**ESRI GIS tools**

Many data sets involve some form of location data or other geographical data. GIS (geographic information system) tools provides a framework to handle this spatial analysis, and ESRI’s contribution is one of the most popular GIS toolboxes for Hadoop.

This framework is built on a java library called the “ESRI Geometry API for Java.” This library provides methods to create geometric objects, compute spatial operations, and to find topological relations (e.g. crosses, touch, and contains). These methods can be used directly in Java for MapReduce jobs, but they are also utilized by another component of the GIS tools.

The Spatial Framework for Hadoop provides UDFs for Hive that allows for queries to call upon the methods mentioned above. With this framework geographical data in Hive tables can be easily queried by properties like distance to another object and if an object touches another object. The full list of UDFs can be found at <https://github.com/Esri/spatial-framework-for-hadoop/wiki/UDF-Documentation>. The github page for the spatial sdk (<https://github.com/Esri/spatial-framework-for-hadoop>) has descriptions of the hive UDFs available and the geometry API page ([https://github.com/Esri/geometry-api-jav](https://github.com/Esri/geometry-api-java)a) has a wiki and javadoc pages for the framework’s classes.

Other features are focused on ESRI’s software ARCGIS, including SerDes and InputFormats for ARCGIS’s JSON format.

**Installation of ESRI GIS tools**

For both the ESRI Geometry API and the Spatial Framework to be used with Hive the files need to be loaded in jar format with Hive’s ‘add’ command. Both can be built with Maven or Ant (<https://github.com/Esri/spatial-framework-for-hadoop> & <https://github.com/Esri/geometry-api-java>) or found as prebuilt jars on the tutorial’s github repository (<https://github.com/Esri/gis-tools-for-hadoop>). In order to use the Geometry API in MapReduce jobs, the com.esri.geometry package can be included as a maven dependency.

**Task 1: Closest Store**

Use the Spatial Framework’s UDFs in Hive to determine the closest store in the data set to a given line made up of two points. The Hive script should create a table to hold the data from the input file. It should also return the name and distance (in latitude/longitude distance) of the closest store in the set and should use parameter substitution to accept the input points as well as the input location of the data set. The input should be in the form of two parameters, point1 and point2, each being a string of the form “x y”, where x and y are the longitude and latitude of the points.

Sample Input:

The Food Store,39.3001,-87.6298

Groceries Inc.,40.4029,-86.4030

Kroger,39.8205,-85.5023

**Task 2: Geodesic Distance**

Write a custom UDF to return the [Manhattan distance](https://en.wiktionary.org/wiki/Manhattan_distance) between two points. Use this UDF to write a hive script that returns the Manhattan distance between an input point and the coordinates in each row.

For example, the input

spotA,3,10

spotB,5,2

spotC,2,9

With the input values of a=0 and b=0 should return

spotA,13

spotB,7

spotC,11

Hint: look at the source code of a UDF such as the ST\_Distance UDF and the OperatorProximity2D class that is a part of the geometry API for java.

### MiniProject

Write a custom UDF to return a linestring made up of an input point and the closest point on a second input geometry. Then use this UDF with the [ST\_GeodesicLengthWGS84](https://github.com/Esri/spatial-framework-for-hadoop/wiki/UDF-Accessors#st_geodesiclengthwgs84) UDF to find the geodesic distance (in meters) between the points and LineStrings of the input:

ContrivedInput1 -87.66925367 41.74025138 linestring(-87.65435458422664 41.77415375292839, -87.65434535483662 41.77381927867472, -87.65433458178138 41.773415584072985, -87.65432489479052 41.77305492655605, -87.65431910654507 41.77278038042004, -87.65430967282559 41.77233294690428)

ContrivedInput2 -87.641824 41.93286042 linestring(-87.63829660445927 41.67785424456207, -87.63804769090487 41.6778563554925, -87.63775078230799 41.677859827477235, -87.63730802052346 41.67786466585596, -87.63709282715357 41.677866996515284, -87.63684966462363 41.677869628626155)

ContrivedInput3 -87.667008027 41.977125251 linestring(-87.66139078534161 41.764942871815165, -87.66122767217554 41.76494610552368, -87.66077021333673 41.76495341777175, -87.66057972196394 ,41.764956328816766, -87.66042743219155 41.76495865561849, -87.6601786662065 41.76496102601809)

Hint: Check the OperatorFactoryLocal if you can’t figure out how to instantiate the proximity operator. The [javadoc site](http://esri.github.io/geometry-api-java/javadoc/) should have information about these classes, but source code is useful as well.

Sample Output:

ContrivedInput1 3773.7794251563287  
ContrivedInput2 28324.92673262132  
ContrivedInput3 23571.875619643484

### Project: U.S. States with the 100 Most Populous Cities

Use the Esri Geometry API and Spatial Framework’s UDFs and your knowledge of Hive to determine the number of cities from the top 100 most populous cities in the US for each state. Two datasets are provided for you: data on the 100 most populous US cities and data describing the outline of each state’s borders. You should write a hive script that creates tables for each dataset. Reference this page for how to import the data into hive: <https://community.hortonworks.com/articles/5129/geospatial-data-analysis-in-hadoop.html>

Note that these datasets are not stored in the same format. The data on US cities is a CSV with the city name, latitude, longitude, and population, while the data on state borders comes as JSON.

The output should be of the form:

State name, number of cities

For example, if the list of cities included Terre Haute and Indianapolis as the only cities from Indiana, the output would include:

Indiana, 2

Hint: You will need to use the ST\_Contains, ST\_Point, and ST\_GeomFromGeoJSON hive UDFs that are part of the Esri spatial framework.

**Correct Output:**

Alabama 1  
Alaska 1  
Arizona 6  
California 17  
Colorado 3  
District of Columbia 1  
Florida 6  
Georgia 1  
Illinois 1  
Indiana 2  
Kansas 1  
Kentucky 2  
Louisiana 2  
Maryland 1  
Massachusetts 1  
Michigan 1  
Minnesota 2  
Missouri 2  
Nebraska 2  
Nevada 4  
New Jersey 2  
New Mexico 1  
New York 4  
North Carolina 5  
Ohio 4  
Oklahoma 2  
Oregon 1  
Pennsylvania 2  
Puerto Rico 1  
Tennessee 2  
Texas 13  
Virginia 3  
Washington 1  
Wisconsin 2

**Things to Watch Out for with ESRI GIS Tools**

ESRI GIS tools are useful for calculating geometric features, but they were created with ESRI’s ArcGIS software in mind, so some features are not as useful as they might seem. The JSON SerDe format is not very standard as it is meant primarily for output from ArcGIS. Instead the ST\_GeomFromGeoJson or ST\_GeomFromJson (depending on the data’s format) UDF is usually a simpler solution.

When doing geographic calculations the toolkit treats all geometric structures as planar. Unfortunately if your data is in common non-planar format such as Latitude and Longitude then many of the UDFs need to be modified or else they cannot be used. This is because Latitude and Longitude are only planar units very close to the Earth’s equator. The framework does provide a geodesic length UDF, but this only accepts lines as an input, so it may be difficult to find the distances between types of geometries besides points.

When adding jars to Hive, the esri-geometry-api jar must be added before the spatial-sdk-hive jar as there are dependencies.