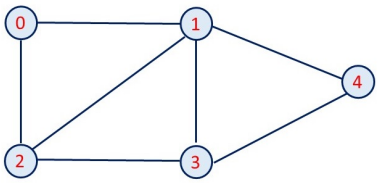
INP7079233 - BIG DATA COMPUTING (Proff. A: Pietracaprina and F. Silvestri) 2022-2023

Assignment of Homework 1 (DEADLINE April 12, 23:59)

The **purpose of this first homework** is to get acquainted with Spark and with its use to implement MapReduce algorithms. In preparation for the homework, you must set up your environment following the instructions given in Moodle Exam, in the same section as this page. After the set up is complete, test it using the WordCountExample program (Java or Python), and familiarize with the Spark methods it uses. The Introduction to Programming in Spark may turn out useful to this purpose.

TRIANGLE COUNTING. In the homework you must implement and test in Spark two MapReduce algorithms to count the number of distinct triangles in an undirected graph $G = (V, E)$, where a triangle is defined by 3 vertices u, v, w in V, such that $(u, v), (v, w), (w, u)$ are in E. In the right image, you find an example of a 5-node graph with 3 triangles. Triangle counting is a popular primitive in social network analysis, where it is used to detect communities and measure the cohesiveness of those communities. It has also been used is different other scenarios, for instance: detecting web spam (the distributions of local triangle frequency of spam hosts significantly differ from those of the non-spam hosts), uncovering the hidden thematic structure in the World Wide Web (connected regions of the web which are dense in triangles represents a common topic), query plan optimization in databases (triangle counting can be used for estimating the size of some joins).



3 DISTINCT TRIANGLES: (0,1,2), (1,2,3), (1,3,4)

Both algorithms use an integer parameter $C \geq 1$, which is used to partition the data.

ALGORITHM 1: Define a **hash function** h_C which maps each vertex u in V into a color $h_C(u)$ in $[0, C-1]$. To this purpose, we advise you to use the hash function

$$h_C(u) = ((a \cdot u + b) \mod p) \mod C$$

where $p = 8191$ (which is prime), a **is a random integer in** $[1, p-1]$**, and** b **is a random integer in** $[0, p-1]$**.**

Round 1:

- Create C subsets of edges, where, for $0 \leq i < C$, the i-th subset, $E(i)$ consist of all edges (u, v) of E such that $h_C(u) = h_C(v) = i$. Note that if the two endpoints of an edge have different colors, the edge does not belong to any $E(i)$ and will be ignored by the algorithm.
- Compute the number $t(i)$ triangles formed by edges of $E(i)$, separately for each $0 \leq i < C$.

Round 2: Compute and return $t_{final} = C^2 \sum_{0 \leq i < C} t(i)$ as final estimate of the number of triangles in G.

In the homework you must develop an implementation of this algorithm as a method/function ***MR_ApproxTCwithNodeColors.*** *(More details below.)*

ALGORITHM 2:

Round 1:

- Partition the edges at random into C subsets $E(0), E(1), \ldots E(C-1)$. Note that, unlike the previous algorithm, now every edge ends up in some $E(i)$.
- Compute the number t(i) of triangles formed by edges of E(i), separately for each $0 \leq i < C$.

Round 2: Compute and return $t_{final} = C^2 \sum_{0 \leq i < C} t(i)$ as final estimate of the number of triangles in G.

In the homework you must develop an implementation of this algorithm as a method/function ***MR_ApproxTCwithSparkPartitions*** *that, in*

*Round 1, **uses the partitions provided by Spark**, which you can access through method mapPartitions. (More details below.)*

SEQUENTIAL CODE for TRIANGLE COUNTING. Both algorithms above require (in Round 1) to compute the number of triangles formed by edges in the subsets $E(i)$'s. To this purpose you can use the methods/functions provided in the following files: CountTriangles.java (Java) and CountTriangles.py (Python).

DATA FORMAT. To implement the algorithms assume that the vertices (set V) are represented as 32-bit integers (i.e., type Integer in Java), and that the graph G is given in input as the set of edges E stored in a file. Each row of the file contains one edge stored as two integers (the edge's endpoints) separated by comma (','). Each edge of E appears exactly once in the file and E does not contain multiple copies of the same edge.

TASK for HW1:

1) Write the method/function MR_ApproxTCwithNodeColors which implements **ALGORITHM 1**. Specifically, MR_ApproxTCwithNodeColors must take as input an RDD of edges and a number of colors C and must return an estimate t_{final} of the number of triangles formed by the input edges computed through transformations of the input RDD, as specified by the algorithm. *It is important that the local space required by the algorithm be proportional to the size of the largest subset $E(i)$* (hence, you cannot download the whole graph into a local data structure). **Hint:** define the hash function h_C inside MR_ApproxTCwithNodeColors, but before starting processing the RDD, so that all transformations of the RDD will use the same hash function, but different runs of MR_ApproxTCwithNodeColors will use different hash functions (i.e., defined by different values of a and b).

2) Write the method/function MR_ApproxTCwithSparkPartitions which implements **ALGORITHM 2**. Specifically, MR_ApproxTCwithSparkPartitions must take as input an RDD of edges and the number of partitions C, and must return an estimate t_{final} of the number of triangles formed by the input edges computed through transformations of the input RDD, as specified by the algorithm. In particular, the input RDD must be subdivided into C partitions and each partition, accessed through one of the mapPartitions methods offered by Spark, will represent one of the subsets $E(i)$ appearing in the high-level description above. If the RDD is passed to the method already subdivided into C partitions, it is not necessary to repartition it. *It is important that the local space required by the algorithm be proportional to the size of the largest subset Spark partition.*

3) Write a program GxxxHW1.java (for Java users) or **GxxxHW1.py** (for Python users), where xxx is your 3-digit group number (e.g., 004 or 045), which receives in input, as command-line arguments, 2 integers C and R, and a path to the file storing the input graph, and does the following:

- Reads parameters C and R
- Reads the input graph into an RDD of strings (called **rawData**) and transform it into an RDD of edges (called **edges**), represented as pairs of integers, partitioned into C partitions, and cached.
- Prints: the name of the file, the number of edges of the graph, C, and R.
- Runs R times **MR_ApproxTCwithNodeColors** to get R independent estimates t_{final} of the number of triangles in the input graph.
- Prints: the median of the R estimates returned by MR_ApproxTCwithNodeColors and the average running time of MR_ApproxTCwithNodeColors over the R runs.
- Runs **MR_ApproxTCwithSparkPartitions** to get an estimate t_{final} of the number of triangles in the input graph.
- Prints: the estimate returned by MR_ApproxTCwithSparkPartitions and its running time.

File OutputFBsmallC2R5.txt shows you how to format your output. Make sure that your program complies with this format.

4) Test your program using the datasets that we provide in the same section as this page, together with the outputs of some runs of our program on the datasets. Note that while using $C = 1$ should give the exact count of triangles (unique for each graph), using $C > 1$ provides only an approximate count. So, for $C = 1$ your counts should be the same as ours, but for $C > 1$ they may differ. **Fill the table given in this word file** HW1-Table.docx with the required values obtained when testing your program.

SUBMISSION INSTRUCTIONS. Each group must submit a **zipped folder GxxxHW1.zip** (xxx is the group ID). The folder must contain the program (**GxxxHW1.java** or **GxxxHW1.py**) and a file **GxxxHW1able.docx** with the aforementioned table. Only one student per group must submit the zipped folder in Moodle Exam using the link provided in the Homework1 section. Make sure that your code is free from compiling/run-time errors and that you use the file/variable names in the homework description, otherwise your score will be penalized.

*If you have questions about the assignment, contact the teaching assistants (TAs) by email to bdc-course@dei.unipd.it . The subject of the email must be "**HW1 - Group xxx**", where xxx is your group ID. If needed, a zoom meeting between the TAs and the group will be organized.*

Last modified: Friday, 31 March 2023, 4:28 PM

◄ Machine Setup

Jump to...

Submission form for HW1 ►

You are logged in as LUISETTO FEDERICO (Log out)
2022-IN2547-003PD-2022-INP7079233-G2GR1

Data retention summary