Sabancı University

Faculty of Engineering and Natural Sciences

CS204 Advanced Programming

Fall 2020-2021

Homework #5

Due: 04/01/2021 - 23:55

PLEASE NOTE:

Your program should be a robust one such that you have to consider all relevant programmer mistakes and extreme cases; you are expected to take actions accordingly!

You HAVE TO write down the code on your own.
You CANNOT HELP any friend while coding.
Plagiarism will not be tolerated!!

1 Introduction

Sketches are probabilistic data structures that can provide approximate results within mathematically proven error bounds while using orders of magnitude less memory than data itself. They are tailored for streaming data analysis on architectures even with limited memory such as single-board computers that are widely exploited for IoT and edge computing. Aim of this homework is to implement several different sketches using **inheritance** and **polymorphism**. This will in turn, alleviate testing process of your newly implemented sketches with streaming data.

2 Background & Implementation Details

Let $\mathcal{U} = \{1, \cdots, n\}$ be the universal set where the elements in the stream are coming from. Let N be size of the stream $\mathtt{s}[.]$ where $\mathtt{s}[i]$ denotes the ith element in the stream. We will use f_x to denote the frequency of an item. Hence,

$$f_x = |\{x = \mathtt{s}[i] : 1 \leq i \leq N\}|.$$

Given two parameters ϵ and δ, a `Sketch` is constructed as a two-dimensional counter table with $w = \lceil \ln(1/\delta) \rceil$ rows and $d = \lceil e/\epsilon \rceil$ columns. Initially, all the counters inside the sketch are set to 0.

There are two fundamental operations for a `Sketch`; the first one is *insert*(x) which updates internal sketch counters to process the items in the stream. To insert $x \in \mathcal{U}$, the counters $\mathtt{sktch}[i][h_i(x)]$ are incremented or decremented for $1 \leq i \leq w$, i.e., **a counter from each row is incremented or decremented where the column IDs are obtained from the hash values. Note that every row is associated with a unique hash function.** This process is shown in Fig. 1

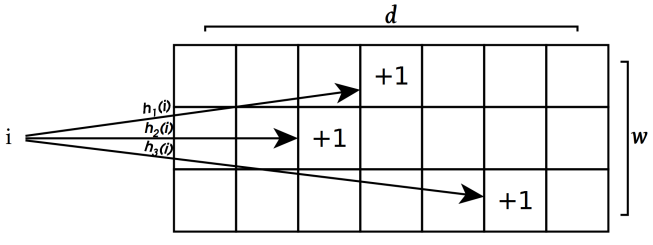

Figure 1: Insertion of an element from stream $\mathtt{s}[.]$ to the table of `CountMinSketch`

The second operation for `Sketch` is *query*(x), which estimates the frequency of $x \in \mathcal{U}$. *query*(x) could be considered as reverse of *insert*(x). For an element $x \in \mathcal{U}$, *query*(x) is needs to be **executed over all rows of `Sketch` and values of counters where *insert*(x) is updated for x needs to be gathered**. Then a heuristic exclusive to every `Sketch` *guesses* frequency of x. In order to fulfill this homework, you need to implement five classes and three different `Sketch`. More details are given below.

- `Sketch` is an abstract class and the base class for `FreqSketch`, has private variables:
  - unsigned no_rows
  - unsigned no_cols
  - StrHash* hashes
  - long long int* table
  - time

  implements:
  - `Sketch` constructor
  - get()
  - add()
  - insert_to_row() *Pure virtual function*
  - insert() *Pure virtual function*
  - query() *Pure virtual function*
  - name()
  - reset()
  - get_no_rows()
  - get_no_cols()
  - getError()
  - print()
  - add_to_time()
  - get_time
  - `Sketch` destructor
- `FreqSketch` is a derived class from `Sketch`, it implements:
  - `FreqSketch` constructor
- `CountSketch` is a derived class from `FreqSketch`, has private variables:
  - StrHash* g_hashes;
  - int* results;

  implements:
  - `CountSketch` constructor
  - insert_to_row() (Alg 2)
  - insert() (Alg 1)
  - query() (Alg 3)
  - name() *Overridden*
  - `CountSketch` destructor
- `CountMinSketch` is a derived class from `FreqSketch`. It implements:
  - `CountMinSketch` constructor
  - insert_to_row() (Alg 4)
  - insert() (Alg 1)

- query() (Alg 5)
- name() *Overridden*
- `CountMinSketch` destructor

- `CountMinMeanSketch` is a derived class from `CountMinSketch`, has private variables:
  - int* results

  implements:
  - `CountMinMeanSketch` constructor
  - insert_to_row() (Alg 4)
  - insert() (Alg 1)
  - query() (Alg 6)
  - `CountMinMeanSketch` destructor

As could be seen, each `Sketch` has their own implementation of *insert_to_row*(x), *insert*(x),*query*(x) and *name()*. Moreover, crucial information of sketch such as table and hashes are initialized in the constructor of `Sketch`, therefore every sketch must call constructor of `Sketch`. For every sketch, **you can only initialize their own private variables in their counstructor** for this homework please note that. Algorithms for *insert_to_row*(x) and *insert*(x) of each sketch is given below.

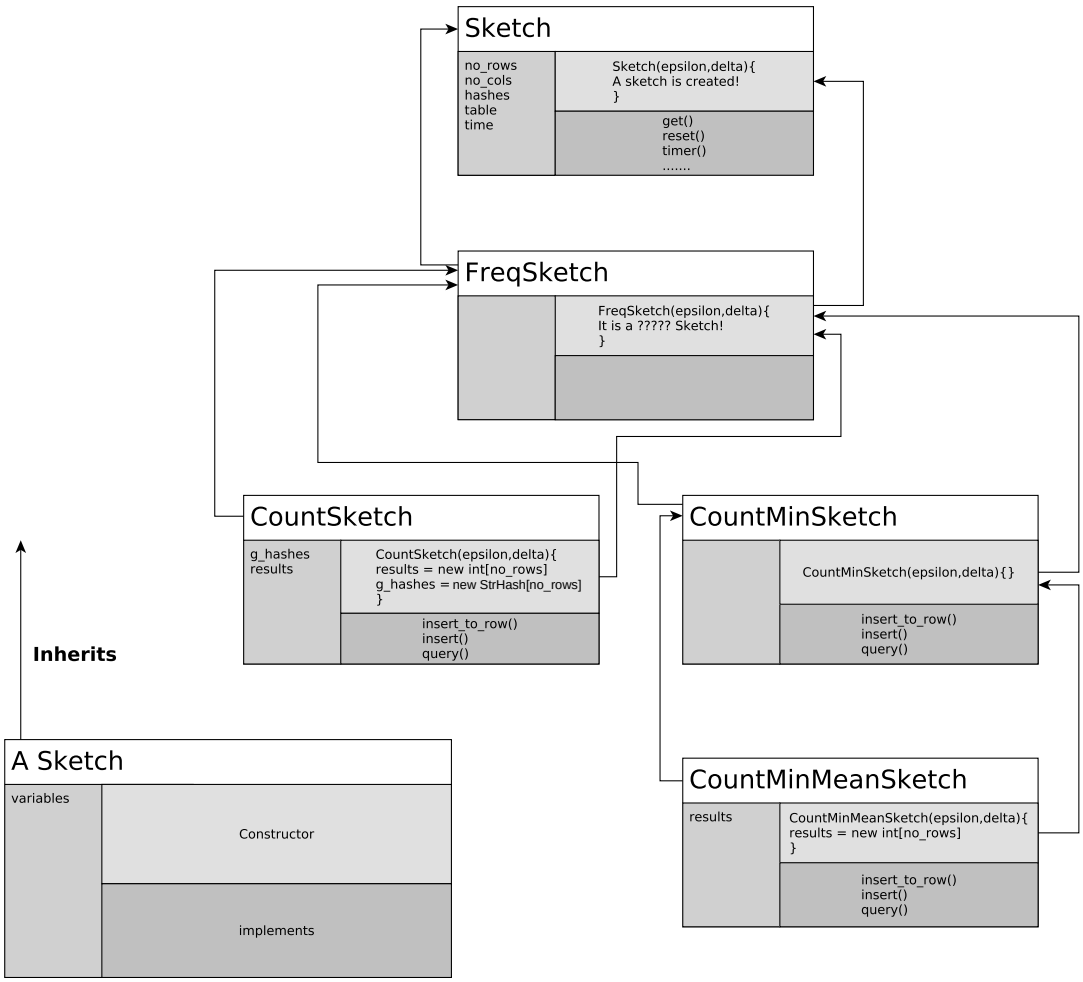

Figure 2: A **summary** of the desired structure.

Algorithm 1: INSERT

```
Input: data: string to be hashed
Output: table is updated
for i ← 0 to no_rows do
    insert_to_row(data, i)
```

Algorithm 2: COUNTSKETCH INSERT_TO_ROW

```
Input: data: string to be hashed
       row_id: row to be updated
Output: table[row_id][hash(data)] is updated
col_id ← hashes[row_id].hash(data)
contrib ← 2 * (g_hashes[row_id].hash(data)%2) − 1
add(row_id, col_id, contrib)
```

Algorithm 3: COUNTSKETCH QUERY

```
Input: data: string to be processed
Output: frequency estimation of data
h_col, coef ← 0
reset results
for i ← 0 to no_rows do
    h_col ← hashes[i].hash(data)
    coef ← 2 * (g_hashes[i].hash(data)%2) − 1
    results[i] ← get(i, h_col) * coef
    if results[i] < 0 then
        results[i] = 0
std::sort(results, results + no_rows)
return results[no_rows / 2]
```

Algorithm 4: COUNTMINSKETCH INSERT_TO_ROW

```
Input: data: string to be hashed
       row_id: row to be updated
Output: table[row_id][hash(data)] is updated
col_id ← hashes[row_id].hash(data)
add(row_id, col_id, 1)
```

Algorithm 5: COUNTMINSKETCH QUERY

```
Input: data: string to be processed
Output: frequency estimation of data
r_freq, freq ←
std::numeric_limits<long long int>::max()
h_col ← 0
for i ← 0 to no_rows do
    h_col ← hashes[i].hash(data)
    r_freq ← get(i, h_col)
    if r_freq<freq then
        freq ← r_freq
return freq
```

Algorithm 6: COUNTMINMEANSKETCH QUERY

```
Input: data: string to be processed
Output: frequency estimation of data
no_stream, r_freq, noise ← 0
h_col ← 0
reset results
for i ← 0 to no_rows do
    h_col ← hashes[i].hash(data)
    r_freq ← get(i, h_col)
    no_stream ← 0
    for j ← 0 to no_cols do
        no_stream += get(i, j)
    noise = ((no_stream - r_freq) / (no_cols -1))
    if r_freq>noise then
        results[i] ← r_freq − noise
std::sort(results, results+no_rows)
return results[no_rows / 2]
```

Data to insert on sketches, base class `Sketch`, hasher class, and parts of main.cpp is provided for this homework. What you need to do is, implementing `CountSketch, CountMinSketch` and `CountMinMeanSketch` classes as described here and fill the gaps in main.cpp to **get the exaxt same result provided in Sample Runs except from times**. Please note that you can't make changes on given parts of main.cpp and implemented `Sketch` class. Also, give attention to declaration order of sketches in main.cpp to get exact same result because values of the hash fuctions are determined by this order.

3 Sample Runs

```
A sketch with 8 rows and 211 columns is created
--> It is a Count-Min Sketch!
A sketch with 8 rows and 211 columns is created
--> It is Count-Min-Mean Sketch!
A sketch with 8 rows and 211 columns is created
--> It is a Count Sketch!
A sketch with 8 rows and 2003 columns is created
--> It is a Count-Min Sketch!
A sketch with 8 rows and 2003 columns is created
--> It is Count-Min-Mean Sketch!
A sketch with 8 rows and 2003 columns is created
--> It is a Count Sketch!
A sketch with 16 rows and 2003 columns is created
--> It is a Count-Min Sketch!
A sketch with 16 rows and 2003 columns is created
--> It is Count-Min-Mean Sketch!
A sketch with 16 rows and 2003 columns is created
--> It is a Count Sketch!

REPORT
--------------------------------------------------
Table Size: 8 Rows x 211 Cols
Errors: CMS: 6.82581 -- CMMS: 2.85205 -- CS: 3.8137
Times: CMS: 1.67589 -- CMMS: 1.68353 -- CS: 2.73099
--------------------------------------------------
Table Size: 8 Rows x 2003 Cols
Errors: CMS: 0.196869 -- CMMS: 0.85463 -- CS: 0.310479
Times: CMS: 1.85028 -- CMMS: 1.83496 -- CS: 2.82895
--------------------------------------------------
Table Size: 16 Rows x 2003 Cols
Errors: CMS: 0.089356 -- CMMS: 0.835899 -- CS: 0.168567
Times: CMS: 3.45893 -- CMMS: 3.45071 -- CS: 5.4101
--------------------------------------------------

TOP3 FREQUENCIES
--------------------------------------------------
REAL-->  WORD: the   COUNT: 176163
-----
Table Size: 8 Rows x 211 Cols
CMS: 181172 -- CMMS: 169347 -- CS: 178847
-----
Table Size: 8 Rows x 2003 Cols
CMS: 176284 -- CMMS: 175100 -- CS: 176186
-----
Table Size: 16 Rows x 2003 Cols
CMS: 176228 -- CMMS: 174856 -- CS: 176254
--------------------------------------------------
REAL-->  WORD: he   COUNT: 89739
-----
Table Size: 8 Rows x 211 Cols
CMS: 92573 -- CMMS: 82937 -- CS: 94203
-----
Table Size: 8 Rows x 2003 Cols
```

```
CMS: 89739 -- CMMS: 88775 -- CS: 89708
-----
Table Size: 16 Rows x 2003 Cols
CMS: 89766 -- CMMS: 88362 -- CS: 89846
-------------------------------------------------
REAL-->  WORD: to   COUNT: 83798
-----
Table Size: 8 Rows x 211 Cols
CMS: 88474 -- CMMS: 74581 -- CS: 85184
-----
Table Size: 8 Rows x 2003 Cols
CMS: 83881 -- CMMS: 82295 -- CS: 83810
-----
Table Size: 16 Rows x 2003 Cols
CMS: 83864 -- CMMS: 82554 -- CS: 83694
-------------------------------------------------
```

4 Some Important Rules

In order to get a full credit, your programs must be efficient and well presented, presence of any redundant computation or bad indentation, or missing, irrelevant comments are going to decrease your grades. You also have to use understandable identifier names, informative introduction and prompts. Modularity is also important; you have to use functions wherever needed and appropriate.

When we grade your homeworks we pay attention to these issues. Moreover, in order to observe the real performance of your codes, we may run your programs in *Release* mode and **we may test your programs with very large test cases.**

What and where to submit (PLEASE READ, IMPORTANT): You should prepare (or at least test) your program using MS Visual Studio 2012 or 2019 C++. We will use the standard C++ compiler and libraries of the above mentioned platform while testing your homework. It'd be a good idea to write your name and last name in the program (as a comment line of course). Submissions guidelines are below. Some parts of the grading process are automatic. Students are expected to strictly follow these guidelines in order to have a smooth grading process. If you do not follow these guidelines, depending on the severity of the problem created during the grading process, 5 or more penalty points are to be deducted from the grade.
Compress your source file into a zip and name your zip file that contains your program as follows:

SUCourseUserName_YourLastname_YourName_HWnumber.zip

Your SUCourse user name is actually your SUNet username that is used for checking sabanciuniv e-mails. Do NOT use any spaces, non-ASCII and Turkish characters in the file name. For example, if your SUCourse user name is cago, name is Çağlayan, and last name is Özbugsızkodyazaroğlu, then the folder name must be:

cago_Ozbugsizkodyazaroglu_Caglayan_hw1.zip

Do not add the data provided with homework to your submission file. Do not add any other character or phrase to the folder name. Make sure that it contains the last version of your homework program. Compress this folder using WINZIP or WINRAR program. Please use "zip" compression. **"rar" or another compression mechanism is NOT allowed.**. Our homework processing system works only with zip files. Therefore, make sure that the resulting compressed file has a zip extension. Check that your compressed file opens up correctly and it contains your source files.

Submit via SUCourse ONLY! You will receive no credits if you submit by other means (e-mail, paper, etc.).

Successful submission is one of the requirements of the homework. If, for some reason, you cannot successfully submit your homework and we cannot grade it, your grade will be 0.

Good Luck!

CS204 Team (Fatih Taşyaran, Kamer Kaya)