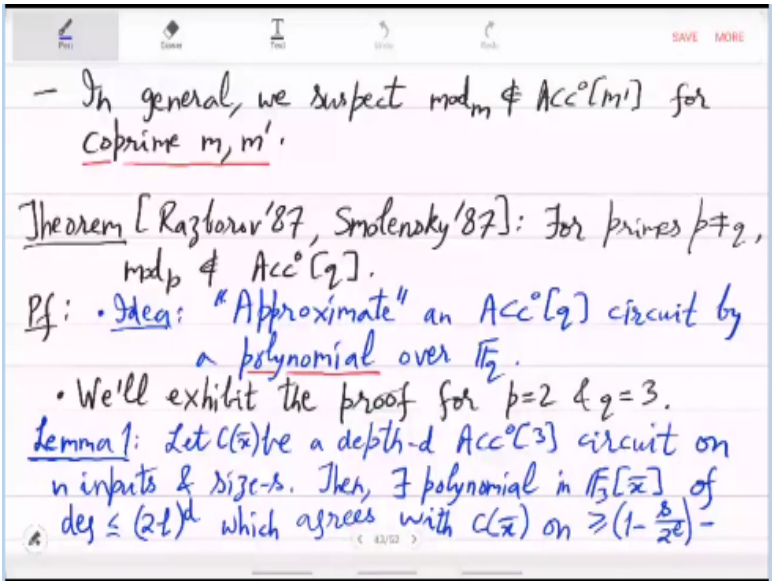
Randomized Methods in Complexity
Prof. Nitin Saxena
Department of Computer Science & Engineering
Indian Institute of Technology - Kanpur

Lecture - 07
Monotone Circuits

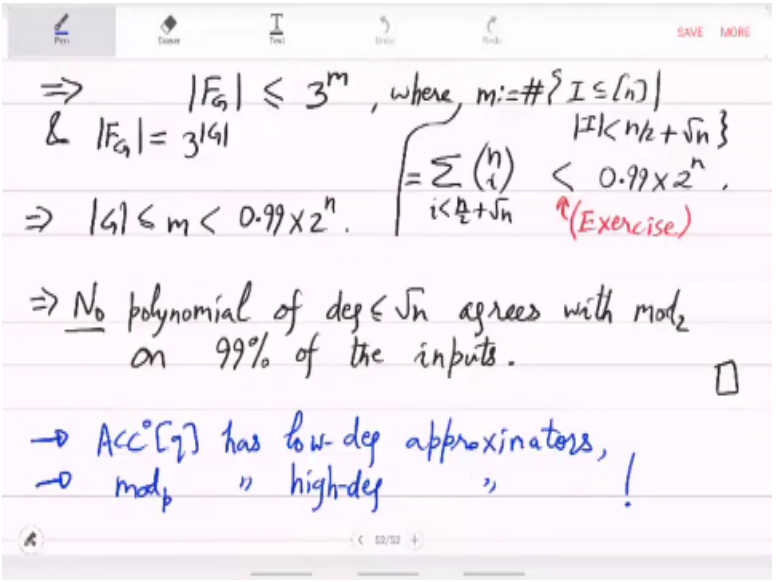
(Refer Slide Time: 00:13)



Last time, we finished the proof of the theorem due to Razborov and Smolensky showing that if you look at constant depth Boolean circuits, ACC^0 together with mod_q gates, then to compute mod_p function, you will require exponential size. Constant depth would need exponential size.

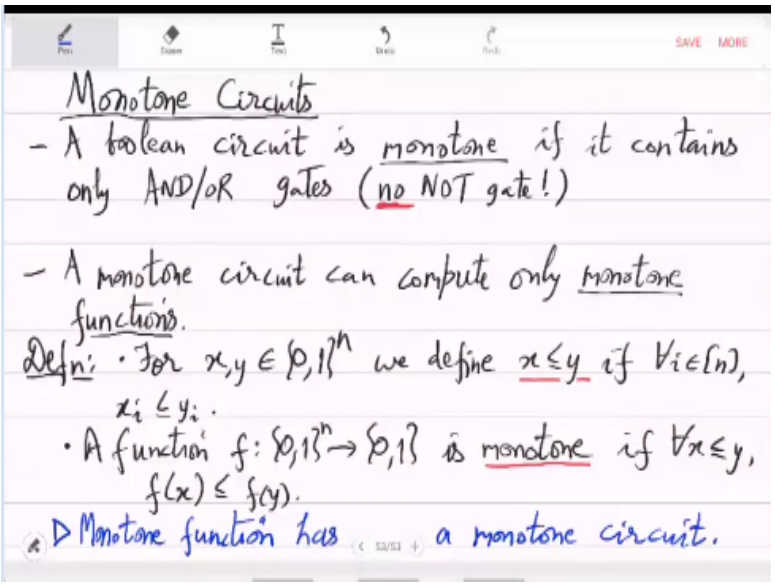
Note that if you allow more than constant depth, then in log depth you can compute it efficiently in polynomial size but in constant depth you will require exponential size. The tools used in the proof were very interesting and new.

(Refer Slide Time: 01:12)



We learned that $ACC^0[q]$ has low degree approximators which are polynomials and we learned that mod_p has high degree approximators. It does not have low degree approximators.

(Refer Slide Time: 01:34)



Now we will move to the next topic which is again a lower bound for Boolean circuits. This will be for the model of Monotone Circuits. It is a Boolean circuit which contains only AND/OR gates. So, monotone means that there is no NOT gate. These circuits in the absence of NOT gate are called monotone circuits.

A monotone circuit can compute only monotone functions. A monotone function as the name suggests, means that if you increase the input, then the output can only increase. Here increase means that if in the input you make any bit from 0 to 1 then in the output it cannot happen that 1 becomes 0. So, 0 can become 1 but 1 cannot become 0.

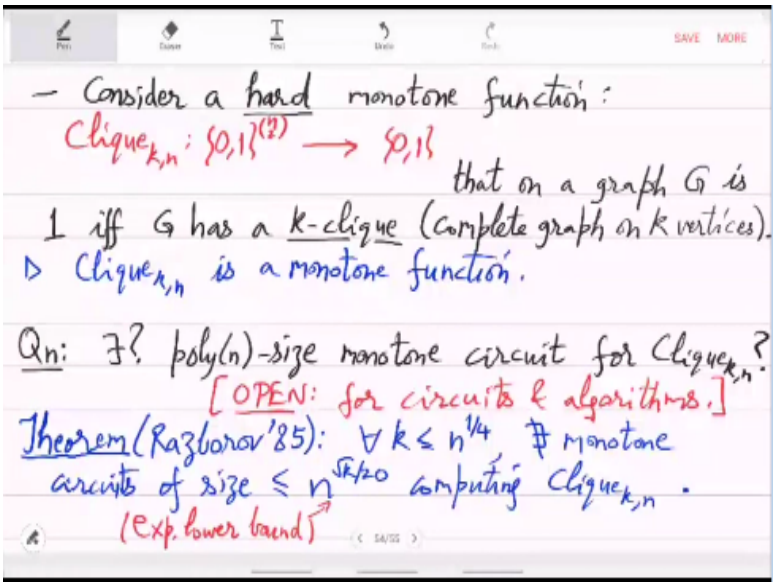
Monotone means that it is increasing in one direction or it is decreasing. If you decrease the input, then the output cannot increase and if you increase the input, then the output cannot

decrease. There is a relationship between input and output. This is because there is no NOT gate. So, for n-bit strings x and y, we define $x \leq y$ if for every bit i-th bit $x_i \leq y_i$. This defines what we mean by $\leq$ for strings.

A n-bit Boolean function is called monotone if for every $x \leq y$ the same relationship holds for the output value. Then we say that it is monotone. You can show that monotone functions always have monotone circuits but in general the circuits will be very large but they do exist that you can show.

In other words, all the monotone functions can be expressed as monotone circuits but the representation may be large. For n-bits you might require 2^n gates, but it exists. On the other hand, non-monotone functions cannot have monotone circuit. This is actually an if and only if condition.

(Refer Slide Time: 06:48)



Let us think of a monotone function, which seems hard. Consider a hard monotone function and a good example is this NP-hard problem called k-clique. This function takes $\binom{n}{2}$ bits in the inputs and output a single bit that based on a graph G is 1 if and only if G has a k-clique. Where k-clique means that there is a complete graph on k vertices. If you remember your theory of computation or algorithms course, k-clique is supposed to be an NP-hard problem.

It is supposed to be very difficult to solve practically. Here k is something which is also growing. The Brute-force algorithm on this graph will be to look at all possible subsets of k vertices, which will take time $\binom{n}{k} \sim n^k$. We believe clique is a hard function.

It is believed to be hard in general or for an algorithm, then you will also believe that the circuits also will be large size and hence you will believe that the monotone circuits will also be of large size. The important point why I took this example is that clique is a monotone function.

Why is that? Well! Because in a graph which is the input if you add more bits in the input or if you add more vertices or edges in the input, then it cannot happen that a clique which existed before a k-clique disappears. The clique can only improve, it cannot disappear by addition of more edges or vertices. This is a monotone function and the question is, does there exist a polynomial sized monotone circuit for clique?

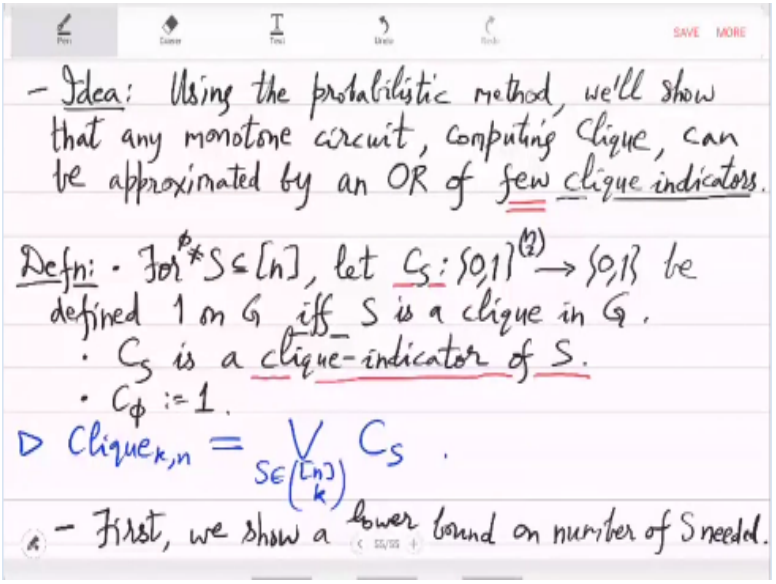
This is a question which we will answer negatively. That monotone circuit complexity for clique is indeed exponential. But the same question is open for Boolean circuits and for algorithms. We will basically prove this theorem by Razborov. Pick your k to be a variable function of n, say, $n^{1/4}$. Then there does not exist monotone circuit of size smaller than $n^{\sqrt{k}/20}$ computing k-clique.

That is the theorem. That monotone circuits for clique problem have to be of size $n^{\sqrt{k}}$, which is very large. This is something like $n^{n^{1/8}}$ which is what we call exponential size. It is an exponential lower bound.

So, how will we show this? This seems to be a pretty strong result. We have to understand two things. We have to understand properties of monotone circuits which we can use to somehow show that monotone circuits have a weakness. And then at the same time, we also have to show that clique is a hard function. And then we have to match the two things.

We have to show that monotone circuit is a weak model and we have to show that clique is a strong model and then we have to compare the parameters of the two. What is that parameter or what is that potential function that will give you a separation.

(Refer Slide Time: 14:19)



We will use here the probabilistic model like we used in ACC^0. Using the probabilistic method, we will show that any monotone circuit computing clique can be approximated. Like in the previous result, we showed that they can be approximated by polynomials low

degree polynomials. Here, we will show that they can be approximated by an OR of few clique indicators.

Instead of polynomial, we will use something else here. We will remain in the Boolean world or in the world of propositional formulas. It can be approximated by an OR of few clique indicators. Clique indicators will be simple functions of low circuit complexity and they will be few or a few. Overall this monotone circuit model somehow will imply a weakness of clique.

On the other hand, will also show that clique is actually not weak. When you match the two, then you will get the lower bound. But, let us first define what is a clique indicator. For a subset of vertices, vertices are 1 to n. For a subset of the vertices, let clique indicator C_s be a Boolean function which takes a graph edges, $\left(\frac{n}{2}\right)$. Also you can assume it to be an undirected graph – with no self-loop.

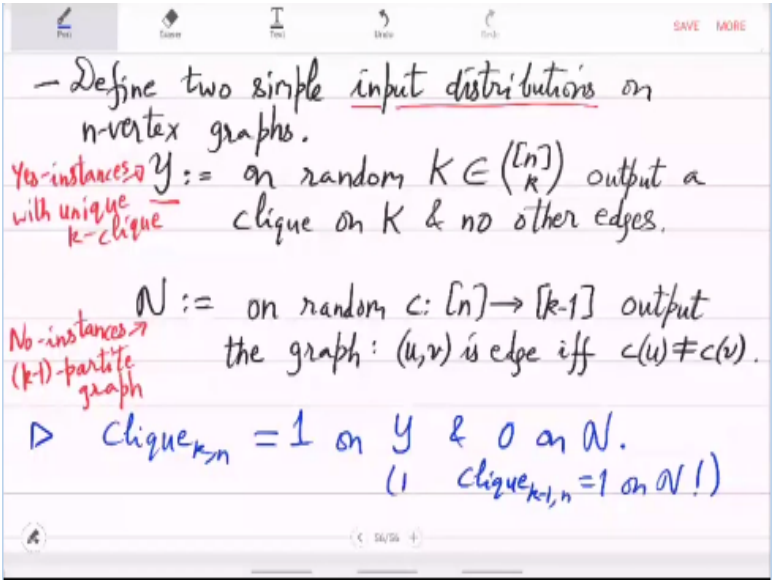
So, the edge vector is given to you defines a graph uniquely. Now, in the graph you test whether the vertices S form a clique. So, 1 on G if and only if S is a clique in G. Just test this and this is called, C_s is called clique indicator of S and as a notation C_Φ, for the empty set when you don't take anything, you assume it to be a clique of size 0.

Conventionally we can set C_Φ to 1 and for all the other non-empty subsets, check whether there is a clique on the vertices defined by the subset S. What is immediate from this is that the clique function, this is an OR of clique indicators. Go over all the subsets of size k. This follows from the definition of clique because $clique_{k,n}$ will be 1 if and only if there is a k-clique.

If there is a k-clique, then there would be a subset of k vertices and that would be S. So, C_s will be 1 and it is an OR. If one of these C_s is 1 the answer is 1. This is what was meant by OR of clique indicators and the weakness of the monotone model would mean that to show that clique actually can be approximated by few of these C_s; not all C s are needed, just few will be enough.

First we show a lower bound on the number of clique indicators on the number of S needed to compute clique. In fact, we will use a simpler input than a general graph. We will use yes instances to be very simple graphs or very simple input instances. They will basically just be randomly chosen k-cliques.

(Refer Slide Time: 21:46)



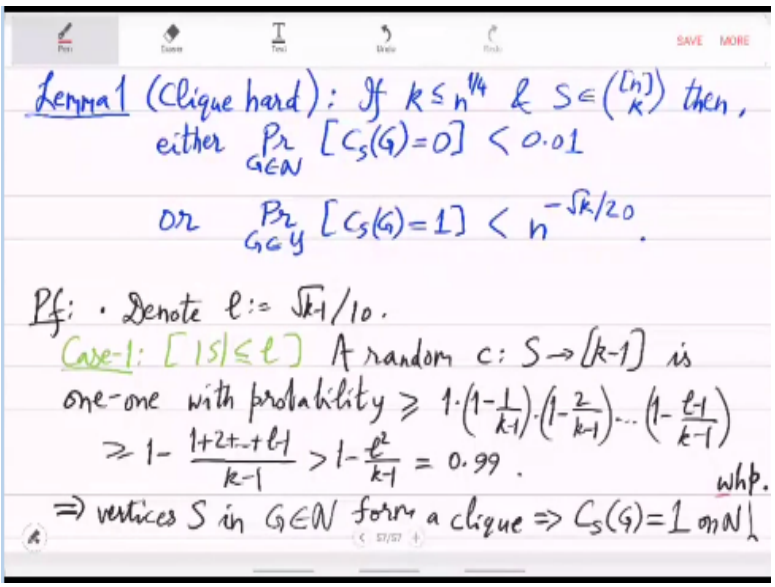
So, let us define two simple distributions on n vertex graphs. First is for the yes instances of input. This distribution is on a random subset big K of size small k. So, on a random subset output a clique and no other edges. Yes instances with unique k-clique. For the no instances you want k-clique to be absent.

How do you avoid that? This is non-trivial. It is a more clever construction than the first one or more clever definition. What you do is on a random function that is k - 1 valued on the vertices, randomly assign a label 1 to k-1 to every vertex. Now you have basically clustered the vertices in k - 1 clusters in a random way. And what is the graph that you define?

So, u, v is an edge of the graph if and only if the labels are different. Basically you have these k - 1 clusters to join across. This is a k - 1 multipartite or k - 1 partite graph. These are no instances. You are just outputting a random k - 1 partite graph. Can it have a k-clique? Well! It cannot have a k-clique, because there are only k - 1 parts but it has a lot of k - 1 cliques.

That if you pick vertices one each from each of the k - 1 parts, then all possible edges are present. You can immediately see that clique function is 1 on the yes instances and 0 on the no instances. On the yes instances, there is always a k-clique; and on the no instances there is never a k-clique but there are k - 1 cliques. These are your yes and no instances. Let us first show that clique is hard.

(Refer Slide Time: 27:09)



Basically what we will show is that clique requires many indicators. It will suffice to show that if you just use one indicator, then the yes instances with high probability will fail, which means that your indicator will output 0 on yes instances and the opposite thing on the no instances. On most of the no instances, your clique indicator will say yes. That means, the answers will be wrong with high probability.

This is what we want to show, which will mean that you need many clique indicators even to work correctly on yes and no distributions. So, if k is let us say $n^{1/4}$ and S is a subset k size, then either the probability on no instances of being 0, which is the correct answer, that probability is quite small or on the yes instances, saying one, which is the right answer is extremely small.

Either on the no instances, there will be mistake but this mistake is 99%. That is on 99% or 0.99 fraction of the no instances, the answer will be 1. This mistake is more than 99% but in the yes instances the probability is exponentially small. Almost everywhere it is making a mistake. Second thing is a bigger problem, which shows that clique is actually hard.

To show this, we will use some probability. We will use a parameter $l = \sqrt{k-1}/10$. Suppose the set S is smaller than l. See either the set S is of size less than l or more than l. When it is less than l, it will make mistake on one side; when it is more than l it will make mistake on the other side. For example, when the set S is of size less than l, it is a small set, then what will happen is then we have to look at the no instance distribution.

In that distribution, although the k-clique is absent since there are many k - 1 cliques. This indicator will usually answer 1 because the set is small. When the set is large, then we will show that mistake happens in the other side. Remember that S defines your clique indicator. So, if you take a small set, then you will have these false positives, say and if you take your indicator to be based on S, then you will have the opposite problem.

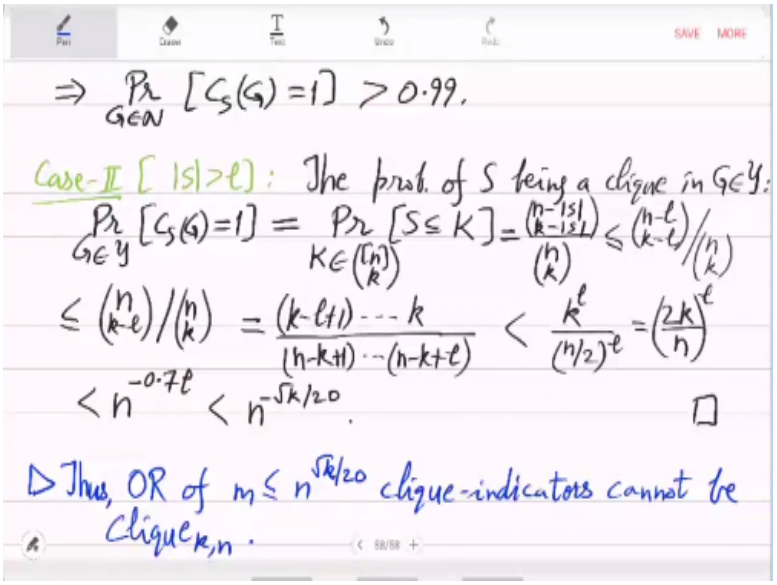
We will do it step by step. Let us first handle this small s case. So, what happens when the set is small or your indicator is small is that a random C from S → k - 1 will be one to one

with high probability. The first element in S, you can freely label but then the second one, you cannot use the label which you used in the first vertex.

That will be $(1 - (1/k - 1)) \cdot (1 - (2/k - 1)) \cdots (1 - (l - 1/k - 1))$. This is at least $1 - (1 + 2 + \cdots + l - 1)/k - 1$ which is at least $1 - \left(l^2/k - 1\right)$. Here $l^2/k - 1$ by definition we have picked it to be 1/100. So, we get 0.99. Almost always your random labeling will keep S distinct. The labels will be distinct, which means that in the (k – 1) partite graph, you will see a clique on S.

Your answer will be 1 but there was actually no k-clique. So, your answer for 99 percent of the cases, is incorrect. Vertices S in the no instances form a clique, which means that the clique indicator will answer 1 with high probability on no instances that is one case.

(Refer Slide Time: 35:14)



Let us go to the next case when the set is large. Basically you have chosen a big indicator. Now what will happen is, you might be missing the yes instances because you have chosen a bigger indicator. Let us calculate the probability of S being a clique in the yes instances. S will be a clique only when this graph of in the yes input distribution, it is contained in the clique.

This G is already a k-clique. S has to be a subset of that if you want to see a 1 in the answer. This is the same as the probability that S is contained in this subset K, the way G was defined. Graph G was defined with respect to some subset K of size k, and S should be contained in that subset of that the indicator output 1. And what is that probability?

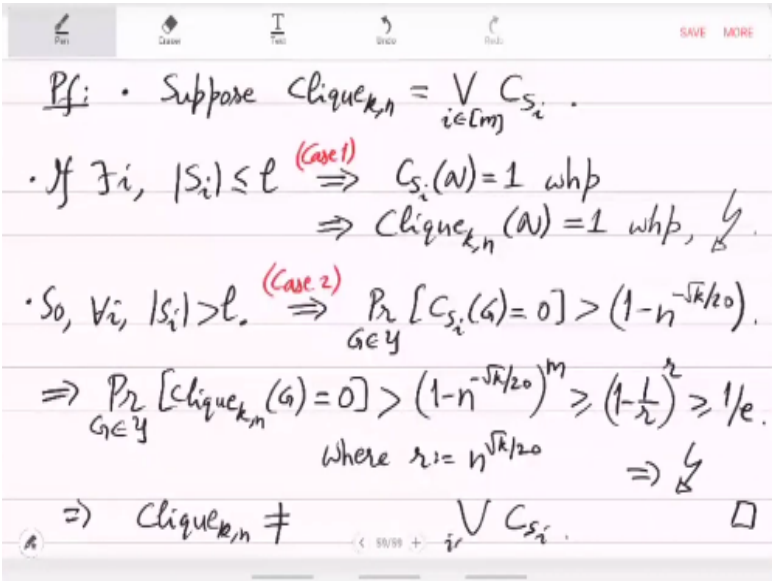
We can first give the exact probability, which is you want to pick a random subset K but you also have to put big S in it. That is total number of choices is $\left(\frac{n}{k}\right)$ and favourable ones are $\left(\frac{n-|S|}{k-|S|}\right)$, which is less than equal to $\left(\frac{n-l}{k-l}\right)/\left(\frac{n}{k}\right)$, which again is less than equal to $\left(\frac{n}{k-k}\right)/\left(\frac{n}{k}\right)$ and that is exactly the binomial coefficient expansion $((k - l + 1) \cdots k)/((n - k + 1) \cdots (n - k + l))$.

Now, all the numbers dividing the numerator are at most k. So, you get k^l as an upper bound. All the numbers that appear in the denominator, they are at least n/2. So, you get $(2k/n)^l$. Further, k/n is smaller than $n^{\{-0.7l)}$.

Recall that l is $\sqrt{k}/10$. You get smaller than$-\sqrt{k}/20$. In case-2, you get success probability or the error probability on yes instances to be very large. In case-1, you get error probability to be large on no instances. You can see the probabilities match our claim.

We will actually use the lemma 1 to show that if you just take $n^{\sqrt{k}/20}$ or fewer than that indicators, it cannot be clique. This almost immediately follows but, let us just do the formality.

(Refer Slide Time: 42:02)



First you write clique as this OR of clique indicators I from 1 to m. And if there is an i here, such that size of S_i subset is smaller than that l then the case-1 applies. If there is an i such that set S_i is smaller than l, then your case-1 will give you problem in the no instance. Then C_{S_i} on the no instance is 1 with high probability, which means that clique function is also 1 with high probability. Because clique is an OR of these C_{S_i}'s.

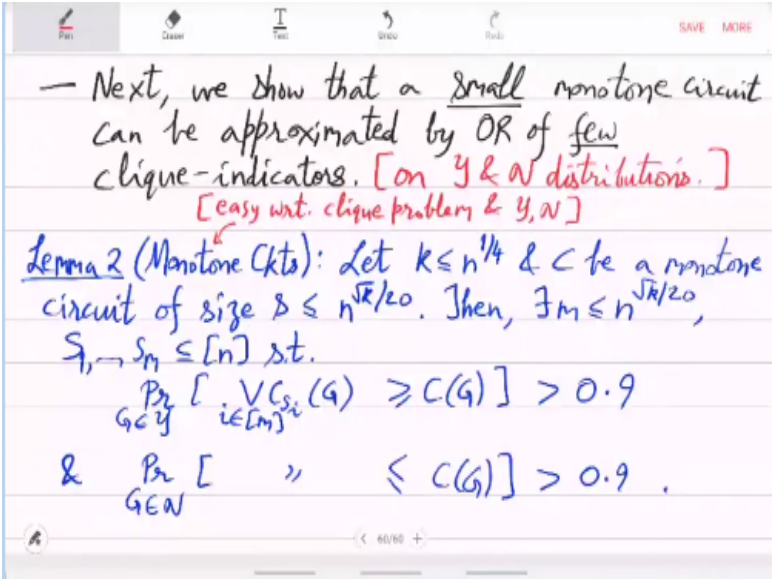
If one of them is 1, the whole thing is 1. That is a contradiction. This cannot happen. Let us look at the other case. So, for all i, the subsets are large, and since the indicators are large you can apply case-2 on the yes instances. There is a problem in the yes instance. The probability on the yes instances that C_{S_i} is 1 or a 0. This probability we calculated to be quite high.

We had $1 - n^{\sqrt{k}/20}$as the probability, which means that probability on these graphs of your indicator being 0. Clique function is 0 when all these indicators output 0, each indicator output 0 with this much of probability. So, you multiply them. This is $(1 - 1/r)^r$ where

$r = n^{\sqrt{k}/20}$ and it is more than $1/e$. Here m was picked to be smaller than that equal to that actually.

With a decent probability on the yes instances, you are getting 0, which means that it cannot be clique function. That is also a contradiction. This finishes the proof. It means that $\text{Clique}_{k,n}$ is not C_{S_i}. We have shown that if you use only $n^{\sqrt{k}}$ clique indicators, then you will not be able to compute clique exactly. It is a statement about exact computation. Remember it is not approximation at this point.

(Refer Slide Time: 47:04)



Next lemma that we want to show is to do with weakness of the monotone circuit model with respect to clique indicators. We show that a small monotone circuit can be approximated by an OR of few clique indicators. Here, we will talk about approximation and we will show something opposite to what we showed for the clique function that even few indicators fewer than $n^{\sqrt{k}/20}$ that we had before, fewer than that many clique indicators are enough to approximate a monotone circuit.

By the way, this will not be on all input instances. This will be on the same input instance distribution as in lemma 1. This actually result is not strong enough to work for all inputs. I should remark that for distributions yes and no, this is important. Let us compute the probabilities on these distributions and also we have to specify which clique indicators will work.

Lemma 1 was about clique hard. This is now about monotone circuits. This will show that again same $k \leq n^{1/4}$ and size you can think of as $n^{\sqrt{k}/20}$; same parameters. Then there exists m subsets or clique indicators where m is smaller than $n^{\sqrt{k}/20}$.

Subsets S_1 to S_m of 1 to n such that we will now talk about success. We want to show that these indicators will well approximate the value of monotone circuit. Actually the probability now is large of success. On the yes instances when you look at this OR of clique indicators, these m clique indicators, it is actually.

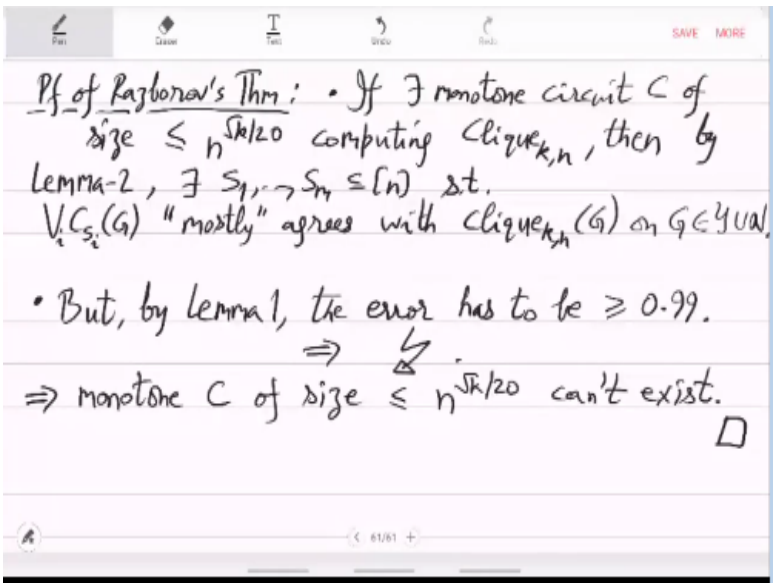
So, on the yes instances, it is actually 1 or it is whatever the circuit was outputting, the value of the circuit monotone circuit C on this G. It may be outputting 0, or 1 but whatever it was outputting the OR of the indicators is at least that much. It matches the answer of C with high probability on most of the yes instances according to the distribution and symmetric thing on the no input distribution.

For no instances this indicator is at most the value of the circuit. Iff the value was 0, this is also 0 with probability 0.9. If the value was 1, then the OR of indicator can be 0 or 1 but the point is that it matches the monotone circuit C value. On the yes instance in 90 percent of the cases, it is. So, if C(G) is 1, then this or of indicators will be 1. If C(G) was 0, then then it can be 0 or 1.

When it is 0, it is at least matching C(G). If it is 1, then it is matching the yes instance, the correct answer. Either it is the correct answer or it matches C(G) with high probability. The point is that with respect to clique, we have shown that monotone circuits are approximable. So it is easy with respect to clique distributions that is what this lemma is saying, it is not a general result for monotone circuits. It is only for the clique problem.

The function that we are interested in is the clique function and the input that we are interested in is only yes and no instances. Only then this this thing makes sense. First prove the main theorem using lemma 1 lemma 2.

(Refer Slide Time: 55:20)



Proof of Razborov's theorem, you see that if there exists a monotone circuit C of size smaller than what was claimed in the theorem statement computing clique, then by lemma 2 there will be clique indicators such that the OR of them mostly agrees with the clique function for

the yes and no instances, because lemma 2 talks about both. Graphs both in the yes and the no input distributions, the clique function is being well approximated.

For 90% of the cases it is correct with this OR of clique indicator. This is the approximation result you get for clique by lemma 2 but. Now in lemma 1, it was shown that whenever OR of indicators agrees with clique, there has to be a lot of indicators. Compare with this probability that you got error probability of 1%. The match cannot be more than 1% that is what lemma 1 said.

But by lemma 1, the error is at least 99%, which implies a contradiction. Lemma 1 and lemma 2 contradict. So, lemma 2 is saying that m clique indicators are agreeing with clique on 90% of the cases. On the other hand, lemma 1 is saying that they can agree on at most 1% of the cases. t is a big contradiction, which means that monotone circuit of this much size cannot exist.

That is the impossibility result. If once we have shown lemma 2, then lemma 1 and lemma 2 together prove Razborov's theorem. Next time we will do lemma 2.