OpenTripPlanner Code Overview: Focusing on Routing

Marcy Gordon

3-15-11

Code Revision: 1196

Projects within OTP

- opentripplanner-routing
  - Core routing algorithms, data-structures, and libraries
- opentripplanner-api-webapp
  - Web app providing a REST API to the trip planning engine
- opentripplanner-webapp
  - Web app providing web-based UI to trip planning engine
- opentripplanner-graph-builder
  - Command line tool for configuring and building the trip planner graph

Projects within OTP

- opentripplanner-gui
  - Graph visualizer for development and troubleshooting
- opentripplanner-integration
  - Integration tests
- opentripplanner-api-extended
  - Webapp that can optionally be enabled to display a system map; requires a geoserver
- opentripplanner-utils
  - Encodes polylines (probably for shapefiles)
- opentripplaner-geocoder
  - Webapp that provides a REST API used by OTP to geocode places; uses other geocoders – doesn't actually geocode

opentripplanner-graph-builder

GRAPH BUILDER PROJECT

Graph Builder Project

- graph-builder.xml
  - Configuration file that specifies graph builder tasks
- Graph builder tasks
  - Usually beans of class org.opentripplanner.graph_builder.impl.????GraphBuilderImpl
  - Important tasks
    - GtfsGraphBuilderImpl, OpenStreetMapGraphBuilderImpl, NEDGraphBuilderImpl, ShapefileStreetGraphBuilderImpl

Building the Graph

- org.opentripplanner.graph_builder
  - GraphBuilderMain.java
    - Gets each GraphBuilderTask from the config file and calls run
  - GraphBuilderTask.java
    - Run method builds the graph and creates a ContractionHierarchySet

opentripplaner-routing

ROUTING PROJECT

Routing Algorithms

- org.opentripplanner.routing.algorithm
  - A-Star and Dijkstra used for regular graph
- org.opentripplanner.routing.contraction
  - Used for contracted hierarchies
  - Modified bi-directional Dijkstra's algorithm
  - Magic happens in org.opentripplanner.routing.contraction.ContractionHierarchy.getShortestPath(…)

Edges

- org.opentripplanner.routing.edgetype
- Edges usually extend AbstractEdge class and implement another edge class
- Most implement StreetEdge which is an interface with 3 important methods
  - canTraverse(…);
  - getLength()
  - getPermission()

org.opentripplanner.routing.edgetype

- Alight.java
- ArrayTripPattern.java
- BasicTripPattern.java
- Board.java
- DummyableEdge.java
- Dwell.java
- EdgeWithElevation.java
- EndpointVertex.java
- FixedModeEdge.java
- FreeEdge.java
- Hop.java
- OnBoardForwardEdge.java
- OnBoardReverseEdge.java
- OutEdge.java
- PathwayEdge.java
- PatternAlight.java
- PatternBoard.java
- PatternDwell.java
- PatternEdge.java
- PatternHop.java
- PatternInterlineDwell.java
- PlainStreetEdge.java
- PreAlightEdge.java
- PreBoardEdge.java
- StreetEdge.java
- StreetTransitLink.java
- StreetTraversalPermission.java
- StreetVertex.java
- TransferEdge.java
- TripPattern.java
- TurnEdge.java

Vertices

- OTP uses a edge-based graph but it does have vertices
- org.opentripplanner.routing.core
  - GenericVertex.java
  - GraphVertex.java
  - Vertex.java
  - VertexIngress.java
- org.opentripplanner.routing.edgetype
  - EndpointVertex.java
  - StreetVertex.java

AFFECTING ROUTING

How to Affect Routing

- Building the graph
  - Read in new data and store it in the graph
  - May need to modify edge classes in the routing project to store new info
  - OSM tags are not stored in the graph
- Routing on the graph
  - Adjust how weights are calculated based on information stored in the graph

OSM Graph Builder Task

- /opentripplanner-graph-builder/src/main/java/org/opentripplanner/graph_builder/impl/osm/OpenStreetMapGraphBuilderImpl.java
- OSM tags are read here to affect:
  - Street permissions
  - Safety features
  - Creative names
  - Wheelchair accessibility

Traversing

- Traversing – an edge (or vertex) is considered for routing – when weight and time are calculated – done after graph is built, during routing
- Can adjust weight and time based on mode and other variables
- Important class for traversing: /opentripplanner-routing/src/main/java/org/opentripplanner/routing/edgetype/PlainStreetEdge.java

Traversing

- doTraverse(…) – usually in an edge class
- private TraverseResult doTraverse(State s0, TraverseOptions options, boolean back) {
- State – holds current routing state: time, walk distance, # of boardings
- TraverseOptions – holds options for routing (some user specified): wheelchair accessible, max walk distance, modes of travel
- Back – true if retreating back(?)