GSLetterNeo vol.143

2020年6月

GTFS 、駅が同じルートを見る

松原 伸人 matubara@sra.co.jp

はじめに

Vol.137、139、141と GTFS データを読み込んで時刻と経路を Web ブラウザで地図上に 3D 的に表示する方法を紹介してきました。今回は GTFS データにどのようなルートがあるのか見ていきます。

ルートに関するファイルとフィールド

ルートに関する情報は routes.txt と trips.txt と stop_times.txt と stops.txt の4つのファイルに書かれています。それぞれ定義があり作成するデータとして必須のフィールドが決められています。

routes.txt にルートの名称と ID が route_id、route_short_name、route_long_name の3つのフィールドへ記載されています。このうち route_id は trips.txt にも書かれており、どのルートの旅程なのかが分かるようになっています。

trips.txt は各ルートの旅程を定義しています。route_id、service_id、trip_id の3つのフィールドが必須となっています。service_id はcalendar.txt に定義している日付のセットを識別する ID です。月曜から金曜の平日に運行するとか、土日のみとか、特定の祝祭日限定とか運行する日や曜日で運行ダイヤを表しています。trips.txt の任意フィールドに必要に応じて、11:34発東京行きとか、山手線内回りの何時発の列車なのかなど旅程に関する情報が記載されます。trip_id は stop_times.txt にも書かれていて、どの旅程に関する発着時刻の情報なのか識別できるようになっています。

stop_times.txt には、各旅程で始発駅から終着駅までにとおる停車駅ごとに、停車順序と発着時刻が書かれています。必須フィールドの trip_id、arrival_time、departure_time、stop_id、stop_sequence の4つが必須項目です。arrival_time が到着時刻、departure_time が発車時刻、stop_sequence は停車順序、stop_id は stops.txt に記載の駅を参照する駅の ID です。

stops.txt は駅の ID と名前や駅の場所の経緯度など駅に関する情報が書かれています。

同じルートでも停車する駅は様々

各ルートがどの駅にどういう順番に停車するのかは、ルートの名称が同じでも旅程により様々です。例えば各駅停車のときと特急の場合で同じ線路をとおるとしても停車する駅としない駅がちがうことがあります。上りと下りで停車する駅は同じですが進行方向は反対なので停車順序はちがいます。ある旅程の始発から終点までに停車する駅の順序は、旅程の trip_id が書かれている発着時刻情報を stop_times.txt から発着時刻の配列を取り出して stop_sequence に書いてある停車順に並べることでわかります。

リファレンス | GTFS

リファレンス | *Static Transit | Google Developers*

```
let route_ids = Object.keys(gtfs.routeByRoute_id)
let route_id = route_ids[0]
let trip = gtfs.tripByRoute_id[route_id][0]
let stopsOfTrip = gtfs.stop_timeByTrip_id[trip.trip_id].sort((s1, s2) => {
    return parseInt(s1.stop_sequence) - parseInt(s2.stop_sequence)
})
console.log(collectStopsOfTrip(trip))
```

gtfs.routeByRoute_id は、routes.txt から読み込んだルートの route_id をキーとして route_id ごとにルートの配列を登録してあります。gtfs.tripByRoute_id には trips.txt から読み込んだ旅程の route_id をキーとして route_id ごとの旅程を配列にして登録しています。登録されている1つ目のルートの1つ目の旅程の駅を停車順に並べ替えた配列を得てコンソールに出力しています。gtfs.stop_timeByTrip_id は、stop_times.txt から読み込んだ発着時刻情報の trip_id をキーとして trip_id ごとに発着時刻情報の配列を登録してある object です。

次のコードでは、全旅程につき停車順の駅の ID 配列を得ます。

```
let stopsByTrip_id = {}
route_ids.forEach((route_id) => {
    let trips = gtfs.tripByRoute_id[route_id]
    if (!trips) {
        return
    }
    trips.forEach((trip) => {
        let stopsOfTrip = gtfs.stop_timeByTrip_id[trip.trip_id].sort((s1, s2) => {
            return parseInt(s1.stop_sequence) - parseInt(s2.stop_sequence)
        }).map(stop_time => stop_time.stop_id)
        stopsByTrip_id[trip.trip_id] = stopsOfTrip
    })
})
console.log(stopsByTrip_id, Object.keys(stopsByTrip_id).length)
```

停車する駅の連なりをくらべて旅程をグループにする

各旅程についてそれぞれ始発から終点までに停車する駅の ID の配列が得られました。この駅の ID の配列同士を比べて、同じ順番に同じ ID が並んでいれば、2つの旅程の停車する駅の連なりが同じだとわかります。

```
let tripsByRouteByStops_id = {}
let trip_ids = Object.keys(stopsByTrip_id)
trip_ids.forEach((trip_id) => {
    let stops = stopsByTrip_id[trip_id].stops
    let trip = stopsByTrip_id[trip_id].trip
    let stops_id = stops.join('->')
    let tripsByRoute = tripsByRouteByStops_id[stops_id]
    if (!tripsByRoute) {
        tripsByRoute = {}
        tripsByRouteByStops_id[stops_id] = tripsByRoute
    }
    let trips = tripsByRoute[trip.route_id]
    if (!trips) {
        trips = []
        tripsByRoute[trip.route_id] = trips
    }
    trips.push(trip)
})
console.log(tripsByRouteByStops_id, Object.keys(tripsByRouteByStops_id).length)
```

停車駅の ID の配列を '->' 文字列でつないで連結した文字列を作成し、これを停車駅の連なりの ID とします。あとはこの ID をキーとして同じ ID ごとに旅程を配列にまとめます。

ルートごとに見ると停車駅が同じで、発着時刻がちがうルートがたくさん出てくるので、停車駅の連なりでグループ分けしてさらにルート ID ごとのグループに分けています。

停車する駅を parent_station の連なりにかえる

コンソールに出力されたグループ数を見ると、ルートの本数とあまり変わらないことがあります。GTFS では stops.txt に駅のデータが書かれていますが、駅には5種類あり stop.location_type に定義されています。location_type が 0 の 場所は乗り降りする停車地を表します。stop_times.txt の stop_id はすべて停車地の ID が書かれています。そのため、同じ駅なのに違う駅のならびになっています。stop.parent_station に停車地の親階層の駅舎とかターミナルを表す stop_id が書かれています。

乗降場所の違いを気にせずに同じ駅としてグループ分けする場合は parent_station で駅の連なりを作ります。

```
let tripsByRouteByStops_id2 = {}
trip_ids.forEach((trip_id) => {
    let stops = stopsByTrip_id[trip_id].stops
    let trip = stopsByTrip_id[trip_id].trip
    let stops_id = stops.map(stop_id => gtfs.stopByStop_id[stop_id].parent_station || stop_id).join('->')
    let tripsByRoute = tripsByRouteByStops_id2[stops_id]
    if (!tripsByRoute) {
        tripsByRoute = {}
        tripsByRouteByStops_id2[stops_id] = tripsByRoute
    }
    let trips = tripsByRoute[trip.route_id]
    if (!trips) {
        trips = []
        tripsByRoute[trip.route_id] = trips
    }
    trips.push(trip)
})
console.log(tripsByRouteByStops_id2, Object.keys(tripsByRouteByStops_id2).length)
```

前のプログラムとの違いは 5行目 で駅の ID を '->' つなぐ際に parent_station を使用しているところだけです。これにより、例えば、同じ駅に停車するけれど、旅程によって乗り場が異なるような路線も同じ停車の連なりとしてグループ分けできます。

逆にすると停車する駅の連なりが同じになる旅程もグループにする

上りと下りとで停車する駅がちがうことが時々ありますが、知らずに出かけて降車できずに乗り過ごすなんてことがあるかもしれません。停車駅の連なりの ID の逆順に並び替えて作った文字列と比較すると、このような旅程を見つけられます。

```
let tripsByRouteByStops_id3 = {}
trip_ids.forEach((trip_id) => {
    let stops = stopsByTrip_id[trip_id].stops
    let trip = stopsByTrip_id[trip_id].trip
    let stops_id = stops.map(stop_id => gtfs.stopByStop_id[stop_id].parent_station || stop_id).join('->')
    let stops_id_reverse = stops.map(stop_id => gtfs.stopByStop_id[stop_id].parent_station || stop_id).reverse().join('->')
    let tripsByRoute = tripsByRouteByStops_id3[stops_id]
    if (!tripsByRoute) {
        tripsByRoute = tripsByRouteByStops_id3[stops_id_reverse]
    }
    if (!tripsByRoute) {
        tripsByRoute = {}
        tripsByRouteByStops_id3[stops_id] = tripsByRoute
    }
    let trips = tripsByRoute[trip.route_id]
    if (!trips) {
        trips = []
        tripsByRoute[trip.route_id] = trips
    }
    trips.push(trip)
})
console.log(tripsByRouteByStops_id3, Object.keys(tripsByRouteByStops_id3).length)
```

今回は駅の連なりが同じになるルートを見つけかたをいくつか紹介しました。掲載したプログラムは次の URL で試せるようになっています。

https://www3.sra.co.jp/ktl/beam/test-output_stops_of_trip.html

URL を開き GTFS ファイルが入っているフォルダを選ぶと上記の5つのプログラムを実行しコンソールに結果を出力します。

GTFS の運行情報にかぎらずデータのならびに特徴があるようなデータ、例えば、文字が並んでいる文とかテキスト、ある程度データの種類が決められるようなデータは、同じようにデータの連なりを探すことができると思います。特別な道具を用意せずとも割と短いコードで書けるので少し試してみるようなときなどにいかがでしょうか。

GSLetterNeo vol.143
発行日 2020年6月20日
発行者 株式会社 ＳＲＡ 先端技術研究所
編集者 土屋 正人
バックナンバー https://www.sra.co.jp/gsletter/

お問い合わせ
gsneo@sra.co.jp
〒171-8513 東京都豊島区南池袋2-32-8
株式会社SRA