CS 6240 LARGE SCALE DATA PROCESSING
HOMEWORK 1

SNEH GURDASANI
NUID: 001399060

pseudo-code for Twitter-follower count program implementation in MapReduce:

```
Map (String key, String value):
// key: document file
// value: document contents
for each user_id in value:
    // Output intermediate <key, value>(user_id, 1) pairs
    EmitIntermediate(user_id, "1");

Reduce (String key, Iterator values):
// key: UserID
// values: a list of counts
int total_followers = 0;
for each v in values:
    total_followers += ParseInt(v);
// Output the total value for each user_id
Emit(AsString(total_followers));
```

There are two different parts in the program:

1. **Mapper Phase:**

The input of the mapper is a csv file which describes each user_id following another user_id. So, in the mapper phase, the algorithm reads the csv and for each row, it ignores the first value(followers) and takes the second value(the users) as key and then creates an intermediate <key, value> pair it gives an stores the value ‘1’ for the followed Twitter user_id(key).

2. **Reducer Phase:**

The input to the reducer phase is a collection of rows for different user_ids as key and the count of their followers as values. The reducer sums up the counts of followers for respective followed Twitter user_id.

Question: Measure the running time of each program.
Running time of program 1(first run on AWS): 472720 miliseconds
Running time of program 2(second run on AWS): 481250 miliseconds

Question: <u>Report</u> the amount of data transferred to the Mappers, from Mappers to Reducers, and from Reducers to output.

Program 1 credentials:

Amount of data transferred to Mappers: 85331845 records.
Amount of data transferred from Mappers to Combiners: 85331845 records.
Amount of data transferred from Combiners to Reducers: 15362582 records.
Amount of data transferred from Reducers to output: 6626985 records.

Program 2 credentials:

Amount of data transferred to Mappers: 85331845 records.
Amount of data transferred from Mappers to Combiners: 85331845 records.
Amount of data transferred from Combiners to Reducers: 15362582 records.
Amount of data transferred from Reducers to output: 6626985 records.

Question: <u>Argue</u> briefly, why or why not your MapReduce program is expected to have good speedup. Make sure you discuss (i) *how many tasks* were executed in each stage and (ii) if there is a part of your program that is *inherently sequential* (see discussion of Amdahl's Law in the module.)

Program 1:
Launched map tasks=20
Launched reduce tasks=10

Program 2:
Launched map tasks = 20
Launched reduce tasks = 10

(i). MapReduce is a simple programming model for enabling distributed computations, including data processing on very large input datasets, in a highly scalable and fault-tolerant way. It is more efficient than normal sequential computation because it can execute tasks in distributed system that is the task of finding the followers from a large file is split into multiple groups of rows where each mapper will count the number of followers for the group of user_ids assigned to it parallelly. Also, the reducers aggregate the counts obtained from the Shuffle and sort output groups (groups of user_ids) and work parallelly to return the number of followers for each user_id.

(ii). In the MapReduce architecture, the Master node assigns each worker (or mapper) tasks. So, the assigning of tasks is the sequential part of follower count program as the Master node assigns tasks to each mapper one-by-one. According to Amdahl's law, each task can be split into sequential part and the parallel part. So, the total running time of the program will be equal to the time taken to execute the sequential part and the time taken by the Mapper-Reducer program to run.