CSC2420 Fall 2012: Algorithm Design, Analysis and Theory

Allan Borodin

March 26, 2015; Lecture 11

Annoucements and Todays Agenda

- Announcements
  1. I added one more question to assignment 3 and (upon request) extended the due date to Tuesday, April 7.
  2. If you are an undergraduate planning to graduate this term, then please email me so that I can be sure that your assignments are graded and a grade is calculated in time for you to graduate.
- Todays agenda
  1. Sublinear space: A slight detour into complexity theory
  2. Streaming model
  3. The weighted majority paradigm
  4. Start of miscellaneous topics (continuing next into final lecture)
     - ★ The Miller-Rabin randomized primality test
     - ★ Monotone submodular maximization subject to matroid and independence constraints and the return of non-oblivious local search.
     - ★ The Lovasz Local lemma and the Moser-Tardos algorithm for finding a satisfying instance of an exact k-SAT formula in which every clause C shares a variable with at most $d < 2^k/e$ other clauses.
     - ★ Spectral methods

Sublinear space: A slight detour into complexity theory

- Sublinear space has been an important topic in complexity theory since the start of complexity theory. While not as important as the $P = NP$ or $NP = co - NP$ question, there are two fundamental space questions that remain unresolved:
  1. Is $NSPACE(S) = DSPACE(S)$ for $S \geq \log n$?
  2. Is P contained in $DSPACE(\log n)$ or $\cup_k SPACE(\log^k n)$? Equivalently, is P contained in polylogarthmic parallel time.
- Savitch [1969] showed a non deterministic S space bounded TM can be simulated by a deterministic S^2 space bounded machine (for space constructible bounds S).
- Further in what was considered a very surprising result, Immerman [1987] and independently Szelepcsényi [1987] $NSPACE(S) = co - NSPACE(S)$. (Savitch's result was also considered suprising by some researchers when it was announced.)

USTCON vs STCON

We let *USTCON* (resp. *STCON*) denote the problem of deciding if there is a path from some specified source node s to some specified target node t in an unidirected (resp. directed) graph G.

- As previously mentioned the Aleliunas' et al [1979] random walk result showed that *USTCON* is in $RSPACE(\log n)$ and after a sequence of partial results about *USTCON*, Reingold [2008] was eventually able to show that *USTCON* is in $DSPACE(\log n)$
- It remains open if
  1. *STCON* (and hence $NSPACE(\log n)$) is in $RSPACE(\log n)$ or even $DSPACE(\log n)$.
  2. $STCON \in RSPACE(S)$ or even $DSAPCE(S)$ for any $S = o(\log^2 n)$
  3. $RSPACE(S) = DSPACE(S)$.

The streaming model

- In the data stream model, the input is a sequence A of inputs $a_1, \ldots, a_m$ which is assumed to be too large to store in memory.
- We usually assume that m is not known and hence one can think of this model as a type of online or dynamic algorithm that is maintaining (say) current statistics.
- The space available $S(m, n)$ is some sublinear function. The input streams by and one can only store information in space S.
- In some papers, space is measured in bits (which is what we will do) and sometimes in words, each word being $O(\log n)$ bits.
- It is also desirable that that each input is processed efficiently, say $\log(m + n)$ and perhaps even in time $O(1)$ (assuming we are counting operations on words as $O(1)$).

The streaming model continued

- The initial (and primary) work in streaming algorithms is to approximately compute some function (say a statistic) of the data or identify some particular element(s) of the data stream.
- Lately, the model has been extended to consider "semi-streaming" algorithms for optimization problems. For example, for a graph problem such as matching for a graph $G = (V, E)$, the goal is to obtain a good approximation using space $\tilde{O}(|V|)$ rather than $O(|E|)$.
- Most results concern the space required for a one pass algorithm. But there are other results concerning the tradeoff between the space and number of passes.

An example of a deterministic streaming algorithms

As in sublinear time, it will turn out that almost all of the results in this area are for randomized algorithms. Here is one exception.

The missing element problem

Suppose we are given a stream $A = a_1, \ldots, a_{n-1}$ and we are promised that the stream A is a permutation of $\{1, \ldots, n\} - \{x\}$ for some integer x in $[1, n]$. The goal is to compute the missing x.

- Space n is obvious using a bit vector $c_j = 1$ iff j has occured.
- Instead we know that $\sum_{j \in A} = n(n+1)/2 - x$.
  So if $s = \sum_{i \in A} a_i$, then $x = n(n+1)/2 - s$.
  This uses only $2 \log n$ space and constant time/item.

Generalizing to k missing elements

Now suppose we are promised a stream A of length $n-k$ whose elements consist of a permutation of $n-k$ distinct elements in $\{1,\ldots,n\}$. We want to find the missing k elements.

- Generalizing the one missing element solution, to the case that there are k missing elements we can (for example) maintain the sum of j^{th} powers $(1 \le j \le k)$ $s_j = \sum_{i \in A}(a_i)^j = c_j(n) - \sum_{i \notin A} x_i^j$. Here $c_j(n)$ is the closed form expression for $\sum_{i=1}^{n} i^j$. This results in k equations in k unknowns using space $k^2 \log n$ but without an efficient way to compute the solution.
- As far as I know there may not be an efficient small space streaming algorithm for this problem.
- Using randomization, much more efficient methods are known; namely, there is a streaming alg with space and time/item $O(k \log k \log n)$; it can be shown that $\Omega(k \log(n/k))$ space is necessary.

Some well-studied streaming problems

- Computing frequency moments. Let $A = a_1 \dots a_m$ be a data stream with $a_i \in [n] = \{1, 2, \dots n\}$. 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.
  1. $F_1 = m$, the length of the sequence which can be simply computed.
  2. F_0 is the number of distinct elements in the stream
  3. F_2 is a special case of interest called the repeat index (also known as Ginis homogeneity index).
- Finding k-heavy hitters; i.e. those elements appearing at least n/k times in stream A.
- Finding rare or unique elements in A.

What is known about computing F_k?

Given an error bound ϵ and confidence bound δ, 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 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.

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.

Before moving on

- Streaming is an important field of recent algorithmic research. We have just considered one problem in one particular input model, the so-called time series model. There are two more general input models in which this and other similar problems can be studied.
  1. Cash register model: the input stream is a sequence $(I_1, I_2, , I_n)$ where $I_t = (a_j, c_t)$ and $c_t \geq 1$ representing how much to increase the (integral) count for item a_j. The current count state $(n_1(t), \ldots, n_m(t))$ at time t is then $n_i(t) = n_i(t-1) + c_t$ if $I_t = (i, c_t)$ and otherwise $n_i(t) = n_i(t-1)$.
  2. Turnstile model: this is the same model but now $|c_t| \geq 1$ allowing (integral) decrements as well as increments.

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

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

% We vote with weighted majority; arbitrary if tie

For $i = 1..n$

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 prediction?

Theorem : Perfomance 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) $M(t) \leq \frac{2 \ln n}{\epsilon} + 2(1+\epsilon)m_i(t)$.

- 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 $w_i(t)/\sum_i w_i(t)$

Theorem: Performance of Randomized WM

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

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

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

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

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 n features; that is by points $\mathbf{a}$ in $\Re^n$. We are given m labelled examples $(\mathbf{a}_1, \ell_1), \ldots, (\mathbf{a}_m, \ell_m)$ 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 classifer; 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 n features and gains specified by the m 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 $\lceil 4\rho^2 \ln n/\delta^2 \rceil$ iterations.

Miscellaneous topics to end the term

Clearly "algorithm design, analysis, and theory" is a extremely broad subject (and one might say it is much of what CS does) so clearly we have only discussed a few topics, mainly ' some standard algorithmic approaches for optimization problems, and even then only discussed them briefly. Here are a few possible topics which with we will conclude the course:

1. The Miller-Rabin randomized primality test.
2. A brief return to (non-oblivious) local search and its power.
3. The constructive Lovasz Local Lemma for 3SAT when variables do not appear in too many clauses.
4. spectral methods

Primality testing

- I now want to briefly 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 text.
  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 algorithms (excluding the streaming algorithm for F_k) 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 comes 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 $a \neq 0(modN), a^{N-1} = 1(modN)$.
  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 $g^i \neq 1(modN)$ 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 $v_1 = w(modN_1)$ and $v_2 = w(modN_2)$.