Algorithm Design: 5th Hands On

M. Baglioni, L. Crociani, A. F. Montagnoli, G. Trapani

22 Apr 2024

1 Exercise 1

Consider the counters $F[i]$ *for* $1 \leq i \leq n$*, where* n *is the number of items in the stream of any length. At any time, we know that* $||F||$ *is the total number of items (with repetitions) seen so far, where each* $F[i]$ *contains how many times item* i *has been so far. We saw that CM-sketches provide a FPTAS* $F'[i]$ *such that* $F[i] \leq F'[i] \leq F[i] + \epsilon||F||$*, where the latter inequality holds with probability at least* $1 - \delta$*.*

Consider now a range query (a, b)*, where we want* $F_{ab} = \sum_{a \leq i \leq b} F[i]$*. Show how to adapt CM-sketch so that a FPTAS* F'_{ab} *is provided:*

- *Baseline is* $\sum_{a \leq i \leq b} F'[i]$*, but this has drawbacks as both time and error grows with* $b - a + 1$*.*
- *Consider how to maintain counters for just the sums when* $b - a + 1$ *is any power of 2 (less or equal to* n*):*
  - *Can we now answer quickly also when* $b - a + 1$ *is not a power of two?*
  - *Can we reduce the number of these power-of-2 intervals from nlogn to* $2n$*?*
  - *Can we bound the error with a certain probability? Suggestion: it does not suffices to say that it is at most* δ *the probability of error of each individual counter; while each counter is still the actual wanted value plus the residual as before, it is better to consider the sum* V *of these wanted values and the sum* X *of these residuals, and apply Markov's inequality to* V *and* X *rather than on the individual counters.*

1.1 Solution

We shall assume w.l.o.g. $|F| = n$ is a power of 2 and build a vector of $\log_2 n$ CM-sketches. Now we explain how to implement the Update and the Range-Query operations.

Update

We assume we have an operator `sketch_update(cm, h, x)` which applies the Update operation as defined in class on the CM-sketch `cm` with the hash function `h(x)`.

We can now give the pseudocode for the Update operation on our sketches, which takes as input the CM-sketches `cms`, the r hash functions `hs` and the value `x` we get from the stream.

```
def Update(cms, hs, x):
    for i in [0; log_2(n) - 1]:
        for j in [0; r - 1]:
            sketch_update(
                cms[i], // We update the i-th sketch
                hs[j], // We apply the j-th hash function
                floor(x / pow(2, i))
            )
```

RangeQuery(a, b):

We can reduce every range-query to $2 \log n$ point-queries over the CM-sketches defined as above.

Therefore we can partition RangeQuery(a, b) in $2 \log n$ point-queries and compute the sum of the results.

Complexity: $O(\log n \times r) = O\left(\log n \ \times \log \frac{1}{\delta}\right)$

1.1.1 Error Analysis

We know with probability $\geq 1 - \delta$ when we compute query(i), we get the following estimation

$$\tilde{F}[i] \leq F[i] + \epsilon ||F||$$

We also know that

$$\tilde{F}[i] = F[i] + X_{ji}$$

with X_{ji} r.i.i.d which measures the error for the i-th element in row j.

We know the expected value of X_{ji} is

$$E[X_{ji}] = \frac{\epsilon}{e} ||F||$$

Therefore the expected value for the additive error of RangeQuery(a, b) which we compute calling the query method $2 \log n$ times is

$$2 \log n \frac{\epsilon}{e} ||F||$$

Now, let Y_{ji} be the error for the RangeQuery method.

Using Markov's inequality we can say that

$$\Pr\left[Y_{ji} > 2\log n\epsilon||F||\right] \leq \frac{E[Y_{ji}]}{2\log n\epsilon||F||} = \frac{2\log n\frac{\epsilon}{e}||F||}{e \times 2\log n \times E[X_{ji}]} = \frac{2\log nE[X_{ji}]}{2e\log nE[X_{ji}]} = \frac{1}{e}$$

Since in every CM-sketch we have $r = \ln\delta^{-1}$ rows:

$$\prod_{j\in[r]} \Pr\left[Y_{ji} > 2\log n\epsilon||F||\right] < \left(\frac{1}{e}\right)^r = \delta$$

Therefore we demonstrated that:

$$\text{RangeQuery}(F, a, b) \leq \text{RangeQuery}(\tilde{F}, a, b)$$

and that with probability $1 - \delta$ we have:

$$\text{RangeQuery}(\tilde{F}, a, b) \leq \text{RangeQuery}(F, a, b) + \epsilon \times 2\log n||F||$$

□

We observe that in order to estimate with correctness up to $\epsilon'||F||$ with probability $1 - \delta$ it is sufficient to choose a value $\epsilon = \dfrac{\epsilon'}{2\log n}$.

2 Exercise 2 (Bonus)

Show (and prove correctness) that there is a deterministic streaming algorithm that works in O(1) space and finds the most frequent item if the latter appears strictly more than half of the times in the stream.

2.1 Solution

The address the problem we have to iterate O(n) times on the stream. Follows a pseudo-coded algorithm:

```
def find_majority(stream):
    maj = None
    counter = 0
    while stream.has_next():
        elem = stream.next()
        if counter == 0 or maj is None or elem == maj:
            maj = elem
            counter += 1
        else:
            counter -= 1
    return maj
```

2.1.1 Correctness

We shall prove the correctness of this algorithm via induction.

Inductive hypothesis: the algorithm returns the majority element up to length $< l$ if it exists.

Base Case: the length of the stream is 1, there is only one element in the stream, and it is returned correctly.

Inductive step: We distinguish two cases:

- In this case the element we are considering is not the majority element. The counter becomes 0 at some point. As it becomes 0, the algorithm "resets" and starts from scratch on a length $< l$.

- In this case the element we are considering was the majority element. We can have three different subcases:

  - The counter increases. The majority element is the element we are considering now, and it is returned.
  - The counter decreases but it is still greater than 0. The majority element is the element we are considering now, and it is returned.
  - The counter decreases but it becomes 0. This is the first case we discussed: the element we are considering now is not the majority element.