# CSCI4180 (Fall 2015)

### Assignment 2: HBase and Parallel Dijkstra's Algorithm

Due on November 19, 2015, 23:59:59

## 1 Part 1: Counting Initial Sequences of N-Grams with HBase (50%)

This part is composed of three sub-parts.

**Import:** You need to first configure HBase correctly. Both HBase and HDFS must be hosted in fully distributed mode. The import process is to load the source data into a HBase table. Specifically, you need to create a table, in which each row stores a *single line of words*. You can design your own schema format for storing the original text.

Write a helper program called *HBaseImport.java* that takes a command-line argument "directory" and loads all text files in the directory into HBase.

**Counting:** After that, you need to count the occurrences of the initial sequences of $N$-grams, whose definition is the same as in Assignment 1. Here, we restrict $N = 2$ or $N = 3$. Also, we restrict ourselves to English alphabets (i.e., [a-zA-Z]).

The output will be stored in another HBase table, in which each row stores the tuple *(initial sequences of $N$-gram, count)*. The output table should be named **ngraminitial_result**. The key of the table is the initial sequence of an $N$-gram (e.g., "t w", for $N = 2$). You should store the count of each initial sequence in the column **result:count**. Marks will be deducted if you violate the naming convention.

Write a MapReduce program called *HBaseNgramInitial.java* that loads the data from HBase, counts the initial sequences, and writes the results to HBase.

**Export:** You should write a program that dumps the result from HBase. The program will take a command-line argument $\theta$, which only outputs the initial sequences that have the count at least $\theta$ to stdout in the following format.

**Sample Output for $\theta = 5$:**
q b 6
b f 5
f j 5

Write a helper program called *HBaseExport.java* that outputs the results from HBase.

**Notes:**

- You are allowed to submit an HBase script (e.g. init.hb) to create the table schema (but not inserting data). We will execute the script by calling the command `hbase shell init.hb` before the demo.

- Please refer to the lecture notes, tutorial notes, and the source code on the course website for the compilation and execution methods on HBase.

## 2 Part 2: Single-source Shortest Path Lengths by Parallel Dijkstra's Algorithm (50%)

In this part, you will need to write a map-reduce program to compute the **shortest path lengths** from a given **source node** in a graph dataset extracted from Twitter. We implement the algorithm via the parallel Dijkstra's algorithm. Details about the algorithm will be covered in tutorial. The Twitter dataset is obtained from the following reference:

- Haewoon Kwak, Changhyun Lee, Hosung Park, Sue Moon.
  "What is Twitter, a Social Network or a News Medium".
  19th World-Wide Web (WWW) Conference, April 2010.
  URL: http://an.kaist.ac.kr/traces/WWW2010.html

**Definition:** We model the Twitter network as a directed graph. Each user is represented as a *node* with a unique positive integer as the nodeID. When user 1 "follows" user 2 on Twitter, an *edge* is created from node 1 to node 2 in the graph. Also, we attach an integer weight to each edge, where the weight is between 1 and 50 inclusively.

**Problem:** Given a graph $G = (V, E)$ and a source node $v_s \in V$, find the shortest path distance from $v_s$ to every other reachable node in $V$. The source node $v_s$ is provided as a command-line argument.

**Input Format:** Each line contains a tuple of (nodeID, nodeID, weight), separated by spaces. Each tuple indicates a directed edge from the former node to the latter node.

**Output Format:** Each line contains a tuple of (nodeID, distance), separated by spaces. Only output tuples for nodes that are reachable from $v_s$ which is given as a command-line argument.

**Sample Input:**
1 2 7
1 3 20
2 3 3
3 1 5
4 1 9
5 6 10

**Sample Output:**
1 0
2 7
3 10

**Notes:**

- Since there is no path going to node 4, 5 or 6 from $v_s$, the tuples corresponding to these nodes should not be shown in the output.
- You will need to implement a class of the node structure (call it *NodeWritable.java*) to define the node attributes, such as adjacency lists.
- Your program (call it *ParallelDijkstra.java*) will take a command-line argument `Iterations` to indicate the number of map-reduce iterations. If `Iterations` equals 0, it means that the program will keep iterating until the shortest path distances are found.

- For this problem, you can actually implement multiple map-reduce iterations through a single program *ParallelDijkstra.java*. Throughout the iterative map/reduce process, feel free to store and retrieve any intermediate files in your own format on HDFS. We would only look at your final output for grading. The TA will give you instructions.

**Time Limit:** Note that the program should be completed within a reasonable timeframe for a given dataset. The time limit is set by the TAs. Marks will be deducted if the program runs too long.

## Bonus (5%)

The top 3 groups who have the total shortest running time for their programs in *both Parts 1 and 2* will receive the bonus marks. You may consider optimizing your programs or configuring some parameters in Hadoop to make the programs perform better. If more than 3 groups have the best performance, we will still give out the 5% bonus to each group. Note that the program must return the correct answer in order to be considered for the bonus mark.

## Submission Guidelines

You must at least submit the following files, though you may submit additional files that are needed:

**Part 1:**

- HBaseImport.java
- HBaseNgramInitial.java
- HBaseExport.java

**Part 2:**

- NodeWritable.java
- ParallelDijkstra.java

Your separate MapReduce scripts must be implemented in Java.

The assignment is due on November 19, 2015, 23:59:59. Demo will be arranged on the next day. Have fun! :)