assignment3

October 28, 2024

1 Assignment 3 - Ishaan Sathaye

1.0.1 1.

Consider two files. One file has information about students (ID, name, address, phone number, courses taken):

1, John, 123 Main, 233 223 5566, (CSC365 CSC369 CSC469) is an example.

Consider a second files that has information about courses and their difficulty:

(CSC365, 1)

(CSC369, 1)

(CSC469, 2) is an example.

Your goal is to print the names and addresses of students that have taken all the top N most difficult classes.

Solution:

- Job 1: Find the top N most difficult classes.
  - Mapper
    - (?, "course_id, difficulty") -> (null, Course)
    - TreeSet of Course class (to keep track of the top N most difficult classes)
      - if TreeSet.size() > N, remove the last
    - Course class consists of course_id and difficulty
      - compareTo: compareTo() on difficulty if not equal, else comapreTo() on course_id
  - Reducer
    - (null, Course) -> (null, "course_id, …") (output top N courses)
    - Keep track of the top N most difficult classes using a TreeSet of Course class.
- Job 2: Join files
  - topN Mapper
    - (?, "course_id …") -> ((null, 1), ("course_id, …", "topN"))
    - 1 so that topN courses appears first in the reducer
  - student Mapper
    - (?, "ID, name, address, phone number, courses taken") -> ((null, 2), ("ID, name, address, phone number, courses taken", "student"))
  - Reducer
    - ((null, 1), ("course_id, …", "topN"))

        * ((null, 2), ("ID, name, address, phone number, courses taken", "student")) ->
        * ("ID, name, address", null)
        * Parsing to compare the courses taken (rest of values) with the top N courses

1.0.2 2.

Consider an input file that has information about students (ID, name, address, phone number, (course taken, grade)):

1, John, 123 Main, 233 223 5566, ((CSC365 A) (CSC369 A) (CSC469 B)) is an example.

The problem is to print the N students with the highest GPA. You can assume that you get 4 points for A, 3 points for a B, 2 points for a C, and 1 point for a D, and 0 points for an F. The average GPA will be a real number between 0 and 4.

Solution:

- Job 1: Calculate GPA for each student.
  - Mapper
    * (?, "student_id, name, address, phone, ((course, grade))") -> (student_id, GPA_Record)
    * GPA_Record class consists of name, address, and GPA.
    * For sorting, it first compares the gpa, then name, and finally the id.
    * Convert the grades to points and calculate the GPA.
  - Reducer
    * (student_id, GPA_Record) -> (student_id, name, address, GPA)
    * Pass through the student_id and GPA to aggregate the results.
- Job 2: Find the top N students with the highest GPA.
  - Mapper
    * (?, "student_id, name, address, GPA") -> (null, "student_id, name, address, GPA")
    * TreeSet to keep track of the top N students, sorted by GPA in descending order.
    * Send to a singe reducer.
  - Reducer
    * (null, "student_id, name, address, GPA") -> (null, "student_id, name, address, GPA")
    * Output the top N students using the TreeSet of GPA_Record class

1.0.3 3.

Consider the following input file.

Enrolled (student id, course name). For example: (1,CSC354) means that John is enrolled in CSC354.

The problem is to print the top N most popular classes (i.e., classes with the highest enrollment).

Solution:

- Job 1: Count the number of students enrolled in each class.

  - Mapper
    - (?, “student_id, course_name”) -> (course_name, 1)
  - Reducer
    - (course_name, count) -> (course_name, count)
    - Aggregate the counts for each course.
- Job 2: Find the top N most popular classes.
  - Mapper
    - (? , “course_name, count”) -> (null, “course_name, count”)
    - Use a TreeSet of Course class to keep track of the top N classes.
  - Reducer
    - (null, “course_name, count”) -> (null, “course_name, count”)
    - Output the top N classes using a TreeSet of Course class sorted by count in descending order.