- ✓ Given start and end location
- ✓ check has offline data for start location
- ✓ find nearest way to start location
- ✓ start dijkstra
- Dijkstra over sequences
- ✓ order by sequence quality, pop first
- ✓ select way; push nodes on stack; push predecessor way on stack
- ✓ for nodes(...) on stack
- ✓ check has offline data for node
- ✓ for ways(...) connected to node
- ✓ solve dijkstra until end location found

- ✓ for location: calculate download tile coordinate
- ✓ download json file for tile coordinate
- ✓ filter ways on applicable types
- ✓ filter nodes on coverage by ways
- ✓ Split ways nodes and polylines
- ✓ check old tile merge algorithm
- ✓ merge split ways into local osm routing model
- ✓ update OSMNode to way mapping on insert of new ways
- ✓ add ways in raw form
- ✓ lazy import ways for coordinate
- ✓ model method update ways for node with download options and async callback. OSMNode.ways becomes private to model
- ✓ implement OSMWaySegment for routing
- ✓ write FastCoded elements dict as cache and as marker for has offline data for coordinate region

OSMTile: writes OSMWays and OSMNode child elements for given tile coordinate.
OSMNodes: may write as pseudo polyline with array containing identifiers. Will be very compact representation. with identifiers being aliased in fastCoder

OSMRoutingModel

progressively imports and merges Json tiles for index coordinates

OSMNode
- ✓ nodeID
- version
- ✓ coordinate
- ✓ ways

OSMWay
- ✓ wayID
- version
- ✓ nodes
- polyline
- ✓ wayType
- qualifiedFor
- ✓ tags
- directionAllowed

OSMSequence
- ✓ segments
- ✓ quality

OSMSegment
- ✓ way
- ✓ startNodeIndex
- ✓ endNodeIndex
- ✓ transitCoordinate

OSMPath
- ✓ nextPath
- ✓ track

OSMRoutingModel

On memory warning first flush model , may extend with more requests. Second return memory error

reading and writing on different queues with writing having lower priority

OSMSequence

quality by direction to target, diff on air distance sequence distance, routing type and osm way type

OSMWaySegment

✓ describes a sub part of a given way where start and end node index can be into either direction

TRM Routing Integration

TNRoutingEditor

updateRoute:didEndBlock

TNRoutingRequest

TNRoutingFetchRouteController

executeFetchRequest:didFinishBlock:

setupSegmentFetchQueueWithRoutingFetchRequestsAndCachedSegmentRequests

_segmentFetchQueue

TNRoutingSegmentRequest

fetchRequest.source = TNRoutingFetchSourceGeoService

fetchRequest.indexSet = indexSet;

executeGeoFindRouteSegmentFetchRequestsWithCallbackBlock:

TNRoutingSegmentRequest

executeGeoFindRouteFetchRequest:callbackBlock:

GeoFindRouteRequest