2023-IN2547-003PD-2023-INP7079233-G2GR2
Participants
Grades
INP7079233 - BIG DATA COMPUTING (A-B) 2023-2024 - Prof. Pietracaprina and Silvestri
Spark installation
Homework 1
Homework 2
Homework 3
Home
Dashboard
Calendar
My courses
2022-IN0508-000ZZ-2020-INQ1097239-N0
2023-IN2547-003PD-2023-INP7079233-G2GR2
2023-IN2547-001PD-2022-INQ0091300-N0

INP7079233 - BIG DATA COMPUTING 2023-2024 (prof. Pietracaprina and Silvestri)

Home / My courses / 2023-IN2547-003PD-2023-INP7079233-G2GR2 / Homework 3 / Assignment of Homework 3 (DEADLINE: June 18, 23.59)

Assignment of Homework 3 (DEADLINE: June 18, 23.59)

In this homework, you will use the Spark Streaming API to devise a program which processes a stream of items and compares the effectiveness of two methods to identify frequent items: (1) the method based on reservoir sampling; and (2) the method based on sticky sampling. Both methods have been discussed in class.

Spark streaming setting that will be used for the homework

For the homework, we created a server which generates a continuous stream of **integer items**. The server has been already activated on the machine **algo.dei.unipd.it** and emits the items (viewed as strings) on specific **ports (from 8886 or 8889)**. Your program must first define a **Spark Streaming Context sc** that provides access to the stream through the method **socketTextStream** which transforms the input stream, coming from the specified machine and port number, into a *Discretized Stream* (**DStream**) of **batches of items**. A batch consists of the items arrived during a time interval whose duration is specified at the creation of the context sc. **Each batch is viewed as an RDD of strings**, and a set of RDD methods are available to process it. A method **foreachRDD** is then invoked to process the batches one after the other. Typically, the processing of a batch entails the update of some data structures stored in the driver's local space (i.e., its working memory) which are needed to perform the required analysis. The beginning/end of the stream processing will be set by invoking **start/stop** methods from the context sc. Typically, the stop command is invoked after the desired number of items is processed.

To learn more about Spark Streaming you may refer to the official Spark site. Relevant links are:

- Spark Streaming Programming Guide (full documentation)
- Transformations on Streams (list of transformations applicable to the RDDs in a DStream)

Running the program and template

Your program will be run in local mode on your PC, exactly as the one devised for Homework 1. The **master should be set to local[*]**.

In order to see a concrete application of the above setting you can download and run the following **example program** which takes as input the port number (*port*) and the number of elements (*threshold*) to be processed, and computes the exact number of distinct elements among the first *threshold* elements of the stream coming from port number *port*:

- (**Java version**) DistinctItemsExample.java
- (**Python version**) DistinctItemsExample.py

We strongly encourage to use this program as a template for your homework.
WARNING: When executing your programs, if you receive an error message such as ***ERROR ReceiverTracker: Deregistered receiver for stream 0: Stopped by driver*** do not worry. This is triggered by the stop signal and is due to an unclean management of the signal. However, it **has no consequence on the correctness of the execution.**

TASK for HW3.

You must write a program **GxxxHW3.java** (for Java users) or **GxxxHW3.py** (for Python users), where xxx is your 3-digit group number (e.g., 004 or 045), which receives in input the following **5 command-line arguments (in the given order):**

- **An integer** n: the number of items of the stream to be processed
- **A float** *phi*: the frequency thresold in $(0, 1)$
- **A float** *epsilon*: the accuracy parameter in $(0, 1)$
- **A float** *delta*: the confidence parameter in $(0, 1)$
- **An integer** *portExp*: the port number

The program must process the items in the batches up to and including the first batch which contains the n-th item of the stream Σ emitted by **machine algo.dei.unipd.it** at port *portExp*, and it must compute the following information **relative to the first *n* processed items** (*all items after the n-th one should be ignored*):

- The true frequent items with respect to the threshold *phi*
- An m**-sample** of Σ using **Reservoir Sampling** of, with $m = \lceil 1/phi \rceil$
- The ***epsilon*-Approximate Frequent Items** computed using **Sticky Sampling** with confidence parameter *delta*

If some action must be performed with a probability p, generate a random number in $x \in [0, 1]$ and perform the action only if $x \leq p$. Use the random generators by provided by Java and Python.

The program should print:

- The input parameters provided as command-line arguments
- For the EXACT ALGORITHM:
  - The size of the data structure used to compute the true frequent items
  - The number of true frequent items
  - The true frequent items, in increasing order (one item per line)
- For the algorithm based on RESERVOIR SAMPLING:
  - The size m of the Reservoir sample
  - The number of estimated frequent items (i.e., distinct items in the sample)
  - The estimated frequent items, in increasing order (one item per line). Next to each item print a "+" if the item is a true freuent one, and "-" otherwise.
- For the algorithm based on STICKY SAMPLING
  - The size of the Hash Table
  - The number of estimated frequent items (i.e., the items considered frequent by Sticky sampling)
  - The estimated frequent items, in increasing order (one item per line). Next to each item print a "+" if the item is a true freuent one, and "-" otherwise.

These files shows how to format your output. Make sure that your program complies with the input and output format.

The program that you submit should run without requiring additional files. Test your program on your local or virtual machine using various configurations of parameters, and **report your results using the table given in this word file.**

PORTS IN algo.dei.unipd.it

The ports from 8886 to 8889 of algo.dei.unipd.it generate four streams of 32-bit integers:

- 8887: it generates a stream where a few elements are very frequent, while all the remaining are randomly selected in the 32-bit integer domain.
- 8889: it generates a stream where a few elements are very frequent, some elements are moderately frequent, and all the remaining are randomly selected in the 32-bit integer domain.
- 8886: it is the "deterministic" version of the stream 8887, meaning that it generates the exact same stream every time you connect to this port. It should be used to test your algorithm.
- 8888: it is the "deterministic" version of the stream 8889, meaning that it generates the exact same stream every time you connect to this port. It should be used to test your algorithm.

SUBMISSION INSTRUCTIONS. Each group must submit a zipped folder GxxxHW3.zip, where xxx is your group number. The folder must contain the program (GxxxHW3.java or GxxxHW3.py) and a file GxxxHW3table.docx with the aforementioned table. Only one student per group must do the submission using the link provided in the Homework 3 section. Make sure that your code is free from compiling/run-time errors and that you comply with the specification, otherwise your grade 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 "HW3 - Group xxx", where xxx is your group number. If needed, a zoom meeting between the TAs and the group will be organized.

Last modified: Thursday, 6 June 2024, 2:37 PM

◀ Output uber-large.csv 3 100 16 (with 16 executors)
Jump to... ⬍
Submission form for HW3 ▶