CSC2420: Algorithm Design, Analysis and Theory
Fall 2023
An introductory (i.e. foundational) level graduate course.

Allan Borodin

November 21, 2023

Week 10

Announcements:

- I posted 2 questions for the third and last assignment. I hope to add one or two questions. The assignment will be due Friday, December 8 at 11 AM.
- I have posted slides by Denis Pankratov on LP theory and LP duality.

Todays agenda
We will discuss the following topics:

- The set cover problem. This is where we left off last week.
- Primal dual algorithms and primal dual fitting.
- Begin streaming algorithms.

Solving the f-frequency set cover

In the f-frequency set cover problem, each element is contained in at most f sets. We can solve the f-frequency cover problem by obtaining an optimal solution $\{x_j^*\}$ to the (primal) LP and then rounding to obtain $\bar{x}_j = 1$ iff $x_j^* \geq \frac{1}{f}$.

The set cover problem as an IP/LP

minimize $\sum_j w_j x_j$
subject to $\sum_{j:u_i \in S_j} x_j \geq 1$ for all i; that is, for all $u_i \in U$
$x_j \in \{0,1\}$ (resp. $x_j \geq 0$)

This is a conceptually simple method but requires solving the LP. We will see that the primal dual method allows us to achieve the same approximation without solving the LP.

Duality: See Vazirani and Shmoys/Williamson texts, and Williamson article

- For a primal maximization (resp. minimization) LP in standard form, the dual LP is a minimization (resp. maximization) LP in standard form.
- Specifically, if the primal $\mathcal{P}$ is:
  - Minimize $\mathbf{c} \cdot \mathbf{x}$
  - subject to $A_{m \times n} \cdot \mathbf{x} \geq \mathbf{b}$
  - $\mathbf{x} \geq 0$
- then the dual LP $\mathcal{D}$ with dual variables $\mathbf{y}$ is:
  - Maximize $\mathbf{b} \cdot \mathbf{y}$
  - subject to $A^{tr}_{n \times m} \cdot \mathbf{y} \leq \mathbf{c}$
  - $\mathbf{y} \geq 0$
- Note that the dual (resp. primal) variables are in correspondence to primal (resp. dual) constraints.
- If we consider the dual $\mathcal{D}$ as the primal then its dual is the original primal $\mathcal{P}$. That is, the dual of the dual is the primal.

An example: set cover

The set cover problem as an IP/LP

minimize $\sum_j w_j x_j$
subject to $\sum_{j:u_i \in S_j} x_j \geq 1 \quad$ for all $u_i \in U$
$x_j \in \{0,1\}$ (resp. $x_j \geq 0$)

The dual LP

maximize $\sum_i y_i$
subject to $\sum_{i:u_i \in S_j} y_i \leq w_j \quad$ for all j
$y_i \geq 0$

If all the parameters in a standard form minimization (resp. maximization) problem are non negative, then the problem is called a covering (resp. packing) problem. Note that the set cover problem is a covering problem and its dual is a packing problem.

Duality Theory Overview

- An essential aspect of duality is that a finite optimal value to either the primal or the dual determines an optimal value to both.
- The relation between these two can sometimes be easy to interpret. However, the interpretation of the dual may not always be intuitively meaningful.
- Still, duality is very useful because the duality principle states that optimization problems may be viewed from either of two perspectives and this might be useful as the solution of the dual might be much easier to calculate than the solution of the primal.
- In some cases, the dual might provide additional insight as to how to round the LP solution to an integral solution.
- Moreover, the relation between the primal $\mathcal{P}$ and the dual $\mathcal{D}$ will lead to primal-dual algorithms and to the so-called dual fiiting analysis.
- In what follows we will assume the primal is a minimization problem.

Strong and Weak Duality

Strong Duality

If x^* and y^* are (finite) optimal primal and resp. dual solutions, then $\mathcal{D}(\mathbf{y}^*) = \mathcal{P}(\mathbf{x}^*)$.

Note: Before it was known that solving LPs was in polynomial time, it was observed that strong duality proves that LP (as a decision problem) is in $\mathbf{NP} \cap \mathbf{co} - \mathbf{NP}$ which strongly suggested that LP was not NP-complete.

Weak Duality for a Minimization Problem

If $\mathbf{x}$ and $\mathbf{y}$ are primal and resp. dual solutions, then $\mathcal{D}(\mathbf{y}) \leq \mathcal{P}(\mathbf{x})$.

- Duality can be motivated by asking how one can verify that the minimum in the primal is at least some value z.

Motivating duality

Consider the motivating example in V. Vazirani's text:

Primal	Dual
minimize $7x_1 + x_2 + 5x_3$	maximize $10y_1 + 6y_2$
subject to	subject to
• (1) $x_1 - x_2 + 3x_3 \geq 10 \quad \%(y_1)$	$y_1 + 5y_2 \leq 7$
• (2) $5x_1 + 2x_2 - x_3 \geq 6 \quad \%(y_2)$	$-y_1 + 2y_2 \leq 1$
	$3y_1 - y_2 \leq 5$
• $x_1, x_2, x_3 \geq 0$	$y_1, y_2 \geq 0$

Adding (1) and (2) and comparing the coefficient for each x_i, we have:

$7x_1 + x_2 + 5x_3 \geq (x_1 - x_2 + 3x_3) + (5x_1 + 2x_2 - x_3) \geq 10 + 6 = 16$

Better yet,

$7x_1 + x_2 + 5x_3 \geq 2(x_1 - x_2 + 3x_3) + (5x_1 + 2x_2 - x_3) \geq 26$

For an upper bound, setting $(x_1, x_2, x_3) = (7/4, 0, 11/4)$

$7x_1 + x_2 + 5x_3 = 7 \cdot (7/4) + 1 \cdot 0 + 5 \cdot (11/4) = 26$

This proves that the optimal value for the abobe primal and ithe dual solution $(y_1, y_2) = (2, 1)$ must be 26.

Easy to prove weak duality

The proof for weak duality

$$\begin{aligned}
&\mathbf{b} \cdot \mathbf{y} = \sum_{j=1}^{m} b_j y_j \\
&\leq \sum_{j=1}^{m} (\sum_{i=1}^{n} A_{ji} x_i) y_j \\
&\leq \sum_{i=1}^{n} \sum_{j=1}^{m} (A_{ji} y_j) x_i \\
&\leq \sum_{i=1}^{n} c_i x_i = \mathbf{c} \cdot \mathbf{x}
\end{aligned}$$

Primal dual for f-frequency set cover

We know that for a minimization problem, any dual solution is a lower bound on any primal solution. One possible goal in a primal dual method for a minimization problem will be to maintain a fractional feasible dual solution and continue to try improve the dual solution. As dual constraints become tight we then set the corresponding primal variables.

Suggestive lemma

We want the following property of our algorithm: Let $\{y_i^*\}$ be an optimal solution to the dual LP and let $\mathcal{C}' = \{S_j | \sum_{e_i \in S_j} y_i^* = w_j\}$. Then $\mathcal{C}'$ is a cover.

Primal dual for f-frequency set cover continued

This suggests the following algorithm:

Primal dual algorithm for set cover

Set $y_i = 0$ for all i; $\mathcal{C}' := \varnothing$
While there exists an e_i not covered by $\mathcal{C}'$
    Increase the dual variables y_i until there is some j : $\sum_{\{k:e_i \in S_j\}} y_i = w_j$
    $\mathcal{C}' := \mathcal{C}' \cup \{S_j\}$
    Freeze the y_i associated with the newly covered e_i
End While

Theorem: Approximation bound for primal dual algorithm

The cover formed by tight constraints in the dual solution provides an f approximation for the f-frequency set cover problem.

Proof of the approximation ratio for the primal dual f-frequency set cover algorithm

Here again I am completely following the exposition in the Vazirani text and the notation for the general statement of the primal and dual LPs.

The proof follows as an application of the relaxed primal and dual complementary slackness conditions and how those yield an approximation ratio. See Section 15.1 in Vazirani.

Primal complementary slackness conditions:
Let $\alpha \geq 1$. $\forall j$ either $x_j = 0$ or $c_j/\alpha \leq \sum_i a_{ij} y_i \leq c_j$.

Dual complementary slackness:
Let $\beta \geq 1$, $\forall i$ either $y_i = 0$ or $b_i \leq \sum_j a_{ij} x_j \leq \beta b_i$.

Theorem If the relaxed complemnetary slackness conditions hold then the primal dual algorith is an $\alpha \cdot \beta$ approximation.
For the f-frequency problem, the algorithm satisfies $\alpha = 1$ and $\beta = f$.

Comments on the primal dual algorithm

- What is being shown is that the integral primal solution is within a factor of f of the dual solution which implies that the primal dual algorithm is an f-approximation algorithm for the f-frequency set cover problem.
- In fact, what is being shown is that the integraility gap of this IP/LP formulation for f-frequency set cover problem is at most f.
- In terms of implementation we would calculate the minimum ϵ needed to make some constraint tight so as to chose which primal variable to set. This ϵ could be 0 if a previous iteration had more than one constraint that becomes tight simultaneously.

More comments on primal dual algorithms

- We have just seen an example of a basic form of the primal dual method for a minimization problem. Namely, we start with an infeasible integral primal solution and feasible (fractional) dual. (For a covering primal problem and dual packing problem, the initial dual solution can be the all zero solution.) Unsatisfied primal constraints suggest which dual constraints might be tightened and when one or more dual constraints become tight this determines which primal variable(s) to set.
- Some primal dual minimization algorithms extend this basic form by using a second (reverse delete) stage to achieve minimality. Some primal dual maximization algorithms use a reverse delete to enforce feasibility. There is some (for me not precise) relation between primal dual and local ratio algorithms (see Bar-Yehuda and Rawitz)
- **NOTE:** In the primal dual method we are not solving any LPs. Primal dual algorithms are viewed as "combinatorial algorithms" and in some cases they might even suggest an explicit greedy algorithm.

Using dual fitting to prove the approximation ratio of the greedy set cover algorithm

We have already seen the following natural greedy algorithm for the weighted set cover problem:

The greedy set cover algorithm

$\mathcal{C}' := \varnothing$

While there are uncovered elements

Choose S_j such that $\frac{w_j}{|\tilde{S}_j|}$ is a minimum where

$\tilde{S}_j$ is the subset of S_j containing the currently uncovered elements

$\mathcal{C}' := \mathcal{C}' \cup S_j$

End While

We wish to prove the following theorem (Lovasz[1975], Chvatal [1979]):

Approximation ratio for greedy set cover

The approximation algorithm for the greedy algorithm is H_d where d is the maximum size of any set S_j.

The dual fitting analysis

The greedy set cover algorithm setting prices for each element

$\mathcal{C}' := \varnothing$

While there are uncovered elements

Choose S_j such that $\frac{w_j}{|\tilde{S}_j|}$ is a minimum where

$\tilde{S}_j$ is the subset of S_j containing the currently uncovered elements

%Charge each element e in $\tilde{S}_j$ the average cost $price(e) = \frac{w_j}{|\tilde{S}_j|}$

% This charging is just for the purpose of analysis

$\mathcal{C}' := \mathcal{C}' \cup S_j$

End While

- We can account for the cost of the solution by the costs imposed on the elements; namely, $\{price(e)\}$. That is, the cost of the greedy solution is $\sum_e price(e)$.

Dual fitting analysis continued

- The goal of the dual fitting analysis is to show that $y_e = price(e)/H_d$ is a feasible dual and hence any primal solution must have cost at least $\sum_e price(e)/H_d$.
- Consider any set $S = S_j$ in $\mathcal{C}$ having say $k \leq d$ elements. Let $e_1, \ldots, e_k$ be the elements of S in the order covered by the greedy algorithm (breaking ties arbitrarily). Consider the iteration is which e_i is first covered. At this iteration $\tilde{S}$ must have at least $k - i + 1$ uncovered elements and hence S could cover cover e_i at the average cost of $\frac{w_j}{k-i+1}$. Since the greedy algorithm chooses the most cost efficient set, $price(e_i) \leq \frac{w_j}{k-i+1}$.
- Summing over all elements in S_j, we have $\sum_{e_i \in S_j} y_{e_i} = \sum_{e_i \in S_j} price(e_i)/H_d \leq \sum_{e_i \in S_j} \frac{w_j}{k-i+1} \frac{1}{H_d} = w_j \frac{H_k}{H_d} \leq w_j$. Hence $\{y_e\}$ is a feasible dual.

The streaming model

- In the data stream model, the input is a sequence A of input items (or input elements) $a_1, \ldots, a_n$ which is assumed to be too large to store in memory. Let $a_i \in [1, D]$
  Small notational dilemma: A seminal paper in the streaming area is the Alon, Matias and Szegedy (AMS) paper for computing the frequency moment problem. Their notation (and for many following papers) is that a stream is a sequence $a_1, \ldots a_m\}$ with $a_i \in \{1, 2, \ldots n\}$. In the text, we are trying to consistently use n for the length of the sequence and D for the domain then $m = |D|$ for the size of the input domain. But for the frequency moments results I will stay with the AMS notation. We had the same dilemma for online bipartite matching where the literature convention is mainly to have n be the number of offline nodes.
- We usually assume that the length n of the stream is not known. and one can think of this model as a type of online or dynamic algorithm.

Streaming and online algorithms

The streaming model is similar to the online model in that an algorithm has no control over the order of arrival of input items, but that is mainly where the similarity ends.

While the online setting focuses on irrevocable immediate decisions for each input arrival, the streaming setting focuses on the amount of memory required to process the input stream.

Throughout a streaming computation, the available space, $S(n, |D|)$, is a sublinear (often logarithmic) function. The input items stream by and one can only store information bounded by space S.

As mentioned, the streaming model is most often used to approximately compute statistics about the data stream. And here the goal is to achieve good estimates while maintaining very small space.

In contrast, online algorithms are usually considered with respect to optimization (or search) problems.

Semi-streaming

Streaming algorithms have been considered for graph optimization problem but now the speace requirement is relaxed. Namely, for graphs $G = (V, E)$, the goal is to achieve approximations using only space $\tilde{O}(|V|)$ rather than space $O(|E|)$ or space $O(n)$.

These are called *semi-streaming algorithms*. More generally (beyond graph problems), we can say semi-streaming means that the space is $\tilde{O}(OPT)$ rather than space usage that could entail the entire input stream.

In semi-streaming algorithms, "revocable decisions" are clearly possible since there is no requirement that decisions are irrevovable. Later, we will compare the semi-straaming model and online algorithms (with and without revoking) where the comparison makes more sense.

Although time and space complexity are always important, competitive analysis does not impose any complexity requirements. Similarly, streaming and semi-streaming does not impose bounds on the time to process an input item; in practice, we want fast time per input item.

The streaming model continued

- In some papers, space is measured in bits and sometimes in words, while understanding that each word would cost $O(\log D)$ bits.
- As mentioned, it is desirable that that each input item is processed efficiently, say $\log(n) + \log(m)$ time, and perhaps even in time O(1) (assuming we are counting operations on words as $O(1)$).
- The initial (and primary) work in streaming algorithms is to approximately compute some function (say a statistic) of the data or identify some particular item(s) in 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 results concerning multi-pass algorithms and also 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 item 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 $[1, n]$. (Here $D = n$.) 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 $2 \log m$ space and constant time/item.

Generalizing to k missing elements

Now suppose we are promised a stream A of length $n-k$ whose input 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 $(1 \leq j \leq k)$ $s_j = \sum_{i \in A}(a_i)^j = c_j(n) - \sum_{i \notin A} x_i^j$. Here $c_j(m)$ 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 *deterministic* 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 also a special case of interest called the repeat index (also known as Gini's homogeneity index).
- Finding k-heavy hitters; i.e. those elements appearing more than n/k times in stream A of lenfgth n.
- Finding rare or unique elements in A.

The majority element problem

While most streaming algorithms concern one pass over the input stream, there are results that use two or more passes.

One relatively easy (but still very interesting) result is the Misra-Gries algorithm for computing k heavy hitters. As a special case, we have the majority problem (i.e. the k-hitter problem for $k = 2$).

There is a temptation to solve this problem by divide and conquer; divide the sequence in half, find the heavy hitters in each half and then check.

The streaming model fascilitates thinking about a much better solution. In the case of majority, lets just try to maintain one possible candidate in the first pass and then check to see if the candidate is a majority item in the second pass. See the Chakrabarti Lecture notes.

The Misra-Gries algorithm

Maintain a candidate for the majority element and a counter for that candidate.

When the counter is empty, the next element in the stream becomes the candidate.

Every time the next element inG the stream is the candidate increase the counter by 1. If the next element is not the candidate decrease the counter by 1.

Claim: If there is a majority element then it has to be the current candidate.

We can use a second pass over the elements to check if the candidate occurs more than $n/2$ times.

The space used is $O(\log n + \log D)$ and the time is $(\log n)$ (or $O(1)$ if counting element comparisons) per input element.

Some comments on the Misra-Gries algorithm

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

As suggested the majority algorithm can be extended to solve the k heavy hitters problem for any small k.

Consider the following extension to the ϵ-approximate ϕ-hitters problem where the algorithm is required to return a set S of elements such that
1) Every element in S appears at least ϕn times.
2) S does not contain any elements occuring 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 ϵ-approximate ϕ-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.

Suppose we just want the counts for the ϕ-heavy hitters. We want to get all the counts and have the guarantee that $Prob[count \geq truecount + \epsilon m]$ is small.

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 $Prob[h(v) = h(v'] = 1/k$ so any two distinct values v and v'.

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

Frequency Moments: 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 sketch the (non optimal) AMS results for F_k for $k > 2$ and the result for F_2.
- For k = 0, counting the number of distinct items, there is a wonderful new *simple* algorithm due to Chakraborty, Vinodchandran and Meel (a new UT faculty member).

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.

The CVM algorithm for F_0

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 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 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 $1 - \frac{1}{e}$ competitive *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.

Comparing semi-streaming and the online model with and without revocable items

Is there a (natural) optimization problem and for which some online algorithm (with or without allowing revoking) has a competitive ratio s better than any ratio obtainable in the streaming model?. How much can we use the entire history for the first i inputs vs just knowing the current "configuration".

For the unweighted interval selection problem, there is an interesting result. In the semi-streaming model, there is a deterministiic algorithm that is a $\frac{1}{2}$ approximationi while for the online model with revoking, the optimal deterministic competitve ratio is $2k$, where k is the number of different interval lengths. However, the streaming algorithm can be randomized and implemented as an online algorithm with revoking that achieves competitive ratio $\frac{1}{6}$.

Ansering (to some extent) a question I raised in class

I raised the question as to whether there were any natural problems for which there is an online algorithm using much more than $O(OPT)$ memory.

The answer is YES. The new randomized Fahrbach et al OCS based methods for Display Ads (which achieve competitive ratio $\rho > \frac{1}{2}$) use $O(|E|)$ memory. This method does not need to revoke any decisions.

This then raises the question: Is there a semi-streaming algorithm for Display Ads with competitive ratio $\rho > \frac{1}{2}$.