**1 Overview**

In this MP, we are going to build a chatbot that will return the shortest distance between two cities/nodes in a directed graph, with weight equal to 1 for all the edges.

**2 Requirements**

You need a valid AWS account and will be working on Lambda, API Gateway, DynamoDB, Cognito and Lex. Also, you need to be familiar with one of the following programming languages for implementing lambda: Python / Javascript / Java / Go. While we will make every attempt to help out irrespective of your chosen language, we can best assist with python.

**3 Procedure**

**3.1. AWS Graph Creator Lambda:**

You need to create a POST REST API (using AWS API Gateway) that will take a graph and store it inside the DynamoDB database. The following would be the specification for the graph which will be passed into the body of the POST request:

{"graph": "Chicago->Urbana,Urbana->Springfield,Chicago->Lafayette"}

There is a directed edge going from Chicago to Urbana, Urbana to Springfield and Chicago to Lafayette. You need to parse this graph inside the lambda, compute the shortest distance using BFS (Breadth First Search) between each of the vertices and store this state inside DynamoDb. Make sure you flush the respective table of your database before storing this state.

With respect to DynamoDB, you need to create a table which will contain the source, destination and distance parameters. While parsing the graph, this table should be populated with the locations and the distances between them. These value will be retrieved later on through your bot. The following AWS official documentation will help you get started with this:

<https://docs.aws.amazon.com/lambda/latest/dg/getting-started.html>

<https://docs.aws.amazon.com/apigateway/latest/developerguide/getting-started.html>

<https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/GettingStarted.Python.html>

You may also find this unofficial tutorial to be useful:

<https://medium.com/accenture-the-dock/serverless-api-with-aws-and-python-tutorial-3dff032628a7>

Note: The autograder uses the python requests library to send a post request. The body of your post request will appear as json string in event['body'] where event is the first parameter of your lambda handler function. [Cloudwatch](https://docs.aws.amazon.com/lambda/latest/dg/python-logging.html) is a useful tool that lets you observe logs from your lambda function.

**3.2 AWS Lex:** [**https://docs.aws.amazon.com/apigateway/latest/developerguide/getting-started.html**](https://docs.aws.amazon.com/apigateway/latest/developerguide/getting-started.html)

In this step you need to create a chatbot using AWS Lex. Lex is an AWS service for building conversational interfaces for applications using voice and text. A good way to get started with lex is via Amazon's offical documentation: <https://docs.aws.amazon.com/lex/latest/dg/ex1-sch-appt.html>

You need to create a chatbot that can decipher the name of the two cities from text and provide the distance. The interaction with the chatbot will be of the following type:

1. User: "What is the distance from Chicago to Springfield?"

Reply from chatbot: "2"

2. User: "I need to find the distance between two cities?"

Reply from chatbot: "Source?"

User: "Chicago"

Reply from chatbot: "Destination?"

User: "Urbana"

Reply from chatbot: "1"

The autograder uses the above utterances to prompt lex, so please make sure you set the utterances correctly. For slot types, please choose AMAZON.US\_CITY.

Give your bot an [alias](https://docs.aws.amazon.com/connect/latest/adminguide/tutorial1-lex-bot-publish.html) before you publish it.

**3.3 Lex with Lambda:**

You need to link the above created chatbot to an AWS lambda function which when triggered retrieves the shortest distance between two nodes in the graph from the database and return the result back to the chatbot. A lambda function can be linked to a chatbot intent by selecting the fulfillment tab and selecting the appropriate lambda function.

You can learn more about the input and response format from AWS Lex to Lamda from <https://docs.aws.amazon.com/lex/latest/dg/lambda-input-response-format.html>

For testing you can pass in a random graph using the API created in 3.1 and interact with Lex to see if the end to end flow works. After the bot has been build you need to publish your bot and give it an alias name. Please note down the alias name you provide as it needs to be passed to the autograder.

**3.4 Deploying Lex:**

You need to deploy your Lex to make it publicly accessible. There are many ways to do this however we will be using AWS Cognito Identity Pool. The "Setup Amazon Cognito" section from the following link shows how this can be done.

<https://aws.amazon.com/blogs/machine-learning/greetings-visitor-engage-your-web-users-with-amazon-lex/>

After following the tutorial, you should have an identity pool id which will be used by our autograder to verify the functionality of your Lex.

**4 Submission**

Add the necessary info in the payload section in test.py file attached below. If all the test cases passed then you will be able to see your grade on coursera.