Users in Audience Segments

Yarkin Gazi – Software Engineer, Data Ad Products

10/10/2022

INTRO:

I haven't coded in Go Language anytime before this assignment. In our meeting with Mr. Kinshuk Varshney, He mentioned that at New York Times, they are usually using Go Language and he told me that it has adapted some concepts from C language and JAVA. So I wanted to utilize this take home task and learned GO Language and I have completed this system design technical assessment in Go.

SUMMARY:

Given a 2d binary NxM matrix where N is the number of rows of end-users and M is the number of columns of the audience segments. Data can be retrieved from a database, Google BigTable or Cassandra. However, for the sake of creating an algorithm, I initialize a 2D Matrix with random values every time an executable is created. 10 columns are holding the values for segments for 20 different users. Normally users can be millions and segments can be up to 999. In the question, segments are labeled as SEGMENTA, SEGMENTB, …etc. So I have created an algorithm which uses a map. Key values are Segment names and values corresponding are the column index number which the segment is stored. There is no specific pattern declared in the question so I assumed that Segment names are alphabetically increasing as column names are increasing. A - > Z , and then AA - > ZZ etc. Users can enter values as follows:
AND,SEGMENTA,SEGMENTB
AND,SEGMENTA,SEGMENTB,SEGMENTC
AND,SEGMENTA,SEGMENTB,SEGMENTC,SEGMENTD
OR,SEGMENTA,SEGMENTB,SEGMENTC
SEGMENTA

METHODS:

INPUT TYPES:
1)AND/OR OPERATION FOLLOWED BY SEGMENTS UP TO 4: eg. AND, SegmentA, SegmentB
2)One Segment Specification: eg. SegmentA

ASSUMPTIONS:
COLUMNS CONSECUTIVELY COME FROM THE DATABASE AS FOLLOWS:
COLUMN1: Segment1(Corresponds to SegmentA)
COLUMN2: Segment2(Corresponds to SegmentB)
COLUMN3: Segment3(Corresponds to SegmentC)
.
.
.
.
COLUMN27: Segment27(Corresponds to SegmentAA)

COLUMN28: Segment27(Corresponds to SegmentAB)

.

.

.

.

COLUMN676: Segment676(Corresponds to SegmentZZ)

COLUMN677: Segment677(Corresponds to SegmentAAA)

.

.

.

COLUMN999: Segment999(Corresponds to SegmentAKK)

-Based on the documentation, We are assuming there are only 10 columns(segments).

-These segments are SegmentA through SegmentJ(Segment1 through Segment10)

-Based on the input, we can directly go find the index of Segment1 in our data_matrix which is the 0th index. (like Segment2's index in data_matrix is 1 etc.)

-However, this would work only for data which have 10 segments. I will impelement this. Furthermore,

I will propose an algorithm which would handle all the 999 columns if such a large data come from the database.

Firstly, I created a lookup map so that I do not have to iterate all the columns every time a user enters input. We will be iterating in one dimension for sure which will take us O(N) time. However, almost no algorithm is efficient if that algorithm is O(N^2) So, I used the map to look up where the segment is located and only iterate that column/columns if needed. My solution has the notion of storing keys and their indices. This can be enhanced and implemented to a larger scale if there were many more segments, iterating the column which has the segment names and storing each segment name and their corresponding index so that a lookup would be ready for future purposes. So far these are all pros of the algorithm, when it comes to cons, It has got to be preprocessed if data is retrieved in any other formats. Since in the scope of this question, I cannot presume how and in which shape the data is retrieved, I sticked to the base assumption where the main data is in 2D Matrix.

COMPLEXITY ANALYSIS:

Time Complexity is O(N) since there is not much of a time requirement than a for loop iterating N times where N is the number of users.

Space Complexity: Since an auxiliary map is used, it is also O(N), which N is the number of columns. If we take 999 as the maximum it can get and 999 as a constant value, then one can claim it is constant.