Project 2 for CS585/DS 503: Big Data Management Spring 2020

"Big data analytics from spatial analytics to data mining on big data".

Total Points:	**100**
Given Out:	**Monday, 17th Feb, 2020**
Due Date:	**Thursday, 5th March, 2020 (11:59PM).**
Teams:	**Project is to be done in teams -- Project2-team.**
	Team members will again be assigned by CS585/DS503 staff.
	If there are any concerns, notify us promptly!

Project Overview

In this project, you will create datasets and upload them into Hadoop HDFS. Next, you will analyze data in a scalable fashion by writing custom analytics tasks over big data using map-reduce Java code. The tasks span a range of jobs from advanced spatial data analysis to data mining (big data clustering algorithm). Throughout this project, you will work with Hadoop's File Formats.

Project Submission

You will submit the below using CANVAS as one single zip file.

1. You will submit a file containing the **Java or python programs** for creating data files, a small snippet of each data input and output files your program created (but please don't upload the large size input dataset generated, TA will create them using your code) and Java code for your MapReduce queries. In addition, this should include a **report (.pdf or .doc)** containing all documentation (including screenshots of outputs) and explanations of your solutions.

2. In this report, you also indicate the **skills of each team member prior to this project, skills acquired while working on this project, and the relative contributions**[1] of each team member to this project. Team members must in person discuss and agree on the precise statement to be put into the report and that it reflects indeed the division of work by the team members[2].

3. Lastly, we will ask **each of you independently** to submit brief comments to the CS585/DS503 staff via a survey at https://forms.gle/eczSJY2R32CYxU7g6 This concerns your personal assessment of your own contribution and effort as well as the contributions of your team member to this project. These comments will be treated confidentially to the degree possible. *You will not be given a grade, unless and until you submit your survey.*

[1] For instance, if each team member has done the project independently, and then only at the end you pulled the best of the material together, you need to say so. If you have closely collaborated and done the same amount of effort working side by side, report this.

[2] It is not sufficient for one student to only create a data file, for example, but not to be involved in the actual Hadoop analytics tasks. This would not be sufficient to receive credit for this project.

Problem 1 (Spatial Join) [35 points]

Spatial join is a common type of joins in many applications that manage multi-dimensional data. A typical example of spatial join is to have two datasets: ***Dataset P*** (set of POINTS in two-dimensional space such as mobile devices on people or sensors on cars) as shown in Figure 1a, and ***Dataset R*** (set of RECTANGLES in two-dimensional space such as parking structures, buildings or regions in a city) as shown in Figure 1b. The spatial join operation is to join these two datasets and report any pair (rectangle *r*, point *p*) where *p* is contained in *r* (or even in the border of *r*). For instance, this would allow us to determine which parking lot R a given car P is in, or which building R a person P has entered.

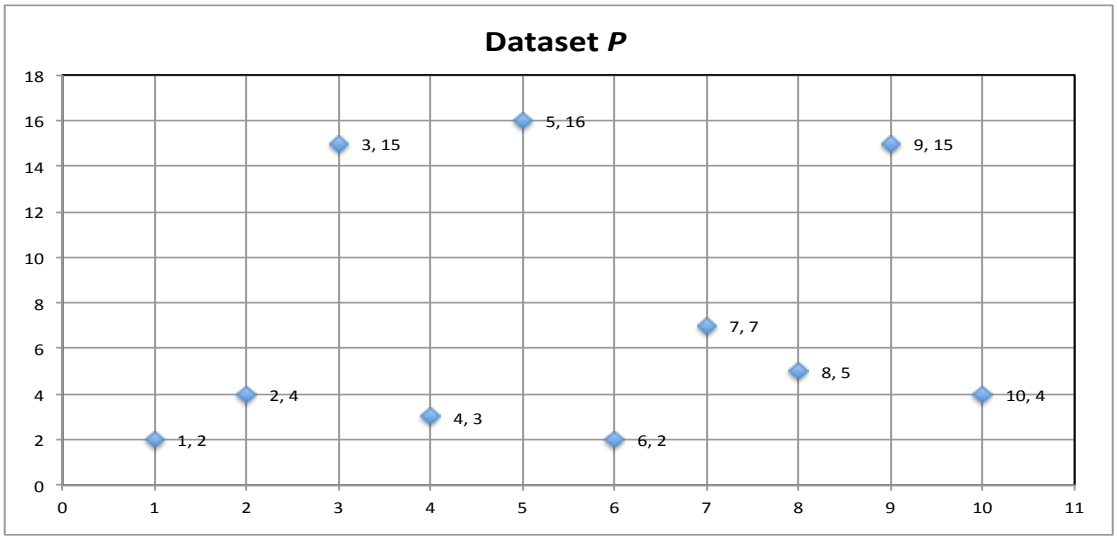


Figure 1a: Data File P of 2D Points

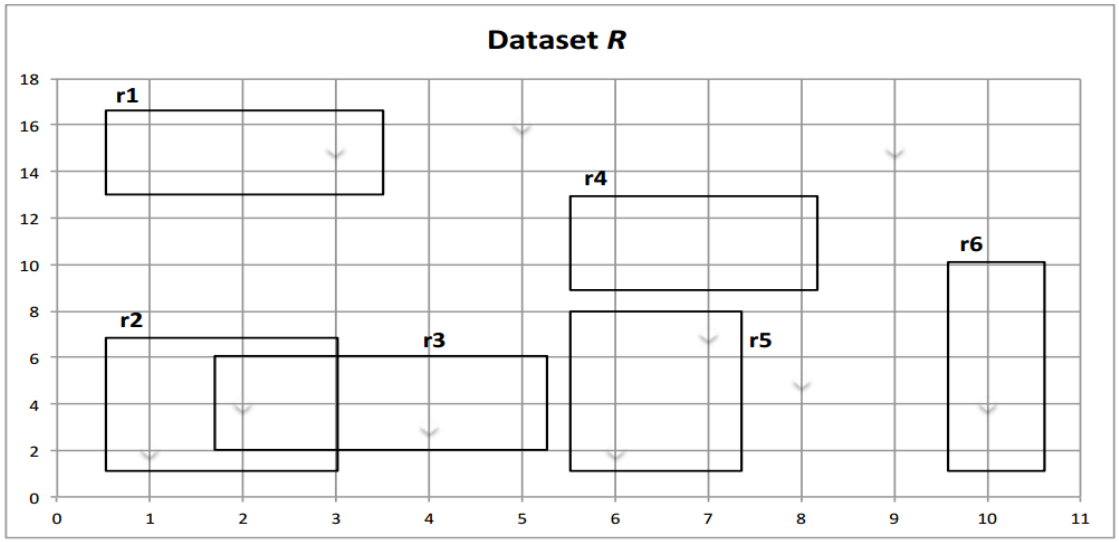


Figure 1b: Data File R of 2D Rectangles

For example, this spatial containment computation (join) between the two example data files shown in Figure 1 will result in:

<r1, (3,15)>
<r2, (1,2)>
<r2, (2,4)>
<r3, (2,4)>
<r3, (4,3)>
<r5, (6,2)>
<r5, (7,7)>
<r6, (10,4)>

Step 1 (Create the Two Datasets)

In this task, you will (use java or python or even hadoop) create the two datasets *P* (set of 2D points) and *R* (set of 2D rectangles). ***Finally, we will also provide you a test data set T in addition to your own data set towards the end of the project period to work with.***

- Each line in data file P should denote an object be <x, y> where x and y are the coordinates (in integer). While for data file R, each line denotes a region as <ri, x-bottom-left, y-bottom-left, x-top-right,y-top-right> where ri denotes the identifier of the region and the rest denote the coordinates of the rectangle.
- The space extends from [1 … 10,000] in both the X and Y axis.
- All data points in P must have coordinates X and Y that are integers as value (fall on a line).
- Scale one of the datasets *P* or *R* to be at least 100MB.
- Choose the appropriate random function (of your choice) to create the points. For the rectangles, you could for example select a point at random (say the bottom-left corner), and then select two random variables that define the *height* and *width* of the rectangle. For example, you should set the height random variable uniformly between [1,20] and the width uniformly between [1,5]. Make sure that all rectangles are fully contained within the given space.

Step 2 (MapReduce job for Spatial Containment Join)

In this task, you develop your big data processing solution.

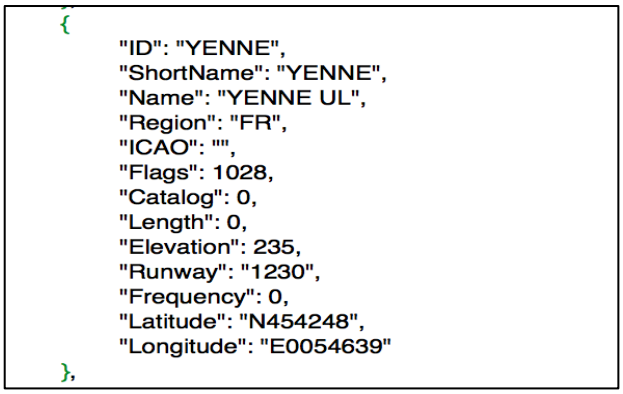
- First, you will write a java map-reduce job that implements the spatial join operation between the two datasets P and R.
- Then, adjust your program to take an optional input parameter W(x1, y1, x2, y2) that indicates a spatial window (rectangle) of interest within which we want to report the joined objects. If W is omitted, then the entire two sets should be joined. For example, referring to Figure 1, if the window parameter is W(1, 3, 3, 20), then the reported joined objects should be:
  <r1, (3,15)>
  <r2, (2,4)>
  <r3, (2,4)>
- If your above solution uses more than one map-reduce job, then rewrite your solution to now be realized as one single map-reduce job. Explain assumption you make, if any.

Problem 2 (Custom Input Format) [30 points]

So far, all of the given assignments use text files as input, and hence you use 'TextInputFormat()' to read the files. In this problem, you will work with Hadoop input formats and you will write your custom program to read the input data.

Step 1 (Data Sets)

You will use the dataset posted in CANVAS (under Project 2) with the file name "airfield.text". This file has records formatted in JSON format. Each record starts with "{"and ends with "}" (no quotes). All attributes in between form one record. Records are separated with ",". For example, the figure below shows one such record.



As first step, you upload the provided file into HDFS.

Step 2 (Map Job with a Custom Input Format)

- In order to work on the above dataset using the standard "TextInputFormat()", the map function would have to be very complex to collect many lines to form a single record.
- A better way to handle such custom structured data is thus to instead write a custom input format, let us call it "***JSONInputFormat***". This input format should read many lines from the input file until it gets a complete record (as the one in the image above). Then it should convert these lines to a list of commas separated values in a single line, and then pass it to the map function for processing. Our objective here is to design a solution so that the map function need not be aware that it is reading a complex custom JSON formatted file. In our case, each line passed to the map function should have the fieldname then a colon and then the actual value, and then a "," separating the fields. For example, we would have: *ID:…, ShortName:…., Name:…, Region:…, …*
- Your task is to write this new "JSONInputFormat".
- Thereafter, you should design a map-reduce job that uses your new JSONInputFormat reader and aggregates the records based on the "Elevation" field, and for each elevation value it reports the number of the corresponding records.

Hint: *You need to understand first the "**FileInputFormat**", "**TextInputFormat**", and "**LineRecordReader**" classes. You can then reuse some of them to build your new one as extension.*

Problem 3 (K-Means Clustering) [35 points]

K-Means clustering is a popular algorithm for clustering similar objects into *K* groups (clusters). It starts with an initial seed of K points (randomly chosen) as centers, and then the algorithm iteratively tries to enhance these centers. Your system should terminate if either of these two conditions become true:

a) The K centers did not change over two consecutive iterations
b) The maximum number of iterations has been reached (parameter R for rounds)

Step 1 (Creation of Dataset):

- Create a dataset that consists of 2-dimenional points, i.e., each point has (x, y) values. X and Y values each range from 0 to 10,000 at a scale of say 100 MB. ***In addition, we will also provide you with a testing data set T towards the end of the project period to work with.***
- Create another file that will contain K initial seed points. ***Make the "K" value as a parameter to your program***, such that your program will generate these K seeds randomly, and then upload the generated file to HDFS.

Step 2 (Clustering the Data):

Please develop the K-means **clustering strategies** described below using java map-reduce jobs. You should document the key differences from one algorithm to the next, i.e., what changes were made at the mapper, at the reducer, at the number of mappers or reducers, or in the main control program. These algorithms should include:

1. A **single-iteration** only Kmeans algorithm (you can accomplish that by setting R=1)
2. A basic **multi-iteration** Kmeans algorithm that terminates after R iterations (e.g., R=6).
3. A more advanced multi-iteration Kmeans algorithm that terminates potentially earlier as soon as it converges based on some threshold (or after R iterations).
4. An even more advanced algorithm that makes use of **optimization** ideas discussed in class, including the use of a combiner, a single reducer, and other optimizations.
5. For your Kmeans in subproblem 4, please design **two output variations** as per below:
   (a) return only cluster centers along with an indication if convergence has been reached; *vs*
   (b) return the final clustered data points along with their cluster centers.
6. Provide a description for each of your above solutions. Describe the relative performance of each of your algorithms. Explain and analyze your findings.

Note 1: You may reference the links below for general ideas about the clustering algorithm:
http://en.wikipedia.org/wiki/K-means_clustering#Standard_algorithm

Note 2: Since the algorithm is iterative, you need your program that generates the map-reduce jobs to also control whether it should start another iteration or not.

- The End -