Chapter 12: Lower Bounds for Online Algorithms

CMSC 858M: Algorithmic Lower Bounds: Fun with Hardness
Instructors: Mohammad T. Hajiaghayi and William Gasarch
Textbook: Fun with Hardness: Algorithmic Lower bounds

Juan Luque

Spring 2021

1 Introduction to Streaming Algorithms

Streaming algorithms share many similarities with online algorithms. For example, they both require decisions without seeing all of the data but streaming algorithms can use a limited amount of memory to defer decision making. Intuitively, this is helpful when parsing very large inputs where it is not possible to store the entirety of input in fast-access memory. In other words, streaming algorithms take a large set of data and process without seeing the entire stream with a limited amount of memory. The study of streaming algorithms began with the seminal paper of [AMS99], which won the Gödel prize.

A brief introduction to communication complexity is given since it is necessary for proving lower bounds for streaming algorithms.

Streaming algorithms are used for processing very long data streams. Online social media such as Facebook, Instagram and Twitter produce such streams.

Example 1. *Consider receiving a stream where each of the numbers in* $\{1, \ldots, n\}$ *appears once except for one of the elements. We wish to output the number that was not received while being constrained to* $O(\log n)$ *space.*

Due to the space limitation, we cannot even hold the entire data in memory, nor create a sort of table to check off seen elements. However, there is a simple algorithm! Simply maintain a running total sum s of numbers seen throughout the stream. Then output $\frac{n(n+1)}{2} - s$. This is precisely the missing number.

2 Communication complexity

This section provides a brief, but necessary, introduction to communication complexity. An excelennet reference that includes all proofs and theorems in this section is the bookby Kushilevitz and Nisan [KN97].

First some definitions

Definition 1. *Let $f : X \times Y \to Z$ be a function. The 2-party communication model consists of two players, Alice and Bob. Alice is given an input $x \in X$ and Bob is given an input $y \in Y$. Their goal is to compute $f(x, y)$. The challenge is that neither player knows the other's input. We are concerned with the number of bits that Alice and Bob must communicate to compute f.*

1. *A protocol for f is an algorithm for Alice and Bob to compute f. Formally, it is a decision tree where what a player sends depends on what they have already seen and their input.*

2. *The best protocol is the one with the smallest worst case needed bits.*

3. *The communication complexity of f is the worst case of the best protocol.*

4. *Protocol variants include (1) deterministic, (2) randomized with a small probability of error, (3) one-way protocol which means that only one player sends information thus the other player only receives. Both roles must be specified and only the receiver needs to be able to compute f.*

Next we introduce three well known problems from communication complexity. These problems will be key to proving lower bounds for streaming algorithms.

Definition 2.

1. *INDEX.*
   Instance: Alice has a string $x \in \{0, 1\}^n$ and Bob has a number $i \in [n]$.
   Question: Bob wants to know x_i. We use the one-way model so Alice sends Bob a string and just from that Bob needs to find x_i.

2. *INDEXSAME.*
   Instance: Alice has a string $x \in \{0, 1\}^n$ and Bob has a number $i \in [n - 1]$.
   Question: Bob wants to know if $x_i = x_{i-1}$. We use the one-way model so Alice sends Bob a string and just from that Bob needs to find x_i.

3. *DISJ (short for disjointness).*
   Instance: Alice has a string $x \in \{0,1\}^n$ and Bob has a string $y \in \{0,1\}^n$.
   Question: Alice and Bob both want to know if the sets (represented by bit vectors) of x and y are disjiont. The communication is 2-way and they have as many rounds as they want.

Note that the difficultly in INDEX arises because Alice does not know Bob's i. Since she doesn't know Bob's i, loosely speaking, she has to send the entirety of the input, result in the following Theorem. DISJ uses as many rounds as they want so it can be used to prove lower bounds for streaming algorithms with an arbitrary number of passess of the data.

Theorem 1.

1. *INDEX and INDEXSAME require $\Omega(n)$ bits (in the 1-way communication model). This lower bound also holds for both deterministic and randomized protocols. (Randomized protocol uses public coins and probability of correctness is $\geq \frac{2}{3}$.)*
2. *DISJ requires $\Omega(n)$ bits (in the 1-way communication model). This lower bounds also holds for both deterministic and randomized protocols. It also holds when promised that $\sum x_i = \sum y_i = \lfloor n/4 \rfloor$. (Randomized protocol uses public coins and probability of correctness is $\geq \frac{2}{3}$.)*

3 Lower bounds on graph streaming problems

With the results from the previous section on communication complexity we can prove lower bounds for problems in the following way. Suppose we have an instance I of Π and we wish to show a lower bound on required memory for I. Then, for the sake of contradiction, suppose there exists an algorithm $\mathcal{A}$ using $o(n)$ bits of memory to solive I. Use this algorithm to design a protocol that solves INDEX, INDEXSAME, or DISJ. Finally this leads to a contradiction with the previous theorem since we would have solved a problem that requires $\Omega(n)$ bits using only $o(n)$.

3.1 Lower bounds using INDEX: Max-Conn-Comp(k)

Definition 3. Max-Conn-Comp(k) *($k \geq 3$).*
Instance: A forest $G = (V, E)$.
Question: Is there a connected component of size $\geq k$. Note that k is not a part of the input. We have defined an infinite set of problems.

Theorem 2. *Let $k \geq 3$. Any single-pass streaming graph algorithm that solves* Max-Conn-Comp(k) *problems on a forest needs $\Omega(n)$ space.*

Proof. Proved by reduction from INDEX to Max-Conn-Comp(k). Assume there is a $o(n)$ space streaming algorithm $\mathcal{A}$ for Max-Conn-Comp(k). We use $\mathcal{A}$ in the following protocol for INDEX. The protocol will use $o(n)$ space which contradicts Theorem 3. Hence no such $\mathcal{A}$ can exist.

1. Alice has $x_1 \ldots x_n \in \{0,1\}^n$ and Bob has $i \in [n]$. Our goal is that Alice gives Bob a strsing of length $o(n)$ and then Bob knows what x_i is.

2. Alice and Bob construc tdifferents parts of a graph. The vertices are $V_l \cup V_r \cup V_d$ where

$$V_l = \{l_1, \ldots, l_n\}, \quad V_r = \{r_1, \ldots, r_n\}, \quad V_d = \{d_1, \ldots, d_{k-2}\}.$$

   (a) Alice constructs the graph on vertices $V_l \cup V_r$ by letting

$$E_A = \{(l_j, r_j) : x_j = 1\}.$$

   (b) Bob construfcts the graph on vertices $\{r_i\} \cup V_d$ by letting

$$E_B = \{(r_i, d_1)\} \cup \{(d_i, d_{i+1}) : 1 \leq i \leq k-3\}.$$

3. Alice runs E_A through the streaming algorithm $\mathcal{A}$. Since it is an $o(n)$ space algorithm, when she is done there is $o(n)$ bits in memory. She tells Bob these bits. Note that this is just $o(n)$ bits.

4. Bob initializes the memory to those bits sent by Alice and then runs the streaming algorithm on E_B.

5. (Comment, not apt of the algorithm.) If $x_i = 1$ then the path $l_i - r_i - d_1 - \cdots - d_{k-3}$ is a connected component of size k. If $x_i = 0$ then the longest connected componenet is of size $k - 1$.

6. If when Bob finishes the streaming algorithm answer is YES then he knows that $x_i = 1$, if NO then he knows that $x_i = 0$. The total 1-way communication is $o(n)$.

■

3.2 Other lower bounds

Proofs are skipped for brevity's sake however we can show similar results for the following problems. Is-Tree(G) is the problem of determining whether a graph G is a tree; Perfect-Matching(G) is the problem of determining if G has a perfect matching; given a graph G and two vertices v, w, Shortest Path is the problem of finding the length of the shortest path from v to w.

Theorem 3. 1. *Any single-pass streaming algorithm solving* Is-Tree(G) *needs* $\Omega(n)$ *memory.*

2. *Any single pass streaming algorithm for* Perfect-Matching(G) *requires* $\Omega(m) = \Omega(n^2)$ *memory.*

3. *Any single pass streaming algorithm that approximates Shortest Path with a factor better than* $\frac{5}{3}$ *needs* $\Omega(n^2)$ *space. (So the algorithm produces a number that it* $< \frac{5}{3}\times$ *the length of the shortest path).*

4 Further results

1. [MV19] For Maximum Coverage, any approximation factor better than $(1 - (1 - 1/k)^k) \approx 1 - 1/e$ in constant passes requires $\Omega(m)$ space for constant k even if the algorithm is allowed unbounded processing time.

2. [EHL$^+$18] Gives an algorithm that, with high probability, estimates the size of a maximum matching within a constant factor using $\tilde{O}(n^{2/3})$ space, where n is the number of vertices.

3. [CGQ15] Gives deterministic and randomized single pass streaming algorithms that obtain a $\Omega(1/p)$ approximation using $O(k \log k)$ space where k is an upper bound on the cardinality of the desired set.

4. [CS14] Gives a $(4 + \epsilon)$ approximation algorithm for weighted graph matching which applies in the semistreaming, sliding window, and MapReduce models.

5. [SSS18] shows that coresets are composable and how to compute them in a streaming setting. Ultimately this leads to scalable algorithms for fair k-means clustering.

5 Chapter suggestions

1. p330 Theorem 18.4.3 uses a different definition of competitive ratio than what is given at the beginning of the chapter. I believe these are reciprocals of each other. A suggestion would be to stick with the convention being used in the Theorem since it seems to be most commonly used in current research.

2. p327: Under Yao's Lemma. The second enumerated item has $\max_{x \in A}$ but it should read $x \in X$. Same thing below in the lemma environment.

   (a) backwards quote right under the Lemma at the top of p328. Should have a "the worst input in . . . .

   (b) Suggested edit for the intuition of Yao's Lemma: cost under the worst input in $p \geq$ cost of the best deterministic algorithm w.r.t. p. Note the case used for the distribution p. Also, the brief explanation should mention we are comparing costs. Otherwise it isn't clear if the $\leq$ means less cost or better.

References

[AMS99] Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. *Journal of Computer and System Sciences*, 58(1):137–147, 1999.

[CGQ15] Chandra Chekuri, Shalmoli Gupta, and Kent Quanrud. Streaming algorithms for submodular function maximization. In *International Colloquium on Automata, Languages, and Programming*, pages 318–330. Springer, 2015.

[CS14] Michael Crouch and Daniel M Stubbs. Improved streaming algorithms for weighted matching, via unweighted matching. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2014)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2014.

[EHL+18] Hossein Esfandiari, Mohammadtaghi Hajiaghayi, Vahid Liaghat, Morteza Monemizadeh, and Krzysztof Onak. Streaming algorithms for estimating the matching size in planar graphs and beyond. *ACM Transactions on Algorithms (TALG)*, 14(4):1–23, 2018.

[KN97] Eyal Kushilevitz and Noam Nisan. Communication complexity. *Cambridge University Press*, 1997.

[MV19] Andrew McGregor and Hoa T Vu. Better streaming algorithms for the maximum coverage problem. *Theory of Computing Systems*, 63(7):1595–1619, 2019.

[SSS18] Melanie Schmidt, Chris Schwiegelshohn, and Christian Sohler. Fair coresets and streaming algorithms for fair k-means clustering. *arXiv preprint arXiv:1812.10854*, 2018.