# CSC2422 Fall 2022: Algorithm Design, Analysis and Theory
# An introductory (i.e. foundational) level graduate course.

Allan Borodin

December 7, 2022; Lecture 11

# Brief Announcements and todays agenda

Announcements

- Assignment 3 is now due Wednesday, December 14 1PM. This is one week beyond the original due date. I am not adding any questions beyond the three posted questions.
- This weeks theory seminar concerns online algorithms with delays. I am posting the announcement on the next slide.

Todays agenda

- Continue streaming algorithms
  - Count min sketch for the heavy hitters problem.
  - Computing the $F_k$ frequency moments
- Semi-streaming vs the traditional online model
- Learning experts: the weighted majority algorithm
- Primality testing

# Seminar: Friday, December 9, 11AM

Speaker: Elizabet Burjons (York University)
Title: "Reservation Models for Online Algorithms"
Abstract: Online algorithms receive their input in an iterative fashion and have to act in an irrevocable way on each piece of the input, with no additional knowledge. The goal, as with offline algorithms, is to optimize some objective function. In order to measure the performance of an online algorithm, we compare the performance to that of an optimal offline algorithm on the same instance. Variants of online models explore different possibilities on how to optimize online algorithms in different settings. In the variant we propose, we allow an algorithm to delay a specific online decision for a fractional cost. We look at two different problems, the knapsack problem and the secretary problem with some promising results.

# Some comments on the Misra-Gries algorithm

It can be shown that no (even randomized) one pass streaming algorithm can return a truly majority element. So the second pass is needed to verify a candidate.

The majority algorithm can be extend to obtain solve the $k$ heavy hitters problem for any small $k$.

Consider the following extension to the $\epsilon$-approximate $\phi$-hitters problem where the algorithm is required to return a set $S$ of elements such that
1) Every element in $S$ appears at least $\phi n$ times.
2) $S$ does not contain any elements less than $(\phi - \epsilon)n$ times.

The Misra-Gries algorithm can be extended to solve this problem by setting $k = \frac{1}{\epsilon}$ and then outputting all elements $a$ that occur at least $(\phi - \epsilon)n$ times.

# Another way to solve an upper bound version of the $\epsilon$-approximate $\phi$-heavy hitters.

The following ideas are another very general approach to solving various streaming problems. We introduce it here as it is an easy case to expose the ideas. And now we are going to use randomization.

The idea is to hash values and then just keep counts of the hashed values. Let $k = 2/\epsilon$. We use a 2-universal hash function $h$ so that $Prob[h(v) = h(v'] = 1/k$ for any two distinct values $v$ and $v'$.

Count Min Sketch uses a number of different hash functions (in paralell) and then takes the min of these counters.

Suppose we just want the counts for the $\phi$-heavy hitters. The Count Min Skecth algorithm has the guarantee that $Prob[count \geq truecount + \epsilon n]$ is small.

# Frequency Moments: What is known about computing $F_k$?

**The notational dilemma:** The seminal paper in the streaming area is the Alon, Matias and Szegedy (AMS) paper for computing the frequency moment problem. Their notation is that a stream is a sequence $a_1, \ldots a_m\}$ with $a_i \in \{1, 2, \ldots n\}$. So many papers follow this convention using $m$ for the length of the stream and $n$ for the size of domain $D$.

To be more consistent with the online literature, I was using $n$ for the length of the sequence and $D$ for the domain of the $a_i$. I will now switch to the more standard AMS streaming notational convention.

Let $m_i$ denote the number of occurences of the value $i$ in the stream $A$. For $k \geq 0$, the $k^{th}$ frequency moment is $F_k = \sum_{i \in [n]} (m_i)^k$. The frequency moments are most often studied for integral $k$.

Given an error bound $\epsilon$ and confidence bound $\delta$, the goal in the frequency moment problem is to compute an estimate $F'_k$ such that $Prob[|F_k - F'_k| > \epsilon F_k] \leq \delta$.

# The Alon et al frequency moment paper

- The seminal paper in this regard is by Alon, Matias and Szegedy (AMS) [1999]. AMS establish a number of results:
  1. For $k \geq 3$, there is an $\tilde{O}(m^{1-1/k})$ space algorithm. The $\tilde{O}$ notation hides factors that are polynomial in $\frac{1}{\epsilon}$ and polylogarithmic in $m, n, \frac{1}{\delta}$.
  2. For $k = 0$ and every $c > 2$, there is an $O(\log n)$ space algorithm computing $F_0'$ such that $Prob[(1/c)F_0 \leq F_0' \leq cF_0$ does *not* hold$] \leq 2/c$.
  3. For $k = 1$, $\log n$ is obvious to exactly compute the length but an estimate can be obtained with space $O(\log \log n + 1/\epsilon)$
  4. For $k = 2$, they obtain space $\tilde{O}(1) = O(\frac{\log(1/\delta}{\epsilon^2})(\log n + \log m))$
  5. They also show that for all $k > 5$, there is a (space) lower bound of $\Omega(m^{1-5/k})$.

# Results following AMS

- A considerable line of research followed this seminal paper. Notably settling conjectures in AMS:
- The following results apply to real as well as integral $k$.
  1. An $\tilde{\Omega}(m^{1-2/k})$ space lower bound for all $k > 2$ (Bar Yossef et al [2002]).
  2. Indyk and Woodruff [2005] settle the space bound for $k > 2$ with a matching upper bound of $\tilde{O}(m^{1-2/k})$
- The basic idea behind these randomized approximation algorithms is to define a random variable $Y$ whose expected value is close to $F_k$ and variance is sufficiently small such that this r.v. can be calculated under the space constraint.
- We will just sketch the (non optimal) AMS results for $F_k$ for $k > 2$ and the result for $F_2$.

# The AMS $F_k$ algorithm

Let $s_1 = (\frac{8}{\epsilon^2} m^{1-\frac{1}{k}})/\delta^2$ and $s_2 = 2 \log \frac{1}{\delta}$.

## AMS algorithm for $F_k$

The output $Y$ of the algorithm is the median of $s_2$ random variables $Y_1, Y_2, ...., Y_{s_2}$ where $Y_i$ is the mean of $s_1$ random variables $X_{ij}, 1 \leq j \leq s_1$ . All $X_{ij}$ are independent identically distributed random variables. Each $X = X_{ij}$ is calculated in the same way as follows: Choose random $p \in [1, \ldots, m]$, and then see the value of $a_p$. Maintain $r = |\{q | q \geq p \text{ and } a_q = a_p\}|$. Define $X = m(r^k - (r-1)^k)$.

- Note that in order to calculate $X$ , we only require storing $a_p$ (i.e. $\log n$ bits) and $r$ (i.e. at most $\log m$ bits). Hence the Each $X = X_{ij}$ is calculated in the same way using only $O(\log n + \log n)$ bits.
- For simplicity we assume the input stream length $m$ is known but it can be estimated and updated as the stream unfolds.
- We need to show that $\mathbf{E}[X] = F_k$ and that the variance $Var[X]$ is small enough so as to use the Chebyshev inequality to show that $Prob[|Y_i - F_k| > \epsilon F_k$ is small.

# Clarifying the indices $i, j$ in $X_{ij}$ in the AMS algorithm

The $i$ (for $1 \leq i \leq s_2$) refers to the index $i$ in the r.v. $Y_i$.

To compute a $Y_i$, we take the mean of $s_1$ trials where $j$ (for $1 \leq j \leq s_1$) is the index for one of these $s_1$ random trials.

Such a trial consists of picking a random index $p$ (for $1 \leq p \leq m$) where following the AMS notation, $m$ is the length of the input sequence.

Then we compute the number $r$ of occcuences of $a_p$ following (and including) the occurence in the $p^{th}$ symbol of the input sequence; that is $r = |\{q : a_q = a_p, q \geq p\}|$.
Finally, the r.v. $x_{ij} = m(r^k - (r-1)^k)$ is for this particular i$j^{th}$ trial. .

# AMS analysis sketch

- Showing $E[X] = F_k$.

$$\frac{m}{m}[(1^k + (2^k - 1^k) + \ldots + (m_1^k - (m_1 - 1)^k)) +$$

$$(1^k + (2^k - 1^k) + \ldots + (m_2^k - (m_2 - 1)^k)) + \ldots ... +$$

$$(1^k + (2^k - 1^k) + \ldots + (m_n^k - (m_n - 1)^k))]$$

(by telescoping)

$$= \sum_i^n m_i^k$$

$$= F_k$$

# AMS analysis continued

- $Y$ is the median of the $Y_i$. It is a standard probabilistic idea that the median $Y$ of identical r.v.s $Y_i$ (each having constant probability of small deviation from their mean $F_k$) implies that $Y$ has a high probability of having a small deviation from this mean.
- $E[Y_i] = E[X]$ and $Var[Y_i] \leq Var[X]/s_1 \leq E[X^2]/s_1$.
- The result needed is that $Prob[|Y_i - F_k| > \epsilon F_k] \leq \frac{1}{8}$
- The $Y_i$ values are an average of independent $X = X_{ij}$ variables but they can take on large vales so that instead of Chernoff bounds, AMS use the Chebyshev inequality:

$$Prob[|Y - E[Y]| > \epsilon E[Y]] \leq \frac{Var[Y]}{\epsilon^2 E[Y]}$$

- It remains to show that $E[X^2] \leq kF_1F_{2k-1}$ and that $F_1F_{2k-1} \leq n^{1-1/k}F_k^2$

# Sketch of $F_2$ improvement

- They again take the median of $s_2 = 2\log(\frac{1}{\delta})$ random variables $Y_i$ but now each $Y_i$ will be the sum of only a constant number $s_1 = \frac{16}{\epsilon^2}$ of identically distibuted $X = X_{ij}$.
- The key additional idea is that $X$ will not maintain a count for each particular value separately but rather will count an appropriate sum $Z = \sum_{t=1}^{n} b_t m_t$ and set $X = Z^2$.
- Here is how the vector $< b_1, \ldots, b_n > \in \{-1, 1\}^n$ is randomly chosen.
- Let $V = \{v_1, \ldots, v_h\}$ be a set of $O(n^2)$ vectors over $\{-1, 1\}$ where each vector $v_p =< v_{p,1}, \ldots, v_{p,n} > \in V$ is a 4-wise independent vector of length $n$.
- Then $p$ is selected uniformly in $\{1, \ldots, h\}$ and $< b_1, \ldots, b_n >$ is set to $v_p$.

# The semi-streaming model

Feigenbaum et al [2005] introduced "semi-streaming" in order to consider sparse graph problem in the streaming model. As such we can view semi-streaming as an online model where we are not required to make *immediate* irrevocable (or revocable) decisions but can only maintain a limited amount of information about the input stream.

Like (tradional) online algorithms, we have a graph $G = (V, E)$ with $|V| = n, |E| = m$ and vertices or edges arrive online in sequence.

If we are interested in producing a solution (i.e., a coloring, an independent set, a matching, etc.) we need at lease $\Omega(n)$ space just for the solution.

The goal is to maintain $\tilde{O}(n)$ space.

As we saw in the Misra-Gries majority algorithm, a different model can leave us open to different ways of thinking about a problem. Another example is a semi-streaming algorithm for unweighted interval selection (call control on the line) due to Emek et al [2016] that is shown to work in the online model that allows for revoking previous acceptances.

# Comparing the semi-streaming and (tradititional) online models

It might seem that the semi-streaming model is less restrictive than the online setting since immediate decisions are not required.

But is it obvious that every online algorithm can be simulated in the semi-streaming model?

# Comparing the semi-streaming and (tradititional) online models

It might seem that the semi-streaming model is less restrictive than the online setting since immediate decisions are not required.

But is it obvious that every online algorithm can be simulated in the semi-streaming model?

An online algorithm does not have to limit itself to $\tilde{O}(n)$ memory. It could remember all edges seen this far and the order in which they arrived. And it can use any amount of spacxe in order to make decisions.

However, it it is not clear when and how to ultilize the unbounded space enjoyed by an online algorithm.

# Bipartite matching in the semi-streaming mpdel

In the edge arrival model,

- There is no (randomized) semi-streaming algorithm with worst case competitive ratio better than $\frac{1}{2}$. That ratio is optimal since any greedy algorithm (i.e. always make a match when possible) achieves $\frac{1}{2}$ in the streaming model as well as in the online model.
- A slightly better ratio exists when edges arrive in random order.

In the vertex arrival model,

- The randomized KVV algorithm also easily carries over to the semi-streaming model (either as a deterministic random order algorithm or as a randomized adversarial order algorithm).
- Surpringly, Goel et al [2011] show that there is a *deterministic* semi-streaming algorithm with adversarial order input sequences. This tends to reinforce our intuition that the semi-streaming model is less restrictive than the tradititional online model.

Is there a (natural) optimization problem for which the competitive ratio of some online algorithm is better than any ratio that can be obtained in the sgtreaming model?. How much can we use the entire history for the first $i$ inputs vs just knowing the current "configuration".

# New topic: the weighted majority algorithm

I am following a survey type paper by Arora, Hazan and Kale [2008]. To quote from their paper: "We feel that this meta-algorithm and its analysis should be viewed as a basic tool taught to all algorithms students together with divide-and-conquer, dynamic programming, random sampling, and the like".

- The weighted majority algorithm and generalizations
  The "classical" WMA pertains to the following situation:
  Suppose we have say $k$ expert weathermen (or maybe "expert" stock market forecasters) and at every time $t$, they give a binary prediction (rain or no rain, Raptors win or lose, dow jones up or down, Canadian dollar goes up or down.
- Now some or all of these experts may actually be getting their opinions from the same sources (or each other) and hence these predictions can be highly correlated.
- Without any knowledge of the subject matter (and why should I be any different from the "experts") I want to try to make predictions that will be nearly as good (over time $t$) as the BEST expert.

# The weighted majority algorithm

## The WM algorithm

Set $w_i(0) = 1$ for all $i$

**For** $t = 0...$

Our $(t+1)^{st}$ predication is

0: if $\sum_{\{i:\text{ expert } i \text{ predicts } 0 \text{ for day } t+1\}} w_i(t) \geq (1/2)\sum_i w_i(t)$

1: if $\sum_{\{i:\text{ expert } i \text{ predicts } 1 \text{ for day } t+1\}} w_i(t) \geq (1/2)\sum_i w_i(t)$ ; arbitrary o.w.

% We vote with weighted majority; arbitrary if tie

**For** $i = 1..k$

**If** expert $i$ made a mistake on $(t+1)^{st}$ prediction

**then** $w_i(t+1) = (1-\epsilon)w_i(t)$;

**else** $w_i(t+1) = w_i(t)$

**End If**

**End For**

**End For**

# How good is our uninformed MW prediction?

**Theorem : Performance of WM**

Theorem: Let $m_i(t)$ be the number of mistakes of expert $i$ after the first $t$ forecasts, and let $M(t)$ be the number of our mistakes. Then for any expert $i$ (including the best expert) $M(t) \leq \frac{2 \ln k}{\epsilon} + 2(1+\epsilon)m_i(t)$ .

- That is, we are "essentially" within a multiplicative factor of 2 plus an additive term of the best expert (without knowing anything).
- Using randomization, the factor of 2 can be removed. That is, instead of taking the weighted majority opinion, in each iteration $t$, choose the prediction of the $i^{th}$ expert with probability $w_i(t)/\sum_i w_i(t)$. Where have we seen this "natural randomization before"?

**Theorem: Performance of Randomized WM**

For any expert $i$, $\mathbf{E}[M(t)] \leq \frac{\ln k}{\epsilon} + (1+\epsilon)m_i(t)$

# Proof of deterministic WM

Let's assume that $\epsilon \leq 1/2$. It follows that

$$-\epsilon - \epsilon^2 \leq \ln(1-\epsilon) < -\epsilon$$

Let $w_{i,t}$ be the weight of the $i^{th}$ expert at time $t$ and let $m_i(t)$ be the number of mistakes made by expert $i$. . Consider the potential function $\Phi(t) = \sum_i w_{i,t}$. Clearly

$$\Phi(t) \geq w_{i,t} = (1-\epsilon)^{m_i(t)}$$

We now need an upper bound on $\Phi(t)$. Since each time the WM algoriithm makes a mistake, at least half of the algorithms make a mistake so that $\Phi(t) \leq (1-\epsilon/2)\Phi(t-1)$. Starting with $\Phi(0) = n$, by induction

$$\Phi(t) \leq n \cdot (1-\epsilon/2)^{M(t)}$$

Putting the two inequalities together and taking logarithms

$$\ln(1-\epsilon)m_i(t) \leq \ln k + M(t)\ln(1-\epsilon/2)$$

The argument is completed by rearranging, using the above facts concerning $\ln(1-\epsilon)$ and then dividing by $\epsilon/2$.

# What is the meaning of the randomized impovement?

- In many applications of randomization we can argue that randomization is (provably) necessary and in other applications, it may not be provable so far but current experience argues that the best algorithm in theory and practice is randomized.
- For some algorithms (and especially online algorithms) analyzed in terms of worst case performance, there is some debate on what randomization is actually accomplishing.
- In a [1996] article Blum states that "Intuitively, the advantage of the randomized approach is that it dilutes the worst case". He continues to explain that in the determinstic algorithm, slightly more than half of the total weight could have predicted incorrectly, causing the algorithm to make a mistake and yet only reducing the total weight by $1/4$ (when $\epsilon = 1/2$). But in the randomized version, there is still a .5 probability that the algorithm will predict correctly. Convincing?

# An opposing viewpoint

- In the blog LessWrong this view is strongly rejected. Here the writer makes the following comments: "We should be especially suspicious that the randomized algorithm guesses with probability proportional to the expert weight assigned. This seems strongly reminiscent of betting with 70% probability on blue, when the environment is a random mix of 70% blue and 30% red cards. We know the best bet and yet we only sometimes make this best bet, at other times betting on a condition we believe to be less probable.
  Yet we thereby prove a smaller upper bound on the expected error. Is there an algebraic error in the second proof? Are we extracting useful work from a noise source? Is our knowledge harming us so much that we can do better through ignorance?" The writer asks: "So what's the *gotcha* ... the improved upper bound proven for the randomized algorithm did not come from the randomized algorithm making systematically better predictions - doing superior cognitive work, being more intelligent - but because we arbitrarily declared that an intelligent adversary could read our mind in one case but not in the other."

# Further defense of the randomized approach

- Blum's article expresses a second benefit of the randomized approach: "Therefore the algorithm can be naturally applied when predictions are 'strategies' or other sorts of things that cannot easily be combined together. Moreover, if the 'experts' are programs to be run or functions to be evaluated, then this view speeds up prediction since only one expert needs to be examined in order to produce the algorithm's prediction ...."
- We also know (in another context) that ROM ordering can beat any deterministic priority order say for the online bipartite matching problem.

# Generalizing: The Multiplicative Weights algorithm

The Weighted Majority algorithm can be generalized to the multiplicative weights algorithm. If the $i^{th}$ expert or decision is chosen on day $t$, it incurs a real valued cost/profit $m_i(t) \in [-1, 1]$. The algorithm then updates $w_i(t+1) = (1 - \epsilon m_i(t))w_i(t)$. Let $\epsilon \leq 1/2$ and $\mathbf{\Phi}(t) = \sum_i w_i(t)$. On day $t$, we randomly select expert $i$ with probability $w_i(t)/\Phi(t)$.

**Performance of The MW algorithm**

The expected cost of the MW algorithm after $T$ rounds is $\sum_{t=1}^{T} \mathbf{m}(t) \cdot \mathbf{p}(t) \leq \frac{\ln k}{\epsilon} + \sum_{t=1}^{T} m_i(t) + \epsilon \sum_{t=1}^{T} |m_i(t)|$ for any expert $i$.

Aside: In learning theory, one often uses $n$ for the number of experts and $T$ for the number of online events. I am sticking with $T$ but avoining the use of $n$ to hopefully avoid confusion.

# Reinterpreting in terms of gains instead of losses

We can have a vector $\mathbf{m}(t)$ of gains instead of losses and then use the "cost vector" $-\mathbf{m}(t)$ in the MW algorithm resulting in:

**Performance of The MW algorithm for gains**

$\sum_{t=1}^{T} \mathbf{m}(t) \cdot \mathbf{p}(t) \geq -\frac{\ln n}{\epsilon} + \sum_{t=1}^{T} m_i(t) - \epsilon \sum_{t=1}^{T} |m_i(t)|$

By taking convex combinations, an immediate corollary is

**Performance wrt. a fixed distribution p**

$\sum_{t=1}^{T} \mathbf{m}(t) \cdot \mathbf{p}(t) \geq -\frac{\ln k}{\epsilon} + \sum_{t=1}^{T} \mathbf{m}(t) - \epsilon |\mathbf{m}(t)|)\mathbf{p}$

# An application to learning a linear binary classifier

Instead of the online application of following expert advice, let us now think of "time" as rounds in an iterative procedure. In particular, we would like to compute a linear binary classifier (when it exists).

- We are trying to classsify objects characterized by $k$ features; that is by points $\mathbf{a}$ in $\Re^k$. We are given $n$ labelled examples $(\mathbf{a}_1, \ell_1), \ldots, (\mathbf{a}_n, \ell_n)$ where $\ell_j \in \{-1, +1\}$
- We are going to assume that these examples can be "well classified" by a linear classifier in the sense that there exists a non negative vector $\mathbf{x}^* \in \Re^n$ (with $x_i \geq 0$) such that $sign(\mathbf{a}_j \cdot \mathbf{x}^*) = \ell_j$ for all $j$.
- This is equivalent to saying $\ell_j \mathbf{a}_j \cdot \mathbf{x}^* \geq 0$ and furthermore (to explain the "well") we will say that $\ell_j \mathbf{a}_j \cdot \mathbf{x}^* \geq \delta$ for some $\delta > 0$.
- The goal now is to learn some linear classifier; ie a non negative $\mathbf{x} \in \Re^n$ such that $\ell_j \mathbf{a}_j \cdot \mathbf{x}^* \geq 0$. Without loss of generality, we can assume that $\sum_i x_i = 1$.
- Letting $\mathbf{b}_j = \ell_j \mathbf{a}_j$, this can now be veiwed as a reasonably general LP (search) problem.

# Littlestone's Winnow algorithm for learning a linear classifier

- Litlestone [1987] used the multiplicative weights approach to solve this linear classification problem.
- Let $\rho = \max_j ||\mathbf{b}_j||_\infty$ and let $\epsilon = \delta/(2\rho)$
- The idea is to run the MW algorthm with the decisions given by the $k$ features and gains specified by the $n$ examples. The gain for feature $i$ with respect to the $j^{th}$ example is defined as $(\mathbf{b}_j)_i/\rho$ which is in [-1,1]. The $\mathbf{x}$ we are seeking is the distribution $\mathbf{p}$ in MW.

## The Winnow algorithm

Initialize $\mathbf{p}$
**While** there are points not yet satisfied
  Let $\mathbf{b}_j \cdot \mathbf{p} < 0$  % a constraint not satisfied
  Use MW to upate $\mathbf{p}$
**End While**

## Bound on number of iterations

The Winnow algorithm will terminate in at most $\lceil 4\rho^2 \ln k/\delta^2 \rceil$ iterations.

# Some additional remarks on Multiplicative Weights

The survey by Arora, Hazan and Kale [2012] discusses other modifications of the MW paradigm and numerous applications. In terms of applications, they sketch results for

- Approximately solving (in the sense of property testing) the decision problem for an LP; that is, given linear constraints expressed by $A\mathbf{x} \geq \mathbf{b}$, the decision problem is to see if such a non-negative $\mathbf{x}$ exists (or more generally, if $\mathbf{x}$ is in some given convex set). The algorithm either returns a $\mathbf{x} : \mathbf{A_i x} \geq \mathbf{b_i} - \delta$ for all $i$ and some additive approximation $\delta$ or says that the given LP was infeasible.
- Solving zero sum games approximately.
- The AdaBoost algorithm of Shapire and Freund
- Some other specific applications including a class of online algorithms.

# Primality testing

- I now want to turn attention to one of the most influential randomized algorithms, namely a poly time randomized algorithm for primality (or perhaps better called compositeness) testing. Let $PRIME = \{N | N \text{ is a prime number}\}$ where $N$ is represented in say binary (or any base other than unary) so that $n = |N| = O(\log N)$.
- History of polynomial time algorithms:
  1. Vaughan 1972 showed that *PRIMES* is in *NP*. Note that co-PRIMES (i.e. the composites) are easily seen to be in *NP*.
  2. One sided error randomized algorithms (for compositeness) by Solovay and Strassen and independently Rabin in 1974. That is, $Prob[ALG \text{ says } N \text{ prime } | N \text{ composite}] \leq \delta < 1$ and $Prob[ALG \text{ says } N \text{ composite } | N \text{ prime}] = 0$
  3. The Rabin test is related to an algorithm by Miller that gives a deterministic polynomial time algorithm assuming a conjecture that would follow from (the unproven) ERH. The Rabin test is now called the Miller-Rabin test.
  4. Goldwasser and Killian establish a 0-sided randomized algorithm.
  5. In 2002, Agarwal, Kayal and Saxena show that primality is in deterministic polynomial time.

# Why consider randomized tests when there is a deterministic algorithm?

- Even though there is now a deterministic algorithm, it is not nearly as efficient as the 1-sided error algorithms which are used in practice. These randomized results spurred interest in the topic (and other number theoretic algorithms) and had a major role in cryptographic protocols (which often need random large primes). Moreover, these algorithms became the impetus for major developments in randomized algoritms.
- While many of our previous randomized algorithms might be considered reasonably natural (or natural extensions of a deterministic algorithm), the primality tests require some understanding of the subject matter (i.e. a little number theory) and these algorithms are not something that immediately come to mind.

# Some basic number theory we need

- $Z_N^* = \{a \in Z_N : gcd(a, N) = 1\}$ is a (commutative) group under multiplication mod $N$.
- If $N$ is prime, then
  1. For $a \neq 0 (mod\ N), a^{N-1} = 1 (mod\ N)$.
  2. $Z_N^*$ is a cyclic group; that is there exists a generator $g$ such that $\{g, g^2, g^3, \ldots, g^{N-1}\}$ (all mod $N$) is the set $Z_N^*$. This implies that $g^i \neq 1 (mod N)$ for any $1 \leq i < N - 1$.
  3. There are exactly two square roots of 1 in $Z_N^*$, namely 1 and -1.
- The Chinese Remainder Theorem: Whenever $N_1$ and $N_2$ are relatively prime (i.e. $gcd(N_1, N_2) = 1$), then for all $v_1 < N_1$ and $v_2 < N_2$, there exists a unique $w < N_1 \cdot N_2$ such that $v_1 = w (mod\ N_1)$ and $v_2 = w (mod\ N_2)$.

# A simple but "not quite" correct algorithm

We also need two basic computational facts.

1. $a^i$ mod $N$ can be computed efficiently.
2. $gcd(a, b)$ can be efficiently computed.

The following is a simple algorithm that works except for an annoying set of numbers called Carmichael numbers.

**Simple algorithm ignoring Carmichael numbers**

Choose $a \in Z_N$ uniformly at random.
If $gcd(a, N) \neq 1$, then Output Composite
If $a^{N-1}$ mod $N \neq 1$, then Output Composite
Else Output Prime

# When does the simple algorithm work?

- $S = \{a | gcd(a, N) = 1 \text{ and } a^{N-1} = 1\}$ is a subgroup of $Z_N^*$
- If there exists an $a \in Z_N^*$ such that $gcd(a, N) = 1$ but $a^{N-1} \neq 1$, then $S$ is a proper subgroup of $Z_N^*$.
- By Lagrange's theorem, if $S$ is a proper subgroup, $|S|$ must divide the order of the group so that $|S| \leq \frac{N-1}{2}$
- Thus the simple algorithm would be a 1-sided error algorithm with probabiltiy $< \frac{1}{2}$ of saying Prime when $N$ is Composite.
- The only composite numbers that give us trouble are the Carmichael numbers (also known as *false primes*) for which $a^{N-1} mod\ N = 1$ for all $a$ such that $gcd(a, N) = 1$.
- It was only recently (relatively speaking) that in 1994 it was proven that there are an infinite number of Carmichael numbers.
- The first three Carmicahel numbers are 561, 1105, 1729

# Miller-Rabin 1-sided error algorithm

Let $N-1 = 2^t u$ with $u$ odd  %Since wlg. $N$ is odd, $t \geq 1$
Randomly choose non zero $a \in Z_N$  %Hoping that $a$ will be composite certificate
If $gcd(a, N) \neq 1$ then report Composite
$x_0 = a^u$  %All computation is done mod $N$
For $i = 1 \ldots t$
    $x_i := x_{i-1}^2$
    If $x_i = 1$ and $x^{i-1} \notin \{-1, 1\}$, then report Composite
End For
If $x_t \neq 1$, then report Composite  $\%x_t = x^{N-1}$
Else report Prime

# Analysis sketch of Miller-Rabin

- Let $S$ be the set of $a \in N$ that pass (i.e. fool) the Rabin-Miller test.
- $S$ is a subgroup of $Z_N^*$. We want to show that $S$ is a proper subgroup and then as before by Langrange we will be done.
- It suffices then to find one element $w \in Z_N^*$ that will not pass the Miller-Rabin test.
  Case 1: $N$ is not Carmichael and then we are done.
  Case 2: $N$ is Carmichael and hence $N$ cannot be a prime power.
  - $N = N_1 \cdot N_2$ and $gcd(N_1, N_2) = 1$ and of course odd
  - The non-certificates must include some $b$ such that $b^{2^i u} = -1(modN)$ and hence $b^{2^i u} = -1(modN_1)$
  - By the Chinese Remainder Theorem, there exists $w = v(modN_1)$ and $w = 1(modN_2)$
  - Hence $w^{2^i u} = -1(modN_1)$ and $w^{2^i u} = 1(modN_2)$
  - This implies $w^{2^i u} \notin \{-1, 1\} \pmod N$