Code Structure of the Project:

1) The main data structure of the project is UNION-FIND data structure that contains a range of numeric numbers that helps you look after a single element, and it consists of nodes:

1.1) In every single node in the UNION-FIND structure there is a binary tree which consists of nodes with range of numeric numbers:

1.1.1) The range of the root of the tree is similar to UNION-FIND node range, that it belongs to:

1.1.2) The range of every two "Brother" nodes, is divided:

if a father node range is: [MIN\_range,MAX\_range], then the left child range is [MIN\_range,(MIN\_range+MAX\_range)/2] and the right child range is [(MIN\_range+MAX\_range)/2 + 1 , MAX\_range]

1.1.3) In every node in the binary tree, there's a count-min sketch.

1.1.4) Each node also includes two Boolean parameters isLastupper that indicates if there is an active right son if right is not null, and isLastlower that indicates if there is an active left son if left is not null.

2) INSERTING - "update (x)":   
Calculating the hash function on element x, using xxhash, the returned value is moded (using %) to the specific range of the whole structure, and the inserting is done by searching for the relevant UF node that have the range that x belongs to, and it will be inserted to the deepest leaf that is "Active".

\*A leaf is "Active" if he hasn't been selected to get shrinked.

3) Asking for frequency - "QUERY (x)":   
It searches in the UF structure for the range that contains, the hash value of xxhash on x. When the node is found, we sum all the outcomes for every Count-min sketch in the path from the root until the leaf.

4) Shrink:  
When the available memory is low, the user can decide to shrink the structure by writing “shrink” in the input. The data structure will search for the least loaded active leaf and deactivate it by changing the isLastlower parameter in the parent node to true if it was the left son of the parent, or isLastupper parameter in the parent node to true if it was the right son of the parent. This means that the data that’s already stored in this node will not be changed and while calling query it will return its part of the query as we explained in section 3, but when update is called, the element will not be entered to this node as long as the node is inactive. By shrinking, the allowed error will be doubled, new epsilon = 2 \* epsilon.

5) Expand:  
When the available memory is high, the user can decide to expand the structure by writing   
“expand” in the input. The data structure will search for the most loaded active leaf, if it already has sons, it will activate them. If it does not have sons, it will create two sons and their ranges will be as explained in 1.1.2. In both cases the isLastlower and isLastupper of the parent will be false. By expanding, the allowed error will be reduced by a factor equal to the number of the sketches divided by the number of the sketched plus one.   
new epsilon = epsilon \* (number of sketches) / (number of sketches + 1).

Dependencies:

In our implementation, we used count-min sketch, and the xxhash function. Those files are attached to the project itself.

Also, if you would like to use our script “run.py” you should install the python libraries numpy and matplotlib.