция RailNet.findCrash собирает все «прибытия» всех паровозов (engine) на все станции в массив arrivals. Каждое прибытие представлено объектом класса

class Arrival (val time:Int, val station:Int, val engine: Int)

Затем массив arrivals сортируется в порядке возрастания значений <time, station, engine>. Эта сортировка выстраивает «одновременные прибытия» в непрерывные участки массива и их легко выбрать оттуда за один проход по этому (отсортированному) массиву.

Заметим, что столкновения не разрушают паровозы и станции. После столкновения паровозы как ни в чем ни бывало продолжают движение по своим маршрутам и могут участвовать в новых столкновениях. Пример example/crash1.xml определяет сеть из 11 станций и 10 паровозов, в которой происходит 14 столкновений.

Конфигурация сети

Ж/д сеть определяется файлом в формате XML.

Корневой узел специфицирует число станций и паровозов, например:

<rail_net StatNum="11" EngineNum="10">

….

</rail_net>

тут определяется сеть из 11 станций и 10 паровозов.

Перегоны между станциями соответствуют узлам <branch>, например:

<branch From="1" To="10" Length="2"/>

отражает существование перегона со станции 1 на станцию 10, длина которого равна 2.

Узлы <route> и <track> специфицируют маршруты паровозов. Например конструкция:

<route Engine ="3">

<track Stat="0"/>

<track Stat="10"/>

<track Stat="0"/>

<track Stat="5"/>

</route>

говорит, что паровоз 3 начинает движение со станции 0, затем идет на станцию 10, возвращается на станцию 0 и потом устремляется на станцию 5, где он «исчезает».

Запуск программы

Под SBT прога запускается командой

run <path to XML network definition>

По содержимому XML файла строится объект класса RailNet, который отражает конфигурацию данной сети. При этом отлавливаются критически важные ошибки конфигурации (например, отсутствие перегонов между станциями, через которые проходит маршрут паровоза). При обнаружении ошибки, на консоль выдается соответствующее сообщение и выполнение прекращается.

Если сеть признана (статически) корректной, то в ней запускается поиск столкновений (RailNet.findCrash). При обнаружении таковых, на консоль выдается инфа о каждой трагедии (время, станция, паровозы).

Примеры

В папку Example я положил несколько примеров сетей с различными результатами выполнения.

Все примеры — маленькие. Я поленился генерить сети с большим (сотни-тысячи) числом объектов. Но если «заказчик» желает, то я сделаю это. Думаю, что с обработкой большой сети у данной примитивной проги никаких проблем не будет.