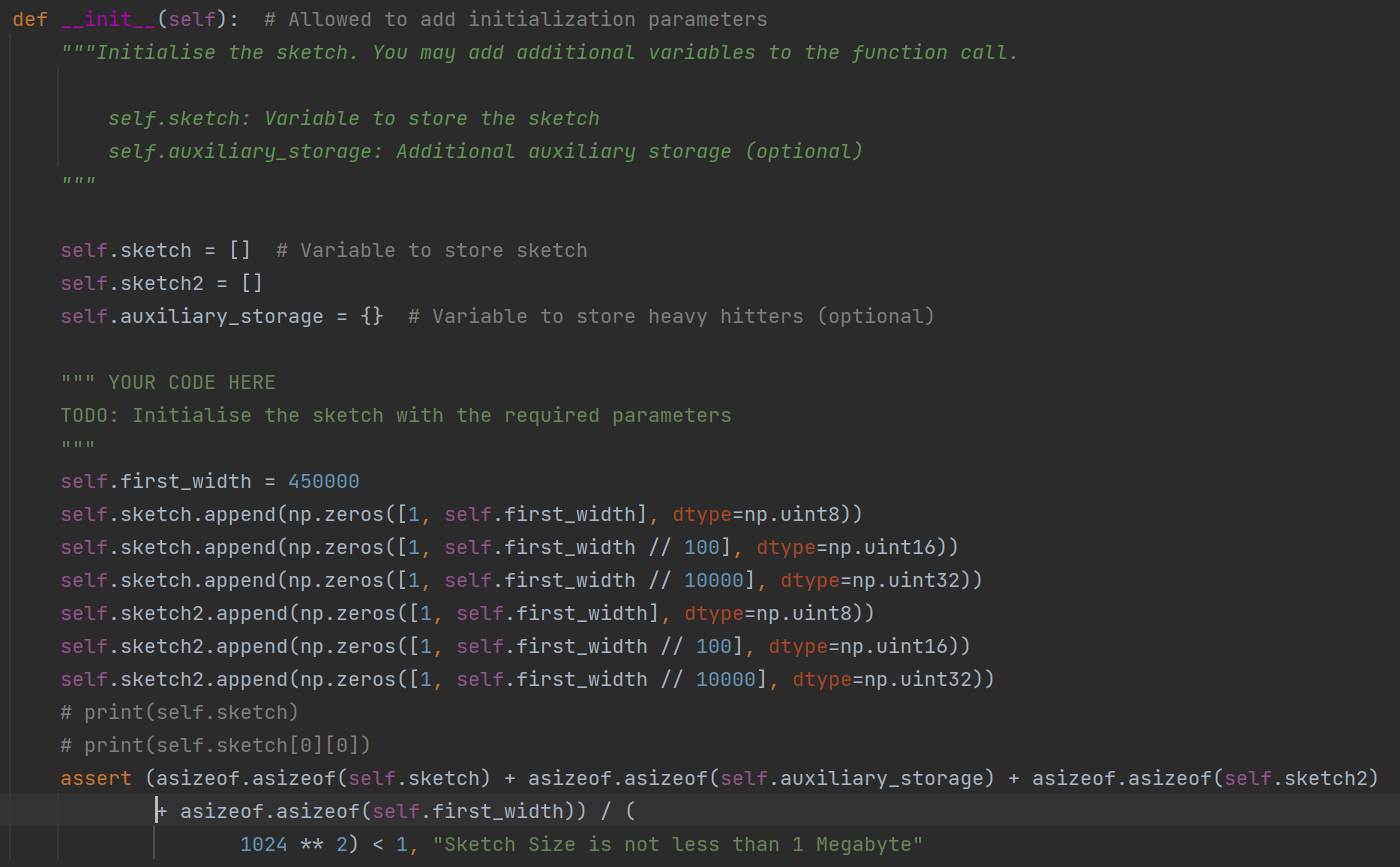
Name: WEI YILEI, Student ID: A0276571W

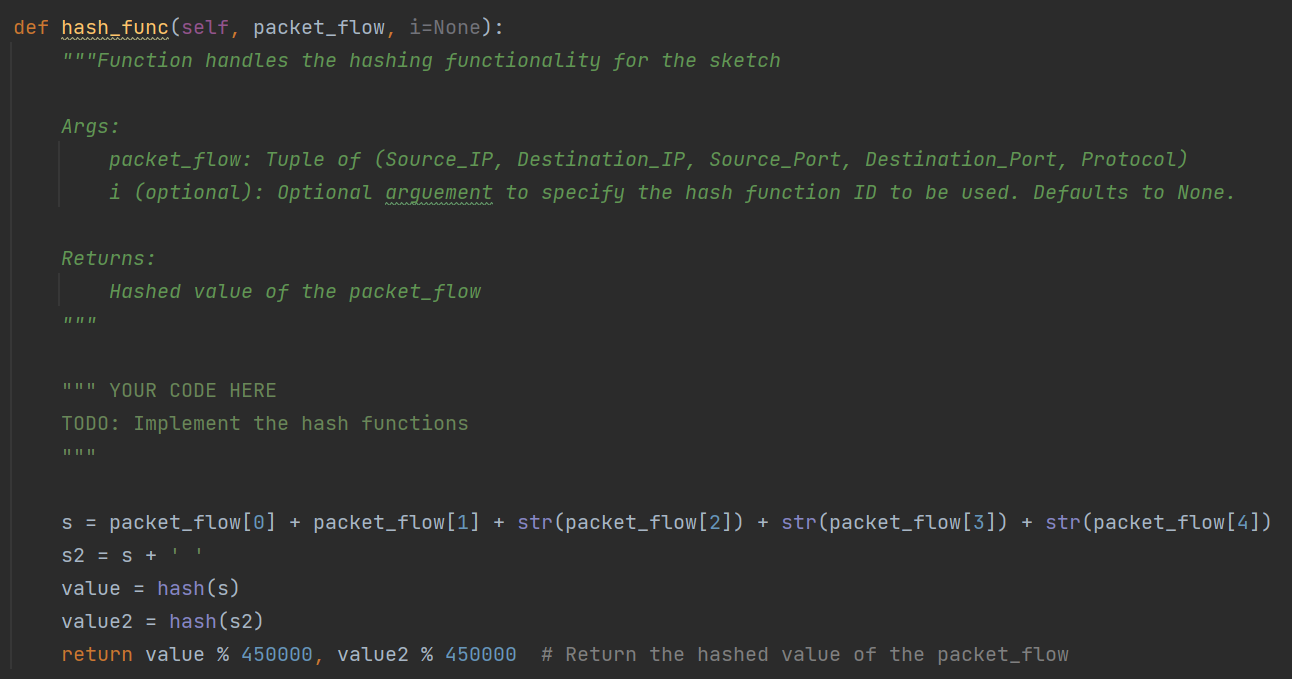
Part1: explanation and implementation of code

After trying basic Count Min Sketch, Elastic Sketch, and Single-Tree FCM-Sketch, the MEAN error results were never very good, and I finally chose to try 2-Tree FCM-Sketch.

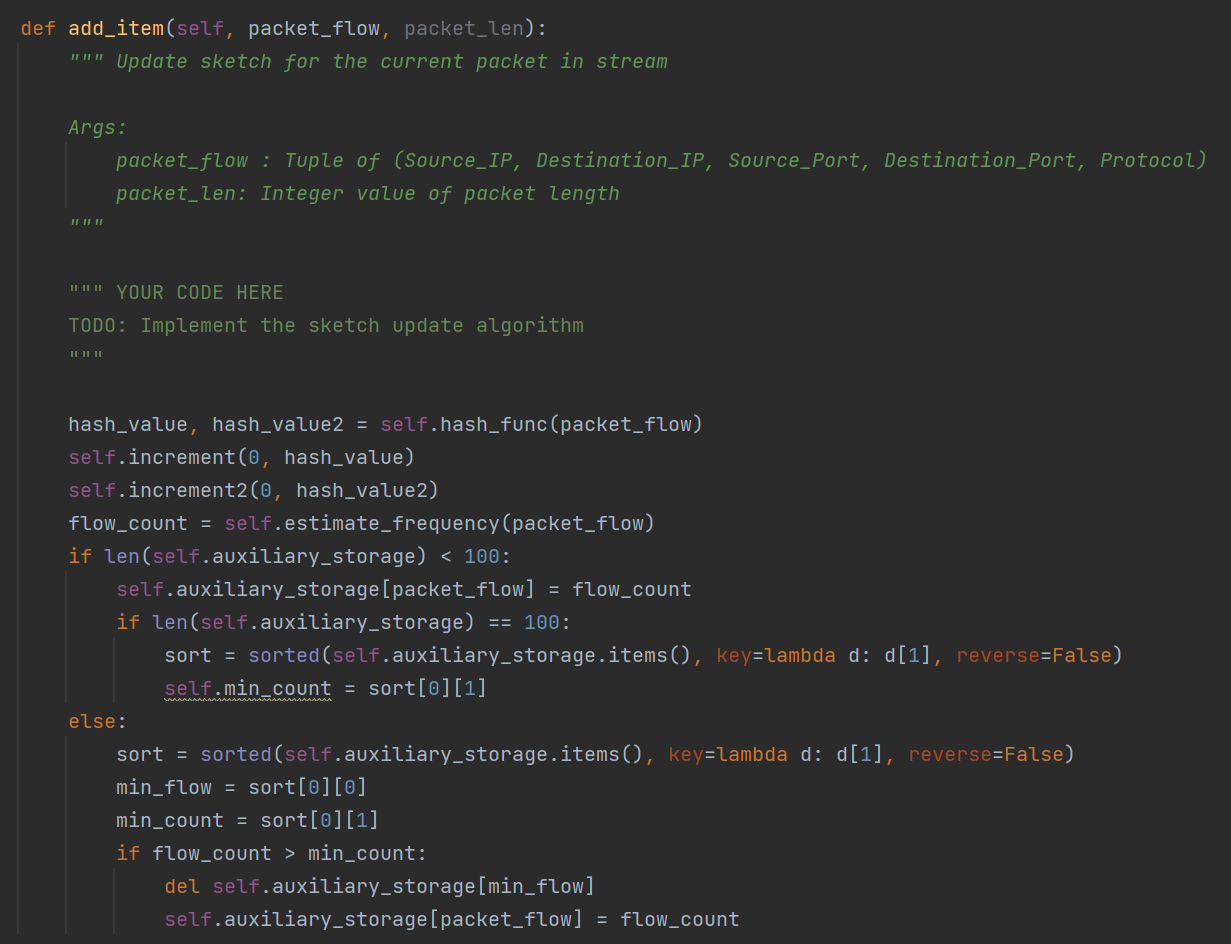
When initializing FCM-Sketch, initialize two k-ary trees with the same structure inside the sketch, because the uint8 array designed with the size of (202000\*5) when trying basic Count Min Sketch before can almost fill up 1MB, and the number of small packets in the network is much more than the number of large packets, so in order to save memory space while minimizing the possibility of hash collisions, after the corresponding calculation I used the first layer of 450000 nodes, the second layer of 4500 nodes, the third layer of 45 nodes of the 100-ary tree. Auxiliary\_storage is a dictionary used to maintain the current top-100 records of the flow data in the process of constantly adding packets.



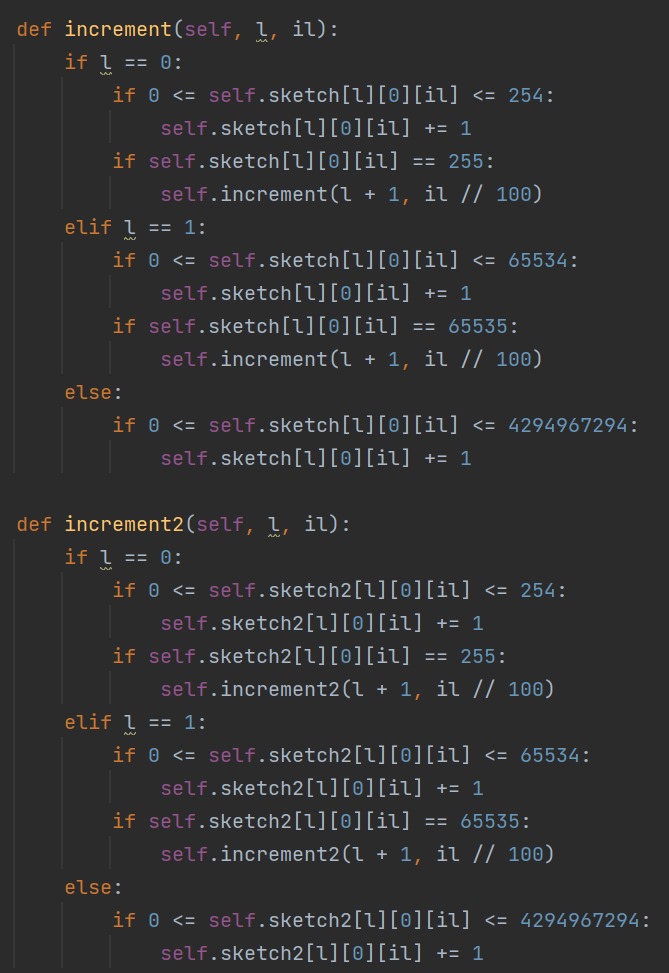
Since I have two trees, I need two hash functions accordingly. In the process of processing input tuple, I tried to put all the characters together, and then use int(str, base=16) way to convert it into an integer, but the final hash result is not ideal, so finally used string hash, for each packet\_flow tuple, put all the elements together into a string s, then one of the hash function calculate hash(s) to get value, another hash function calculate hash(s + ' ') to get value2, the final output will be [value mod 450000, value2 mod 450000]



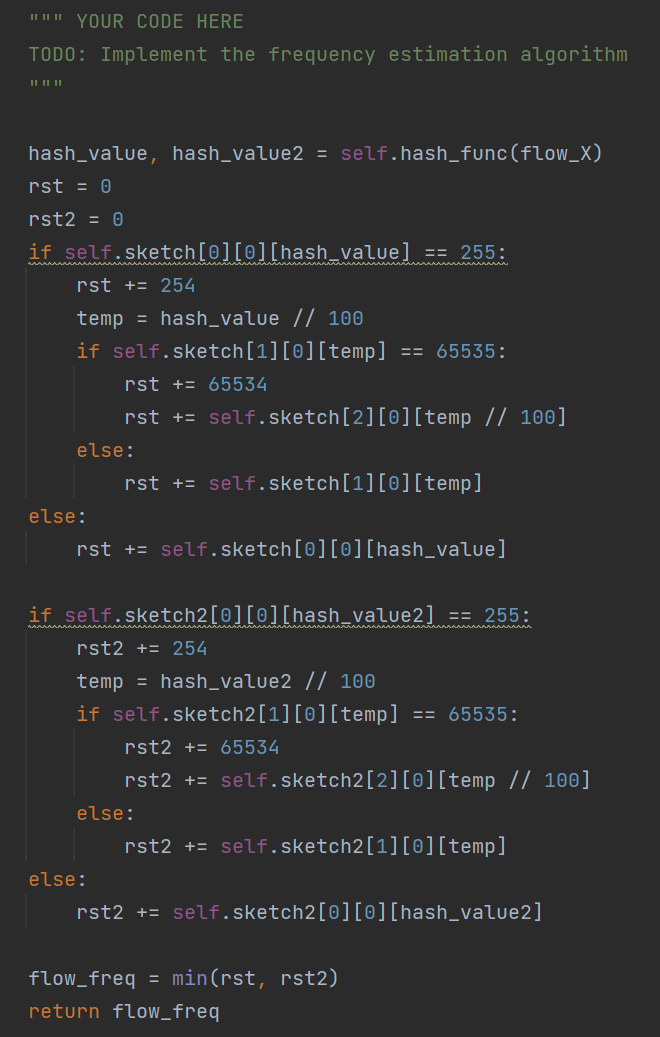
For each packet added, two hash values, hash\_value and hash\_value2 correspondingly, are first computed by hash\_func(), and then these two sketch trees are updated by increment(0, hash\_value) and increment2(0, hash\_value2), respectively. At the beginning of the process, the sketched top-100 dict does not yet have 100 elements, at this time the encountered packet\_flow can all be added into the dict. Once the top-100 dict has 100 elements, we need to identify whether the new flow size of each packet after it is processed will be a member of the new top-100 dict, and if it will, add the new flow and its size to the new top-100 dict, remove the smallest flow from the dict meantime.



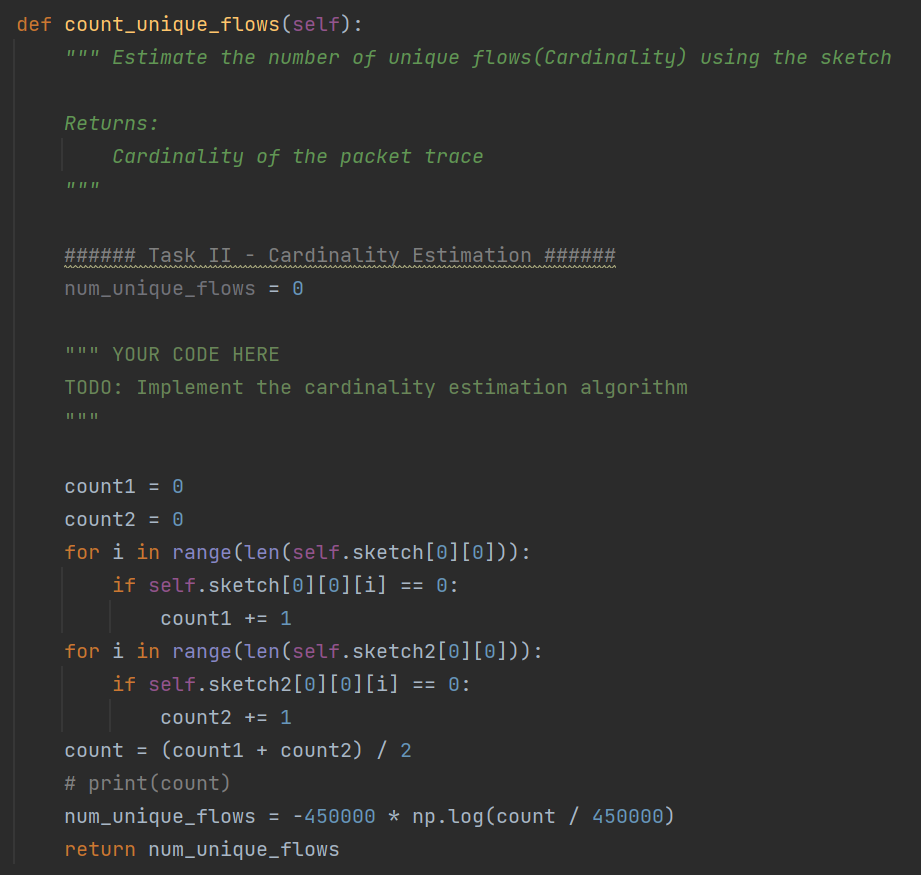
Increment() and increment2() are implemented exactly as described in the paper, will query the node's count of the corresponding layer according to the input values, layer l and hash\_value il. If there is no overflow, the count of node will be directly added by 1 and then exit. If there exists overflow and current layer is not the final layer, then call increment(l + 1, il // 100) or increment2(l + 1, il // 100), which will go to the next layer until there is no overflow.



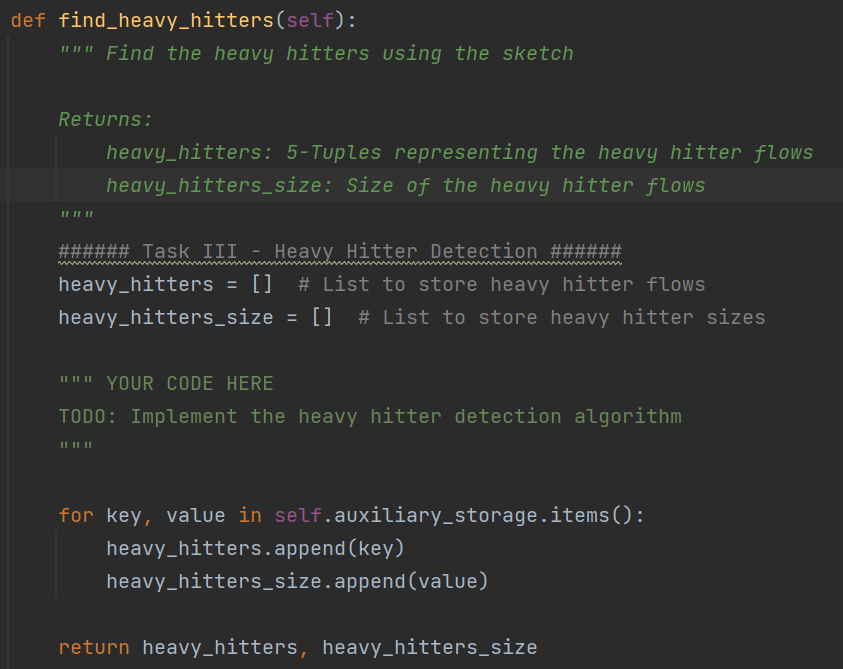
estimate\_frequency(self, flow\_X) query a flow size, first call hash\_func(flow\_X) to get two hash values hash\_value and hash\_value2, and then go to the corresponding sketch tree to query the flow size. First query the node count of layer 0, if it is equal to 28-1(255), which represents the node overflow, we make rst plus 254 and then query its parent node, if its parent node count of layer 1 is equal to 216-1(65535), on behalf of the node overflow, rst will plus 65534 and then query parent node of current node, recursively. If any node is not overflow during the query process, immediately add the node's count to rst and exit the query. Finally the query results of these two sketch trees are rst and rst2, take the flow\_freq for min (rst, rst2) as the output.



Count1 is the empty node counts of sketch tree1 layer0, count2 is the empty node counts of sketch tree2 layer0, take the average of both to get count, which is the average number of empty leaf nodes among those at stage 1. According to the formula, we get num\_unique\_flows = -450000 \* np.log(count / 450000)

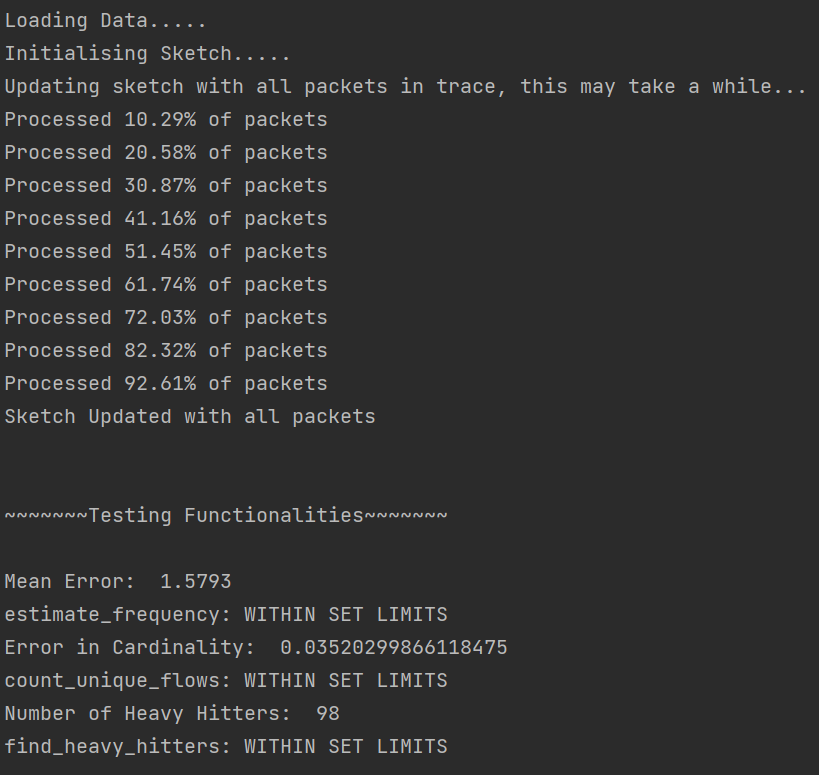


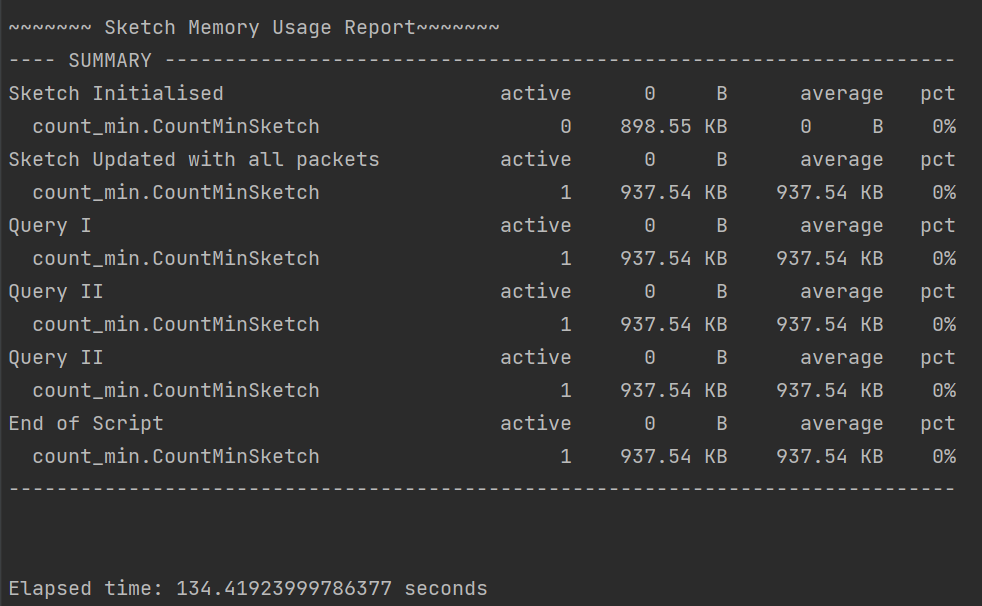
To compute the top-100 heavy hitter, it is sufficient to directly query the keys and values of the top-100 dictionary maintained by the program runtime, where keys is the flow tuples and values is the flow sizes.

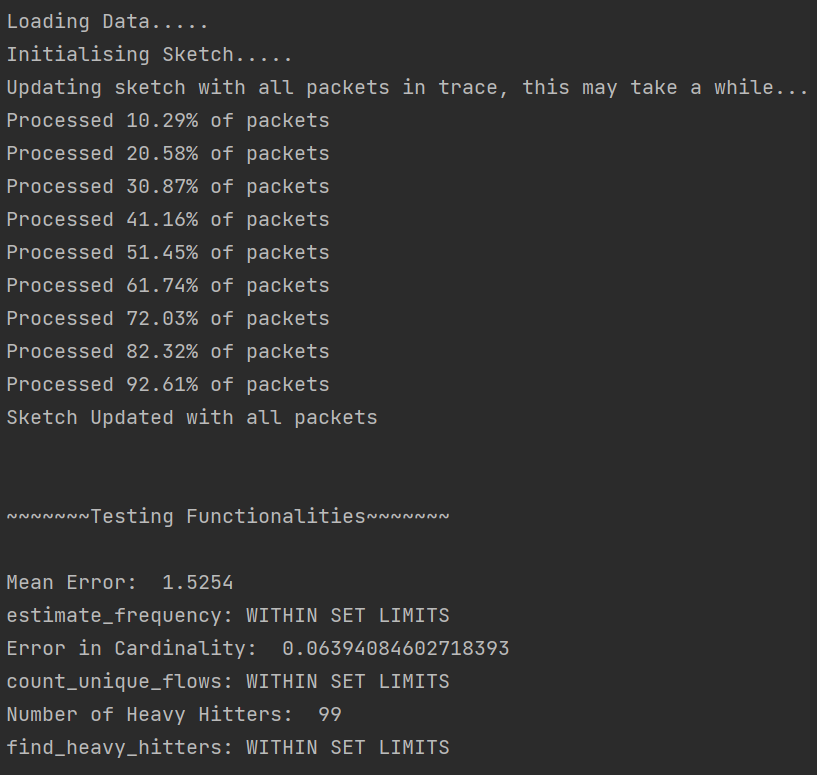


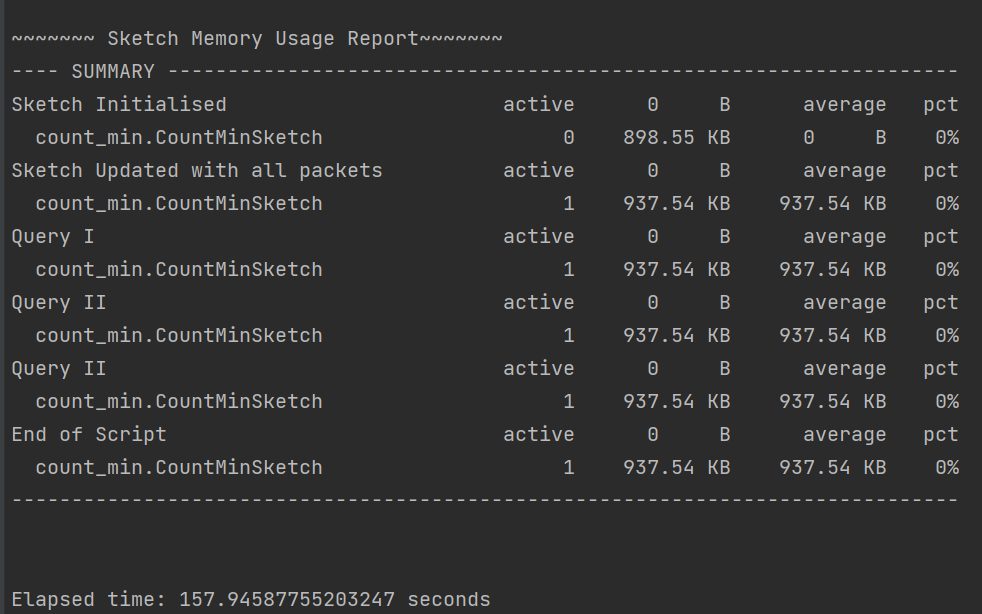
Part2: implementation results

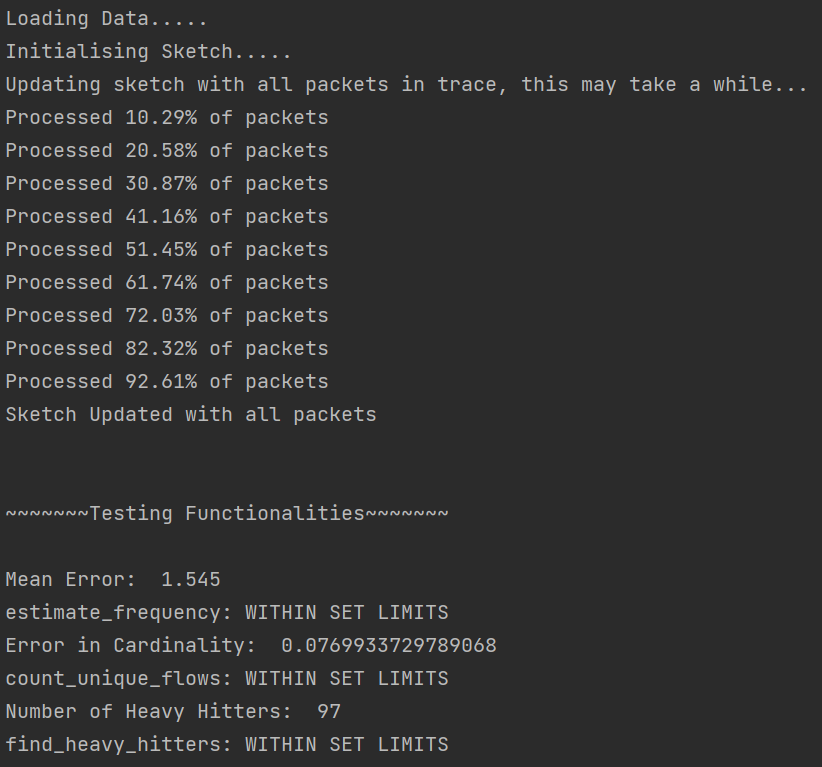
1. Packet-trace.csv(3 times)

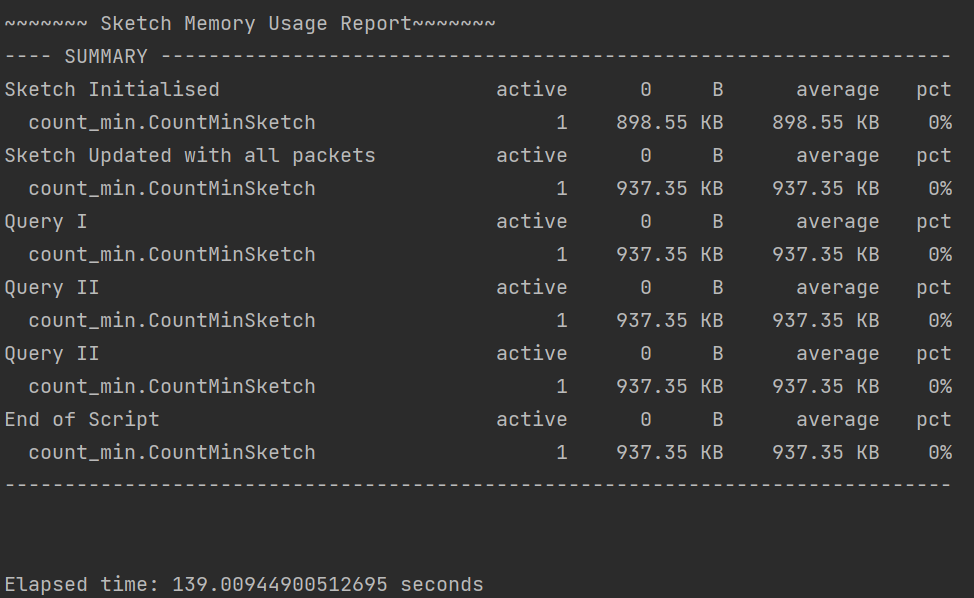




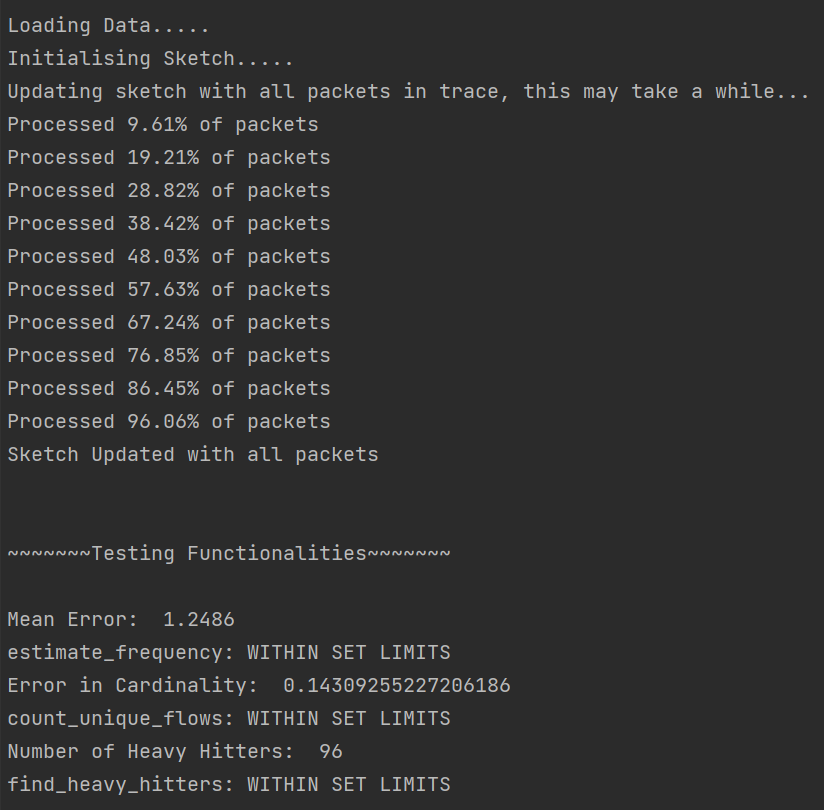


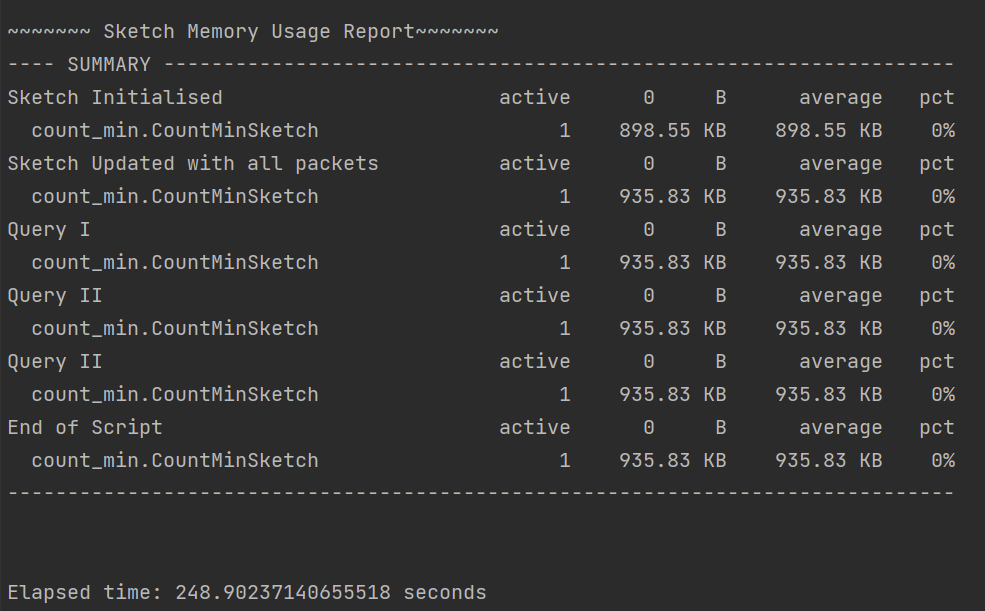


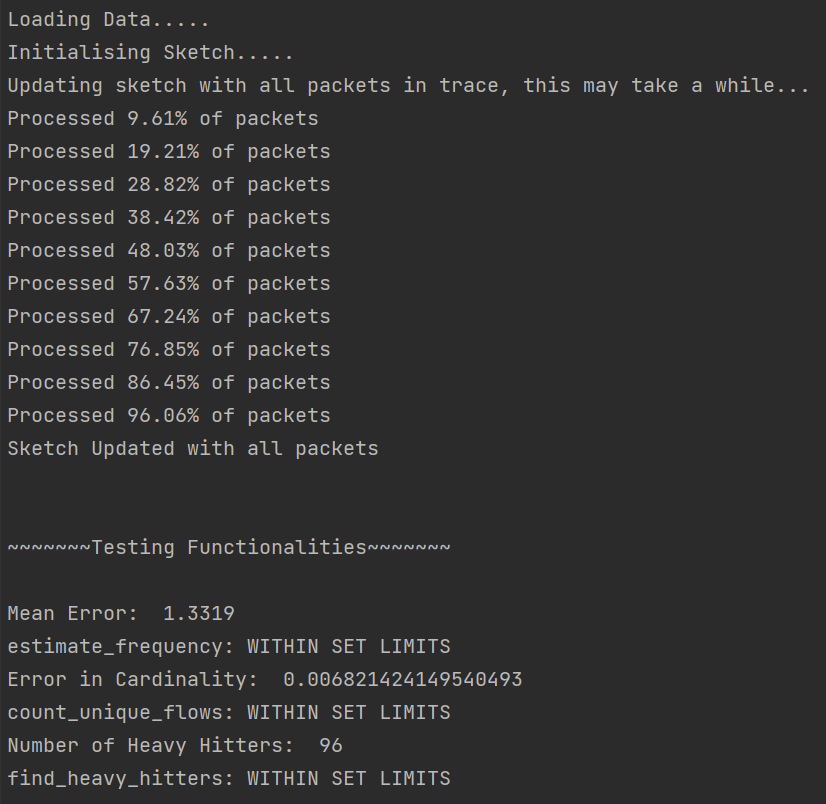


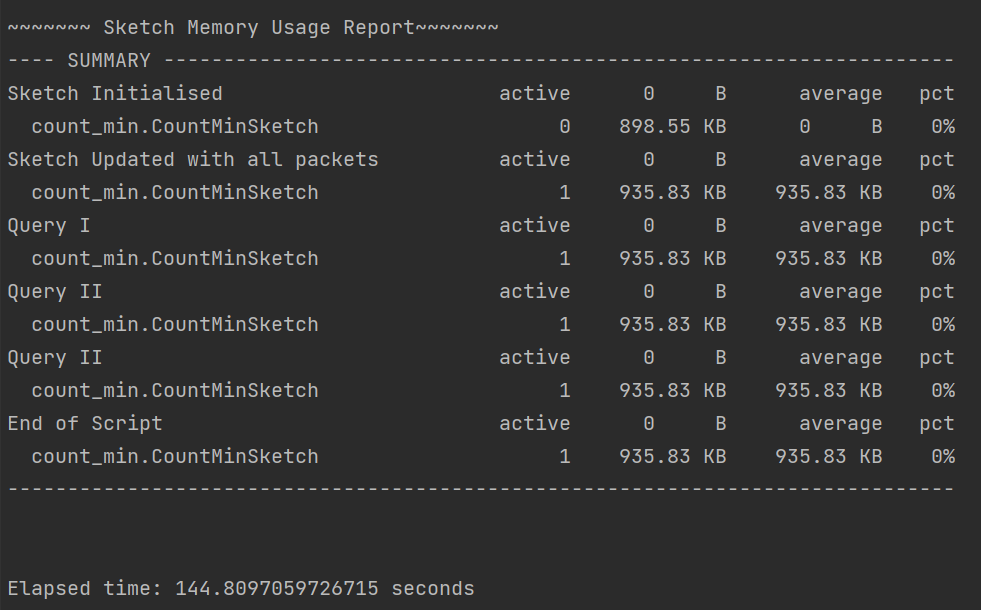


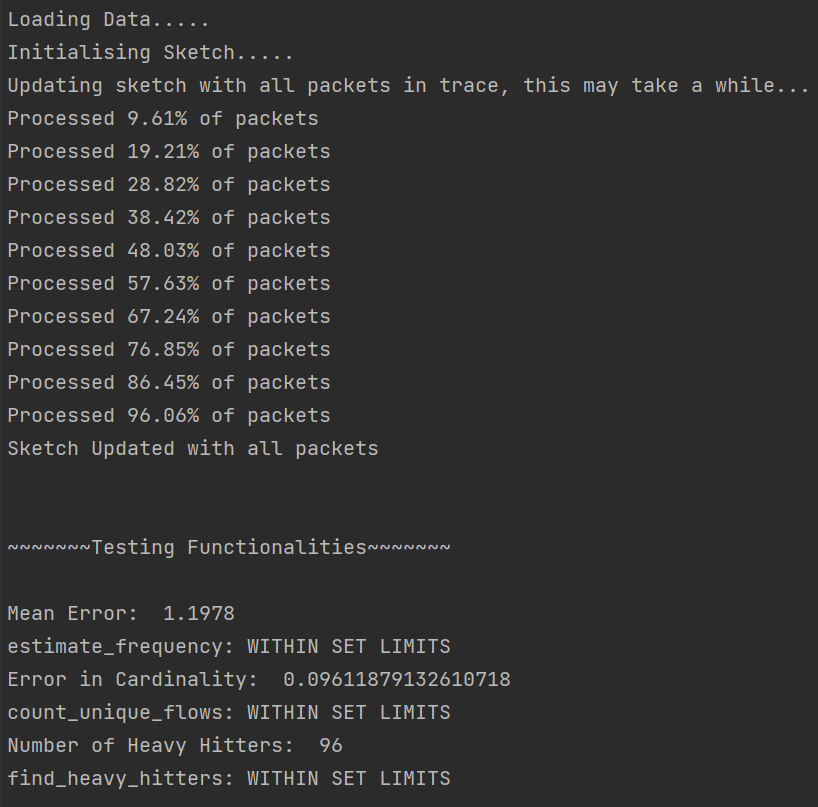
1. Packet-trace2.csv(3 times)

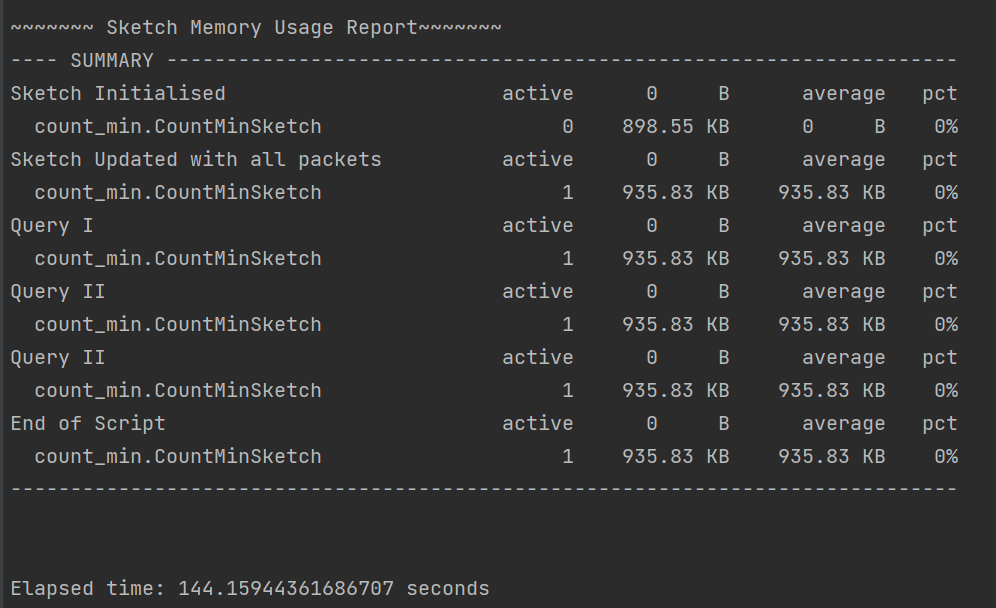












1. Packet-trace3.csv

