**Методичні вказівки**

до виконання лабораторних робіт

з курсу

“Обробка надвеликих масивів даних”

Київ – 2018

**Лабораторна робота 1-2**

Вступ до hadoop та spark

**План:**

1. Отримати розуміння інструментів, необхідних для виконання дій з Hadoop, Spark

2. Встановити Hadoop, Spark

2. Запустити і зрозуміти логіку роботи програми “word count” (на Hadoop та Spark)

3. Pеалізація та розуміння алгоритму “inverted index” (на Hadoop та Spark)

**План виконання роботи:**

Завантажити Hadoop - це можна зробити кількома методами. Дистрибутив можна загрузити з офіційного сайту: <http://hadoop.apache.org/> , також можна скористатися докер контейнером або дистрибутивом від Cloudera (<https://www.cloudera.com/downloads.html>)

Налаштувати Hadoop:

Hadoop використовує файл hadoop-site.xml для налаштувань

Переконайтеся що Hadoop встановлено та ви маєте змогу під’єднатися до HDFS. Для цього, в терміналі виконайте наступну команду:

*$ bin/hadoop dfs -ls /*

Ви повинні отримати вивід приблизно такого змісту:

*Found 2 items*

*drwxr-xr-x - hadoop supergroup 0 2017-10-01 17:56 /tmp*

*drwxr-xr-x - hadoop supergroup 0 2017-10-02 05:10 /user*

(Перевірте змінну JAVA\_HOME - вона має вказувати шлях до актуальної інсталяції Java. Для того щоб встановити її використайте наступну команду:

*$ export JAVA\_HOME=</path/to/your/jdk>*)

Для завантаження Spark, виконайте інструкції:

https://spark.apache.org/downloads.html

**Word count вікористовуючи Hadoop**

Hadoop добре використовувати для обробки файлів які можна розбити на строчки (наприклад, текстові файли). Наприклад, для підрахунку слів у файлах

Приклад:

Наприклад у нас є 2 файла "A.txt" і "B.txt" з таким змістом:

*A.txt*

*--------------*

*This is the A file, it has words in it*

*B.txt*

*--------------*

*Welcome to the B file, it has words too*

*The algorithm we use to count the words must fit in MapReduce. We will implement the*

*following pseudo-code algorithm:*

*Mapper: takes as input a line from a document*

*foreach word w in line:*

*emit (word, 1)*

*Reducer: takes as input a key (word) and a set of values (all of which will be "1")*

*sum = 0*

*foreach v in values:*

*sum = sum + v*

*emit (word, sum)*

Маперам передаються всі рядки з кожного файлу, по одному за раз. Ми розбиваємо їх на слова і виділяємо (слово, 1) пари - вказуючи, що в той момент ми побачили це слово один раз. Потім ред’юсер збирає всі "1", і конкатенує файли для того шоб порахувати слова.

Відповідно на виході з маперів ми отримаємо:

Мапер для файлу A.txt:

*< This, 1 >*

*< is, 1 >*

*< the, 1 >*

*< A, 1 >*

*< file, 1 >*

*< it, 1 >*

*< has, 1 >*

*< words, 1 >*

*< in, 1 >*

*< it, 1 >*

Мапер для файлу B.txt:

*< Welcome, 1 >*

*< to, 1 >*

*< the, 1 >*

*< B, 1 >*

*< file, 1 >*

*< it, 1 >*

*< has, 1 >*

*< words, 1 >*

*< too, 1 >*

Кожне слово має свій власний процес reduce. Наприклад, для слова "it" він буде виглядати так:

*key="it"; values=iterator[1, 1, 1]*

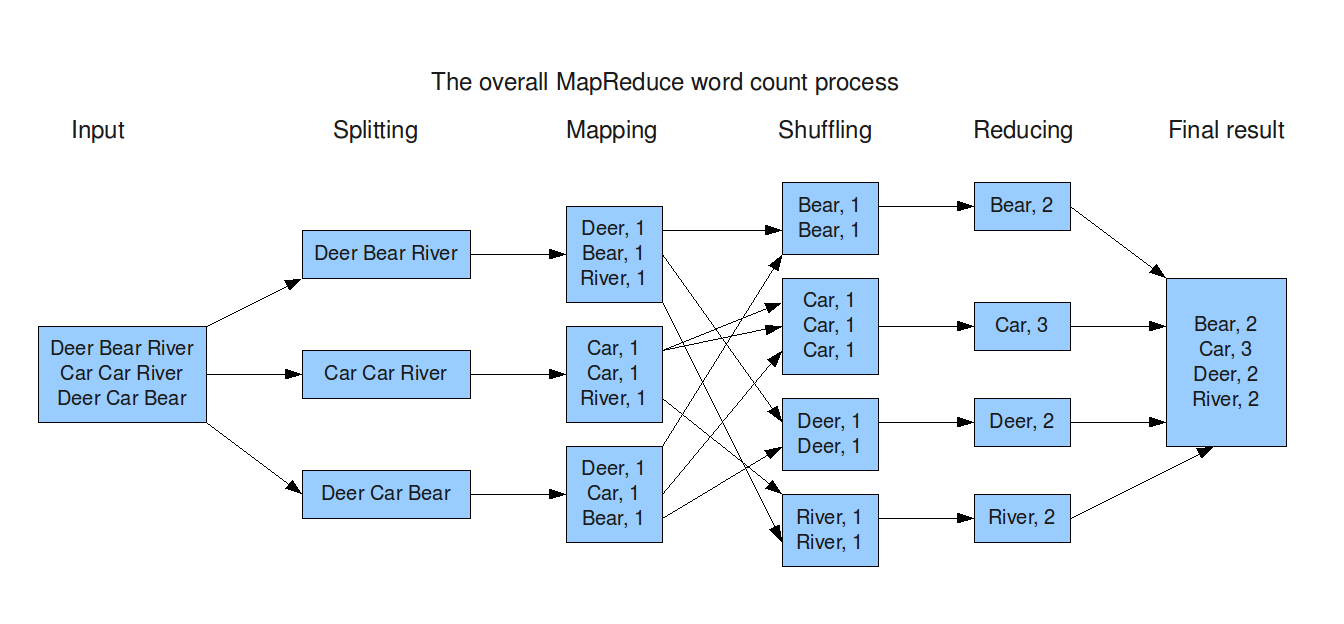
І отримаємо на виході:

*< it, 3 >*

Порядок виконання (Word Count):

Використовуючи вхідний текст, запустіть Word Count за допомогою Hadoop, щоб отримати зведену кількість слів у кожному з декількох документів.

Основна концепція Word Count описана вище, а також зображена на малюнку нижче. З детельним туторіалом можна ознайомитися на офіційній сторінці: *https://hadoop.apache.org/docs/r2.7.1/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html#Example:\_WordCount\_v2.0*



(https://cs.calvin.edu/courses/cs/374/exercises/12/lab/)

**Inverted Index**

Інвертований індекс — структура даних, в якій для кожного слова колекції документів у відповідному списку перераховані всі документи в колекції — в яких воно зустрілося. Інвертований індекс використовується для пошуку за текстами.

Є два варіанти інвертованого індексу:

- індекс, який містить лише список документів для кожного слова,

- індекс, додатково включає позицію слова в кожному документу

Вам необхідно створити інвертований індекс для файлів які ви використовували в word count.

З урахуванням вхідного тексту індексатор зсуву використовує Hadoop для створення індексу всіх слів у тексті. Для кожного слова індекс має список усіх місць, де з'являється це слово, і, можливо, текстовий витяг кожного рядка, де слово з'являється. Запустіть індексатор ліній на творах Шекспіра:

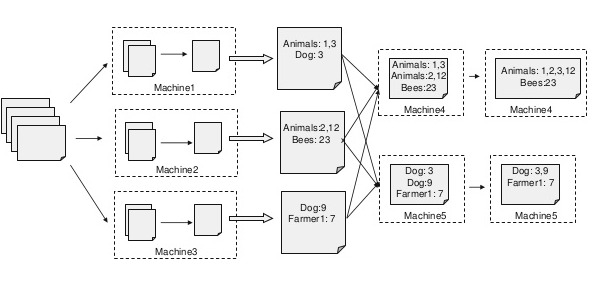
*lucrece.txt, offset 38624: To cipher what is writ in learned books,*

*orlando.txt, offset 66001: ORLANDO Which I take to be either a fool or a cipher.*

*would produce the following output for the word "cipher:"*

*cipher lucrece.txt@38624,orlando.txt@66001,...*

Порядок виконання (Inverted Index):



1. Створіть новий проект
2. Створіть клас мапера який буде приймати такі дані:
3. LongWritable key - побайтовий зсув поточної строчки в файлі
4. Text value - поточна строчка
5. OutputCollector - output - .collect метод для виведення <key, value> пари
6. Reporter reporter

Мапер має на виході отримувати пари <"word", "filename@offset">

1. Створіть клас reduce який буде приймати дані від мапера
2. Створіть драйвер - програму що буде імплементувати main() функцію (за тим же принципом що і в випадку з word count)

**Питання:**

1) Що робить програма підрахунку слів за допомогою ключа, наданого mapper?

2) Що таке HDFS? Як вона працює?

3) Що таке парадигма MapReduce?

4) Для яких даних використовується hadoop?

**Лабораторна робота 3**

**Алгоритм PageRank**

**План:**

1. Отримати розуміння алгоритму і як він використовує MapReduce

2. Імплементувати алгоритм за допомогою Hadoop, Spark

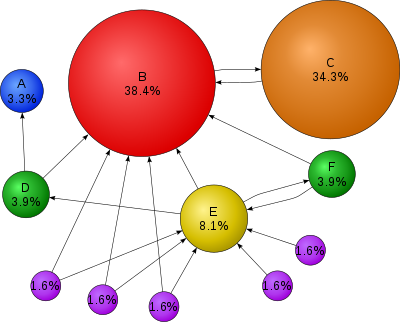
3. Спробувати запустити алгоритм на Вікіпедії і оцінити важливість різних сторінок

PageRank — сімейство алгоритмів оцінки важливості веб-сторінок за допомогою розв'язання систем лінійних рівнянь. Для кожної сторінки обчислює дійсне число, чим більше число — тим «важливіша» сторінка.

Замість прямого підрахунку кількості посилань PageRank інтерпретує посилання сторінки A на сторінку Б як голос сторінки A на користь сторінки Б. Після цього PageRank оцінює рейтинг сторінки відповідно до кількості отриманих голосів.

PageRank також враховує значимість кожної сторінки, що отримала голос, адже голоси деяких сторінок є важливішими, і відповідно до цього підвищується значущість сторінки, посилання на яку вони містять. Важливі сторінки отримують більш високу оцінку PageRank і відображаються на перших позиціях результатів пошуку. Для визначення значущості сторінки технологія Google використовує колективний інтелект всесвітньої мережі. Людина не бере участі в обробці результатів. Пошукова система Google не спотворює інформацію про позиції платою за результати пошуку.

За основу PageRank був обраний академічний підхід оцінки важливості публікації автора по числу її згадок в бібліографічних посиланнях інших авторів. Для адаптації до застосування в Інтернет в алгоритм були внесені наступні зміни: вага кожного посилання враховується індивідуально і нормується за кількістю посилань на сторінці. Крім того, PageRank може бути інтерпретовано в термінах випадкового блукання.



Основною метою алгоритму було поліпшення *якості* відповідей на пошукові запити. Розробники PageRank зазначили, що станом на листопад 1997 року, лише одна із найпоширеніших пошукових систем була здатна вказати власну сторінку в перших 10 результатах при запиті на власну назву (була здатна знайти сама себе).

Запропонований алгоритм, натомість, брав до уваги структуру і текст гіперпосилань. При цьому:

1. Алгоритм PageRank моделював випадкову подорож користувача інтернет починаючи з випадкової сторінки. Чим більше випадкових відвідувачів діставались сторінки, тим вище її рейтинг. Розробники припустили, що в такий спосіб вдасться уникнути проблем зі спамом за ключовими словами.
2. Вміст сторінки оцінювали не лише за ключовими словами в ній, а й в сторінках, що на неї посилаються. Розробники припустили, що зловмисникові буде важче спотворити вміст сторінок, що посилаються на його сторінку, тільки якщо він не контролює інші сторінки.

Значення PageRank для сторінки *A* обчислюється за такими правилами: нехай  — сторінки, що посилаються (цитують) сторінку *A*. Алгоритм також використовує коефіцієнт демпінгу *d*, значенням якого знаходяться в проміжку між 0 та 1, та зазвичай має значення 0.85. Функція *C(T)* дорівнює кількості посилань, що виходять зі сторінки *T*. Тоді значення PageRank сторінки *A*, *PR(A)*[, дорівнює:](https://uk.wikipedia.org/wiki/PageRank#cite_note-originalpaper-7)

[](https://uk.wikipedia.org/wiki/PageRank#cite_note-originalpaper-7)

При цьому, значення PageRank — це випадкові величини сума яких для всіх сторінок дорівнюватиме 1.

**Кроки MapReduce:**

Крок 1: cтворіть граф посилань

Крок 2: обробка PageRank (у цьому розділі ви повинні розділити фрагменти вхідного PageRank між посиланнями на сторінці, а потім рекомбінувати всі фрагменти, отримані сторінкою, до наступної ітерації PageRank.)

Крок 3: очищення та сортування (на цьому етапі ці дані можуть бути визначені та відсортовані).

**Питання:**

1) Опишіть типи даних, які ви використовували для ключів та значень на різних етапах MapReduce. Якщо ви серіалізували дані в текст і назад, опишіть, як ви заклали вміст тексту в потік даних.

2) Яку масштабованість, якщо така є, ви бачите в цій системі?

3) Опишіть процес тестування вашого коду перед його запуском на великому наборі даних.

**Лабораторна робота – Додаткова**

**Рендер картографічних даних**

Ви будете використовувати географічні дані та дані переписів, щоб відобразити карту основних транспортних / географічних особливостей Сполучених Штатів. Статистичне Бюро Сполучених Штатів (www.census.gov) створює набір даних, що описує всі дороги, шосе, міста, громадські місця, водні об'єкти тощо, в США. (https://catalog.data.gov/dataset/2008-tiger-line-nationwide-dataset)

Цей набір даних називається набором даних TIGER (топологічно інтегрована система географічного кодування та віднесення). Ми використовуємо набір даних "TIGER / Line". Цей проект покладається на обробку вихідних даних TIGER, поєднання його з іншими даними перепису (наприклад, оцінка населення для муніципалітетів) та надання карти США як набору плиток. Ми будемо використовувати Hadoop для виконання цієї обробки паралельно.

Плитки, діапазони, рівні масштабу та набори плиток

Неефективно робити спробу перетворити всю карту на будь-який масштаб, який ви хотіли б, як єдине ціле. Таким чином, ми розбиваємо карту на плитки. **Плитка** - квадратна область карти, яка відповідає найменшій одиниці, яку ми відтворюємо за один раз. Вигляд, який відображається в веб-переглядачі, складається з декількох фрагментів, розташованих у 2D сітці. Ми робимо основне спрощення в цьому завданні, що широта і довгота є прямокутною системою координат. Це призведе до неточності рендеринга, але для масштабу США це прийнятно.

Плитки розташовані в прямокутній сітці по карту. Діапазон - це вся область, яку ми хочемо обробити (наприклад кілька квадратних миль для тестування). Картовий діапазон - квадрат (припускаючи прямолінійну широту / довготу), визначений широтами та довготами її країв. Ми розбиваємо картовий діапазон у 2D масив плитки.

У цьому 2D-масиві кожна плитка має ідентифікатор. Ідентифікатор плитки - це (x, y) що охоплює відображуваний діапазон.

Отже, якщо у нас було 9 плиток у деякому відображеному діапазоні, їхні ідентифікатори плитки можуть бути:

*(0, 0) (1, 0) (2, 0)*

*(0, 1) (1, 1) (2, 1)*

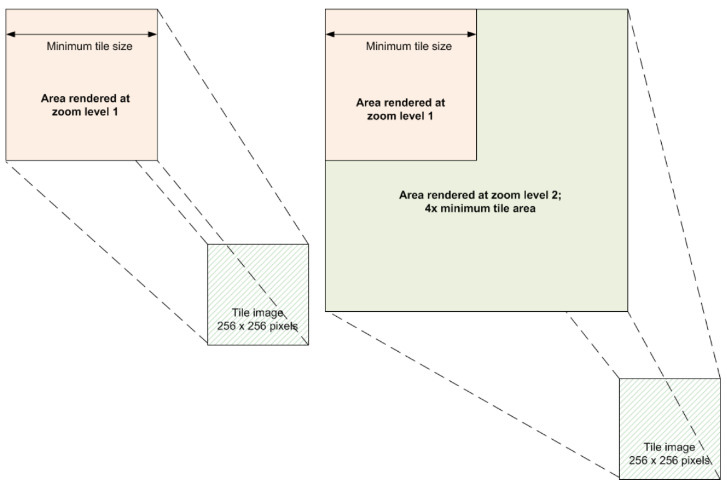
*(0, 2) (1, 2) (2, 2)*

Мапер буде читати всі функції на наших даних карти і видати їх на reducer, відповідальним за розбиття їх на плитки. Ми попередньо обчислимо, які плитки несуть відповідальність за які функції, а маппери надішлють функції на reducer для цих плиток. У reducer ми надамо кожне вихідне зображення.

Ми не хочемо використовувати окремий reducer для кожної плитки. Замість цього кожен reducer буде нести відповідальність за відображення набору плиток. Ці плитки не обов'язково повинні бути сусідні один з одним (хоча, такий обʼєкт, як довгий дорожний сегмент може бути винесений на декілька плиток, і тому було б добре, якщо ми б могли звести до мінімуму обсяг даних, необхідних для переміщення між маперами та reducer).

Мінімальний розмір плитки - це величина, що визначена в коді, яка є найменшою шириною і висотою в градусах, що відповідає окремим плиткам. Не змінюйте цю константу, оскільки зміна цього параметра визначає, наскільки чутлива сітка плитки, і має суттєвий вплив на використання кластера. Кожна плитка визначається як 256 x 256 пікселів.

Рівень масштабування - це ціле число >= 1, яке є експоненціальним мультиплікатором мінімального розміру плитки, який визначає, скільки поверхні площі виводяться на плитку. При масштабуванні 1 рівень мінімальної розміру ділянки для площі повертається до 256x256 пікселів; це дає найближчий масштаб. При масштабуванні 2-го рівня ми беремо ширину у 2 рази завдовжки та 2 рази заввишки, а так само повертаємо же розмір.



Одним з компонентів, який ви маєте реалізувати, є драйвер, який, враховуючи кількість reducer, відображуваного діапазону та передбачуваного рівня масштабування для візуалізації, розбиває відображуваний діапазон на плитки, призначить їм ідентифікатори та групує ідентифікатори плитки в стільки наборів плиток, скільки є reducer’ів. Цей дільник для плитки відповідає за підтримку 2-стороннього відображення: вона повинна мати можливість повернути ідентифікатор плитки у будь-якому (lat, lon) діапазоні та вказати ідентифікатор плитки.

**Типи даних**

Файли даних TIGER постачаються як набір полів, організований у записах. Кожен запис - це одна рядок тексту. У поля фіксована ширина, а це означає, що немає комами або інших обмежувачів, що позначають краї поля. Натомість ми знаємо, скільки символів може займати кожне поле, і весь текст у назвах символів є полем. Дані TIGER містять декілька типів записів.

Ви повинні проаналізувати перші два типи записів. Це:

1 - повні базові дані (дані лінії)

2 - координати фігури (дані багатокутника)

Нижче наводяться сторінки з словника даних TIGER, який чітко визначає записи:

Record Type 1 - Complete Chain Basic Data Record

Field BV Fmt Type Beg End Len Description

RT No L A 1 1 1 Record Type

VERSION No L N 2 5 4 Version Number

TLID No R N 6 15 10 TIGER/Line(R) ID,

Permanent 1-Cell Number

SIDE1 Yes R N 16 16 1 Single-Side Source Code

SOURCE Yes L A 17 17 1 Linear Segment Source Code

FEDIRP Yes L A 18 19 2 Feature Direction, Prefix

FENAME Yes L A 20 49 30 Feature Name

FETYPE Yes L A 50 53 4 Feature Type

FEDIRS Yes L A 54 55 2 Feature Direction, Suffix

CFCC No L A 56 58 3 Census Feature Class Code

FRADDL Yes R A 59 69 11 Start Address, Left

TOADDL Yes R A 70 80 11 End Address, Left

FRADDR Yes R A 81 91 11 Start Address, Right

TOADDR Yes R A 92 102 11 End Address, Right

FRIADDL Yes L A 103 103 1 Start Imputed Address Flag, Left

TOIADDL Yes L A 104 104 1 End Imputed Address Flag, Left

FRIADDR Yes L A 105 105 1 Start Imputed Address Flag, Right

TOIADDR Yes L A 106 106 1 End Imputed Address Flag, Right

ZIPL Yes L N 107 111 5 ZIP Code(R), Left

ZIPR Yes L N 112 116 5 ZIP Code(R), Right

AIANHHFPL Yes L N 117 121 5 FIPS 55 Code

(American Indian/Alaska Native Area/Hawaiian Home Land),

2000 Left AIANHHFPR Yes L N 122 126 5 FIPS 55 Code

(American Indian/Alaska Native Area/Hawaiian Home Land),

2000 Right AIHHTLIL Yes L A 127 127 1

American Indian/Hawaiian Home Land Trust Land Indicator,

2000 Left AIHHTLIR Yes L A 128 128 1

American Indian/Hawaiian Home Land Trust Land Indicator,

2000 Right СENSUS1 Yes L A 129 129 1 Census Use 1

CENSUS2 Yes L A 130 130 1 Census Use 2

STATEL Yes L N 131 132 2 FIPS State Code,

2000 Left (always filled both sides, except at U.S. boundaries)

STATER Yes L N 133 134 2 FIPS State Code,

2000 Right (always filled both sides, except at U.S.boundaries)

COUNTYL Yes L N 135 137 3 FIPS County Code,

2000 Left (always filled both sides, except at U.S. boundaries)

COUNTYR Yes L N 138 140 3 FIPS County Code,

2000 Right (always filled both sides, except at U.S. boundaries)

COUSUBL Yes L N 141 145 5 FIPS 55 Code (County Subdivision),

2000 Left COUSUBR Yes L N 146 150 5 FIPS 55 Code (County Subdivision),

2000 Right SUBMCDL Yes L N 151 155 5 FIPS 55 Code (Subbarrio),

2000 Left SUBMCDR Yes L N 156 160 5 FIPS 55 Code (Subbarrio),

2000 Right PLACEL Yes L N 161 165 5 FIPS 55 Code (Place/CDP),

2000 Left PLACER Yes L N 166 170 5 FIPS 55 Code (Place/CDP),

2000 Right TRACTL Yes L N 171 176 6 Census Tract,

2000 Left TRACTR Yes L N 177 182 6 Census Tract, 2000 Right

BLOCKL Yes L N 183 186 4 Census Block Number,

2000 Left BLOCKR Yes L N 187 190 4 Census Block Number,

2000 Right FRLONG No R N 191 200 10 Start Longitude

FRLAT No R N 201 209 9 Start Latitude

TOLONG No R N 210 219 10 End Longitude

TOLAT No R N 220 228 9 End Latitude

BV (Blank Value):

Yes = Blank value may occur here; No = Blank value should not occur here

Fmt:

L = Left-justified (numeric fields have leading zeros and may be interpreted as character data)

R = Right-justified (numeric fields do not have leading zeros and may be interpreted as integer data)

Type:

A = Alphanumeric, N = Numeric

Record Type 2 - Complete Chain Shape Coordinates

Field BV Fmt Type Beg End Len Description

RT No L A 1 1 1 Record Type

VERSION No L N 2 5 4 Version Number

TLID No R N 6 15 10 TIGER/Line(R) ID,

Permanent 1-Cell Number

RTSQ No R N 16 18 3 Record Sequence Number

LONG1 No R N 19 28 10 Point 1, Longitude

LAT1 No R N 29 37 9 Point 1, Latitude

LONG2 Yes R N 38 47 10 Point 2, Longitude

LAT2 Yes R N 48 56 9 Point 2, Latitude

LONG3 Yes R N 57 66 10 Point 3, Longitude

LAT3 Yes R N 67 75 9 Point 3, Latitude

LONG4 Yes R N 76 85 10 Point 4, Longitude

LAT4 Yes R N 86 94 9 Point 4, Latitude

LONG5 Yes R N 95 104 10 Point 5, Longitude

LAT5 Yes R N 105 113 9 Point 5, Latitude

LONG6 Yes R N 114 123 10 Point 6, Longitude

LAT6 Yes R N 124 132 9 Point 6, Latitude

LONG7 Yes R N 133 142 10 Point 7, Longitude

LAT7 Yes R N 143 151 9 Point 7, Latitude

LONG8 Yes R N 152 161 10 Point 8, Longitude

LAT8 Yes R N 162 170 9 Point 8, Latitude

LONG9 Yes R N 171 180 10 Point 9, Longitude

LAT9 Yes R N 181 189 9 Point 9, Latitude

LONG10 Yes R N 190 199 10 Point 10, Longitude

LAT10 Yes R N 200 208 9 Point 10, Latitude

**Cполучення**

Дивлячись на типи даних вище, ясно, що багатокутникові файли містять полігони, які складають багатокутник, але не кажуть, що насправді є полігоном. ID полігону, однак, містить ідентифікатор Tiger / Line. Це такий самий ідентифікатор, як і запис першого типу, який описує лінію. Цей рядок приєднується до одного з боків багатокутника. Лінійний запис описує, який тип об'єкта представляє полігон (наприклад, вода, школа, парк тощо). Це зберігається у своєму полі CFCC (Class Feature Code Census).

Передача MapReduce має приєднувати багатокутник до пов'язаного з ним рядкового запису, щоб заповнити більш повне тіло запису, щоб визначити тип багатокутника, а отже, як його відтворити. Існує друге сполучення між BGN та наборами даних про кількість населення. У записах BGN міститься ідентифікатор для місцезнаходження (назви міста та штатів), а також координати широти та довготи.

Ви повинні використовувати MapReduce для виконання сполучення, заповнивши широту та довготу з відповідного запису BGN.

Використовуйте Google Maps для рендеринга.

**Питання:**

1. Опишіть роботу власної програми.

2. Опишіть, як працює індекс геокоду для полегшення адресних запитів.

3. Опишіть процес обʼєднання датасетів.