CSCE 689: Special Topics in Modern Algorithms for Data Science Fall 2023

Lecture 9 — September 11, 2023

Prof. Samson Zhou *Scribe: Bokun Wang*

1 The Streaming Model

Consider the following problem: we are given a massive dataset that arrives in a continuous stream. However, we do not have enough space to store all the items to store the whole data.

For example, telescopes may collect more than 1,000 terabytes the images every 5 days! Other application scenarios include the IoTs, financial markets, and traffic network monitoring.

The input, output, and goal of the streaming model can be summarized as follows:

• Input: Elements of an underlying data set S that arrive sequentially.

• Output: Evaluation or approximation of a given function.

• Goal: Sublinear space in the size $m = |S|$.

The core question is: How much space is needed to answer queries of interest? Next, we give several examples of problems in the streaming model.

2 Reservoir Sampling

Given a set of m items $S = \{x_i\}_{i=1}^m$, how can we sample one element from the set uniformly at random? This is easy if we have those n items in hand. However, m may be too large to fit into the memory and those n items could be revealed to us one by one over time, i.e., item x_i arrives at the time step i, $i = 1, 2, \ldots, n$.

To this end, [Vit85] proposed the reservoir sampling algorithm: (1) Initialize a memory slot $s = \perp$; (2) On the arrival of the i-th element of the stream x_i, replace s with x_i with probability $\frac{1}{i}$, for $i = 1, 2, \ldots, m$.

Proposition 1. Reservoir sampling samples one element from S uniformly at random, i.e., $\mathbb{P}[s = x_t] = \frac{1}{m}$ for any $t \in [m]$.

Proof. We use the random variable s_i to represent the status of the memory s at time step i, $i \in [n]$. Note that the event $\{s_m = x_t\}$ is equivalent to $\bigcap_{i=t}^m \{s_i = x_t\}$, i.e., x_t is selected at tine t and not kicked out in the rest of the stream. By independence of the sampling process at each time t and a telescoping product, we have

$$\begin{aligned}\mathbb{P}[\{s_m = x_t\}] &= \mathbb{P}\left[\bigcap_{i=t}^{m}\{s_i = x_t\}\right] \\ &= \mathbb{P}\left[\{s_t = x_t\}\right]\mathbb{P}\left[\{s_{t+1} = x_t\} \mid \{s_t = x_t\}\right]\ldots\mathbb{P}\left[\{s_m = x_t\} \mid \bigcap_{i=t}^{m-1}\{s_i = x_t\}\right] \\ &= \frac{1}{t}\left(1-\frac{1}{t+1}\right)\cdots\left(1-\frac{1}{m}\right) = \frac{1}{m}.\end{aligned}$$

■

3 Frequent Items

Given a set S of m elements from $[n]$, let f_i be the frequency of element i. Define the frequency vector $f = (f_1, f_2, \ldots, f_n)$. The goal is to find the k elements with the largest frequency. Applications of this problem include finding top products/viral objects and identifying IP addresses with high volume traffic.

A natural approach is to store the count for each item and return the k elements with the largest frequency. However, this requires $O(n)$ space.

Now we look at a special case: Find a unique item from $[n]$ that has frequency more than $\frac{m}{2}$, if such an item exists. We consider the following algorithm:

- Initialize item $V = 1$ with count $c = 0$
- For updates $1, \ldots, m$
  - If $c = 0$, set $V = x_i$ and $c = 1$
  - Else if $V = x_i$, increment counter c by 1 $c = c + 1$
  - Else if $V \neq x_i$, decrement counter c by 1 $c = c - 1$

Let M be the true majority element and z be a helper variable such that $z = \begin{cases} 1 & \text{if } x_i = M \\ -1 & \text{if } x_i \neq M \end{cases}$. Then z is positive at the end of the stream. This algorithms only needs $O(\log m + \log n)$ bits of space if we store V and c in binary format. Next class, we will extend the algorithm to general $k \geq 2$ and show the "correctness' of the algorithm.

References

[Vit85] Jeffrey Scott Vitter. Random sampling with a reservoir. *ACM Trans. Math. Softw.*, 11(1):37–57, 1985.