Assignment 2 (3rd April 2017)

The second test post-midsem will be on 3rd April 2017. The test will be based on **Neo4j**. Neo4j is a graph database management system. A graph database is a database that uses graph structures for semantic queries with nodes, edges and properties to represent and store data. A key concept of the system is the *graph* (or *edge* or *relationship*), which directly relates data items in the store. The relationships allow data in the store to be linked together directly, and in many cases retrieved with one operation.

Resources :

a) **Introduction to graph databases and Neo4j:**
https://neo4j.com/developer/graph-database/

b) **Instructions for installing Neo4j are available in the link below :**
https://neo4j.com/docs/operations-manual/current/installation/linux/tarball/

c) **An introductory video to get started with neo4j is available in the link below:**
https://www.youtube.com/watch?v=mBhi_Jsdkt0

Neo4j has its own query language called Cypher. Cypher uses a similar syntax to SQL. You can find help about Neo4j and Cypher in the link below :

http://www.tutorialspoint.com/neo4j/

Problem Statement:

You will be using the same dataset used for Assignment 1, Post Mid-sem. It contains Youtube video information crawled using Youtube api. Dataset includes 500 json files. Each file has information about a single video. So total of 500 files describes some useful properties of 500 different videos. To know about the fields and their description go through the Youtube api documentation.

You are required to create a multi-relational network (***graph***) connecting videos. A network is multi-relational if nodes are connected with more than a relations. You should consider the following relations to connect nodes.

i. Two videos are connected if they are collected from same channel.

ii. Two videos are connected if they share common tags. The number of the common tags is the weight of the edge.

iii. Two videos are connected if they share common terms in the ***Description***. The number of the common term is the weight of the edge.

Further, nodes can be associated with node properties. Consider the following node properties.

"commentCount", "viewCount", "favoriteCount", "dislikeCount", "likeCount"

Neo4j's query language Cypher supports loading data from CSV directly but not from JSON files or URLs. So you can either convert the data into csv format or you can use the **Py2neo** library available for working with Neo4j from within Python applications and from the command line

About Lab Session:

It will be a 1 hour 30 minutes test where you will need to write the retrieval queries for the same data set. So make sure you have Neo4j installed in your machine and in working condition and the dataset provided to you should be inserted into your database.

Tasks to be completed before the lab session :

1. Understanding of graph databases
2. Understanding and installation of Neo4j
3. Understanding of the dataset provided
4. Insertion of data into Neo4j (You have to bring the insertion script/code and submit on the day of the test)

Example Queries :

Insertion of a node

```
CREATE (n1:Person { name: 'YA' })
CREATE (n2:State { name: 'UP' })
CREATE (n3:Person { name: 'MP' })
CREATE (n4:State { name: 'GOA' })
RETURN n1,n2,n3,n4
```

Insertion of a relationship

```
MATCH (p1:Person { name: 'YA' }), (p2:Person { name: 'MP' }),(s1:State { name: 'UP' }),(s2:State { name: 'GOA' })
CREATE (p1)-[r1:CM{ name: p1.name + '<--IsCM-->' + s1.name }]->(s1)
CREATE (p2)-[r2:CM{ name: p2.name + '<--IsCM-->' + s2.name }]->(s2)
RETURN p1,r1,s1,p2,r2,s2
```

Delete all nodes and relationships

```
Match (n)-[r]-()
Delete n,r;
```

Create a new set of nodes and relationships

solution given in ans1_1.txt, ans1_2.txt, ans1_3.txt

Search by node property

a) Return all MLA for a given party

solution given in ans2_1.txt,ans2_2.txt

Search by relationship property

a) Return the state name , whose CM name is YOGI

solution given in 3_1.txt

b) Return all the constituencies for the state name UP

solution given in 3_2.txt