# IERG 4330/ESTR 4316/IEMS 5730 Spring 2022
# Homework 5

Release date: Apr 20, 2022
Due date: 11:59:00 pm, May 10, 2022
***No late homework will be accepted!***

**Every Student MUST include the following statement, together with his/her signature in the submitted homework.**

*I declare that the assignment submitted on Elearning system is original except for source material explicitly acknowledged, and that the same or related material has not been previously submitted for another course. I also acknowledge that I am aware of University policy and regulations on honesty in academic work, and of the disciplinary guidelines and procedures applicable to breaches of such policy and regulations, as contained in the website http://www.cuhk.edu.hk/policy/academichonesty/.*

Signed (Student_____) Date:_____

Name_____ SID_____

**Submission notice:**

- Submit your homework via the elearning system

**General homework policies:**

A student may discuss the problems with others. However, the work a student turns in must be created COMPLETELY by oneself ALONE. A student may not share ANY written work or pictures, nor may one copy answers from any source other than one's own brain.

Each student **MUST LIST** on the homework paper the **name of every person he/she has discussed or worked with**. If the answer includes content from any other source, the student **MUST STATE THE SOURCE**. Failure to do so is cheating and will result in sanctions. Copying answers from someone else is cheating even if one lists their name(s) on the homework.

If there is information you need to solve a problem but the information is not stated in the problem, try to find the data somewhere. If you cannot find it, state what data you need, make a reasonable estimate of its value and justify any assumptions you make. You will be graded not only on whether your answer is correct but also on whether you have done an intelligent analysis.

# Q1 [35 marks]: GraphFrames

In this question, we will analyze the user activities on a Massive Open Online Course (MOOC) [1] using GraphFrames [2]. The MOOC user action dataset is a directed, temporal, bipartite network. The vertices represent users and course activities, and directed edges represent the actions taken by users on the MOOC platform (i.e. edges are always directed from users to course activities). Each edge has a timestamp. The timestamp is standardized to start from timestamp 0.

We provide 2 preprocessed files as follows:

1. mooc_actions.tsv [9]
   The data file is in tab-separated format.
   `SRCID  TARGETID  TIMESTAMP`
   where
   `SRCID`: a unique id for the source vertex.
   `TARGETID`: a unique id for each target vertex.
   `TIMESTAMP`: timestamp for the action in seconds from the beginning.

2. vertices.tsv [10]
   The data file is in tab-separated format.
   `id  type`
   where
   `id`: a unique id for each vertex.
   `type`: the type of the vertices (either “User” or “Course Activity”).

**In this question, you need to submit your Spark application to a Hadoop cluster (either your own Hadoop cluster or the IE DIC Hadoop cluster) or Kubernetes cluster (either your own Kubernetes cluster or the DIC’s Kubernetes cluster).**

**You are recommended to use Spark 3.2.1 with Hadoop 3.3 in this question.**

(a) [10 marks] Read both mooc_actions.tsv and vertices.tsv into Spark DataFrames, construct a GraphFrame from these 2 DataFrames, and report the following:
   (i) the number of vertices,
   (ii) the number of users,
   (iii) the number of course activities,
   (iv) the number of edges,
   (v) the vertex with the largest in-degree, and
   (vi) and the vertex with the largest out-degree.

(b) [5 marks] Use the dropIsolatedVertices() method to create a subgraph that contains all edges with timestamp >= 10000 AND timestamp <=50000. How many nodes are there in the subgraph? How many edges are there in the subgraph?

(c) [20 marks] GraphFrames supports searching for structural patterns in a graph [8]. Use the find() method to search for the following 4 motifs in the **subgraph extracted in question 1b)** and report the number of occurrences for each of the motifs.

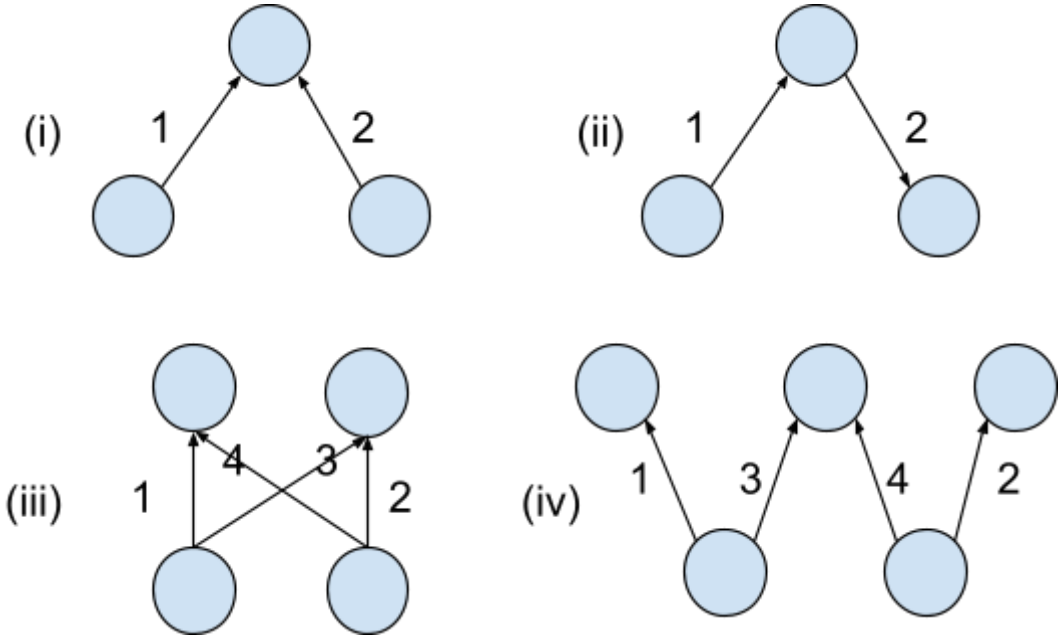


Fig1. Note that edge numbers indicate their temporal order

**Hints:**

- **To restrict named elements to be distinct vertices or edges, use post-hoc filters such as resultDataframe.filter("a.id != c.id").**
- **To restrict edges following the temporal order, use post-hoc filters such as resultDataframe.filter("e1.timestamp <= e2.timestamp").**

Submit your code and result.

# Q2 [35 marks]: GraphX

GraphX [3] is a new component in Spark for graphs and graph-parallel computation. At a high level, GraphX extends the Spark RDD by introducing a new Graph abstraction: a directed multigraph with properties attached to each vertex and edge. To support graph computation, GraphX exposes a set of fundamental operators (e.g., subgraph, joinVertices, and aggregateMessages) as well as an optimized variant of the Pregel API.

In this question, we aim at learning graph processing basics in GraphX. We will process the ogbn-arxiv dataset [5] to extract different common statistics. The ogbn-arxiv dataset is a directed graph, representing the citation network between all Computer Science (CS) arXiv papers. Each node is an arXiv paper and each directed edge indicates that one paper cites another one. The preprocessed data can be downloaded from [6].

**In this question, you need to submit your Spark application to a Hadoop cluster (either your own Hadoop cluster or the IE DIC Hadoop cluster) or Kubernetes cluster (either your own Kubernetes cluster or the DIC's Kubernetes cluster).**

**You are recommended to use Spark 3.2.1 with Hadoop 3.3 in this question.**

(a) [3 marks] Use the GraphLoader.edgeListFile() method available in GraphX to load the given dataset and report the number of vertices, the number of edges, the vertex with the largest in-degree, and the vertex with the largest out-degree using the graph attributes.

(b)

(i) [3 marks] Use the connectedComponents() method available in GraphX to find connected components in the citation network. How many connected components exist in the citation network? How many vertices exist in the largest connected component?

(ii) [3 marks] Similarly, use the stronglyConnectedComponents() method available in GraphX to find the strongly connected components in the citation network. How many strongly connected components exist in the citation network?

(c)

(i) [3 marks] The personalized PageRank algorithm is an extension of the original PageRank algorithm that defines the relevance of each node (to a target node) in the graph, the greater scores will correspond to nodes with greater relevance. Use the personalizedPageRank() with resetProb=0.15 to find the personalized PageRank values for all vertices in the graph with respect to target vertex 4330 and 5730. Report the top 20 nodes with the highest PageRanks with respect to each of these vertices (4330 and 5730)

(ii) [3 marks] Find the top 2000 nodes with the highest PageRanks with respect to the vertex 5730. Use the subgraph() function to construct a subgraph containing these 2000 nodes and all of the edges connecting pairs of vertices in that 2000 nodes. How many edges exist in the subgraph?

(d) [5 marks] Use the LabelPropagation.run() method with maxSteps=50 available in GraphX to find the communities in the citation network. How many communities are there? How many vertices are there in the largest communities?

(e) [15 marks] GraphX provides an API [4] based on Google's Pregel to accomplish iterative algorithms in a concise manner. In fact, many of GraphX's built-in algorithms are implemented in terms of a single Pregel call. In this question, we'll implement an algorithm that finds the vertex with the largest distance from its root in the ogbn-arxiv graph. At the end of the algorithm, we hope to have each vertex labeled with the farthest distance from a root.

Since the ogbn-arxiv contains many cycles, to avoid the situation where a group of vertices continuously sends and receives messages without termination, we

construct the minimum spanning tree of the ogbn-arxiv graph so that the graph becomes a Directed acyclic graph (DAG). The edge list of the minimum spanning tree can be downloaded from [7]

Consider the following implementation using the aggregateMessages() method:

```
def sendMsg(ec: EdgeContext[Int,Int,Int]): Unit = {
   ec.sendToDst(ec.srcAttr+1)
}

// the vertex will contain the highest value, or distance, over all the messages
def mergeMsg(a: Int, b: Int): Int = {
   math.max(a,b)
}

def propagateEdgeCount(g:Graph[Int,Int]):Graph[Int,Int] = {
   val verts = g.aggregateMessages[Int](sendMsg, mergeMsg)
   val g2 = Graph(verts, g.edges)

   // in Tuple2[vertexId, Tuple2[old vertex data, new vertex data]]
   val check = g2.vertices.join(g.vertices).
      map(x => x._2._1 - x._2._2).
      reduce(_ + _)
   println(s"check: $check")

   if (check > 0) // If there are no new information
      propagateEdgeCount(g2)
   else
      g
}
val dag = GraphLoader.edgeListFile(spark.sparkContext, "data/dag_edge_list.csv")
val initialGraph = dag.mapVertices((_,_) => 0)
propagateEdgeCount(initialGraph).vertices.collect
```

Implement the same algorithm with a single Pregel call. Submit the code and the top 20 nodes with the largest distance from the root**.**

# Q3 [30 marks]: HBase

**In this question, you need to use a Hadoop cluster (either your own Hadoop cluster or the IE DIC Hadoop cluster). If you want to use your own Hadoop cluster, you can follow https://hbase.apache.org/book.html#quickstart to install a fully-distributed HBase.**

(a) **[15 marks]** Import the "Google Books 2" dataset in [11] to HBase using Bulk Loading in [12]. It includes two steps: upload the file to HDFS and convert it into HFiles; then use CompleteBulkLoad in [13] to move the generated file into an HBase table. As for the first step, you can either write your own MapReduce code to convert the file to HFiles or use ImportTsv in [14] to convert it. When creating the table, you should make sure the table is split into at least 3 parts so that each part is stored on a different node in the cluster.

(b) **[15 marks]** Perform the following tasks (use either Java code or command-line shell):

1. Insert the following record into the table

   ierg4330 2019 100 4
2. Get all the records in the Year 1671 with occurrences larger than 100. (Filter is needed).
3. Delete the records in part 2.

Hints: You may need to add a unique rowkey for each row in the dataset.

## Q4 [20 bonus marks]: Spark ML

In this question, we will take advantage of the Spark Machine Learning Library MLlib to deal with the Netflix Challenge and implement a Movie Recommender System based on collaborative filtering.

**In this question, you need to submit your Spark application to a Hadoop cluster (either your own Hadoop cluster or the IE DIC Hadoop cluster) or Kubernetes cluster (either your own Kubernetes cluster or the DIC's Kubernetes cluster).**

(a) **[10 marks]** The Netflix Challenge is aimed at predicting user ratings for films, based on previous ratings without any other information about the users or films [15]. In this part, you are asked to set up a recommender system based on collaborative filtering. You can read [16] to find more information about this method.

Fortunately, Spark MLlib already supports collaborative filtering. You can directly use the ALS (Alternating Least Squares) algorithm [17] in MLlib to fill in the missing entries of a user-item association matrix in the GroupLens dataset [18]. Please print out the top 10 favorite movies of user 1, user 1001, and user 10001.

Hints: You do not need to adjust any parameters in the ALS algorithm. You can just use:
val model = ALS.train(ratings, 50, 10, 0.01, 0.01)

(b) **[10 marks]** In this part, you are asked to use cross-validation (CrossValidator) in MLlib to evaluate the performance of the model and choose the best set of parameters (in the ALS algorithm) for this recommender system. You can choose any one of MSE (mean square error), ROC (receiver operating characteristic curve), or AUC (Area under the Curve of ROC) as the indicator to compare the performance of part 1 and part 2. You can read [19] for more information.

Hints: Before cross-validation, you need to use a ParamGridBuilder to construct a grid of parameters to search for the best set of parameters.

Submit your code and result in a SINGLE pdf file.

# Reference:


[1] Social Network: MOOC User Action Dataset https://snap.stanford.edu/data/act-mooc.html
[2] GraphFrames
https://graphframes.github.io/graphframes/docs/_site/index.html
[3] GraphX
https://spark.apache.org/docs/latest/graphx-programming-guide.html
[4] GraphX Pregel API
https://spark.apache.org/docs/latest/graphx-programming-guide.html#pregel-api
[5] ogb-arxiv dataset
https://ogb.stanford.edu/docs/nodeprop/#ogbn-arxiv
[6] preprocessed ogbn-arxiv dataset
https://mobitec.ie.cuhk.edu.hk/ierg4330/static_files/assignments/edge_list.txt
[7] Minimum spanning tree of the ogbn-arxiv dataset
https://mobitec.ie.cuhk.edu.hk/ierg4330/static_files/assignments/dag_edge_list.txt
[8] Motif finding in GraphFrames
http://graphframes.github.io/graphframes/docs/_site/user-guide.html#motif-finding
[9] mooc_actions.tsv
https://mobitec.ie.cuhk.edu.hk/ierg4330/static_files/assignments/mooc_actions.tsv
[10] vertices.tsv
https://mobitec.ie.cuhk.edu.hk/ierg4330/static_files/assignments/vertices.tsv
[11] Google Books 2:
http://storage.googleapis.com/books/ngrams/books/googlebooks-eng-all-1gram-20120701-b.gz
[12] Bulk Loading:
https://hbase.apache.org/book.html#arch.bulk.load
[13] CompleteBulkLoad:
https://hbase.apache.org/book.html#completebulkload
[14] ImportTsv:
https://hbase.apache.org/book.html#importtsv
[15] Netflix Prize
https://en.wikipedia.org/wiki/Netflix_Prize
[16] coursera collaborative filtering
https://www.coursera.org/learn/machine-learning/lecture/2WoBV/collaborative-filtering
[17] ALS in MLlib
https://spark.apache.org/docs/latest/mllib-collaborative-filtering.html
[18] Grouplens 20M dataset:
https://grouplens.org/datasets/movielens/20m/
[19] Spark Cross Validation
https://spark.apache.org/docs/latest/ml-tuning.html