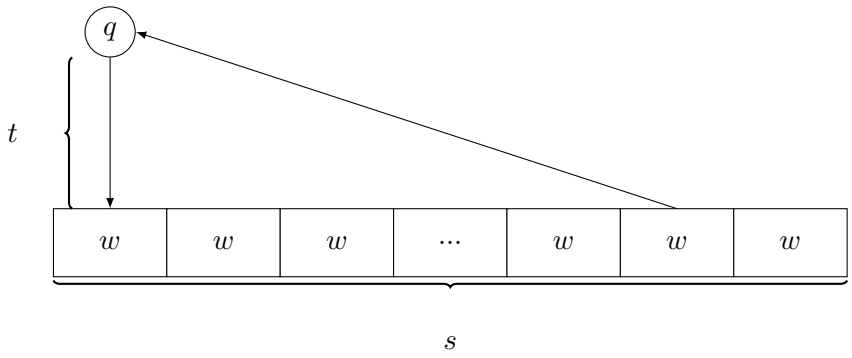
COMS E6998-9: Advanced Data Structures (Spring'19) Feb 21, 2019

Lecture 5: Data Structure Lower Bounds

Instructor: *Omri Weinstein* Scribes: *Jonathan Terry, Mia Brielle Saint Clair*

Introduction

Over the past two lectures, we have delved into the analyses of two data structures: van Emde Boas trees and Fusion Trees. Appropriately, we used the word RAM model to show their respective upper bounds. In this lecture, we switch gears and use the cell probe model to prove a lower bound for predecessor search on integer data structures. Recall that in the cell probe model, the only thing that can "slow down" a data structure is the number of reads and writes to main memory made by the CPU. Recall that in this model we consider a (s, t, w)-DS to use s space, with word size w requiring t reads or writes to memory in order to accomplish a predecessor search query q. Below is a diagram to help visualize the parameters we care about in this model.



In order to analyze the amount of information t transferred between the processor and RAM, communication complexity and the round elimination lemma are introduced to show the lower bound for an even easier problem: Colored Predecessor Search. In the course of the analysis, we see that our two favorite data structures - van Emde Boas trees and fusion trees - rear their heads once again as their core tenants of constant time universe reduction and set reduction (respectively) prove to be all that we can do in order to optimally solve the predecessor search problem. With this in mind, the entire goal of today's lecture is to prove the following lower bound:

Theorem 1. *Predecessor Search Lower Bound*

For all (s, t, w)-DS for $\text{Pred}_{n,w}$ *in the cell probe model using polynomial space $s = n^{O(1)}$:*

$$t = \Omega\left(\min\left\{ \frac{\log(w)}{\log(\log(n))}, \log_w(n) \right\} \right)$$

Introductory Remarks on Predecessor Search

Before delving into the meat of communication complexity, a few remarks need to be made about the bounds to be presented as well as the development of the bounds.

1. Although mentioned above, it bears repeating as it is a deep observation: there is a duality between vEB Trees and Fusion Trees. vEB Trees play this trick where they reduce the universe size in constant time and Fusion Trees play a trick where they reduce the size of the set being searched over in constant time. As the Predecessor Search Problem - $\text{Pred}_{n,w}$ - is parameterized only by word size and the set size, these are the only two tricks you can pull in order to perform a predecessor search.

2. See point 1!!

3. The stated lower bound is essentially useless when $w = O(\log(n))$ as this implies a universe polynomial in the number of items we store in the data structure i.e. $|U| = n^{O(1)}$.

4. The stated lower bound is essentially tight whenever $|U| = n^{\omega(1)} \rightarrow w = \omega(\log(n))$. In other words, we get a tight bound when we have a much larger universe than the number of items stored in the data structure.

5. Beam and Fich [BF99] proved a strong bound that depends purely on the word size w. Given just a tiny bit more than linear space $s = n^{1+\epsilon}$, then the query time is $\Omega\left(\frac{\log(w)}{\log(\log(w))}\right)$.

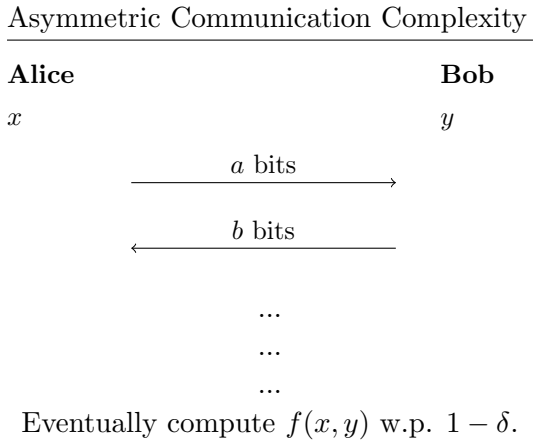
6. Patrascu and Thorup [PT06] showed an even better lower bound using the cell probe model, which stated that $t = \Omega\left(\frac{\log(w)}{\log(\frac{s}{n})+\log(w)}\right)$. One can note that for polynomial space, we return to the Beam and Fich bound. See their paper for a more sophisticated analysis.

Asymmetric Communication Complexity

With the basic ideas in mind and the stage set, we now need to establish the tool used to analyze how many memory accesses are needed in order to solve the Predecessor Search Problem $\text{Pred}_{n,w}$. In comes asymmetric communication complexity, which considers the problem of Alice and Bob attempting to work together in order to compute a function $f(x, y)$ subject to Alice holding x and Bob holding y with their communication governed by a protocol π. Aiding in their effort, Alice and Bob may have access to some shared randomness $z \sim \mathcal{D}_{shared}$. Furthermore, we may assume that their respective inputs are drawn from a known distribution $(x, y) \sim \mathcal{D}_{input}$. The asymmetry comes to play in that subject to the protocol π, Alice may be required to send fewer bits than Bob, or visa versa; additionally, one party is required to transmit information first. The protocol figure on the next page tries to capture what is going on in a diagram. Noting this setup, we can now intelligently define the communication complexity of computing a function.

Definition 2. *Communication Complexity of a Function* $[CC^{\delta}_{\mathcal{D}_{input}}(f)]$

The communication complexity of a function f with inputs $(x, y) \sim \mathcal{D}_{input}$ is the minimal protocol (and therefore number of bits) $\min ||\pi||$ *such that both parties can compute $f(x, y)$ with probability at least* $1-\delta$.



Complexity of Computing Equality (Exact and Approximate)

As a first endeavor into communication complexity, consider the equality function on n-bits, EQ_n, which has the following definition:

$$EQ_n(x,y) = \begin{cases} 0 & \text{for } x \neq y \\ 1 & \text{for } x = y \end{cases}$$

Now consider the scenario where we want to successfully compute the function every single time we attempt the computation. In this case $\delta = 0$, and as such Alice will need to send every single bit of x to Bob so that Bob can compute $EQ_n(x,y)$. Since we are not done until both parties have the result in hand, Bob must return a single bit to Alice so she knows whether or not the two numbers are equal. Therefore,

$$CC^0_{\mathcal{D}_{input}}(EQ_n) = n + 1$$

What if we wanted to admit a small amount of error ϵ in our equality computation? In this case, the parties could use their shared randomness $z \sim \mathcal{D}_{shared}$ in order to take a hash and transmit a smaller number of bits. Simply have Alice compute a single bit $\langle x, z \rangle$, send it over to Bob, and have Bob check the result against $\langle y, z \rangle$. Repeating this process for $\log(\delta^{-1})$ iid random drawings of $z \in \{0,1\}^n$, then upon comparing all $\log(\delta^{-1})$ results returns a single bit to Alice stating the approximate result of the computation. Therefore:

$$CC^\delta_{\mathcal{D}_{input}}(EQ^\epsilon_n) = 1 + \log\left(\frac{1}{\delta}\right)$$

Setting Up the Complexity of (Colored) Predecessor Search

Applying the above model to the Predecessor Search Problem, we should think of Alice as the program running on the CPU which has a query $x \in \{0,1\}^w$ to be executed, and Bob as main memory with y being the integer data structure holding n elements. Computing the predecessor amongst a CPU and RAM is an incarnation of the query-DB problem. As Alice wants to receive elements of the data structure in order to compute $f(x,y)$, then Alice will be sending $a = \log(s)$ bits per message. Likewise, since Bob merely has to return some element in the universe of possible elements, Bob will be required to send w bits per message. As Alice issues the query, she shall speak first. From here, we make a few key observations, and formalize the Colored Predecessor Search.

Definition 3. *Colored Predecessor Search* $\text{Pred}^{\oplus}_{n,w}(q)$

For any integer data structure, augment the integers with a color $c \in \{\text{red}, \text{blue}\}$. *For any query* q, *simply return the color* c *of the Predecessor* $\text{Pred}_{n,w}(q)$.

Observation 4. *In order to solve* $\text{Pred}^{\oplus}_{n,w}(q)$, *Alice still needs to send at most* $a = \log(s)$ *bits of information per query and Bob needs to send* $b = w$ *bits per answer.*

Observation 5. $\text{Pred}^{\oplus}_{n,w}(q)$ *is an easier problem than* $\text{Pred}_{n,w}(q)$

As the colored predecessor search only returns one of two colors, then naively one may guess the answer to the problem with a coin flip. That is to say, without any additional information, we can expect at most a $\delta = 0.5$ *error rate by simply having Alice guess the color at random.*

Observation 6. *If there exists a* $(s, t, w)_\delta$*-DS in the Cell Probe model for* $f(x, y) = \text{Pred}_y(x)$, *then there exists a cheap* $2t$*-round protocol* π *where* $a = \log(s)$, $b = w$, *and Alice speaks first. As a corollary, we simply need to prove a lower bound on the number of communication rounds in order to obtain a lower bound on the query time. That being said, Communication Complexity can never prove* $t \geq \frac{w}{\log(s)}$. *Additionally, this model gives Bob significantly more power than he really has as Bob, in reality, is just a table of cells.*

Proving the Lower Bound

At long last, we have the complete picture of communication complexity that we need to prove the lower bound for the Colored Predecessor Problem $\text{Pred}^{\oplus}_{n,w}(q)$. In the parlance of Communication Complexity, we seek to show the following bound for any $(t, a, b)_\delta$ protocol for $\text{Pred}^{\oplus}_{n,w}(q)$:

$$t \geq \Omega\big(\min\{\log_a(w), \log_b(n)\}\big)$$

You will note that so long as $a = \log(s)$ and $b = \log(w)$, we get the same bound as stated in the first theorem. In order to show this lower bound, we will use (without proof) the Round Elimination Lemma to show that we can eliminate all of the rounds of communication and be left with a $(0, a, b)_\delta$ protocol where $\delta < 1/2$. If this were the case, we could do better than random guessing without any extra information - a clear impossibility.

Round Elimination Lemma

Before we can actually describe the Round Elimination Lemma, we need to consider a certain decomposition of the mutual function f which the parties are trying to compute.

Definition 7. *Direct Sum Problem* f^k

Assume that Alice has k *i.i.d. inputs* $A_1, A_2, ..., A_k$ *and Bob has the entire dataset* B *in addition to an index* $I \in [k]$ *which only he knows. Still restricting the amount of communication per round to* $a = \log(s)$ *bits and* $b = w$ *bits for Alice and Bob respectively, the goal is to compute* $f(A_I, B)$ *when Alice speaks first.*

Here it is critical that Alice speaks first, otherwise Bob can simply supply the index I he wants and Alice can in return send the query that Bob wants to answer. Under these constraints, we can expect any given message to convey $\approx \frac{a}{k}$ bits of information about A_I. In the worst case, I is chosen at random, and without this message Bob would have to guess a/k bits of information. His probability of failure is therefore equal to $1 - 2^{-a/k}$, which is approximately a/k. In the regime that $a << k$, then Alice is effectively sending almost no information, and the first message can be dropped with only incurring a little bit extra error. In reality the extra error incurred is $O(\sqrt{a/k})$. This development leads us to the statement of the round elimination lemma!

Lemma 8. *Round Elimination Lemma*

If one has an $(m, a, b)_\delta$ protocol for the direct sum problem f^k where Alice speaks first, then there exists an $(m-1, a, b)_{\delta+O(\sqrt{\frac{a}{k}})}$ protocol to solve f where Bob speaks first.

With this in hand, we assume a communication protocol to solve f^k (in our case the one defined above) and repeat Round Elimination in such a way that we transmit no messages while still doing better than random (i.e. having a $\delta < 1/2$). As this is impossible, by quantifying the amount of information that would be transmitted without round elimination, we arrive at a measure of *at least* the amount of information needed to solve the problem, and thus a lower bound. One will note that there is an inherent asymmetry to Alice and Bob, and as such eliminating a message each way will require require slightly different techniques. As we will see, Bob will use a vEB-esque argument to justify dropping a message from from Alice, and Alice will use a Fusion-esque argument to justify dropping a message from from Bob. In the vernacular of the direct sum problem, the following two simultaneous relations will be exploited to deal with the asymmetry and to provide the desired lower bound:

$$\text{Pred}^{\oplus}_{n,w} \overset{\text{vEB}}{\approx} \left(\text{Pred}^{\oplus}_{n,w/k}\right)^k$$

$$\text{Pred}^{\oplus}_{n,w} \overset{\text{FT}}{\approx} \left(\text{Pred}^{\oplus}_{n/k,w}\right)^k$$

Eliminating Alice to Bob (Think vEB)

Alice's input to the function (which we denoted as x) has w' bits on a given round. Note that in general the size of the query she has in mind differs from the restriction on what she can transmit in a given round. In order to apply the Round Elimination lemma, she must split her input into k identically sized pieces $x_1, x_2, ..., x_k$. Since we want to incur $O(1/t)$ amount of extra error when dropping the message, and Alice is constrained to send a bits, then we should choose $k = O(at^2)$ to be able to drop the message an incur a small enough amount of error:

$$\delta' = \delta + O\left(\sqrt{\frac{a}{k}}\right) = \delta + O\left(\sqrt{\frac{a}{at^2}}\right)$$

$$\delta' = \delta + O\left(\frac{1}{t}\right)$$

Now lets frame the Alice to Bob message elimination from the perspective of Bob. On the receiving end, Bob has some $I \in [k]$ that he cares about unbeknownst to Alice, and the setting under which RE applies is when the goal is to compute $f(x_I, y)$. In the worst case, the *only* thing differentiating all of the n elements in his data structure lies in the Ith chunk of Alice's input. In this scenario, Alice can never hope to transmit that much information to solve the problem, and as such losing this message becomes admissible. Thinking about this worst case scenario as a tree in Bob's perspective, Bob would visualize a depth $O(at^2)$ tree with the elements in the data structure as the leaves. At the Ith level, we have *all* of the branching occurring - that is to say the remaining pieces of Alice's input $(x_1, x_2, ..., x_{I-1}, x_{I+1}, ..., x_k)$ do not help us compute the Colored Predecessor of x. At this point, the only real work to be done is to compute the predecessor at the node of x_I. If we lack x_I, then who really cares if we get the message or not? This observation justifies the round elimination from Alice to Bob.

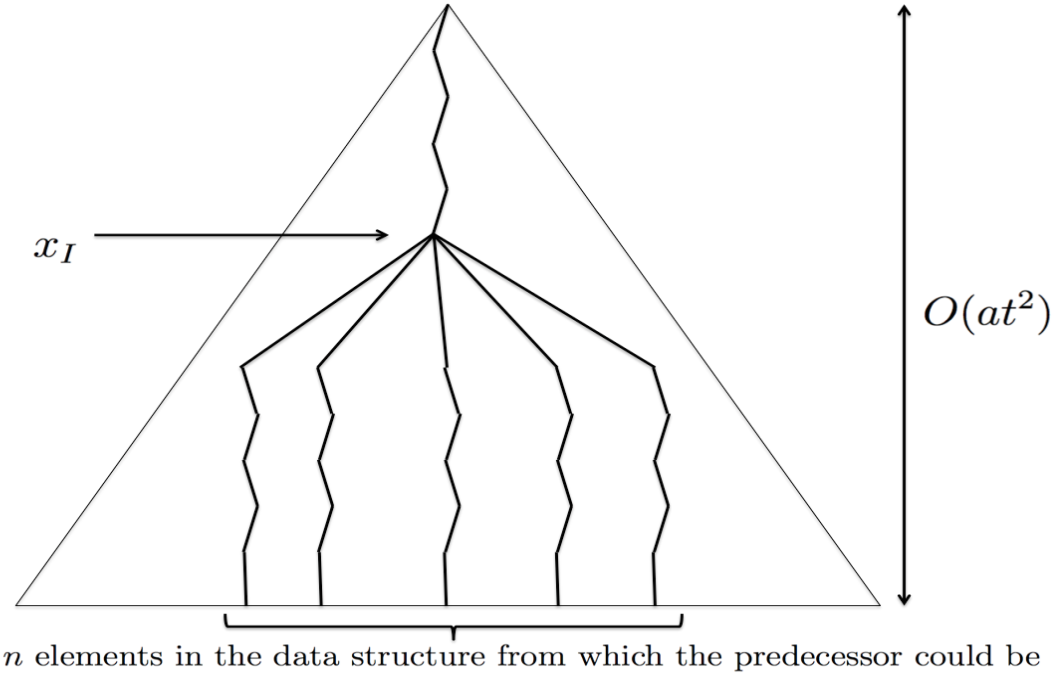


Figure 1: A visualization of what Bob needs to compute $f(x_I, y)$ under the Round Elimination Lemma in the worst case. Note that if Alice completely specifies the Ith chunk, then Bob can compute the function. If she does not, then Bob cannot compute anything.

We note that in this round, we effectively reduced the word size as we chopped up Alice's input and as we reduced the word size in an effort to compute a predecessor problem, this is analogous to the operation of a van Emde Boas tree:

$$w' \to O\left(\frac{w'}{at^2}\right)$$

Eliminating Bob to Alice (Think FT)

Bob's input to the function (which we denoted as y) has n' elements on a given round. In order to apply the Round Elimination lemma, he must split his input into k identically sized pieces $y_1, y_2, ..., y_k$. Since we want to incur $O(1/t)$ amount of extra error when dropping the message, and Bob is constrained to send b bits, then we should choose $k = O(bt^2)$ to be able to drop the message an incur a small enough amount of error:

$$\delta' = \delta + O\left(\sqrt{\frac{b}{k}}\right) = \delta + O\left(\sqrt{\frac{b}{bt^2}}\right)$$

$$\delta' = \delta + O\left(\frac{1}{t}\right)$$

Once again, let us frame this elimination from the point of the message recipient, which is Alice in this case. Alice knows the bits of her query x and as such has an *exact* idea of what can and cannot be a predecessor. For instance if $x = 0b01...$, then something of the form $0b01...$ or $0b00...$ may be a predecessor, while any element of the form $0b11...$ simply cannot be a predecessor. This is totally analogous to Bob's concept of needing the Ith chunk of bits to complete the computation! Instead of "zooming" in on the bits of the query that we need to differentiate elements in the data structure, we are now given the bits and instead need to zoom in on the elements contained in the data structure which could be candidate predecessors. With a sufficiently high number of candidate sub-trees within which the predecessor could reside (i.e. $O(bt^2)$), then we expect Bob's message to contain almost no information on how to solve the problem in the worst case, and as such can be dropped without much harm. Now, we can see that by eliminating this message, we reduce the number of elements we are searching over which is analogous to a Fusion Tree with $n' \rightarrow O(n'/(bt^2))$

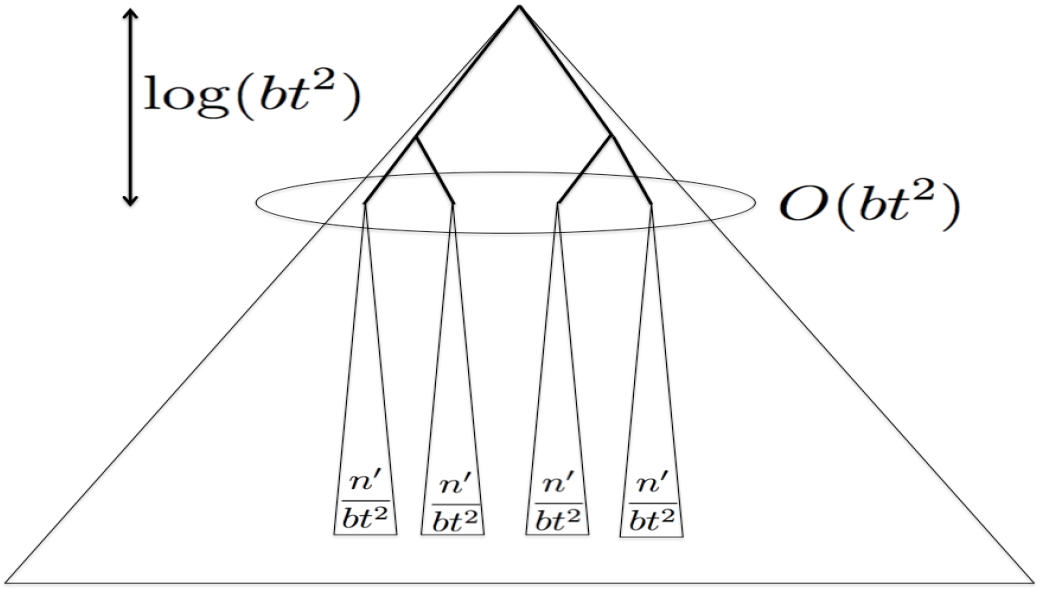


Figure 2: A visualization of what Alice needs to compute $f(x, y_I)$ under the Round Elimination Lemma. Note that if Bob completely specifies the correct subtree of potential predecessors (i.e. y_I), then Alice can compute the function. Since the data structure may be huge, even transmitting a summary is relatively hopeless and this is what allows his message to be dropped.

Completing the Proof

By applying the Round Elimination Lemma a total of $2t$ times with $k = \max\{at^2, bt^2\}$, then with each round (i.e. cell probe) we modify the parameters in the following way:

$$w' \to \frac{w'}{at^2} - \log(bt^2)$$

$$n' \to \frac{n'}{bt^2}$$

The algorithm terminates when there is sufficiently few data points to be searched over (i.e. $n' < 2$) or the universe size is less than that needed to uniquely address the any element (i.e. $w' < \log(n)$). As we divide by at^2 or bt^2 each time, then the number of cell probes required to complete the query is logarithmic and more over determined by which parameter hits the termination criterion the fastest. However, remember that in eliminating a total of $2t$ messages, we now change the probability of error to something upper bounded by a constant:

$$\delta \to \delta + 2t * O\left(\frac{1}{t}\right)$$

$$\delta \to \delta + O(1)$$

Therefore, for any set of integers and any query, if we chunk things up in the right way we can make that $O(1)$ term additive to delta such that $\delta + O(1) < 0.5$. As it is impossible to do better than a random coin toss to report the color of the predecessor, then this shows that we need at least

$$t > \Omega\big(\min\{\log_{at^2}(w), \log_{bt^2}(n)\}\big)$$

Noting that $at^2 < a^3$ and $bt^2 < b^3$, then we can see that (up to constants) we have proven the desired lower bound:

$$t > \Omega\big(\min\{\log_{a^3}(w), \log_{b^3}(n)\}\big)$$

$$t > \Omega\big(\min\{\log_{a}(w), \log_{b}(n)\}\big)$$

□

References

BF99 Paul Beame and Faith E. Fich. Optimal bounds for the predecessor problem and related problems. Journal of Computer and System Sciences, 65(1):3872, 2002. See also STOC99.

PT06 Mihai Patrascu, Mikkel Thorup. Time-space trade-offs for predecessor search. STOC 2006

D12 Erik Demaine. Lecture 13 of 6.851. https://courses.csail.mit.edu/6.851/fall17/lectures/