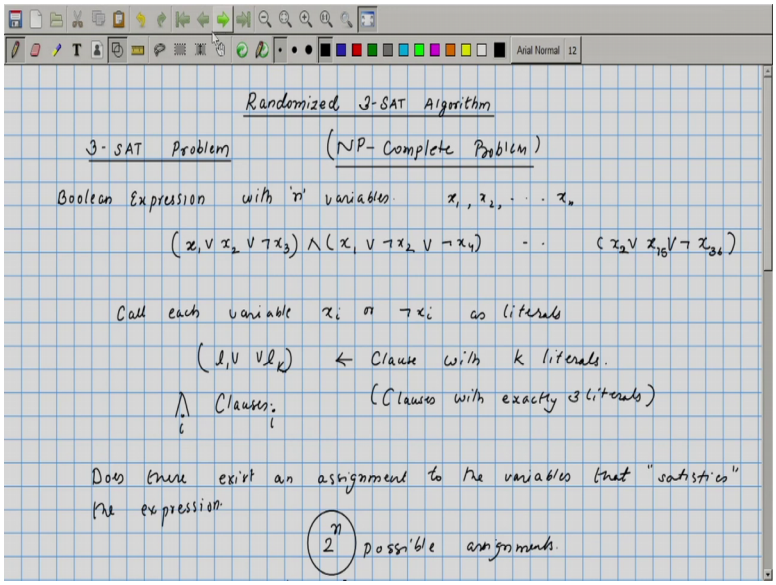
Randomized Algorithms
Prof. Benny George Kenkireth
Department of Computer Science & Engineering
Indian Institute of Technology, Guwahati

Lecture – 18
3-SAT and Markov Chains

(Refer Slide Time: 00:30)



So, in this lecture, we will learn about an algorithm for the 3-SAT problem. So, let us understand what this problem is. So, we are given a Boolean expression with n variables ok. So, this Boolean expression is in a particular format. So, let me call the variables as x 1, x 2, dot dot dot, x n. Now, I can combine these variables and get a Boolean expression, so this kind ok.

So, in this, this expression is an AND of OR's. So, we will call each variable x i or its negation not x i as literals ok. And then we can combine them by OR ok, so take the OR of two or more literals. So, if l 1, l k, r literals, if you take OR of them that is going to be called as a clause with k literals and OR formula is going to be a special type of formula, where we will take the AND of clauses ok. So, if you denote each clause by i, so you are taking an AND of these clauses that is the formula that we are Boolean expression that we are interested in.

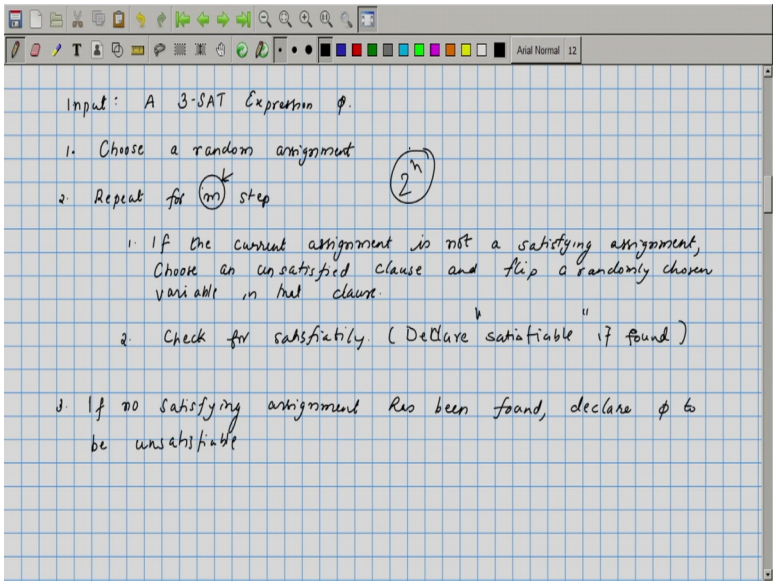
Now, the question is does there exist an assignment to a Boolean assignment to the literals to the variables that satisfies the expression, satisfies means when you evaluate

the Boolean expression, you should get 1 ok. And we will look at even special kind of expressions, where each clause will have at more at most 3 literals or clauses with exactly 3 literals ok. So, this is the kind of problem that this is the exact problem that we are working with.

We have a Boolean expression which is an AND of clauses. And each clause is an OR of exactly 3 literals. And we want to know whether there is a Boolean assignment to the variables, which will make all clauses simultaneously true. When all clauses are simultaneously true, by some particular assignment we will call that as a satisfying assignment ok. So, this is the problem that we are interested in.

Clearly there are 2 raise to n possible assignments. So, one way would be try out all the 2 raise to n possible assignments, but that will be prohibitively expensive. So, we want to design an algorithm, which works faster than just try all possible assignments. We will do we will construct a randomized algorithm, which will do that. It will still be an exponential time algorithm; this problem is known to be an NP-complete problem ok. You can just understand that as this is a problem which is known to be very, very difficult to solve. And people do not expect that this algorithm will have a polynomial this problem will have a polynomial time algorithm.

(Refer Slide Time: 05:46)



So, we will design a randomized algorithm on the lines of a similar randomized algorithm for 2-SAT. So, 2-SAT is the version where each clause has exactly 2 literals

that problem is known to be polynomial time solvable. But, we will design the algorithm for 3-SAT the randomized algorithm for 3-SAT on the lines of a similar randomized algorithm for 2-SAT ok.

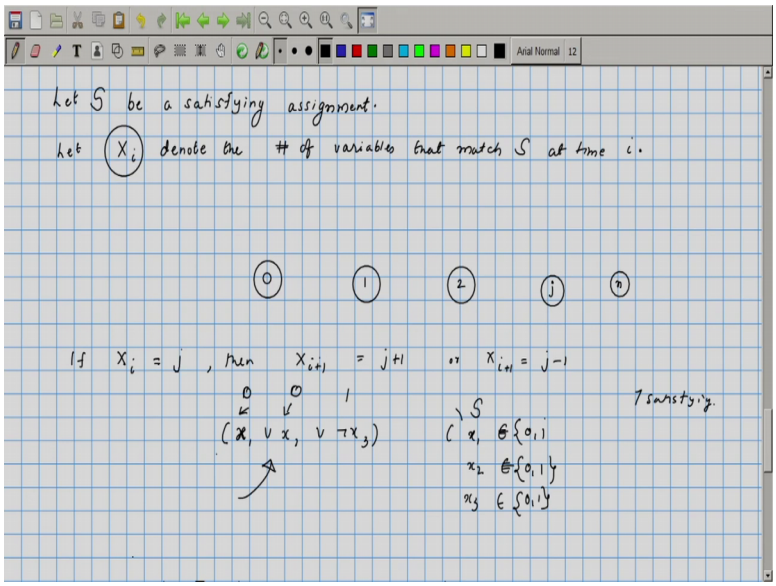
So, let me just describe the algorithm. So, the input is 3-SAT expression, let us call it as say phi ok. And what we will do is choose a random assignment. And then repeat for let us say some number of steps m steps. So, if the current assignment is not satisfying assignment choose an unsatisfied clause, and flip randomly chosen variable in that clause. Then we will check for satisfiability.

So, this is being repeated. If no satisfying assignment has been found, declare phi to be unsatisfiable ok. So, in this step when we check for satisfiability, if you find the satisfying assignment, we will say declare satisfiable. If found if a satisfying assignment is found ok, so the choice of m is going to be crucial ok. So, it is simple algorithm, choose a random assignment. And then if this random assignment is not satisfiable, it means there is one clause. At least one clause which is not satisfied by the current assignment; so that clause, will have three variables inside it. Choose one of those variables, and flip its value ok.

So, if the current assignment and assigned 1, then you changed it to 0. The current assignment has assigned it 0 change it to 1 ok. So, you flip the value of a randomly chosen variable in and in any unsatisfied clause ok. So, this is the algorithm that we have in mind. Let us first analyse how good or how bad this algorithm is. So, how many times, we will we have to repeat this algorithm, in order to get a and say a satisfying assignment.

Now, one feature of this algorithm is if you take an unsatisfiable expression, if the 3-SAT expression was unsatisfiable, this algorithm will always give the correct answer. For a 3-SAT expression which this satisfiable; we might find the satisfying assignment only on our lucky day. We would not know what is the probability, there it will be our lucky day or we want to know what is the probability that satisfiable 3-SAT expression will be found to be satisfiable by this algorithm ok. And we could increase m in order to increase the probability of success, so we want to really know how much how min the relationship between I mean how many steps should we run this algorithm, so that we will get some success in some significant success probability.

(Refer Slide Time: 11:27)



So, in order to analyse this algorithm. Let us look at a satisfying assignment ok. So, we need to bother only about those inputs which were satisfiable, because on the inputs which were not satisfiable, we are guaranteed that this algorithm will return the correct answer. So, let us be a satisfying or satisfies or satisfiable expression sorry satisfying assignment.

Now, what we will determine is the probability that exactly this particular satisfying assignment is found by our algorithm, we will bound that ok. It is possible that the algorithm the input has multiple satisfying assignment and one of them has discovered ok. But, what we will analyse is the probability of the algorithm finding one particular satisfying assignment that probability will surely be a lower bound on our success probability ok.

So, in order to find that probability, what we will do is introduce a stochastic process ok. So, let X i denote the number of variables that match S at time i ok. So, let us look at this algorithm. So, for the time being let us say that we will repeat this for an infinite number of steps ok. We will keep we will never stop the algorithm ok.

And at any time we will look at how many I mean how many variables I mean, so we will define a random variable X i which denotes the number of variables that match S at time i. Of course, there is no way of determining what is X i, because we do not know

what the satisfying assignment is that is not a problem we can we can just. So, this is the random variable that we have defined ok.

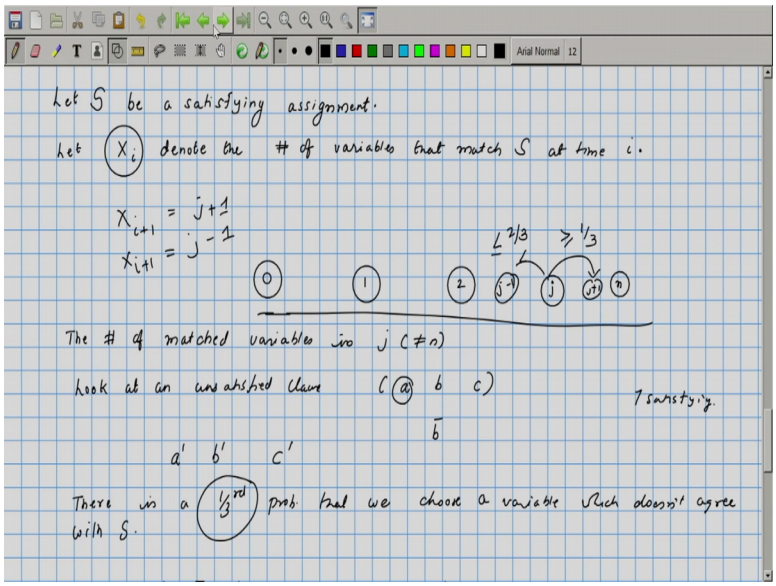
Since, there is a satisfying assignment, this X i has some particular value. The moment X i is equal to n, we know that all the variables match S, and therefore it is a satisfying assignment. And therefore, the algorithm would stop, when X i equals then. We want to know how many steps will it take for excite to reach the value n ok. So, let us set up a chain or a stochastic process, where the states are the values that X i's can X i can take. X i can be 0, X i can be 1, X i can be 2, X i can be n.

Now, suppose somebody tells us that X i's value is let us say j. In the next step, what are the values that x i can take. So, if X i equals j at time let us say i mean, so X i mean value at 2 the value of X i is j. Then x i plus 1 can be equal to either j plus 1 or j minus 1 so ok. These are the only two values that it can take, because so we are assuming j is not equal to n. If j is not equal to n, then what happens is we are choosing a clause which is not a satisfied clause. And in that we flip one of the variables ok.

Now, when you flip one variable that can probably increase or decrease, the so let us say this is a clause let us say small x 1 OR x 2 OR naught X 3 ok. So, only J variables match the satisfying assignment, therefore there is one particular clause which is unsatisfied. In that clause, you look at the values of the variables ok. So, let us say this is 1, 1, 0 it is a 0, 0.

So, if this is unsatisfied, then the value of it is one particular value, it cannot have multiple values, so this 0, 0, 1. Now, in s if you look at the variables, the value of the variables x 1, x 2, and x 3. If they are matching, let us say x i equals I mean 0, 1, these two possibilities are there. X 2 could also have been 0 or 1. And x 3 also belongs to 0 comma 1, but the satisfying assignments would have been there are seven satisfying assignments.

(Refer Slide Time: 18:36)



If we choose X 1 and flip it, then of course sorry so let us look at a so this is a scenario that we have at some time the number of matched variables is j ok, and j not equal to n. Therefore, the current assignment its so matched variable this j, so current assignments certainly is not matching the satisfying assignment. So, there is one particular there is at least one particular clause on which it is not a satisfying assignment, I mean I it is not it is at least one clause on which it is different from s. As if you take grid with s on every particular on every variable, then j would have been n.

So, let us look at an unsatisfied clause, so look at an unsatisfied clause ok. And let us say the values assigned to it was a, b, and c, we will flip one of these ok. So, if you choose b, it becomes b bar. We choose c, it will be c bar and so on. Now, what happens to the number of variables that match s, maybe a I mean in the assignment that we are looking at, we had let us say a bar, b bar, and c bar.

We know that it is not the case that a bar equals a, and b bar equals b, and c bar equals c, at least one of them must be different. So, the randomly chosen one is going to be different from s with probability of at least one-third. So, we can say there is a one-third probability that we choose a variable, which does not agree with s. Of course, it could be the case at all the three of them disagrees with this ok.

In that case if we randomly pick, we will with probability one get a variable which does not agree with s. But, we can always say that at least one-third probability is there,

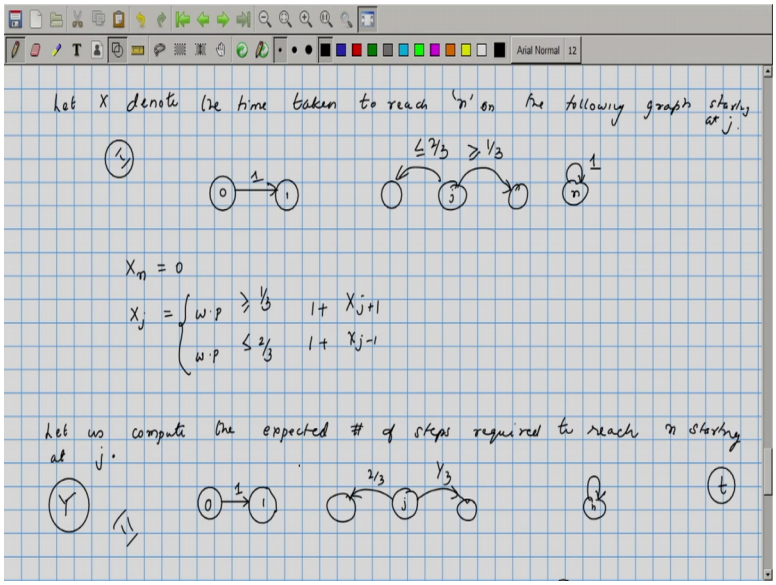
because the assignment that we currently have is not matching with the assignment as on some particular clause. So, there is at least a one-third probability. So, if we pick that particular variable and flip it, it is now going to increase the number of variables that agree. So, X j plus X i plus 1 would become j plus 1 in that case.

Now, if we were unfortunate, and if we picked a variable which did agree with s, then when we flip we reduce the number of matches. So, in that case x i plus 1 at the next instance, the value of the random variable is going to be j minus 1 ok. So, these are the two possible values that X i plus 1 can take. If you know that X i plus 1 is equal to X i is equal to j.

So, from j it can go to j plus 1 with probability greater than or equal to one-third, and it could go back to j minus 1 with probability less than or equal to two-third. In case of 2-SAT, these probabilities would have been half and half ok. If you look at the 2-SAT case, the random walk you can imagine a random walk on this particular graph, it has a tendency to go forward or backward with equal probabilities, but in this particular setup where we are reconsidering 3-SAT.

The graph has a larger has a higher tendency ten density to move towards 0. We need to ascertain how much is the success probability that is if we walk on this particular graph, what if you do a random walk on this particular graph, what is the probability that you will hit n, when you hit n it means we have reached the satisfying assignment s.

(Refer Slide Time: 23:19)



So, let x denote the time taken to reach n on the following graph ok. So, if you are at 0, you go to 1 with probability 1. And at any other node and at n you stay there with probability a 1. And at any other node let us say j, you go forward with probability greater than or equal to one-third. And you come back with probability less than or equal to two-third ok.

So, if you start at some random place inside this particular graph ok, let us say you start at so instead of so this is when you define the random variable x, and the time taken to reach n, we also need to specify the starting point, so let us say starting at j ok. So, if you start at the position n, if j equals n, then you will reach in no time. And for the others, it is going to depend on the particular structure ok.

So, we can say that x n is equal to 0, x j is going to depend on. So, x j is going to be with probability greater than one-third, this is going to be equal to 1 plus X j plus 1 ok. Whatever is the time taken to reach from j plus 1, add 1 to it that is going to be the value of X j with probability one-third? And with probability less than two-third, it is going to be 1 plus X j minus 1 ok. So, these are the random variables that are involved.

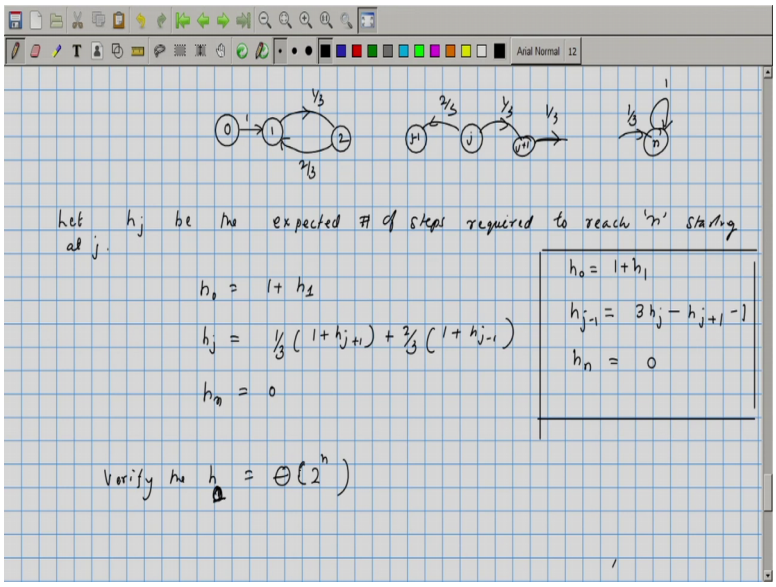
Let us compute the expected time to reach n from a starting point j. So, let us compute, the expectations of these X i's. The expected number of steps required to reach n starting at j. Now, when these probabilities are not known, this calculation we cannot really do, we can only bound the probabilities. So, instead what we will do is since we are looking at reaching n, we will define a new random variable y or a new random walk on which you are going forward you going forward with probability.

So, if you are at some particular node j, you go forward with probability one-third, and you go backward with probability two-third ok. So, this is the different random process than the one we had discussed, because there we only knew that this probability is at least one-third, and the reverse probability is at most two-third. But, if you look at this modified one, we can say that since this is a propensity to move forward, I mean it has at least the same tendency as x when so let us call this is the first random process, and this is the second random process.

The expected time taken by the second random process to reach n starting at any point is going to be surely greater than the expected time taken in the case of the first random process ok. So, this is having a tendency to move forward with I mean probability greater

than one-third, this we are putting it to be exactly one-third ok. So, if we get Y, if we get a upper bound on y ok, we know that we can use that same bound for x as well. If Y is shown to be at most let us say some particular amount t ok, the expected value of y is t, x value is only going to be the expected value of x is only going to be less than or equal to t. And this is much more easier to analyse than the other one. So, we will analyse this particular random process.

(Refer Slide Time: 29:01)



So, let h j be the expected number of steps required to reach n starting at j. So, we can write the recurrence relations for these h j's h 0 is equal to 1 plus h 1. And h j is equal to 1 by 3 into 1 plus h j plus 1. If taken one step, and gone to h j plus 1 that happens with one-third probability plus 2 by 3 into 1 plus h j minus 1. And h n is equal to 0 ok.

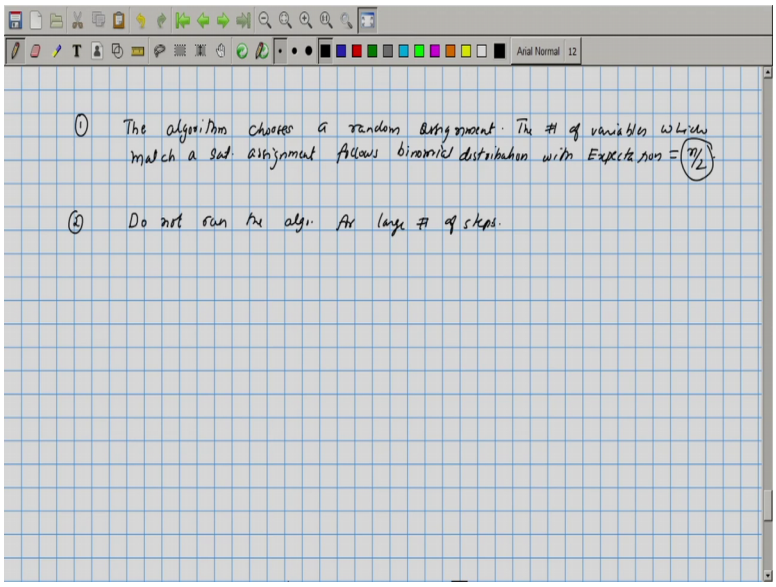
So, these are recurrence relations, we can write it as h 0 is equal to 1 plus h 1, and h j minus 1 is equal to 3 h j minus h j plus 1 minus 1, and h n is equal to 0 ok. So, you can solve this a linear recurrence system we can solve it, and one can show that h n is approx h 1 starting at 0 or starting at 1, how much time would we take. So, you can verify that h 0 or it is approximately I mean it is equal to this, we can write it as is equal to O of 2 raise to n ok. We can show that it is going to take this much that is theta of 2 raise to n. So, you can show that this recommend solves to something like this.

So, if we use this, the number of steps we have to perform I mean if we use this analysis, the number of iterations that our algorithm will have to do. In order to reach a satisfying

assignment, the value of m will be 2 raise to n. We could have just tried all possible 2 raise to n assignments that would have been a deterministic algorithm. So, we are not gaining anything by doing this particular, I mean by following this particular algorithm. But, there are few things that we can change in this algorithm, and that is going to give us better bound than 2 raise to n.

In case of 2-SAT, this does give wonderful results, because we had half probability of going forward and backward ok. So, if there was a satisfying assignment, we can say that there is a reasonable chance of there is a significant chance of hitting the satisfying assignment. Here since the bias is one-third in the forward direction, and two-third in the backward direction, the tendency of the algorithm is to move away from the satisfying assignment ok, so what is the way out.

(Refer Slide Time: 33:36)



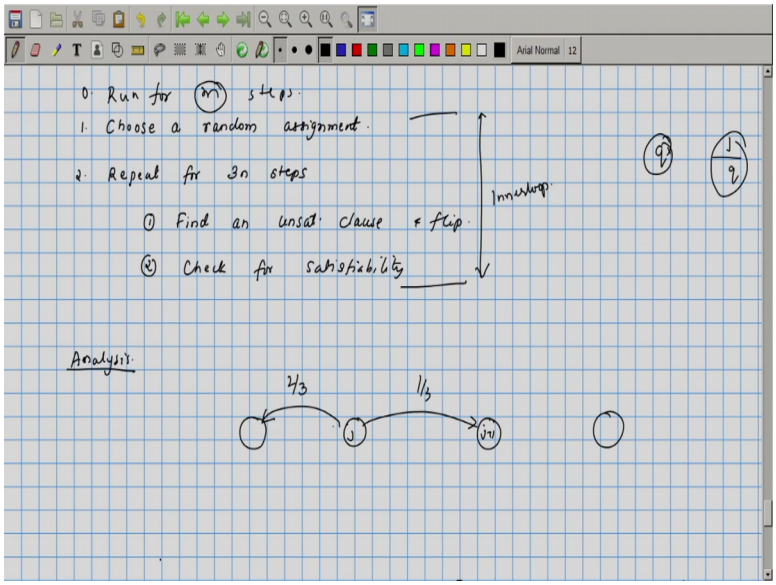
So, first thing is the algorithm chooses a random assignment ok, when it chooses a random assignment, the number of variables which match a satisfying assignment follows the binomial distribution with expectation equal to n by 2 ok. If you had chosen each into the variables by tossing a coin, you can expect that the number of variables in the assignment that you have come up with the initial assignment, it will match roughly 2 raise to I mean roughly n by 2 variables, now that happens with small probability.

But, one can say that the there is not small, but not so insignificant probability that you will start with some assignment which is lot more than n by 2 sat mean n by 2 matches

ok. Matches when our assignment; the randomly chosen assignment is agreeing with a satisfying assignment on a particular variable ok. If we randomly choose although, we expect the number of matches to be n by 2, it could be I mean greater than that and that can happen with signal I mean a non-trivial probability ok.

So, what we will do is we will not keep on running the algorithm for long steps, because if you keep running the algorithm for long steps, so what happens when you run the algorithm for large number of steps is, it will draw your assignment your random assignment towards zero matches with the satisfying assignment. So, instead what we will do is we will run it for some number of steps. And if it has not hit n that means, it if it is not found n matches yet or if it has not found a satisfying assignment, what we will do is we will just reset, and start with a fresh assignment ok. And that process we will repeat for many times, so that will be our modified algorithm.

(Refer Slide Time: 36:36)



So, let me just write down the modified algorithm. First step is as say I mean similar to the earlier one. Choose a random assignment ok, and then repeat for say 3 n steps ok. In our earlier algorithm, we are running for m steps that m now is going to be 3 n ok. So, what do we repeat, we will find an unsatisfied clause and flip. And then check for satisfiability ok.

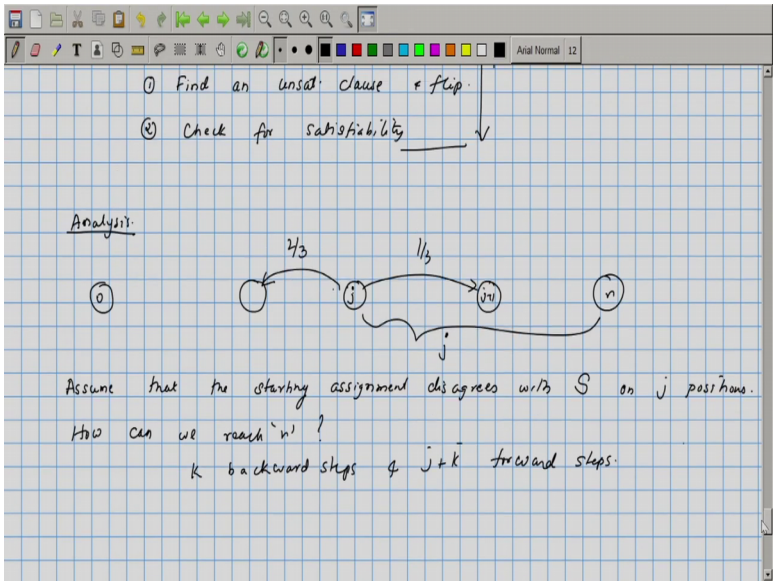
And then this entire sequence, let me say let us 0th step, wherein reached I will restart with a new ok. So, this I will run for let us say m steps ok. So, there are two loops, so this

m we will fix later. So, in each iteration of the outer loop, I will choose a random assignment and try for 3 n steps. If it does not succeed in 3 n steps, I will again choose another random assignment, and then run on that random assignment for 3 n steps, and keep on doing this.

We need to analyse what is the probability that this algorithm succeeds ok. So, what is remaining us the analysis of this particular algorithm, how is it better than the other algorithm ok. So, what we will do is we will just analyse the inner loop ok, this will call as the inner loop ok. What is the success probability for the inner loop ok? If the success probability of the inner loop is q, we need to run the algorithm for 1 by q steps ok, because this is like a geometric random variable.

Each run of the in inner loop is independent of the other iterations ok. So, we can run it for 1 by q steps, and that will be our algorithm that 1 by q will be m, so what we will determine is q. And similar to the earlier thing, what we are interested in this particular chain from j to j plus 1 with probability one-third; and to j minus 1 with probability two-third.

(Refer Slide Time: 40:03)

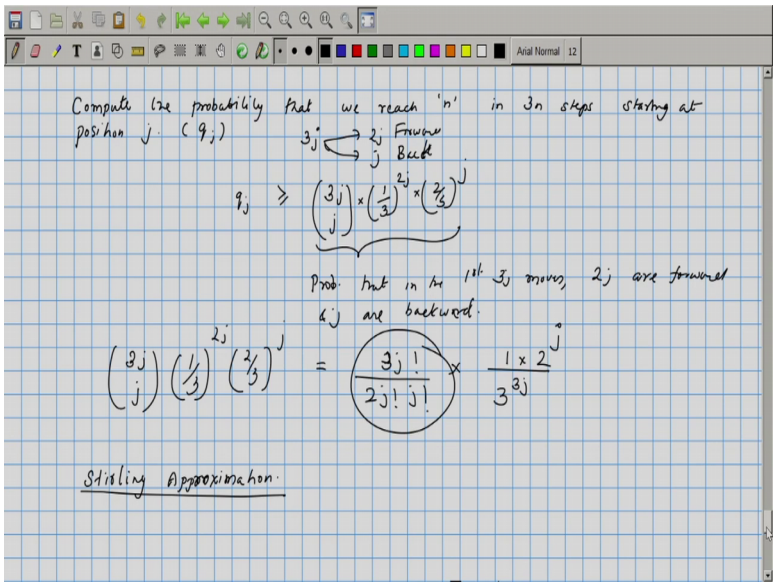


So, when you look at this particular chain, we want to know the so suppose you start at jth position, what is the probability that you will hit n. In fact, instead of starting mean calling those indexes j, let us call this length as j ok. So, assume that the starting

permutation or the star sorry the starting assignment disagrees with s, s is known to be a satisfying assignment.

So, the starting assignment disagrees with this on j positions ok. So, if we get j forward moves immediately, then we will reach n. The other possibility is we have so how can we reach n, so k backward steps, and j plus k forward steps ok. So, at some point if you have j plus k backward steps, and sorry if you have j plus k forward steps, and k a backward steps, you would have gone j steps forward ok. And then you would have reached a satisfying assignment, what is the probability of that happening ok. So, what we want to determine is if you had if you condition on starting at an initial assignment, which has only j positions unmatched with satisfying assignment s, then what is the probability that we will reach n in 3 n steps ok.

(Refer Slide Time: 42:17)



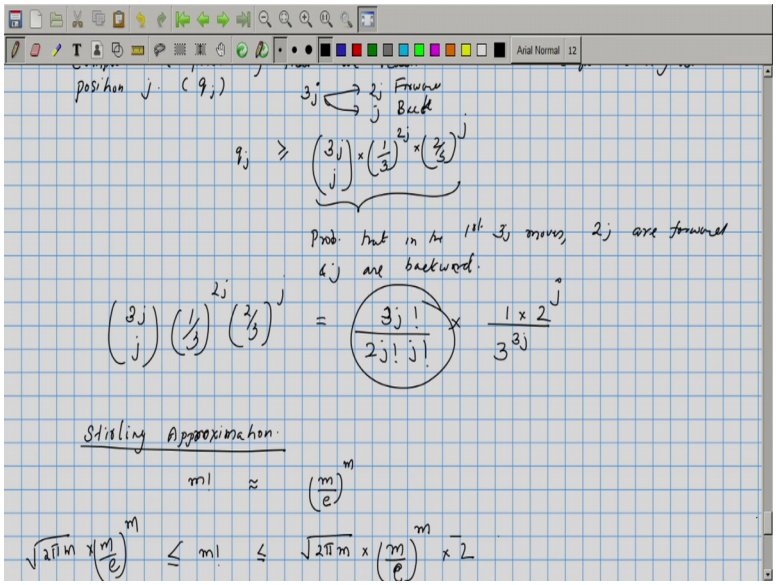
We want to compute, the probability that we reach n in 3 n steps starting at position j ok. So, let us call that probability as q j. So, q j is certainly greater than so let us say that out of 3 j steps. 2 j are forward, and j are backward steps. So, 2 j forward and j backward ok. So, let us just imagine the first 3 j moves, as j is less than n 3 j is going to be certainly less than 3 n.

Let us look at the first 3 j moves in the first 3 j moves what is the probability that 2 j of the moves are forward, and j moves are backward well that is going to be 3 j choose j, so these are the moves chosen for the backward move. And once the backward moves have

been fixed, the forward moves is going to be the remaining. So, this into forward moves happen with probability 1 by 3 raise to 2 j times 2 by 3 raise to j ok.

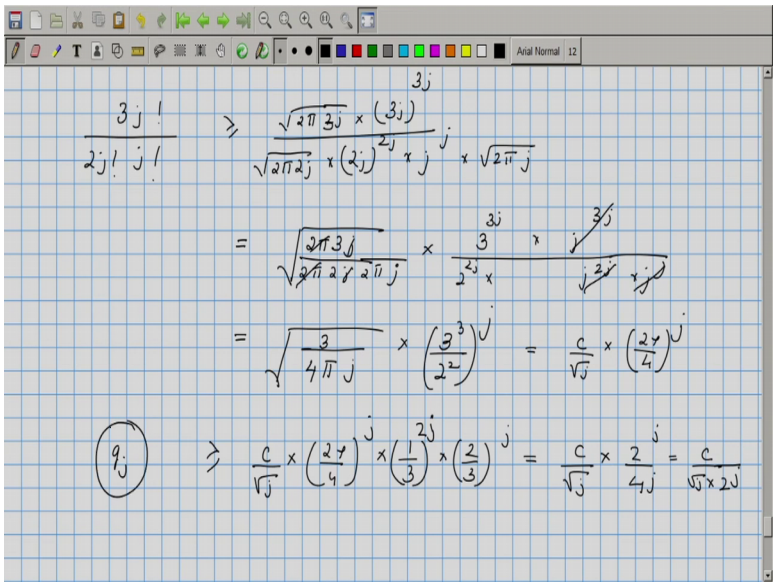
So, this quantity here is the probability that in the first 3 j moves, 2 j are forward, and j are backward ok. And if this event happens, then surely we would have found a satisfying assignment ok. So, this probability is a lower bound on the probability that you will find a satisfying assignment starting at position j. So, we will need to estimate how small or large is we have to bound this; we have to compute a lower bound for this quantity. So, 3 j choose j into 1 by 3 raise to 2 j times 2 by 3 raise to j is the quantity of interest, this is equal to 3 j factorial by 2 j factorial into j factorial into 1 by 3 raise to 3 j, and 2 raise to j.

(Refer Slide Time: 45:53)



In this quantity, we need to estimate we will use Stirling approximation for that. So, if you look at m factorial, m factorial is approximately m by e raise to m ok, but we can get a slightly better bound ok. We can say that m factorial is less than root 2 pi m times this quantity m by e m divided by e raise to m into 2 ok, and it is greater than root 2 pi m into m by e raise to m ok, so it lies between these two numbers. If you think of this as t, this is 2 t. So, m factor relies between t and 2 t, where t is this particular quantity ok. So, 3 j factorial divided by 2 j factorial into j factorial is what we want to compute ok.

(Refer Slide Time: 47:07)



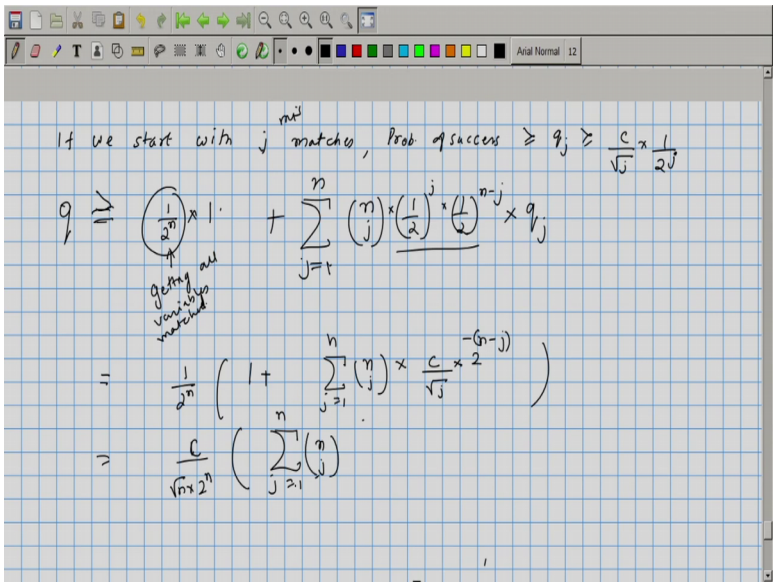
And we want to find a lower bound for that ok. So, 3 j factorial by 2 j factorial into j factorial the by e we can ignore, because the same number of is comes in the numerator and denominator. So, this we can say is greater than the new numerator, we will apply the lower bound which is the root 2 pi times 2 j sorry 3 j into 3 j raise to 3 j 3 j by e is what we should have taken, but that by e will cancel out in the numerator and denominators divided by root 2 pi into 2 j into 2 j raise to 2 j times j raise to j into root 2 pi j ok.

So, this is going to be equal to so under root 2 pi 3 j divided by 2 pi 2 j 2 pi j times, the j's are going to cancel out. So, we will get 3 raise to 3 j into j raise to 3 j divided by 2 j raise to 2 j 2 raise to 2 j into j raise to into j raise to 2 j into j raise to j, so these quantities cancel out. So, I will get this to be equal to root 2 root 3 by 4 pi j into 3 raise to 3 by 2 raise to 2 the whole raise to j ok.

So, this I can write it as some constant divided by root j into 27 by 4 raise to j ok. And the additional term, so q j that the additional terms was so you can put plug in those values. So, q j is greater than c by root j into 27 by 4 raise to j into there was a 1 by 3 raise to 2 j, and 2 by 3 raise to j 1 by 3 raise to j into 2 by 3 raise to 2 j ok. So, this is equal to c by the root j into 27 raise j is 3 raise to 3 j that cancels out from numerator and denominator.

So, what remains is 2 raise to 2 j divided by sorry this is 2 raise to j, so you had a mistake this is raise 2 j, and this quantity is j. So, overall you will get 2 raise to j divided by 4 raise to j, this is equal to c by root j times 2 raise to j that is a probability of q j.

(Refer Slide Time: 51:44)



So, if you start at jth position, we can summarize this as if we start with j matches, probability of success is greater than q j which itself is greater than c by root j times 1 by 2 raise to j. Now, what is the probability that you will get j matches, when we randomly choose an assignment; well that is again binomial distribution. So, out of n we have 1 by 2 raise to n probability of getting everything correct.

And so this is the probability of getting all variables matched plus if you had to have exactly j matches, then that will be n choose j into 1 by 2 raise to j times 1 by 2 raise to n minus j ok. So, this is the probability of getting one match, this is the probability of getting j match. If you get j matches, then the total probability of success would be this summed over all values of j multiplied by q j. And this when you get all matched, you will get it as one. So, this sum is a probability of success.

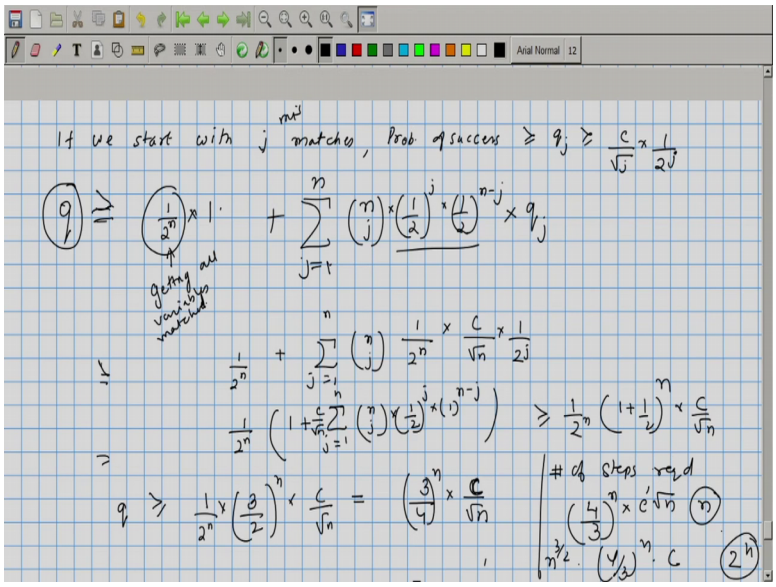
The overall probability of success, if you denote it by q, q is going to be greater than or equal to 1 by 2 into 1 plus j going from 1 to n minus 1 ok.

So, yeah j is the number of mismatched I mean so if we think of j as the number of matches, this goes from j equals 0 to n minus 1 ok. So, this summation sorry let us say

we had stated j to be the number of mismatches, the number of misma[tch]- so this is for when the number of when all variables gets matched, then the number of mismatches is 0 ok. So, 1 by 2 into 1. And in the other case, it will be j equals 1 to mismatch being n ok.

So, this one can sum it up as summation j equals 0 to n. Since, this one can sum it up by taking 1 by 2 raise to an outside 1 plus summation j equals 1 to n, n choose j these two multiply and given raise to n into q j. So, q j we can replace it with C by root j into 2 raise to n minus j ok. So, this can be thought of as so root j is at most root n. So, 1 by root n times 2 raise to n times c summation j equals 0 to n n choose j 2 raise to n minus j or 2 raise to minus n.

(Refer Slide Time: 56:10)



So, this can be written as so if you take root n, and c outside, you can write this as c by root n into 2 raise to n into summation j equals 1 to n n choose j. So, we just plug in the values 1 by 2 raise to n plus summation j equals 1 to n n by j 1 by 2 raise to j, and 1 by 2 raise to n minus j will get 1 by 2 raise to n into q j is going to be c by root j. So, instead of root j, we can say we can just replace it by root n into 1 by 2 raise to j ok.

So, here from this we can take 1 by 2 raise to n outside 1 plus summation j equals 1 to n, so c by root n is common. n choose j into 1 by 2 raise to j into 1 raise to n minus j ok. So, this one can say is to say I mean one can just apply binomial theorem, this is going to be greater than 1 by 2 into 1 plus 1 by 2 raise to n times c by root n. So, if you expand this out, you can compare term by terms, and say that it is greater than this ok.

So, we will get q to be greater than ok, so this 1 by 2 raise to n also outside, 1 by 2 raise to n into 3 by 2 raise to n times c by root n, which is equal to 3 raise to n or 3 by 4 raise to n into 1 by root n. So, if you had to convert the algorithm, by running it multiple times the success probability of which is only q, you would run it 1 by q times the number of times number of steps required will be 4 by 3 raise to n into say c root n c prime root n. This is the number of steps required, each step takes let us say mean n's n if you assume that each step takes n time units, the total running time is going to be n raise to 3 by 2 into 4 by 3 raise to n time c still an exponential algorithm, but it is significantly better than say 2 raise to n ok, so that is the randomized 3-SAT algorithm.