Big Data Analytics

Stony Brook University  
CSE545 - Spring 2022

**Assignment 1**

**Assigned: 2/11/2022;   Due: 2/25/2022 11:59pm**

[**Overview**](https://docs.google.com/document/u/3/d/e/2PACX-1vTCmVF4Q4LaPFdYi0AAMXgrnClkrI9L0rPkIQ4lRUexeyfiFLI7etix-vCqObWIKpmLzcQ-N1JneHf0/pub#h.k7js7on60mn7)

[**Part I. Streaming (50 points)**](https://docs.google.com/document/u/3/d/e/2PACX-1vTCmVF4Q4LaPFdYi0AAMXgrnClkrI9L0rPkIQ4lRUexeyfiFLI7etix-vCqObWIKpmLzcQ-N1JneHf0/pub#h.pvn219uxc1h4)

[**Part II. MapReduce (50 points)**](https://docs.google.com/document/u/3/d/e/2PACX-1vTCmVF4Q4LaPFdYi0AAMXgrnClkrI9L0rPkIQ4lRUexeyfiFLI7etix-vCqObWIKpmLzcQ-N1JneHf0/pub#h.mfqja726bk8j)

[**Submission**](https://docs.google.com/document/u/3/d/e/2PACX-1vTCmVF4Q4LaPFdYi0AAMXgrnClkrI9L0rPkIQ4lRUexeyfiFLI7etix-vCqObWIKpmLzcQ-N1JneHf0/pub#h.udpy7l5v81q)

**Overview**

|  |
| --- |
| **Objectives:**   * To implement prototypical streaming algorithms. * To think through implementation of a novel streaming algorithm under the constraints of limited memory. * To implement parts of a map reduce system in order to become familiar with what it must do behind the scenes. * To implement a map reduce version of a provided algorithm.   **Requirements.** You must use Python version 3.7 or later.  You do not need a distribution of MapReduce for this assignment as we will use our own basic simulator (see task II.A.) -- future assignments will use MapReduce on a cluster.  **Templates and Python Libraries.** Template code is provided for each part of the assignment. Each template includes all of the data science, machine learning, or statistics libraries that are necessary. Other data science, machine learning, or statistics related libraries are prohibited unless listed below --  **ask if unsure.** The intention is for you to implement the algorithms we have gone over and problem solve in order to best understand the concepts of this course and their practical application.  Within the templates, all provided method names and classes must be used as provided with the same parameters. However, you may also use additional methods to keep your code clean.  Additional approved libraries that are not in the template will be listed here (if any):  import random from collections import defaultdict  import numpy as np #for numeric algebra and arrays  import math |
| **Copying code from other students, online or other resources is prohibited** and will result in at least a zero on the assignment and report to graduate program director with possibility for more consequences. Please see syllabus for additional policies.  A word to the wise: As is tradition in CSE545 at SBU, parts of this assignment are completely novel, never given before. |

**Part I. Streaming (50 points)**

|  |
| --- |
| Here, you will be fed a stream of integers representing yearly incomes of individuals (in 10s of thousands; e.g. 1 represents $10,000; 234 represents $2,340,000). Your goal is to summarize the stream in three ways: (a) the approximate distinct number of incomes seen, and (b) the median income, and (c) the most frequent value of income.  You can assume the income data approximately follows a power law, the Pareto distribution. As memory you will only be allowed to store a 100 element array.  **Data.** Two versions of the data are provided, (1) a small trial version with only 1000 integers to use while developing your method, and (2) a test that goes over 1 million integers to test your data on a larger dataset:  [trial\_incomes.csv](https://www.google.com/url?q=http://www3.cs.stonybrook.edu/~has/CSE545/a1/trial_incomes.csv&sa=D&source=editors&ust=1654367799120385&usg=AOvVaw0BTSxNNr_u50saNZinknFB)  [test\_incomes.csv.zip](https://www.google.com/url?q=http://www3.cs.stonybrook.edu/~has/CSE545/a1/test_incomes.zip&sa=D&source=editors&ust=1654367799120937&usg=AOvVaw3WmImJGG250VXIyL0y4hau) (unzip for csv after downloading)  **Template Code:**You will work within a fixed template where by you should only edit the methods for each task "def task1ADistinctValues", "def task1BMedian", "def task1CMostFreqValue"  Download the code here:[streamingCSE545sp22\_lastname\_id.py](https://www.google.com/url?q=http://www3.cs.stonybrook.edu/~has/CSE545/a1/streamingCSE545sp22_lastname_id.py&sa=D&source=editors&ust=1654367799121980&usg=AOvVaw0TVun6K_yE4bVmgI2KjdWy)  Steps for code familiarity:   1. Rename "lastname" and "id" in the filename to your lastname and your SBU student id. Also, add them to a comment at the top of the code. 2. The code should run with "python3 streamingCSE545sp22\_lastname\_id.py trial\_incomes.csv" but produce bogus results. Test that it does. 3. Look at main within the code. You will see that beyond some code to read the input file and set limits on memory (to constrain to working with data as a stream). It iterates over elements in the stream and calls each of the three methods passing the value along. At each of the following interactions: 10, 102,103,… 106 it prints the current calculation from the method. 4. Your job will be to edit those methods such that they return the approximate value, and only use the 100 element array (technically, a deque object with a maxsize but it operates as an array) provided to them as a memory. **Your 100 element array may only contain ints or floats as values**(i.e. you are not allowed to store dictionaries, other arrays, or any other objects/data structures as the values in this array)*.* During streaming, the current size of the array will be printed. It should remain < 8,000.   Of course there are ways to work around the memory limitation.  However, your goal is to implement approaches that work well even within this limitation and therefore you may not do anything that allows you to store more than TBD worth of data at once.   1. **Approximate the count of distinct incomes (20 points)**   For this subtask, you must approximate the number of distinct incomes seen thus far in the stream. Use the Flajolet-Martin algorithm as described in class and the book. You must use the *median of means* approach to average the estimates resulting from all hash functions. You should decide and justify the number of hash functions to use, although it must be < 100 since you may only store up to 100 elements at a time in memory.   1. **Approximate the median of the incomes (20 points)**   For this subtask, you must approximate the median of the incomes seen thus far – i.e. the income value, *m*, such that 50% of incomes seen thus far will be < *m* and 50% will have been > *m*. Typically, approximating the median of streaming data without storing many values is quite difficult. However, here you can assume the data follows a Pareto type 1 distribution, and the following is true of such a distribution:   * The function that yields the probability that a value is < a given *x*(i.e. the cumulative distribution function, CDF) is:  Diagram    Description automatically generated with low confidence where one can assume the minimum value is 1. * A picture containing shape    Description automatically generated can be estimated from the following function (which was derived from maximum likelihood estimation):Diagram    Description automatically generated  1. **Approximate the most frequent income value (10 points)**   Theoretically, the "mode" (i.e. most frequent value) of a Pareto distribution should be its smallest possible value. However, practically speaking, real-world data rarely follow the theoretical distribution perfectly and you've been tasked more precisely with finding the most frequent income seen thus far in the stream. Propose an approximate solution for calculating the most frequently witnessed income within the restrictions of the streaming template (i.e. without using more than 1 array of size 100)  *Hint:  There is no perfect solution to this. This subtask is intended to push the boundaries of your thinking. Thus, you may be awarded points for creativity and demonstration of effort even if your approximate answers are not very accurate.* |

**Part II. MapReduce (50 points)**

|  |
| --- |
| Here, you will complete a back-end for a MapReduce system and test it on a couple MapReduce jobs: word count (provided), matrix multiply (provided), and counting incomes by powers of 10 (you must implement).  **Data.**You will use trial\_incomes.csv from part 1 for task c. Data to test completion of tasks A and B are hard-coded into main.  **Template code.**A template to be filled in with your code is provided here: [MRSimulator\_CSE545\_sp22\_v2\_lastname\_id.py](https://www.google.com/url?q=http://www3.cs.stonybrook.edu/~has/CSE545/a1/MRSimulator_CSE545_sp22_v2_lastname_id.py&sa=D&source=editors&ust=1654367799130058&usg=AOvVaw0uxdZ-ZUMECtpkhdMuW_Le)  **Do not edit any blocks of code beyond those within the scope of "#[TODO]#".**Existing Print statements are designed to test your program during different points in the map-reduce process.  You may add additional print statements during your internal tests but you must remove them before submission.  **Steps for code familiarity:**   1. Rename "lastname" and "id" in the filename to your lastname and your SBU student id. Also, add them to a comment at the top of the code. 2. The code should run with "python3 MRSimulator\_CSE545\_sp22\_v2\_lastname\_id.py trial\_incomes.csv" but not produce results. Test that it does. 3. Examine the class MyMRSimulator. Start by reading the segments of the "runSystem" method to see the steps being run. Then examine each additional method. 4. Examine def wordCountMR for an example of a complete map and reduce function. 5. Your job will be to complete 2 segments of code in both the "reduceTask" and "runSystem", as well as write a map-reduce implementation of an algorithm as described below. Within the code look for  "#[TODO]#".   Specifically, you must complete:   1. **ReduceTask (15 points)** Complete the method "reduceTask" to perform the tasks of the reducer: 2. SEGMENT 1: sort such that all values for a given key are in a list for that key     (i.e. from (k1, v1), (k1, v2), (k1, v3) to {k: [v1, v2, v3]}) 3. SEGMENT 2: call self.reduce(k, vs) for each each key, providing its list of values and add the results (if they exist) to the list variable "namenode\_fromR" 4. **RunSystem (20 points)** Complete the “runSystem(self)” method which divides the data into chunks and schedules the running of mapTasks and reduceTasks. The are two places to complete: 5. SEGMENT 2:  Divide up the data into chunks according to num\_map\_tasks, and launch a map task per chunk. 6. SEGMENT 4:  Send each key-value pair to its assigned reducer by placing it in the to\_reduce\_tasks dictionary. 7. **countBy10PowersMR (15 points)** Edit the “map” and “reduce” methods of “ApplyTopicsMR” to implement a map-reduce computation of counting the integers (representing incomes) by powers of 10. That is, for each integer round it down to it's nearest power of 10 (for example 3 map to 1 = 100; 30 would map to 10 = 101. 87 would map to 10 = 101; 870 would map to 100 = 102, 100 would map to 100 = 102 etc….).  Your goal is to count the number of integers between each power of 10.   Here is an example output after the final reduce:          [(1, 446), (10, 258), (100, 133), (1000, 89), (10000, 50), (100000,24)]   This will use the same input as part 1 of the assignment so you can test with both the trial and test data.  \*\***Do not use self.data from the mappers or reducers: they need to work with the key values that they are provided.\*\***  Your code should run in < 30 seconds across all tests. |

**Submission**

|  |
| --- |
| Please use blackboard to submit two files each with your lastname and student id:   1. streamingCSE545sp22\_<lastname>\_<id>.py 2. MRSimulator\_CSE545\_sp22\_v2\_<lastname>\_<id>.py   **Do not upload a zip file. Double-check that your files are there and correct after uploading and make sure to submit.** Uploading files that are zips or any other type than python code files will result in the submission being considered invalid. Partially uploaded files or non-submitted files will count as unsubmitted.    **Questions**: Please post questions to the course piazza page.  **Hints**  As questions come in this location will be used to track suggestions.  See Piazza posts tagged "assignment1" ([https://piazza.com/stonybrook/spring2022/cse545](https://www.google.com/url?q=https://piazza.com/stonybrook/spring2022/cse545&sa=D&source=editors&ust=1654367799141421&usg=AOvVaw2gBiI-YBP4B_B9N8__h0EQ)). |