Heidelberg University — Group Artificial Intelligence for Programming (AIP)
Winter semester 2024/25 — Prof. Dr. Artur Andrzejak, Kim Tuyen Le

Problem Set 5 for lecture Mining Massive Datasets

Due December 02, 2024, 23:59 CET

Exercise 1 (1 point)

List common mechanisms for resolving hash collisions and explain how key-value dictionaries in C++, Java, and Python handle hash collisions, respectively.

Exercise 2 (2 points)

Read Section 2.7 of the book "Algorithms and Data Structures for Massive Datasets" (2022) by Dzejla Medjedovic and Emin Tahirovic[1] about MurmurHash, along with the related materials linked in that section, and answer the following questions:

a) Explain MurmurHash and how the seed value creates independent hash functions

b) Hash functions can be vulnerable to collision attacks, where an attacker deliberately selects input data to generate hash collisions. Is MurmurHash suitable for applications where collision resistance is critical? Explain your answer.

Exercise 3 (2+1 points)

Suppose our stream consists of the integers 3, 1, 4, 1, 5, 9, 2, 6, 5. Our hash functions will all be of the form $h(x) = (ax + b) \mod 32$ for some a and b. You should treat the result as a 5-bit binary integer. Determine (using a spreadsheet or a short program) the number of trailing zeros (the *tail length*) for each stream element and the resulting estimate of the number of distinct elements using the Flajolet-Martin algorithm if the hash function is:

a) $h(x) = (2x + 1) \mod 32$

b) $h(x) = (3x + 7) \mod 32$

c) $h(x) = (4x) \mod 32$.

Bonus point question (1P). Do you see any problems with the choice of the above hash functions? What advice could you give someone who was going to use a hash function of the form $h(x) = (ax + b) \mod 2^k$, and why?

Exercise 4 (1 point)

Suppose we are given the stream 3, 4, 1, 3, 4, 2, 1, 2 to which we apply the Alon-Matias-Szegedy Algorithm to estimate the k-th moment. For each possible position in the stream let X_i be a variable defined at position i. What are the values of $X_i.el$ and $X_i.val$ for each $i = 1, \ldots, 8$? Does it make sense to have a separate variable for each stream position?

[1] free online via HEIDI

Exercise 5 **(3 points)**

Implement a routine (in Python) for computing the k-moment of a stream using the Alon-Matias-Szegedy (AMS) method. The routine receives as input a list of numbers representing the stream, its length n, the degree k of the moment (up to 3, i.e. $k = 0, \dots, 3$), and the desired number v of the auxiliary variables. The output is the estimation of the k-th moment according to AMS, and for debugging/illustration purposes the list of all v variables with their respective data (i.e. $X.el$, $X.val$). For each run, a random set of positions for initializing the variables is picked. Your routine should perform only one pass over the stream data, i.e. you need to update $X.val$ for each already initialized variable X as new stream elements are revealed. However, you can assume that you know n from the beginning. Submit your source code and the logs/reports of the runs.

a) Execute your routine for the 15-elements example stream from the lecture (slide "Example: Auxiliary Variables"; use 1 for a, 2 for b, etc.) for each of $v = 1, 3, 5, 7, 9$ combined with each of $k = 1, 2, 3$ (i.e. you should get 15 results) and report the respective estimates of the moments.

b) Furthermore, compute and state the exact third moment ($k = 3$).

c) What is the impact of v on the accuracy of the estimates?

Exercise 6 **(Bonus, 1 points)**

If we wanted to estimate the fourth moments the Alon-Matias-Szegedy Algorithm, how would we convert $X.val$ to an estimate of the fourth moment, i.e. how does the function $f(X)$ looks like in this case?

Exercise 7 **(6 points)**

Download historical data of the audio platform Audioscrobbler[2] that has been merged with Last.fm in 2005. The file `user_artist_data_small.txt` is a file containing the tab-separated relation "user X has listened to artist Y for Z many times", represented as "<userid> <artistid> <playcount>". Write a Spark program using only DataFrames APIs (i.e., no RDD or SQL APIs) to implement the following queries (submit your code and the logs of your test/runs as a part of the solution).

a) Populate a utility matrix. Be sure to first replace bad artist IDs that are due to known misspellings using the assignments in `artist_alias_small.txt`. Think about how to store the matrix in a reasonable way.

b) Implement a routine that calculates the similarity between users using Pearson correlation coefficient as similarity metric.

c) Write a method that returns the top k most similar users to a given user based on the routine from Part b) (i.e., the k-neighborhood of a user).

d) Create a new artificial user U with a new unique ID of your choice. Fix a set S of your five favorite artists in `artist_data_small.txt.` Add records to the dataset which simulates that U has listened 20 times to artists from S (and only those; the split of the 20 acts of listening on the five artists is your choice). Then use the above routines (and possibly more code) to recommend new artists to user U *and* to user 1029563. Submit as part of your solution your set S, the additional records resulting from this, and the recommendation result for user 1029563.

[2]Available on heiBOX, see the same folder as this pdf, file data/dataset-problemset5-ex7.zip