CS 598CSC: Algorithms for Big Data	**Lecture date: August 26, 2014**
Instructor: Chandra Chekuri	**Scribe: Chandra Chekuri**

1 Introduction/Administrivia

- Course website: `https://courses.engr.illinois.edu/cs598csc/fa2014/`.
- There is no specific book that we will follow. See website for pointers to several resources.
- Grading based on 4-5 homeworks, scribing a lecture and course project. Details to be figured out.

Course Objectives

Big Data is all the rage today. The goal of this course is learn about some of the basic algorithmic and analysis techniques that have been useful in this area. Some of the techniques are old and many have been developed over the last fifteen years or so. A few topics that we hope to cover are:

- Streaming, Sketching and Sampling
- Dimensionality Reduction
- Streaming for Graphs
- Numerical Linear Algebra
- Compressed Sensing
- Map-Reduce model and some basic algorithms
- Property Testing
- Lower Bounds via Communication Complexity

There is too much material in the above topics. The plan is to touch upon the basics so that it will provide an impetus to explore further.

2 Streaming/One-Pass Model of Computation

In the streaming model we assume that the input data comes as a stream. More formally, the data is assumed to consist of m items/tokens/objects $a_1, a_2, \ldots, a_m$. A simple case is when each token is a number in the range $[1..n]$. Another example is when each token represents an edge in a graph given as (u, v) where u and and v are integers representing the node indices. The tokes arrive one by one in order. The algorithm has to process token x_i before it sees the next token. If we are allowed to store all the tokens then we are in the standard model of computing where we have access to the whole input. However, we will assume that the space available to the algorithm is much less than m tokens; typically sub-linear in m or in the ideal scenario polylog(m). Our goal is to (approximately) compute/estimate various quantities of interest using such limited space. Surprisingly one can compute various useful functions of interest. Some parameters of interest for a streaming algorithm are:

- space used by the algorithm as a function of the stream length m and the nature of the tokens
- the worst-case time to process each token
- the total time to process all the tokens (equivalently the amortized time to process a token)
- the accuracy of the output
- the probability of success in terms of obtaining a desired approximation (assuming that the algorithm is randomized)

There are several application areas that motivate the streaming model. In networking, a large number of packets transit a switch and we may want to analyze some statistics about the packets. The number of packets transiting the switch is vastly more than what can be stored for offline processing. In large databases the data is stored in disks and random access is not feasible; the size of the main memory is much smaller than the storage available on the disk. A one-pass or multi-pass streaming algorithm allows one to avoid sophisticated data structures on the disk. There are also applications where the data is distributed in several places and we need to process them separately and combine the results with low communication overhead.

In database applications and such it also makes sense to discuss the multi-pass model or the semi-streaming model where one gets to make several passes over the data. Typically we will assume that the number of passes is a small constant.

3 Background on Probability and Inequalities

The course will rely heavily on proabablistic methods. We will mostly rely on discrete probability spaces. We will keep the discussion high-level where possible and use certain results in a black-box fashion.

Let Ω be a finite set. A probability measure p assings a non-negative number $p(\omega)$ for each $\omega \in \Omega$ such that $\sum_{\omega \in \Omega} p(\omega) = 1$. The tuple (Ω, p) defines a discrete probability space; an event in this space is any subset $A \subseteq \Omega$ and the probability of an event is simply $p(A) = \sum_{\omega \in A} p(\omega)$. When Ω is a continuous space such as the interval $[0,1]$ things get trickier and we need to talk about a measure spaces σ-algebras over Ω; we can only assign probability to certain subsets of Ω. We will not go into details since we will not need any formal machinery for what we do in this course.

An important definition is that of a *random variable*. We will focus only on real-valued random variables in this course. A random variable X in a probability space is a function $X : \Omega \to \mathbb{R}$. In the discrete setting the *expectation* of X, denoted by $\mathbf{E}[X]$, is defined as $\sum_{\omega \in \Omega} p(w) X(\omega)$. For continuous spaces $\mathbf{E}[X] = \int X(\omega) dp(\omega)$ with appropriate definition of the integral. The variance of X, denoted by $\mathbf{Var}[X]$ or as σ_X^2, is defined as $\mathbf{E}[(X - \mathbf{E}[X])^2]$. The standard deviation is σ_X, the square root of the variance.

Theorem 1 (Markov's Inequality) *Let X be a non-negative random variable such that $\mathbf{E}[X]$ is finite. Then for any $t > 0$, $\Pr[X \geq t] \leq \mathbf{E}[X]/t$.*

Proof: The proof is in some sense obvious, especially in the discrete case. Here is a sketch. Define a new random variable Y where $Y(\omega) = X(\omega)$ if $X(\omega) < t$ and $Y(\omega) = t$ if $X(\omega) \geq t$. Y is non-negative and $Y \leq X$ point-wise and hence $\mathbf{E}[Y] \leq \mathbf{E}[X]$. We also see that:

$$\begin{aligned}
\mathbf{E}[X] \geq \mathbf{E}[Y] &= \sum_{\omega : X(\omega) < t} X(\omega) p(\omega) + \sum_{\omega : X(\omega) \geq t} t p(\omega) \\
&\geq t \sum_{\omega : X(\omega) \geq t} p(\omega) \qquad \text{(since } X \text{ is non-negative)} \\
&\geq t \Pr[X \geq t].
\end{aligned}$$

The continuous case follows by replacing sums by integrals. □

Markov's inequality is tight under the assumption. Assume you can construct an example. The more information we have about a random variable the better we can bound its deviation from the expectation.

Theorem 2 (Chebyshev's Inequality) *Let X be a random variable with $\mathbf{E}[X]$ and $\mathbf{Var}[X]$ finite. Then $\Pr[|X| \geq t] \leq \mathbf{E}[X^2]/t^2$ and $\Pr[|X - \mathbf{E}[X]| \geq t\sigma_X] \leq 1/t^2$.*

Proof: Consider the non-negative random variable $Y = X^2$. $\Pr[|X| \geq t] = \Pr[Y \geq t^2]$ and we apply Markov's inequality to the latter. The second inequality is similar by considering $Y = (X - \mathbf{E}[X])^2$. □

Chernoff-Hoeffding Bounds: We will use several times various forms of the Chernoff-Hoeffding bounds that apply to a random variable that is a a finite sum of bounded and *independent* random variables. There are several versions of these bounds. First we state a general bound that is applicable to non-negative random variables and is dimension-free in that it depends only the expectation rather than the number of variables.

Theorem 3 (Chernoff-Hoeffding) *Let $X_1, X_2, \ldots, X_n$ be independent binary random variables and let $a_1, a_2, \ldots, a_n$ be coefficients in $[0,1]$. Let $X = \sum_i a_i X_i$. Then*

- *For any $\mu \geq \mathbf{E}[X]$ and any $\delta > 0$, $\Pr[X > (1+\delta)\mu] \leq \left(\frac{e^\delta}{(1+\delta)^{(1+\delta)}}\right)^\mu$.*
- *For any $\mu \leq \mathbf{E}[X]$ and any $\delta > 0$, $\Pr[X < (1-\delta)\mu] \leq e^{-\mu\delta^2/2}$.*

The following corollary bounds the deviation from the mean in both directions.

Corollary 4 *Under the conditions of Theorem 3, the following hold:*

- *If $\delta > 2e - 1$, $\Pr[X \geq (1+\delta)\mu] \leq 2^{-(1+\delta)\mu}$.*
- *For any U there is a constant $c(U)$ such that for $0 < \delta < U$, $\Pr[X \geq (1+\delta)\mu] \leq e^{-c(U)\delta^2\mu}$. In particular, combining with the lower tail bound,*

$$\Pr[|X - \mu| \geq \delta\mu] \leq 2e^{-c(U)t^2\mu}.$$

We refer the reader to the standard books on randomized algorithms [6] and [4] for the derivation of the above bounds.

If we are interested only in the upper tail we also have the following bounds which show the dependence of μ on n to obtain an inverse polynomial probability.

Corollary 5 *Under the conditions of Theorem 3, there is a universal constant α such that for any $\mu \geq \max\{1, \mathbf{E}[X]\}$, and sufficiently large n and for $c \geq 1$, $\Pr[X > \frac{\alpha c \ln n}{\ln \ln n} \cdot \mu] \leq 1/n^c$. Similarly, there is a constant α such that for any $\epsilon > 0$, $\Pr[X \geq (1+\epsilon)\mu + \alpha c \log n/\epsilon] \leq 1/n^c$.*

Remark 6 *If the X_i are in the range $[0, b]$ for some b not equal to 1 one can scale them appropriately and then use the standard bounds.*

Some times we need to deal with random variables that are in the range $[-1, 1]$. Consider the setting where $X = \sum_i X_i$ where for each i, $X_i \in [-1, 1]$ and $\mathbf{E}[X_i] = 0$, and the X_i are independent. In this case $\mathbf{E}[X] = 0$ and we can no longer expect a dimension-free bound. Suppose each X_i is 1 with probability 1/2 and -1 with probability 1/2. Then $X = \sum_i X_i$ corresponds to a 1-dimensional random walk and even though the expected value is 0 the standard deviation of X is $\Theta(\sqrt{n})$. One can show that $\Pr[|X| \geq t\sqrt{n}] \leq 2e^{-t^2/2}$. For these settings we can use the following bounds.

Theorem 7 *Let $X_1, X_2, \ldots, X_n$ be independent random variables such that for each i, $X_i \in [a_i, b_i]$. Let $X = \sum_i a_i X_i$ and let $\mu = \mathbf{E}[X]$. Then*

$$\Pr[|X - \mu| \geq t] \leq 2e^{-\frac{2t^2}{\sum_{i=1}^n (b_i - a_i)^2}}.$$

In particular if $b_i - a_i \leq 1$ for all i then

$$\Pr[|X - \mu| \geq t] \leq 2e^{-\frac{2t^2}{n}}.$$

Note that $\mathbf{Var}[X] = \sum_i \mathbf{Var}[X_i]$. One can show a bound based of the following form

$$\Pr[|X - \mu| \geq t] \leq 2e^{-\frac{t^2}{2(\sigma_X^2 + Mt/3)}}$$

where $|X_i| \leq M$ for all i.

Remark 8 *Compare the Chebyshev bound to the Chernoff-Hoeffding bounds for the same variance.*

Statistical Estimators, Reducing Variance and Boosting: In streaming we will mainly work with randomized algorithms that compute a function f of the data stream $x_1, \ldots, x_m$. They typically work by producing an unbiased estimator, via a random variable X, for the the function value. That is, the algorithm will have the property that the $\mathbf{E}[X]$ is the desired value. Note that the randomness is internal to the algorithm and not part of the input (we will also later discuss randomness in the input when considering random order streams). Having an estimator is not often useful. We will also typically try to evaluate $\mathbf{Var}[X]$ and then we can use Chebyshev's inequality. One way to reduce the variance of the estimate is to run the algorithm in parallel (with separate random bits) and get estimators $X_1, X_2, \ldots, X_h$ and use $X = \frac{1}{h}\sum_i X_i$ as the final estimator. Note that $\mathbf{Var}(X) = \frac{1}{h}\sum_i \mathbf{Var}(X_i)$ since the X_i are independent. Thus the variance has been reduced by a factor of h. A different approach is to use the *median* value of $X_1, X_2, \ldots, X_h$ as the final estimator. We can then use Chernoff-Hoeffding bounds to get a much better dependence on h. In fact both approaches can be combined and we illustrate it via a concrete example in the next section.

4 Probabilistic Counting and Morris's Algorithm

Suppose we have a long sequence of events that we wish to count. If we wish to count a sequent of events of length upto some N we can easily do this by using a counter with $\lceil \log_2 N \rceil$ bits. In some settings of interest we would like to further reduce this. It is not hard to argue that if one wants an exact and deterministic count then we do need a counter with $\lceil \log_2 N \rceil$ bits. Surprisingly, if we allow for approximation and randomization, one can count with about $\log logN$ bits. This was first shown in a short and sweet paper of Morris [5]; it is a nice paper to read the historical motivation.

Morris's algorithm keeps a counter that basically keeps an estimate of $\log N$ where N is the number of events and this requires about $\log logN$ bits. There are several variants of this, here we will discuss the simple one and point the reader to [5, 3, 1] for other schemes and a more detailed and careful analysis.

```
ProbabilisticCounting:
$X \leftarrow 0$
While (a new event arrives)
  Toss a biased coin that is heads with probability $1/2^X$
  If (coin turns up heads)
      $X \leftarrow X + 1$
endWhile
Output $2^X - 1$ as the estimate for the length of the stream.
```

For integer $i \geq 0$, let X_n be the random variable that denotes the value of the counter after i events. Let $Y_n = 2^{X_n}$. The lemma below shows that the output of the algorithm is an unbiased estimator of the count that we desire.

Lemma 9 $\mathbf{E}[Y_n] = n + 1$.

Proof: Proof by induction on n. The case of $n = 0, 1$ is easy to verify since in both cases we have that Y_n is deterministically equal to $n + 1$. We have for $n \geq 2$:

$$\begin{aligned}
\mathbf{E}[Y_n] = \mathbf{E}[2^{X_n}] &= \sum_{j=0}^{\infty} 2^j \Pr[X_n = j] \\
&= \sum_{j=0}^{\infty} 2^j \left(\Pr[X_{n-1} = j] \cdot (1 - \frac{1}{2^j}) + \Pr[X_{n-1} = j-1] \cdot \frac{1}{2^{j-1}} \right) \\
&= \sum_{j=0}^{\infty} 2^j \Pr[X_{n-1} = j] + \sum_{j=0}^{\infty} (2 \Pr[X_{n-1} = j-1] - \Pr[X_{n-1} = j]) \\
&= \mathbf{E}[Y_{n-1}] + 1 \\
&= n + 1
\end{aligned}$$

where we used induction to obtain the final equality. □

Since $\mathbf{E}[Y_n] = n + 1$ we have $\mathbf{E}[X_n] = \log_2(n + 1)$ which implies that the expected number of bits in the counter after n events is $\log \log n + O(1)$ bits. We should also calculate the variance of Y_n.

Lemma 10 $\mathbf{E}[Y_n^2] = \frac{3}{2}n^2 + \frac{3}{2}n + 1$ *and* $\mathbf{Var}[Y_n] = n(n-1)/2$.

Proof: Proof is by induction on n. Easy to verify base cases $n = 0, 1$ since $Y_n = n + 1$ deterministically. For $n \geq 2$:

$$\begin{aligned}
\mathbf{E}[Y_n^2] = \mathbf{E}[2^{2X_n}] &= \sum_{j \geq 0} 2^{2j} \cdot \Pr[X_n = j] \\
&= \sum_{j \geq 0} 2^{2j} \cdot \left(\Pr[X_{n-1} = j](1 - \frac{1}{2^j}) + \Pr[X_{n-1} = j-1] \frac{1}{2^{j-1}} \right) \\
&= \sum_{j \geq 0} 2^{2j} \cdot \Pr[X_{n-1} = j] + \sum_{j \geq 0} \left(-2^j \Pr[X_{n-1} = j-1] + 4 2^{j-1} \Pr[X_{n-1} = j-1] \right) \\
&= \mathbf{E}[Y_{n-1}^2] + 3\mathbf{E}[Y_{n-1}] \\
&= \frac{3}{2}(n-1)^2 + \frac{3}{2}(n-1) + 1 + 3n \\
&= \frac{3}{2}n^2 + \frac{3}{2}n + 1.
\end{aligned}$$

We used induction and the value of $\mathbf{E}[Y_n]$ that we previously computed. $\mathbf{Var}[Y_n] = \mathbf{E}[Y_n^2] - (\mathbf{E}[Y_n])^2$ and it can be verified that it is equal to $n(n-1)/2$. □

4.1 Approximation and Success Probability

The analysis of the expectation shows that the output of the algorithm is an unbiased estimator. The analysis of the variance shows that the estimator is fairly reasonable. However, we would like to have a finer understanding. For instance, given some parameter $\epsilon > 0$, what is $\Pr[|Y_n - (n+1)| > \epsilon n]$? We could also ask a related question. Is there an algorithm that given ϵ, δ guarantees that the output Y will satisfy the property that $\Pr[|Y - n| > \epsilon n] \leq \delta$ while still ensuring that the counter size is $O(\log \log n)$; of course we would expect that the constant in the $O()$ notation will depend on ϵ, δ.

The algorithm can be modified to obtain a $(1 + \epsilon)$-approximation with constant probability using a related scheme where the probability of incrementing the counter is $\frac{1}{a^X}$ for some parameter a; see [5, **?**]. The expected number of bits in the counter to achieve a $(1 + \epsilon)$-approximation can be shown to be $\log \log n + O(\log 1/\epsilon)$ bits.

Here we describe two general purpose ways to obtain better approximations by using independent estimators. Suppose we run the basic algorithm h times in parallel with independent randomness to get estimators $Y_{n,1}, Y_{n,2}, \ldots, Y_{n,h}$. We then output $Z = \frac{1}{h} \sum_{i=1}^{h} Y_{n,i} - 1$ as our estimate for n. Note that Z is also an unbiased estimator. We have that $\mathbf{Var}[Z] = \frac{1}{h}\mathbf{Var}[Y_n]$.

Claim 11 *Suppose* $h = 2/\epsilon^2$ *then* $\Pr[|Z - n| \geq \epsilon n] < \frac{1}{4}$.

Proof: We apply Chebyshev's inequality to obtain that

$$\Pr[|Z - n| \geq \epsilon n] \leq \frac{\mathbf{Var}[Z]}{\epsilon^2 n^2} \leq \frac{1}{h} \frac{\mathbf{Var}[Y_n]}{\epsilon^2 n^2} \leq \frac{1}{h} \frac{n(n-1)}{2\epsilon^2 n^2} < \frac{1}{4}.$$

□

Now suppose we want a high probability guarantee regarding the approximation. That is, we would like the estimator to be a $(1 + \epsilon)$-approximation with probability at least $(1 - \delta)$ for some given parameter δ.

We choose $\ell = c \log \frac{1}{\delta}$ for some sufficiently large constant c. We independently and in parallel obtain estimators $Z_1, Z_2, \ldots, Z_\ell$ and output the *median* of the estimators; lets call Z' the random variable corresponding to the median. We will prove the following.

Claim 12 $\Pr[|Z' - n| \geq \epsilon n] \leq (1 - \delta)$.

Proof: Define an indicator random variable A_i, $1 \leq i \leq \ell$ where A_i is 1 if $|Z_i - n| \geq \epsilon n$. From Claim 11, $\Pr[A_i = 1] < 1/4$. Let $A = \sum_{i=1}^{\ell} A_i$ and hence $\mathbf{E}[A] < \ell/4$. We also observe that $|Z' - n| \geq \epsilon n$ only if $A \geq \ell/2$, that is, if more than half of the Z_i's deviate by more than ϵn. We can apply Chernoff-Hoeffding bound to upper bound $\Pr[A \geq \ell/2] = \Pr[A \geq (1 + \delta)\mu]$ where $\delta = 1$ and $\mu = \ell/4 \geq \mathbf{E}[A]$. From Theorem 3 this is at most $(e/4)^{\ell/4}$. Since $\ell = c \log \frac{1}{\delta}$, the probability is at most δ for an appropriate choice of c. □

The total space usage for running the estimates in parallel is $O(\frac{1}{\epsilon^2} \cdot \log \frac{1}{\delta} \cdot \log \log n)$.

5 Reservoir Sampling

Random sampling is a powerful general purpose technique in a variety of areas. The goal is to pick a small subset S of a set of items N such that S is *representative* of N and often a random sample works well. The simplest sampling procedure is to pick a *uniform* sample of size k from a set of size m where $k \leq m$ (typically $k \ll m$). We can consider sampling with or without replacement. These are standard and easy procedures if the whole data set is available in a random-access manner — here we are assuming that we have access to a random number generator.

Below we describe a simple yet nice technique called reservoir sampling to obtain a uniform sample of size 1 from a stream.

```
UniformSample:
$s \leftarrow$ null
$m \leftarrow 0$
While (stream is not done)
  $m \leftarrow m + 1$
  $x_m$ is current item
  Toss a biased coin that is heads with probability $1/m$
  If (coin turns up heads)
      $s \leftarrow x_m$
endWhile
Output $s$ as the sample
```

Lemma 13 *Let m be the length of the stream. The output of the algorithm s is uniform. That is, for any $1 \leq j \leq m$, $\Pr[s = x_j] = 1/m$.*

Proof: We observe that $s = x_j$ if x_j is chosen when it is considered by the algorithm (which happens with probability $1/j$), and none of $x_{j+1}, \ldots, x_m$ are chosen to replace x_j. All the relevant events are independent and we can compute:

$$\Pr[s = x_j] = 1/j \times \prod_{i>j} (1 - 1/i) = 1/m.$$

□

To obtain k samples *with* replacement, the procedure for $k = 1$ can be done in parallel with independent randomness. Now we consider obtaining k samples from the stream *without* replacement. The output will be stored in an array of S of size k.

```
Sample-without-Replacement($k$):

$S[1..k] \leftarrow$ null
$m \leftarrow 0$
While (stream is not done)
  $m \leftarrow m + 1$
  $x_m$ is current item
  If ($m \leq k$)
      $S[m] \leftarrow x_m$
  Else
      $r \leftarrow$ uniform random number in range $[1..m]$
      If ($r \leq k$)
          $S[r] \leftarrow x_m$
endWhile
Output $S$
```

An alternative description is that when item x_t arrives (for $t > k$) we decide to choose it for inclusion in S with probability k/t, and if it is chosen then we choose a uniform element from S to be replaced by x_t.

Exercise: Prove that the algorithm outputs a random sample without replacement of size k from the stream.

Weighted sampling: Now we consider a generalization of the problem to weighted sampling. Suppose the stream consists of m items $x_1, \ldots, x_m$ and each item j has a weigth $w_j > 0$. The goal is to choose a k sample in proportion to the weights. Suppose $k = 1$ then the goal is to choose an item s such that $\Pr[s = x_j] = w_j/W$ where $W = \sum_i w_i$. It is easy to generalize the uniform sampling algorithm to achieve this and k samples with replacement is also easy. The more interesting case is when we want k samples without replacement. The precise definition of what this means is not so obvious. Here is an algorithm. Obtain first a uniform sample s from $x_1, \ldots, x_m$ in proportion to the weights. Remove s from the set and obtain another uniform sample in the residual set. Repeat k times to obtain a set of k items (assume that $k < m$). The k removed items form the output S. We now want to obtain a random set S according to this same distribution but in a streaming fashion.

First we describe a randomized offline algorithm below.

```
Weighted-Sample-without-Replacement($k$):

For $i = 1$ to $m$ do
  $r_i \leftarrow$ uniform random number in interval $(0, 1)$
  $w'_i = r_i^{1/w_i}$
endFor
Sort items in decreasing order according to $w'_i$ values
Output the first $k$ items from the sorted order
```

We leave it as a simple exercise to show that the above algorithm can be implemented in the stream model by keeping the heaviest k modified weights seen so far. Now for the analysis.

To get some intuition we make the following claim.

Claim 14 *Let r_1, r_2 be independent unformly distributed random variables over $[0,1]$ and let $X_1 = r_1^{1/w_1}$ and $X_2 = r_2^{1/w_2}$ where $w_1, w_2 \geq 0$. Then*

$$\Pr[X_1 \leq X_2] = \frac{w_2}{w_1 + w_2}.$$

The above claim can be shown by doing basic analysis via the probability density functions. More precisely, suppose $w > 0$. Consider the random variable $Y = r^{1/w}$ where r is chosen uniformly in $(0,1)$. The cumulative probabilty function of Y,

$$F_Y(t) = \Pr[Y \leq t] = \Pr[r^{1/w} \leq t] = \Pr[r \leq t^w] = t^w.$$

Therefore the density function $f_Y(t)$ is wt^{w-1}. Thus

$$\Pr[X_1 \leq X_2] = \int_0^1 F_{Y_1}(t) f_{Y_2}(t) dt = \int_0^1 t^{w_1} w_2 t^{w_2 - 1} dt = \frac{w_2}{w_1 + w_2}.$$

We now make a much more general statement.

Lemma 15 *Let $r_1, r_2, \ldots, r_n$ be independent random variables each of which is uniformly distributed random variables over $[0,1]$. Let $X_i = r_i^{1/w_i}$ for $1 \leq i \leq n$. Then for any $\alpha \in [0,1]$*

$$\Pr[X_1 \leq X_2 \ldots \leq X_n \leq \alpha] = \alpha^{w_1 + w_2 + \ldots + w_n} \cdot \prod_{i=1}^{n} \frac{w_i}{w_1 + \ldots + w_i}.$$

Proof: By induction on n. For $n = 1$, $\Pr[X_1 \leq \alpha] = F_{Y_1}(\alpha) = \alpha^{w_1}$. Assuming the lemma holds for all $h < n$ we prove it for n.

$$\begin{aligned}
\Pr[X_1 \leq \ldots \leq X_n \leq \alpha] &= \int_0^\alpha \Pr[X_1 \leq \ldots \leq X_{n-1} \leq t] f_{Y_n}(t) dt \\
&= \int_0^\alpha t^{w_1 + w_2 + \ldots + w_{n-1}} \cdot \left(\prod_{i=1}^{n-1} \frac{w_i}{w_1 + \ldots + w_i} \right) w_n t^{w_n - 1} dt \\
&= w_n \left(\prod_{i=1}^{n-1} \frac{w_i}{w_1 + \ldots + w_i} \right) \int_0^\alpha t^{w_1 + w_2 + \ldots + w_n - 1} dt \\
&= \alpha^{w_1 + w_2 + \ldots + w_n} \cdot \prod_{i=1}^{n} \frac{w_i}{w_1 + \ldots + w_i}.
\end{aligned}$$

We used the induction hypothesis in the second equality. □

Now we are ready to finish the proof. Consider any fixed j. We claim that the probability that X_j is the largest number among $X_1, X_2, \ldots, X_m$ is equal to $\frac{w_j}{w_1 + \ldots + w_n}$. Do you see why? Conditioned on X_j being the largest, the rest of the variables are still independent and we can apply this observation again. You should hopefully be convinced that picking the largest k among the values $X_1, X_2, \ldots, X_m$ gives the desired sample.

Bibliographic Notes: Morris's algorithm is from [5]. See [3] for a detailed analysis and [1] for a more recent treatment which seems cleaner and easier. Weighted reservoir sampling is from [2]. See [7] for more on efficient reservoir sampling methods.

References

[1] Miklós Csürös. Approximate counting with a floating-point counter. *CoRR*, abs/0904.3062, 2009.

[2] Pavlos S Efraimidis and Paul G Spirakis. Weighted random sampling with a reservoir. *Information Processing Letters*, 97(5):181–185, 2006.

[3] Philippe Flajolet. Approximate counting: a detailed analysis. *BIT Numerical Mathematics*, 25(1):113–134, 1985.

[4] Michael Mitzenmacher and Eli Upfal. *Probability and computing: Randomized algorithms and probabilistic analysis.* Cambridge University Press, 2005.

[5] Robert Morris. Counting large numbers of events in small registers. *Communications of the ACM*, 21(10):840–842, 1978.

[6] Rajeev Motwani and Prabhakar Raghavan. *Randomized Algorithms.* Cambridge University Press, 1995.

[7] Jeffrey S Vitter. Random sampling with a reservoir. *ACM Transactions on Mathematical Software (TOMS)*, 11(1):37–57, 1985.