600.464 Randomized and Big Data Algorithms
Homework #3 Answers

Ravindra Gaddipati

November 8, 2016

Problem 1 (3 points), MU 6.16

Use the Lovasz local lemma to show that, if

$$4\binom{k}{2}\binom{n}{k-2}2^{1-\binom{k}{2}} \leq 1$$

then it is possible to color the edges of K_n with two colors so that it has no monochromatic K_k subgraph.

Answer:

To show it is possible to color the graph with two colors such that it has no monochromatic subgraph K_k, we show the following. Let E_i be the event that clique i of size k is a monochromatic subgraph of K.

- $P(E_i) \leq p$ for all i
- The degree of dependency between $E_1, \ldots, E_n$ is bounded by d
- $4dp \leq 1$

We examine each possible clique of size k independently, where there are $\binom{n}{k}$ possible cliques. To obtain a monochromatic subgraph, after coloring an edge, there remains $\binom{k}{2} - 1$ to be colored the same (Section 6.1). Thus $p = 2^{1-\binom{k}{2}}$ represents the probability that the subgraph is mono-chromatically colored.

Two cliques are dependent if they share more than 1 edge, since we fix each edge and look at all the possible subgraph that include this edge. Then, at most, there are $\binom{k}{2}\binom{n}{k-2}$ dependencies, where $\binom{k}{2}$ represents the number of edges and $\binom{n}{k-2}$ represents the number of possible dependent graphs. Using these results, we see that,

$$4dp = 4\binom{k}{2}\binom{n}{k-2}2^{1-\binom{k}{2}} \leq 1 \tag{1}$$

Applying the Lovasz local lemma,

$$P\left(\bigcap_{i=1}^{n} E_i\right) > 0 \tag{2}$$

showing that the probability that there does not exist a subgraph K_k is greater than 0.

Problem 2 (3 points), MU 6.5

We have shown using the probabilistic method that if a graph G has n nodes and m edges, then there exists a partition of the n nodes into sets A and B such that at least $m/2$ edges cross the partition. Improve this result slightly: show that there exists a partition such that at least $mn/(2n-1)$ edges cross the partition.

Answer:

We divide the nodes evenly into the two sets A and B. For an edge to cross the partition, than we need to have the corresponding pair of nodes x,y in different sets. Let a_i be the event that edge i crosses the partition, and a the total number of edges crossing the partition. We look at the case where x is in set A and y is in set B, and vice versa.

$$P(a_i) = 2\left(\frac{n/2}{n}\frac{n/2}{n-1}\right) \tag{3}$$

$$= \frac{n}{2n-2} \tag{4}$$

Than the expected number of edges crossing the boundary is

$$E[a] = \sum_{i \in m} E[a_i] = \frac{mn}{2n-2} \geq \frac{mn}{2n-1} \tag{5}$$

We also look at the case where n cannot be divided equally into sets A and B. In this case, one set has an extra node.

$$P(a_i) = \frac{(n-1)/2}{n}\frac{(n-1)/2+1}{n-1} + \frac{(n-1)/2+1}{n}\frac{(n-1)/2}{n-1} \tag{6}$$

$$= \frac{n+1}{2n} \tag{7}$$

and the expected number of edges crossing the boundary is

$$E[a] = \sum_{i \in m} E[a_i] = \frac{mn+m}{2n} \geq \frac{mn}{2n-1} \tag{8}$$

Since in both cases the expected number of edges crossing the partition is greater than the bound, we can say there exists a partition such that at least $mn/(2n-1)$ edges cross the partition.

Problem 3 (3 points), MU 6.6

We can generalize the problem of finding a large cut to finding a large $k-$cut. A $k-$cut is a partition of the vertices into k disjoint sets, and the value of a cut is the weight of all of the edges crossing from one of the k sets to another. In Section 6.2.1 we considered 2-cuts when all edges had the same weight 1, showing that via the probabilistic method that any graphd G with m edges has a cut with atleast $m/2$. Generalize this argument to show that any graph G with m edges has a $k-$cut with value at least $(k-1)m/k$. Show how to use the derandomization (following the argument of Section 6.3) to give a deterministic algorthim for finding such a cut.

Answer:

Let X_i be an indicator variable, where $X_i = 1$ if edge i connects two sets, otherwise $X_i = 0$. We independently and randomly assign each vertex to one of the k sets.

$$E[X_i] = \frac{k-1}{k} \tag{9}$$

Since after placing the first vertex into a set we have $k-1$ options for the edge to cross sets. Let $C(A)$ be the cut value from set A to all other sets. Than,

$$\begin{aligned} E[C(A)] &= E\left[\sum_{i \in m} X_i\right] & (10)\\ &= \sum_{i \in m} E[X_i] & (11)\\ &= \frac{m(k-1)}{k} & (12) \end{aligned}$$

Thus we can say there exists a cut with at least $C(A) = m(k-1)/k$.

Derandomization

We follow a similar procedure to section 6.3, and prove using induction that $E[C(A)] \leq E[C(A)|x_1, \ldots, x_n]$, where x_i is the placement of a vertex. and that there exists a cut of value at least $m(k-1)/k$. We initially place $k-1$ vertices into $k-1$ sets randomly such that each of the $k-1$ sets has 1 vertex. Then we deterministically place the subsequent vertices into set k_i such that $C(A)$ is maximized. To be a k-cut, than we must place nodes into k sets. Thus we can say our base case is:

$$E[C(A)] = E[C(A)|x_1, \ldots, x_{k-1}] \tag{13}$$

After the initial placement of q vertices, each subsequent vertex is placed into any of the K sets. Let $Y_i = A$ be the random variable representing the placement of vertex x_i into set A. We independent and random placement,

$$E[C(A)|X_1,\ldots,x_q] \leq \frac{1}{k}\sum_{i\in K} E[C(A)|x_1,\ldots,x_q,Y_{q+1}=K_i] \tag{14}$$

Since each edge has positive or 0 weight, we can say that

$$\max\left(\frac{1}{k}\sum_{i\in K} E[C(A)|x_1,\ldots,x_q,Y_{q+1}=K_i]\right) \geq E[C(A)|x_1,\ldots,x_q] \tag{15}$$

Given the cut before the $q+1$ vertex is placed, we can calculate the cut value after each placement and maximize it, giving us the result

$$E[C(A)|x_1,\ldots,x_q] \leq E[C(A)|x_1,\ldots,x_{q+1}] \tag{16}$$

The probability for each node to contribute to the cut depends on the placement of the opposite node in a different set, with probability $(k-1)/k$ since after a node has been placed, there are $k-1$ options to place the partner node such that the edge crosses sets. With m vertices and linearity of expectation, we expect a cut with at least $m(k-1)/k$. With a deterministic approach and greedily choosing set K_i for each vertex x_i that maximizes $C(A)$, we are able to expect a cut value of at least $m(k-1)/k$.

Problem 4 (3 points)

Give a method that receives, as an input, k independent random bits and outputs N pairwise independent bits where $k = O(N)$. Try to make N, as a function of k, as large as possible.

Note: it is possible to make N exponential in k.
Answer:
We take every possible subset of bits, excluding the empty set. With k input bits, we obtain $2^k - 1$ subsets. We than XOR each subset to obtain one bit from each subset. Using XOR, each subsequent bit has probability 1/2 that the result is 0 or 1. Since knowledge of one subset or bit does not determine the output of the XOR of another,this maintains the requirement of independence since each bit in the subset is necessary to generate the result with equal probability. This is done for each subset, yielding N pairwise independent bits. Since there are $2^K - 1$ subsets, N is exponential in k.

Problem 5 (3 points)

Alice and Bob have strings $x, y \in \{0,1\}^n$ and Bob wants to determine whether $x = y$. Suppose Alice sends a message of length m to help Bob.
(1) Alice chooses her message deterministically, prove that $m = \Omega(n)$.
(2) Design a random protocol in which m is as small as possible. You may assume players observe a public random string. Bob should return the correct answer with probability $1 - \delta$.
Answer:
(1)
Suppose there exists an algorithm $A(m)$ that compresses message m with a bound lower than $\Omega(n)$. If we recursively apply A to message m, than the message will decrease in size. As the number of times we apply the algorithm approaches infinity, the message size approaches 0. Since it is not possible to compress a message to size 0 while maintaining all the information of the original algorithm, then by way of contradiction, we show no such algorithm exists.
(2)
The protocol is as follows:

1. Alice and Bob regard the public string S as c strings with length n, i.e. $S_i = S[ic : i(c+1)]$.

2. Alice computes the hamming distance between her string and each S_i. She then sends these messages to Bob with $O(c * \log n)$ since at worst we send the length of the string in the case that all bits differ.

3. Bob computes the hamming distance between his string and all S_i. If the hamming distances are the same as those received from Alice, than we return True. Otherwise False.

If the message as d bits that differ in a message of size n, than the number of ways the bits can differ is $\binom{n}{d}$ (i.e. the number of messages represented by the same hamming distance). Thus the error for each hamming distance d and message of size n is

$$\frac{\binom{n}{d} - 1}{\binom{n}{d}} \tag{17}$$

Since we compute c hamming distances, the error becomes

$$\delta = \left(\frac{\binom{n}{d} - 1}{\binom{n}{d}} \right)^c \tag{18}$$

Bounding by the maximum possible error in the hamming distance,

$$\delta \leq \left(\frac{\binom{n}{n/2} - 1}{\binom{n}{n/2}} \right)^c \tag{19}$$

A larger public string results in a reduced error at the cost of sending c messages, giving a bound of $O(c \log n)$.

Problem 6 (3 points)

Consider the stream of numbers $D = \{p_1, \ldots, p_m\}$ where $p_i \in [n]$. Let $f_i = |\{j : p_j = i\}|$ and let $F_2 = \sum_{i=1}^{n} f_i^2$. Element $i \in [n]$ is an (α, F_2)-heavy hitter if $f_i^2 \geq \alpha F_2$. Design an algorithm that makes a single pass over D uses sublinear space and outputs $I \in [n]$ such that the following is true. If i_0 is the $(0.9, F_2)$-heavy hitter then the algorithm must output $I = i_0$. If there is no $(0.9, F_2)$-heavy hitter then the algorithm is allowed to output any value. The algorithm can err with probability at most 0.1.

Hint: use the AMS sketch.

Answer:

The algorithm is as follows:

- Maintain an AMS sketch A with width determined by parameter $\epsilon = 0.1$, where $\hat{f}_i = A(h(p_i))$
- Maintain an F_2 estimator in counter c. As we get p_i, hash the element to $-1, 1$ and add this to c. Return c^2 as our estimate $\hat{F}_2$.
- When receiving element p_i, update the value of $\hat{f}_i$ in that bin.
- if $\hat{f}_i^2 \geq \alpha \hat{F}_2$, than update our heavy hitter as p_i.
- Otherwise, simply return the element p_i that generated the maximum $\hat{f}_i^2$

We first look at our F_2 estimator.

$$E[c^2] = E\left[\sum_{i,j \in n} x_i x_j h(i) h(j)\right] \tag{20}$$

$$= \sum x_i^2 E[h(i)h(i)] + \sum_{i \neq j} \sum_{i,j \in n} p_i p_j * 0 \tag{21}$$

$$= \sum p_i^2 = F_2 \tag{22}$$

In the case that there is a $(0.9, F_2)$ heavy hitter, the estimate $\hat{f}_i^2$ obtained has a maximum error of $\epsilon = 0.1$ by the construction of the sketch. Since each bin in the AMS sketch is bounded by the ϵ error, we return the heavy hitter that exists within this bound. In the case that there is no heavy hitter, we simply return the maximum estimate for $\hat{f}_i^2$. We are again bounded by the ϵ probability that this is the true heavy hitter. The algorithm is allowed to return any value if no heavy hitter exists-thus it satisfies all the requirements.

Collaborators

I worked with Matthew Ige, Emily Wagner, and Phillipe Piantonne on these problems. In addition, number 6 was discussed with Marc Rosen. Lecture notes and the textbook were used, as well as the following lecture slides.
sublinear.wikischolars.columbia.edu/file/view/Lecture%203.pdf/