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.

```
def __init__(self):  # Allowed to add initialization parameters
    """Initialise the sketch. You may add additional variables to the function call.

        self.sketch: Variable to store the sketch
        self.auxiliary_storage: Additional auxiliary storage (optional)
    """

    self.sketch = []  # Variable to store sketch
    self.sketch2 = []
    self.auxiliary_storage = {}  # Variable to store heavy hitters (optional)

    """ YOUR CODE HERE
    TODO: Initialise the sketch with the required parameters
    """
    self.first_width = 450000
    self.sketch.append(np.zeros([1, self.first_width], dtype=np.uint8))
    self.sketch.append(np.zeros([1, self.first_width // 100], dtype=np.uint16))
    self.sketch.append(np.zeros([1, self.first_width // 10000], dtype=np.uint32))
    self.sketch2.append(np.zeros([1, self.first_width], dtype=np.uint8))
    self.sketch2.append(np.zeros([1, self.first_width // 100], dtype=np.uint16))
    self.sketch2.append(np.zeros([1, self.first_width // 10000], dtype=np.uint32))
    # print(self.sketch)
    # print(self.sketch[0][0])
    assert (asizeof.asizeof(self.sketch) + asizeof.asizeof(self.auxiliary_storage) + asizeof.asizeof(self.sketch2)
            + asizeof.asizeof(self.first_width)) / (
                    1024 ** 2) < 1, "Sketch Size is not less than 1 Megabyte"
```

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]

```python
def hash_func(self, packet_flow, i=None):
    """Function handles the hashing functionality for the sketch

    Args:
        packet_flow: Tuple of (Source_IP, Destination_IP, Source_Port, Destination_Port, Protocol)
        i (optional): Optional arguement to specify the hash function ID to be used. Defaults to None.

    Returns:
        Hashed value of the packet_flow
    """

    """ YOUR CODE HERE
    TODO: Implement the hash functions
    """

    s = packet_flow[0] + packet_flow[1] + str(packet_flow[2]) + str(packet_flow[3]) + str(packet_flow[4])
    s2 = s + ' '
    value = hash(s)
    value2 = hash(s2)
    return value % 450000, value2 % 450000  # Return the hashed value of the packet_flow
```

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.

```python
def add_item(self, packet_flow, packet_len):
    """ Update sketch for the current packet in stream

    Args:
        packet_flow : Tuple of (Source_IP, Destination_IP, Source_Port, Destination_Port, Protocol)
        packet_len: Integer value of packet length
    """

    """ YOUR CODE HERE
    TODO: Implement the sketch update algorithm
    """

    hash_value, hash_value2 = self.hash_func(packet_flow)
    self.increment(0, hash_value)
    self.increment2(0, hash_value2)
    flow_count = self.estimate_frequency(packet_flow)
    if len(self.auxiliary_storage) < 100:
        self.auxiliary_storage[packet_flow] = flow_count
        if len(self.auxiliary_storage) == 100:
            sort = sorted(self.auxiliary_storage.items(), key=lambda d: d[1], reverse=False)
            self.min_count = sort[0][1]
    else:
        sort = sorted(self.auxiliary_storage.items(), key=lambda d: d[1], reverse=False)
        min_flow = sort[0][0]
        min_count = sort[0][1]
        if flow_count > min_count:
            del self.auxiliary_storage[min_flow]
            self.auxiliary_storage[packet_flow] = flow_count
```

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.

```
def increment(self, l, il):
    if l == 0:
        if 0 <= self.sketch[l][0][il] <= 254:
            self.sketch[l][0][il] += 1
        if self.sketch[l][0][il] == 255:
            self.increment(l + 1, il // 100)
    elif l == 1:
        if 0 <= self.sketch[l][0][il] <= 65534:
            self.sketch[l][0][il] += 1
        if self.sketch[l][0][il] == 65535:
            self.increment(l + 1, il // 100)
    else:
        if 0 <= self.sketch[l][0][il] <= 4294967294:
            self.sketch[l][0][il] += 1

def increment2(self, l, il):
    if l == 0:
        if 0 <= self.sketch2[l][0][il] <= 254:
            self.sketch2[l][0][il] += 1
        if self.sketch2[l][0][il] == 255:
            self.increment2(l + 1, il // 100)
    elif l == 1:
        if 0 <= self.sketch2[l][0][il] <= 65534:
            self.sketch2[l][0][il] += 1
        if self.sketch2[l][0][il] == 65535:
            self.increment2(l + 1, il // 100)
    else:
        if 0 <= self.sketch2[l][0][il] <= 4294967294:
            self.sketch2[l][0][il] += 1
```

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 2^8-1(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 2^{16}-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.

```
""" YOUR CODE HERE
TODO: Implement the frequency estimation algorithm
"""

hash_value, hash_value2 = self.hash_func(flow_X)
rst = 0
rst2 = 0
if self.sketch[0][0][hash_value] == 255:
    rst += 254
    temp = hash_value // 100
    if self.sketch[1][0][temp] == 65535:
        rst += 65534
        rst += self.sketch[2][0][temp // 100]
    else:
        rst += self.sketch[1][0][temp]
else:
    rst += self.sketch[0][0][hash_value]

if self.sketch2[0][0][hash_value2] == 255:
    rst2 += 254
    temp = hash_value2 // 100
    if self.sketch2[1][0][temp] == 65535:
        rst2 += 65534
        rst2 += self.sketch2[2][0][temp // 100]
    else:
        rst2 += self.sketch2[1][0][temp]
else:
    rst2 += self.sketch2[0][0][hash_value2]

flow_freq = min(rst, rst2)
return flow_freq
```

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)

```python
def count_unique_flows(self):
    """ Estimate the number of unique flows(Cardinality) using the sketch

    Returns:
        Cardinality of the packet trace
    """

    ###### Task II - Cardinality Estimation ######
    num_unique_flows = 0

    """ YOUR CODE HERE
    TODO: Implement the cardinality estimation algorithm
    """

    count1 = 0
    count2 = 0
    for i in range(len(self.sketch[0][0])):
        if self.sketch[0][0][i] == 0:
            count1 += 1
    for i in range(len(self.sketch2[0][0])):
        if self.sketch2[0][0][i] == 0:
            count2 += 1
    count = (count1 + count2) / 2
    # print(count)
    num_unique_flows = -450000 * np.log(count / 450000)
    return num_unique_flows
```

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.

```python
def find_heavy_hitters(self):
    """ Find the heavy hitters using the sketch

    Returns:
        heavy_hitters: 5-Tuples representing the heavy hitter flows
        heavy_hitters_size: Size of the heavy hitter flows
    """
    ###### Task III - Heavy Hitter Detection ######
    heavy_hitters = []  # List to store heavy hitter flows
    heavy_hitters_size = []  # List to store heavy hitter sizes

    """ YOUR CODE HERE
    TODO: Implement the heavy hitter detection algorithm
    """

    for key, value in self.auxiliary_storage.items():
        heavy_hitters.append(key)
        heavy_hitters_size.append(value)

    return heavy_hitters, heavy_hitters_size
```

Part2: implementation results

① Packet-trace.csv(3 times)

```
Loading Data.....
Initialising Sketch.....
Updating sketch with all packets in trace, this may take a while...
Processed 10.29% of packets
Processed 20.58% of packets
Processed 30.87% of packets
Processed 41.16% of packets
Processed 51.45% of packets
Processed 61.74% of packets
Processed 72.03% of packets
Processed 82.32% of packets
Processed 92.61% of packets
Sketch Updated with all packets


~~~~~~~Testing Functionalities~~~~~~~

Mean Error:  1.5793
estimate_frequency: WITHIN SET LIMITS
Error in Cardinality:  0.03520299866118475
count_unique_flows: WITHIN SET LIMITS
Number of Heavy Hitters:  98
find_heavy_hitters: WITHIN SET LIMITS
```

```
~~~~~~~ Sketch Memory Usage Report~~~~~~~
---- SUMMARY ------------------------------------------------------------------
Sketch Initialised                    active      0     B      average   pct
  count_min.CountMinSketch                 0    898.55 KB      0     B    0%
Sketch Updated with all packets       active      0     B      average   pct
  count_min.CountMinSketch                 1    937.54 KB    937.54 KB    0%
Query I                               active      0     B      average   pct
  count_min.CountMinSketch                 1    937.54 KB    937.54 KB    0%
Query II                              active      0     B      average   pct
  count_min.CountMinSketch                 1    937.54 KB    937.54 KB    0%
Query II                              active      0     B      average   pct
  count_min.CountMinSketch                 1    937.54 KB    937.54 KB    0%
End of Script                         active      0     B      average   pct
  count_min.CountMinSketch                 1    937.54 KB    937.54 KB    0%
-------------------------------------------------------------------------------


Elapsed time: 134.41923999786377 seconds
```

```
Loading Data.....
Initialising Sketch.....
Updating sketch with all packets in trace, this may take a while...
Processed 10.29% of packets
Processed 20.58% of packets
Processed 30.87% of packets
Processed 41.16% of packets
Processed 51.45% of packets
Processed 61.74% of packets
Processed 72.03% of packets
Processed 82.32% of packets
Processed 92.61% of packets
Sketch Updated with all packets


~~~~~~~Testing Functionalities~~~~~~~

Mean Error:  1.5254
estimate_frequency: WITHIN SET LIMITS
Error in Cardinality:  0.06394084602718393
count_unique_flows: WITHIN SET LIMITS
Number of Heavy Hitters:  99
find_heavy_hitters: WITHIN SET LIMITS
```

```
~~~~~~~ Sketch Memory Usage Report~~~~~~~
---- SUMMARY ------------------------------------------------------------
Sketch Initialised                      active      0     B      average   pct
  count_min.CountMinSketch                   0    898.55 KB      0     B    0%
Sketch Updated with all packets         active      0     B      average   pct
  count_min.CountMinSketch                   1    937.54 KB    937.54 KB    0%
Query I                                 active      0     B      average   pct
  count_min.CountMinSketch                   1    937.54 KB    937.54 KB    0%
Query II                                active      0     B      average   pct
  count_min.CountMinSketch                   1    937.54 KB    937.54 KB    0%
Query II                                active      0     B      average   pct
  count_min.CountMinSketch                   1    937.54 KB    937.54 KB    0%
End of Script                           active      0     B      average   pct
  count_min.CountMinSketch                   1    937.54 KB    937.54 KB    0%
-------------------------------------------------------------------------------


Elapsed time: 157.94587755203247 seconds
```

```
Loading Data.....
Initialising Sketch.....
Updating sketch with all packets in trace, this may take a while...
Processed 10.29% of packets
Processed 20.58% of packets
Processed 30.87% of packets
Processed 41.16% of packets
Processed 51.45% of packets
Processed 61.74% of packets
Processed 72.03% of packets
Processed 82.32% of packets
Processed 92.61% of packets
Sketch Updated with all packets


~~~~~~~Testing Functionalities~~~~~~~

Mean Error:  1.545
estimate_frequency: WITHIN SET LIMITS
Error in Cardinality:  0.0769933729789068
count_unique_flows: WITHIN SET LIMITS
Number of Heavy Hitters:  97
find_heavy_hitters: WITHIN SET LIMITS
```

```
~~~~~~~ Sketch Memory Usage Report~~~~~~~
---- SUMMARY ------------------------------------------------------------
Sketch Initialised                    active      0     B      average   pct
  count_min.CountMinSketch                 1    898.55 KB    898.55 KB    0%
Sketch Updated with all packets       active      0     B      average   pct
  count_min.CountMinSketch                 1    937.35 KB    937.35 KB    0%
Query I                               active      0     B      average   pct
  count_min.CountMinSketch                 1    937.35 KB    937.35 KB    0%
Query II                              active      0     B      average   pct
  count_min.CountMinSketch                 1    937.35 KB    937.35 KB    0%
Query II                              active      0     B      average   pct
  count_min.CountMinSketch                 1    937.35 KB    937.35 KB    0%
End of Script                         active      0     B      average   pct
  count_min.CountMinSketch                 1    937.35 KB    937.35 KB    0%
-------------------------------------------------------------------------------


Elapsed time: 139.00944900512695 seconds
```

② Packet-trace2.csv(3 times)

```
Loading Data.....
Initialising Sketch.....
Updating sketch with all packets in trace, this may take a while...
Processed 9.61% of packets
Processed 19.21% of packets
Processed 28.82% of packets
Processed 38.42% of packets
Processed 48.03% of packets
Processed 57.63% of packets
Processed 67.24% of packets
Processed 76.85% of packets
Processed 86.45% of packets
Processed 96.06% of packets
Sketch Updated with all packets


~~~~~~~Testing Functionalities~~~~~~~

Mean Error:  1.2486
estimate_frequency: WITHIN SET LIMITS
Error in Cardinality:  0.14309255227206186
count_unique_flows: WITHIN SET LIMITS
Number of Heavy Hitters:  96
find_heavy_hitters: WITHIN SET LIMITS
```

```
~~~~~~~ Sketch Memory Usage Report~~~~~~~
---- SUMMARY ------------------------------------------------------------------
Sketch Initialised                    active      0     B      average   pct
  count_min.CountMinSketch                 1    898.55 KB    898.55 KB    0%
Sketch Updated with all packets       active      0     B      average   pct
  count_min.CountMinSketch                 1    935.83 KB    935.83 KB    0%
Query I                               active      0     B      average   pct
  count_min.CountMinSketch                 1    935.83 KB    935.83 KB    0%
Query II                              active      0     B      average   pct
  count_min.CountMinSketch                 1    935.83 KB    935.83 KB    0%
Query II                              active      0     B      average   pct
  count_min.CountMinSketch                 1    935.83 KB    935.83 KB    0%
End of Script                         active      0     B      average   pct
  count_min.CountMinSketch                 1    935.83 KB    935.83 KB    0%
-------------------------------------------------------------------------------


Elapsed time: 248.90237140655518 seconds
```

```
Loading Data.....
Initialising Sketch.....
Updating sketch with all packets in trace, this may take a while...
Processed 9.61% of packets
Processed 19.21% of packets
Processed 28.82% of packets
Processed 38.42% of packets
Processed 48.03% of packets
Processed 57.63% of packets
Processed 67.24% of packets
Processed 76.85% of packets
Processed 86.45% of packets
Processed 96.06% of packets
Sketch Updated with all packets


~~~~~~~Testing Functionalities~~~~~~~

Mean Error:  1.3319
estimate_frequency: WITHIN SET LIMITS
Error in Cardinality:  0.006821424149540493
count_unique_flows: WITHIN SET LIMITS
Number of Heavy Hitters:  96
find_heavy_hitters: WITHIN SET LIMITS
```

```
~~~~~~~ Sketch Memory Usage Report~~~~~~~
---- SUMMARY ------------------------------------------------------------
Sketch Initialised                     active      0     B      average   pct
  count_min.CountMinSketch                  0    898.55 KB      0     B    0%
Sketch Updated with all packets        active      0     B      average   pct
  count_min.CountMinSketch                  1    935.83 KB    935.83 KB    0%
Query I                                active      0     B      average   pct
  count_min.CountMinSketch                  1    935.83 KB    935.83 KB    0%
Query II                               active      0     B      average   pct
  count_min.CountMinSketch                  1    935.83 KB    935.83 KB    0%
Query II                               active      0     B      average   pct
  count_min.CountMinSketch                  1    935.83 KB    935.83 KB    0%
End of Script                          active      0     B      average   pct
  count_min.CountMinSketch                  1    935.83 KB    935.83 KB    0%
-------------------------------------------------------------------------------


Elapsed time: 144.8097059726715 seconds
```

```
Loading Data.....
Initialising Sketch.....
Updating sketch with all packets in trace, this may take a while...
Processed 9.61% of packets
Processed 19.21% of packets
Processed 28.82% of packets
Processed 38.42% of packets
Processed 48.03% of packets
Processed 57.63% of packets
Processed 67.24% of packets
Processed 76.85% of packets
Processed 86.45% of packets
Processed 96.06% of packets
Sketch Updated with all packets


~~~~~~~Testing Functionalities~~~~~~~

Mean Error:  1.1978
estimate_frequency: WITHIN SET LIMITS
Error in Cardinality:  0.09611879132610718
count_unique_flows: WITHIN SET LIMITS
Number of Heavy Hitters:  96
find_heavy_hitters: WITHIN SET LIMITS
```

```
~~~~~~~ Sketch Memory Usage Report~~~~~~~
---- SUMMARY ------------------------------------------------------------------
Sketch Initialised                     active      0     B      average   pct
  count_min.CountMinSketch                  0    898.55 KB      0     B    0%
Sketch Updated with all packets        active      0     B      average   pct
  count_min.CountMinSketch                  1    935.83 KB    935.83 KB    0%
Query I                                active      0     B      average   pct
  count_min.CountMinSketch                  1    935.83 KB    935.83 KB    0%
Query II                               active      0     B      average   pct
  count_min.CountMinSketch                  1    935.83 KB    935.83 KB    0%
Query II                               active      0     B      average   pct
  count_min.CountMinSketch                  1    935.83 KB    935.83 KB    0%
End of Script                          active      0     B      average   pct
  count_min.CountMinSketch                  1    935.83 KB    935.83 KB    0%
-------------------------------------------------------------------------------


Elapsed time: 144.15944361686707 seconds
```

③ Packet-trace3.csv

```
PS C:\Users\weiyi\Desktop\assignments\CS5229 Advanced Computer Networks\programmin
e-3.csv
Loading Data.....
Initialising Sketch.....
Updating sketch with all packets in trace, this may take a while...
Processed 48.17% of packets
Processed 96.35% of packets
Sketch Updated with all packets


~~~~~~~Testing Functionalities~~~~~~~

Mean Error:  0.1116
estimate_frequency: WITHIN SET LIMITS
Error in Cardinality:  0.09921312957519267
count_unique_flows: WITHIN SET LIMITS
Number of Heavy Hitters:  99
find_heavy_hitters: WITHIN SET LIMITS


~~~~~~~ Sketch Memory Usage Report~~~~~~~
---- SUMMARY ------------------------------------------------------------------
Sketch Initialised                      active      0     B      average   pct
  count_min.CountMinSketch                   0    898.55 KB      0     B    0%
Sketch Updated with all packets         active      0     B      average   pct
  count_min.CountMinSketch                   1    934.87 KB    934.87 KB    0%
Query I                                 active      0     B      average   pct
  count_min.CountMinSketch                   1    934.87 KB    934.87 KB    0%
Query II                                active      0     B      average   pct
  count_min.CountMinSketch                   1    934.87 KB    934.87 KB    0%
Query II                                active      0     B      average   pct
  count_min.CountMinSketch                   1    934.87 KB    934.87 KB    0%
End of Script                           active      0     B      average   pct
  count_min.CountMinSketch                   1    934.87 KB    934.87 KB    0%
-------------------------------------------------------------------------------


Elapsed time: 27.79170799255371 seconds
```

```
PS C:\Users\weiyi\Desktop\assignments\CS5229 Advanced Computer Networks\programmi
e-3.csv
Loading Data.....
Initialising Sketch.....
Updating sketch with all packets in trace, this may take a while...
Processed 48.17% of packets
Processed 96.35% of packets
Sketch Updated with all packets


~~~~~~~Testing Functionalities~~~~~~~

Mean Error:  0.1163
estimate_frequency: WITHIN SET LIMITS
Error in Cardinality:  0.0965606434938651
count_unique_flows: WITHIN SET LIMITS
Number of Heavy Hitters:  98
find_heavy_hitters: WITHIN SET LIMITS


~~~~~~~ Sketch Memory Usage Report~~~~~~~
---- SUMMARY ------------------------------------------------------------------
Sketch Initialised                      active      0     B      average   pct
  count_min.CountMinSketch                   0    898.55 KB      0     B    0%
Sketch Updated with all packets         active      0     B      average   pct
  count_min.CountMinSketch                   1    934.93 KB    934.93 KB    0%
Query I                                 active      0     B      average   pct
  count_min.CountMinSketch                   1    934.93 KB    934.93 KB    0%
Query II                                active      0     B      average   pct
  count_min.CountMinSketch                   1    934.93 KB    934.93 KB    0%
Query II                                active      0     B      average   pct
  count_min.CountMinSketch                   1    934.93 KB    934.93 KB    0%
End of Script                           active      0     B      average   pct
  count_min.CountMinSketch                   1    934.93 KB    934.93 KB    0%
-------------------------------------------------------------------------------


Elapsed time: 34.68801689147949 seconds
```

```
PS C:\Users\weiyi\Desktop\assignments\CS5229 Advanced Computer Networks\programmi
e-3.csv
Loading Data.....
Initialising Sketch.....
Updating sketch with all packets in trace, this may take a while...
Processed 48.17% of packets
Processed 96.35% of packets
Sketch Updated with all packets


~~~~~~~Testing Functionalities~~~~~~~

Mean Error:  0.1123
estimate_frequency: WITHIN SET LIMITS
Error in Cardinality:  0.11402587011356181
count_unique_flows: WITHIN SET LIMITS
Number of Heavy Hitters:  99
find_heavy_hitters: WITHIN SET LIMITS


~~~~~~~ Sketch Memory Usage Report~~~~~~~
---- SUMMARY ------------------------------------------------------------------
Sketch Initialised                     active      0     B      average   pct
  count_min.CountMinSketch                  0    898.55 KB      0     B    0%
Sketch Updated with all packets        active      0     B      average   pct
  count_min.CountMinSketch                  1    934.87 KB    934.87 KB    0%
Query I                                active      0     B      average   pct
  count_min.CountMinSketch                  1    934.87 KB    934.87 KB    0%
Query II                               active      0     B      average   pct
  count_min.CountMinSketch                  1    934.87 KB    934.87 KB    0%
Query II                               active      0     B      average   pct
  count_min.CountMinSketch                  1    934.87 KB    934.87 KB    0%
End of Script                          active      0     B      average   pct
  count_min.CountMinSketch                  1    934.87 KB    934.87 KB    0%
-------------------------------------------------------------------------------


Elapsed time: 27.915160417556763 seconds
```