Recent advances in Complexity CIS 6930/CIS 4930 August 27th, 2002

Lecture 1

Lecturer: Dr. Meera Sitharam Scribe: Andrew Lomonosov

Schedule and instructions

The following schedule has been agreed upon.

Week	Topic	Theme	Student Presentation	Designated Scribe
Aug 27	Circuit Size (Lower Bounds)	P vs NP	None	Andrew
Sep 3	Circuit Size (Lower Bounds)	P vs NP	None	Andrew
Sep 10	Circuit Size (Lower Bounds)	P vs NP	None	Zia
Sep 17	Circuit Depth (Lower Bounds)	P vs NP	Zia	Zia
Sep 24	Circuit Depth (Lower Bounds)	P vs NP	None	Zia
Oct 1	Circuit Depth (Lower Bounds)	P vs NP	None	Erwin
Oct 8	Proof Complexity	NP vs Co-NP	Erwin	Erwin
Oct 15	Proof Complexity	NP vs Co-NP	None	Erwin
Oct 22	Pseudorandomness	P vs BPP	None	Henry
Nov 5	Derandomization	P vs BPP	None	Henry
Nov 12	Sampling	P vs BPP	None	Srijit
Nov 19	Physics Information Computation	Alt. models	Henry	Henry
Nov 26 (1/2)	Quantum Computation	Alt. models	Srijit	Srijit
Dec 3	Biogeometric Models of Computation	Alt. models	None	Srijit
Dec 10 (1/2)	Biogeometric Models of Computation	Alt. models	Andrew	Andrew

Deadlines for scribes: notes are due by the end of Thursday, and should be in Dr. Sitharam's mailbox on Friday morning. Coordinate by email with Dr. Sitharam on Friday morning, so that any corrections can be communicated to you over the weekend.

Deadlines for speakers: you should meet Dr. Sitharam on Monday, 3PM, 2 weeks before your talk to obtain papers and again next Monday to discuss your presentation.

Deadlines for turning in HW exercises: by 3 wks after day assigned.

What is this course about?

Complexity, Information, Randomness

Problem - no formal definitions exist for these three notions. Also, no (even informal) relationship between Complexity and Randomness is known.

Complexity of functions/sequences/processes

Example 1 *Consider growth of a tree leaf (self-similar fractals e.g). It can be represented as states $s_1, s_2, s_3, \ldots$ of a leaf at moments of time $1, 2, 3, \ldots$. This process P of leaf growth can be described by some program A. Intuitively one measure of complexity of P is the length of A (originally known as Kolmogorov complexity).*

Functions f that we consider in this course, map countably infinite set of finite structures (input $\mathcal{I}$) into another countably infinite set of finite structures (output $\mathcal{O}$).

E.g f could be from $\mathcal{I} = \{0,1\}^*$ (i.e the set of all binary numbers) to $\mathcal{O} = \{0,1\}^*$, or notationally, $f : \{0,1\}^* \to \{0,1\}^*$, or f could be from $\mathcal{I} =$ the set of graphs to $\mathcal{O} = \{0,1\}$, or f could be from $\mathcal{I} \subset N \to N$ etc.

Usually such functions f can be converted into similar functions $f' : N \to N$ by using bijections h and g, see Figure 1. Hence we can think of function f as a sequence of processes unfolding in time $f(1), f(2), \ldots$. But we have to be careful about complexity of such conversions (since properties of f' would depend on properties of f AND of h as well). In general, studying complexity of f' helps in studying complexity of f provided we control complexities of h and g (vice-versa h^{-1}, g^{-1}).

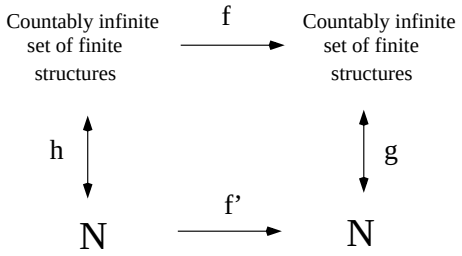

Figure 1: Standardizing functions

Example 2 *To show the existence of bijections such as h and g in this figure 1: how can we map $\mathcal{I} = N^3$ to $\mathcal{O} = N$? Answer - use diagonalization. Figure 2 illustrates diagonalization for mapping from $\mathcal{I} = N^2$ to $\mathcal{O} = N$ (N^3 to N is similar). Here we use bijection h mentioned above, where $h(1,0) = 1, h(0,1) = 2, h(0,2) = 3, h(1,1) = 4, h(2,0) = 5 \ldots$. Thus we can just take $f'(x) = x$ and now $f'(h(x)) = f(x)$. Note that this mapping h is very simple, it takes only a constant number of arithmetic gates to compute h.*

When we talk about complexity, we need to answer following question - *Complexity in terms of what?*. There are three components to it.

- First component is *input parameters*. If we have a function $f, f(\mathcal{I}) \to \mathcal{O}$ then complexity is in terms of the parameters of the input.
- Second component is *model of computation* (see below).
- Third component is robust *resource/model parameter*.

Example 3 *Consider a process of sorting n numbers that you have studied in your Algorithms class. The function f takes a sequence of numbers and maps it to another sequence of numbers. The standard input parameter is length of the*

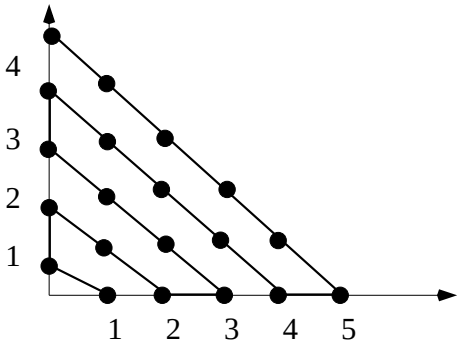

Figure 2: Diagonalization

input (or number of bits in the minimum representation of input). The standard model of computation that you have seen is Turing Machine (TM)/Random Access Machine (RAM) or the regular computer. The resource is time (required to complete execution).

Model of computation

Another possible model of computations is *circuits* (Shannon, 1948). In this case resource parameter is size of the circuit. Circuit is constructed from *circuit bases* - for example logical gates $\vee, \wedge, \neg$ or basis/primitive functions +,-,*,/.

For example, following function $f : N^3 \to N, f(x_1, x_2, x_3) = 2x_1^2x_2 + 3x_1^2x_2$ can be implemented by this circuit in Figure 3.

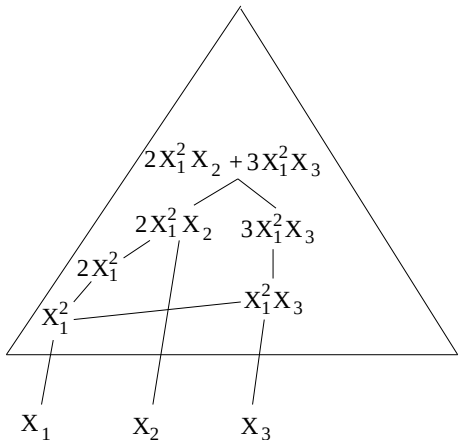

Figure 3: A circuit

One possible parameter of a circuit is it *size*, i.e number of basic gates. Another possible parameter is the *depth* of the circuit, i.e longest distance from root to a leaf, this parameter will be examined in greater detail in Lecture 2.

Circuit depth is related to

Communication complexity

defined as follows. There is a function $f : \{0,1\}^n \to \{0,1\}$. There are say 2 people who communicate with each other by exchanging bits. Communication

complexity of a function is how many bits need to be exchanged in order to compute f (the two people have unbounded computational resources).

f		y			
		00	01	10	11
x	00	0	1	0	0
	01	1	0	1	0
	10	0	0	1	0
	11	1	1	1	1

It is conjectured that communication complexity $C(f) \geq \Theta(\log rank(f))$, where $rank(f)$ is rank of matrix corresponding to f. Clearly $C(f) \leq O(\min(|x|, |y|)$, i.e log(number of rows/columns of matrix corresponding to f), since one player can simply send all her bits to 2^{nd} player who computes f).

Turns out that circuit depth is highly related to communication complexity. A few lectures will be spent on it.

Now we will properly define

Intrinsic complexity and complexity lower bounds

For a function $f : \mathcal{I} \to \mathcal{O}$, there might be several algorithms that compute f. For example algorithm A_f might take $O(n^2)$ RAM time to compute f. But a different algorithm A'_f might take only $O(n \log n)$ time. Which time should be defined as complexity of f then? This requires defining *intrinsic complexity* of f, which is the time complexity of the fastest algorithm for computing f.

Thus intrinsic complexity is sandwiched between times that are not sufficient for computing f (a *lower bound* of f - i.e there is no algorithm that computes f in this time) and times that are sufficient (an *upper bound* of f - i.e there is an algorithm that computes f in this time). See Figure 4.

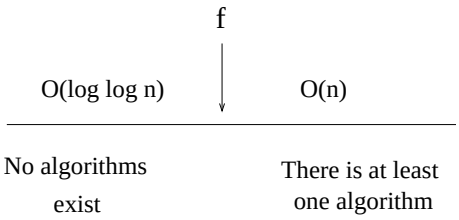

Figure 4: Intrinsic complexity

Proving lower bounds is difficult (need to show that there does not exist a faster algorithm). The process of proving good lower bounds *needs proving good upper bounds too,* i.e finding efficient algorithms and simulations. Typical example for lower bound proof needing upper bounds occurs when we are trying to prove a lower bound of f in model A. And there is another model B, such that it is easier to think about f in B than in A, so we can establish lower bound for f in B. But now complexity of the function h that transforms A into B comes into play. If we don't know how long h takes, but only have an inefficient (very large) upper bound on h then we will end up with inefficient (very small) lower bound on f. Hence in order to improve lower bound of f we need to improve upper bound on h.

The **FIRST MAJOR THEME OF THIS CLASS - i.e, THE QUESTION OF P VS NP** is a lower bound problem (listed in the clay math.

institutes top problems of the century, since we show that there are some problems in NP (all NP-complete problems for example) that cannot be solved in deterministic polynomial time. One particular approach to this theme involves proving circuit size and depth lower bounds, that we will examine in subsequent lectures.

THE SECOND THEME OF THIS CLASS IS NP VS CO-NP is also a lower bound question. This requires the notion of

Proof complexity

Consider a CLIQUE problem $\{(G,k) : \exists$ a clique of size at least k in graph $G\}$. If someone were to give us a possible solution (or a *witness*) of CLIQUE (i.e a subgraph A of size at least k), then this solution would be easy to verify - simply check whether all vertices of A are connected. The class of problems for which such verification is easy is called NP-problems. For a class of complementing problems no easy verification is obvious. Consider for example a CO-CLIQUE problem, i.e given a graph G determine that G does not contain any cliques of size at least k. No obvious witness is apparent.

There are

Other major uses of lower bounds, besides the separation of complexity classes such as P and NP or NP vs. co-NP.

1. The first use is for algorithm design. Any algorithm designer seeking to optimize her/his algorithm will effectively go through the motions of a lower bound proof, if somewhat informally: for example, once a candidate algorithm has been designed, the designer will act as ones own adversary who claims there is a better (less complex) algorithm; then beat the adversary by convincing oneself that for any supposedly better algorithm, there would be some input on which this algorithm would make a mistake.

2. The second use arises in the quantum/Biogeometric models of computation that will be studied as the FINAL THEME in this class. Note that the intrinsic complexity of f is a jump step point, i.e for shorter times f cannot be implemented at all and for any longer time f could be implemented, and there are no intermediate states. Studying the smallest change δ that is needed to transform circuit that cannot compute f into the one that can compute f, requires knowing a formal lower bound proof for f. Such intrinsic knowledge is essential in biotech and molecular engineering and modeling of biological phenomena (theory of evolution by jerks and creeps). Open question discussed towards the end of the class - how can Nature solve problems much faster than our currently standard computational models can?

3. Another use for lower bounds arises in cryptography **THIS USE WAS NOT MENTIONED IN CLASS**.

   Complexity lower bound proofs are needed to establish the hardness of breaking a code or security protocol. I.e, the cryptographic application. A complexity lower bound proof is needed to GUARANTEE that a malicious adversary would need at least a supercomputer running for 2 weeks to break your code.

4. Last use of proving lower bounds on complexity, i.e establishing hardness, is for connecting complexity and randomness (this is **THIRD THEME OF CLASS** i.e pseudorandomness, derandomization etc. See below).

Pseudorandomness and Derandomization

As we have mentioned above the third theme of this class is "Is randomization necessary for computation", i.e $PvsZPP$ or $PvsBPP$. Until VERY recently the algorithm for testing primality of integers (i.e solving PRIMES problem) was a *randomized* polytime algorithm (so-called Las Vegas algorithm) which randomly sampled potential factors to establish composition of a number, thus putting PRIMES in class ZPP ($ZPP = RP \cap Co - RP$, definition should be familiar to people who were in Theory of Computations class). Recently derandomized polynomial time algorithm for PRIMES was constructed (see http://www.cse.iitk.ac.in/news/primality.html), thus placing PRIMES in P.