Advanced topics in TCS

Exercise sheet 3.

CountSketch, Count-Min Sketch, ℓ_0-sampling

Raphaël Clifford

Question 1. CountSketch

Implement the CountSketch algorithm. You will have to choose a method for creating the hash functions needed. In countsketch.py I have shown how g() can be made using MD5. You can similarly make the h() function using SHA256.

```
def g(val, idx):
   if idx > 127:
      print("Run out of bits in g() function")
      quit()
   bits = bin(int.from_bytes(hashlib.md5(val.encode()).digest(),
                                        "little"))[2:].zfill(128)
   return int(bits[idx])*2-1 # Map to {-1, 1}
```

You may prefer to use pairwise independent hash functions instead in which case you will need to store the different (a, b) pairs you create. Each approach has its own advantages and disadvantages would need to be compared experimentally.

The provided code has a function createturnstilesequence (length). This will create an array of pairs (c, ℓ) where c is a positive or negative count and ℓ is a printable character. It is designed so that the counts will broadly speaking follow a Zipf distribution. In other words, some will occur much more frequently than others.

Use your implementation of CountSketch to find out which letters occur most frequently.

Question 2. 1-sparse recovery

Suppose we modified the 1-sparse recovery algorithm to declare $\boldsymbol{f} = \mathbf{0}$ whenever $\ell = z = 0$ without using the value of p. Would this still be correct? Why or why not?

Question 3. s-sparse recovery

1. Consider running the s-sparse recovery algorithm on a stream with more than s items with non-zero frequency. What will the output be?

2. How can the algorithm be modified to detect if the stream has more than s items with non-zero frequency?

Question 4. Counting triangles

Consider an undirected graph with n vertices and t triangles where edges are arriving in a stream. A triangle is a set of 3 vertices such that any two of them are connected by an edge of the graph. We would like to count approximately the number of triangles in the graph. Here is a simple and not very accurate method.

- Randomly pick (uniformly with replacement) k subsets $S_1, \ldots, S_k$ of the vertices each of size 3.

- Let x_S be the number of edges seen between the vertices in set S.

- Let C be the number of indices i for which $x_{S_i} = 3$. That is the number of triangles found.

- Our estimate is $R = \frac{\binom{n}{3}}{k} C$.

Answer the following questions about this triangle counting method:

1. Is R an unbiased estimate of t? Give a proof.

2. Show that $var(R) \in O\left(\frac{tn^3}{k}\right)$.

3. Give an upper bound for the probability that $|R-t| \geq c\sqrt{\frac{tn^3}{k}}$ for $c \geq 1$.