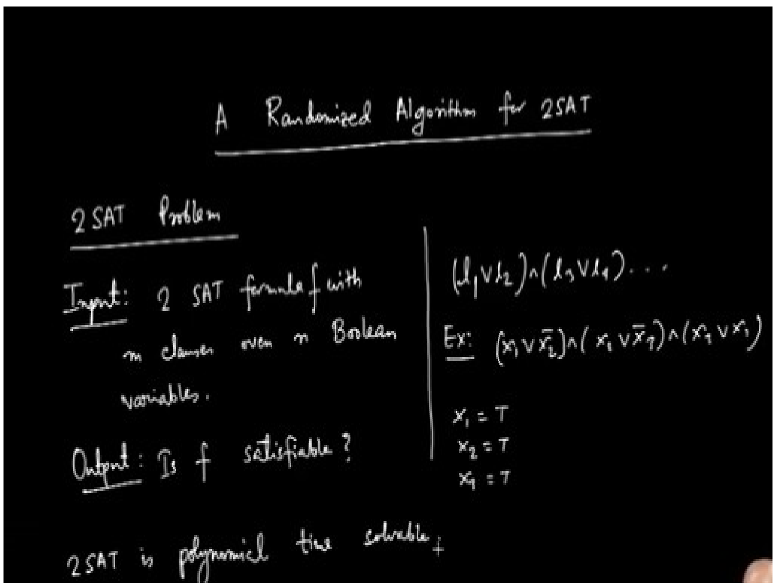
Selected Topics in Algorithm
Prof. Palash Dey
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Module No # 04
Lecture No # 20
Randomized Algorithm for 2SAT

Welcome in today's lecture we will see a very simple randomized algorithm for 2SAT it will be again a Monte Carlo type of randomized algorithm.

(Refer Slide Time: 00:40)

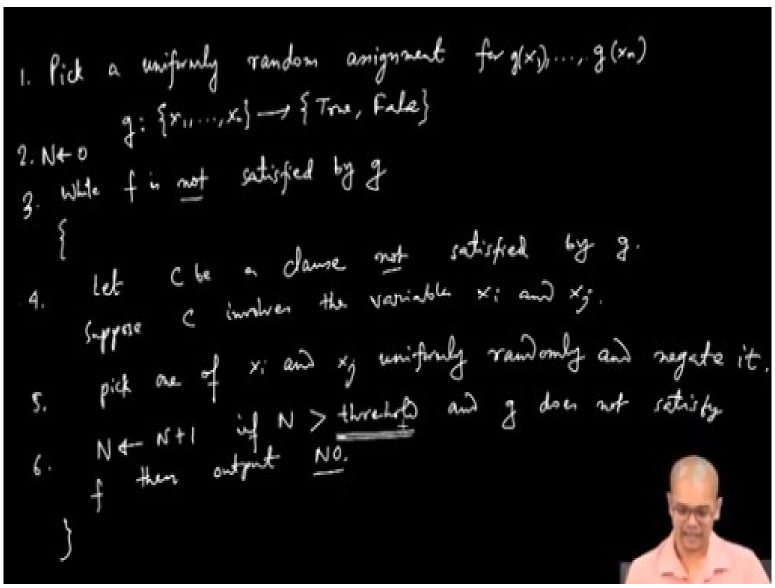


So what is the 2SAT problem? Input is a Boolean formula a 2SAT formula and how does a 2SAT formula look like it is like a literal l_1 or l_2 and some literal l_3 or l_4 and so on. It is an end of literals end of some clauses and each clause is a, or of 2 literals each literal is a variable or its negation. So an example of a 2SAT formula could be $X_1 \vee \bar{X}_2$ and $X_2 \vee \bar{X}_3$ and you know $X_3 \vee X_1$ something like that.

So 2SAT formula over 2SAT formula with m clauses over in Boolean variables. What is the output let us call this formula f is f satisfiable; that means does there exist an assignment to the variables of the formula which makes all clauses turn to true. For example this formula may be satisfiable let us check you can set X_1 to true and X_2 to true maybe X_3 you can make arbitrary with true again. So this particular assignment satisfies this formula and or evaluates this formula to true hence this formula is satisfiable.

So this particular problem is polynomial time soluble 2SAT is polynomial time solvable in contrast you know 3 sat is not parliamentary soluble at least that is to the current understanding and that part will see after a couple of lectures. But now let us focus on 2SAT formulas it has a deterministic algorithm by reducing it to graph and so on but now in today's lecture we will see a very simple randomized algorithm for 2SAT.

(Refer Slide Time: 04:43)



So here is the algorithm so pick a uniformly random assignment for $X_1,\ldots,X_n$ suppose these are the variables $x_1,\ldots,x_n$. So let us call this you know this assignment $f(x_1,\ldots,x_n)$ or let us call some g some other name let us call it g maybe $g(x_1),\ldots,g(x_n)$. So g is basically a function from $x_1,\ldots x_n$ to true and false. While it is not satisfied by g and then so that means if the formula f is not satisfied by g then you keep on trying and we also keep track of how many times we have run the algorithm.

So for that let us call this is some counter you maintain n is 0 and if g is not g is not a satisfying assignment then there exists at least 1 clause which is not satisfied by g. So let see be a clause not satisfied by g pick any such clause. Suppose c involves the variables x_i and x_j that means the clause the literals involved in c is either $x_i \vee \bar{x}_i \vee x_j \vee \bar{x}_j$ and so on. Pick now among between x_i

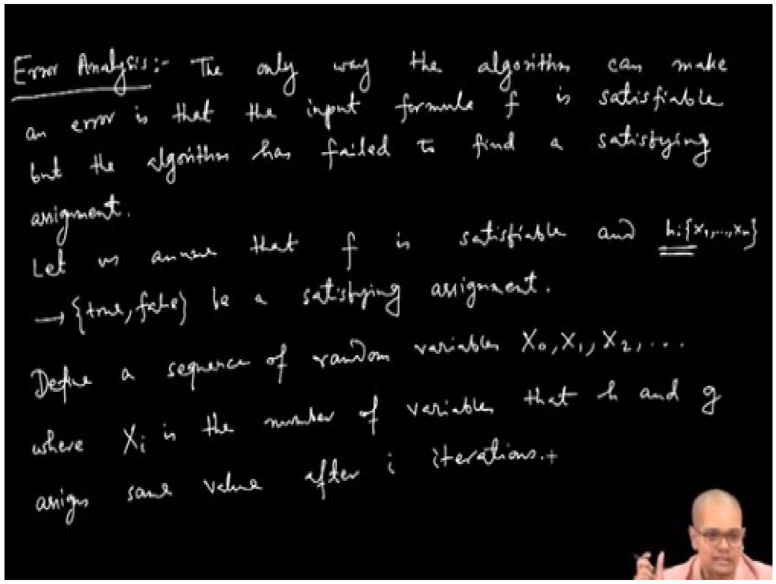
and x_j pick one of them uniformly random and negate them that way this clause c will be satisfied and I get a new assignment.

Pick 1 of x_i and x_j uniformly randomly and negate it of course we increase the counter n is n plus 1 and we so this way we try various assignments. And if we have tried many times and we are still not able to find an assignment then we will then we will simply output no. So if n is greater than some threshold this we will see later in the analysis what should be the threshold? If n is greater than threshold and g does not that means after inverting the x_i or x_j the new g; g does not satisfy f then output no that it is not satisfiable.

Otherwise you again try with some other g and so on. So this is the algorithm is a very simple algorithm so what is the idea? You start with a random assignment if that is not a satisfying assignment then at least 1 clause is not satisfied. So pick any such arbitrary clause which is not satisfied in this step there is no randomness involved. Now this clause because it is a 2SAT clause it involves 2 variables x_i and x_j.

Pick any one of them and invert it, negotiate it and that will that that will satisfy that clause c and we will have a new assignment let us we are that's how g is changed over iterations. And again check if g is a satisfying assignment or not and if not repeat the same process and after trying many number of times. We will see how many if we have not theta satisfying assignment we output no that's the algorithm. So number of iterations is basically is the threshold what we sat and that we will see what should be the threshold and for that let us do the error analysis.

(Refer Slide Time: 11:37)



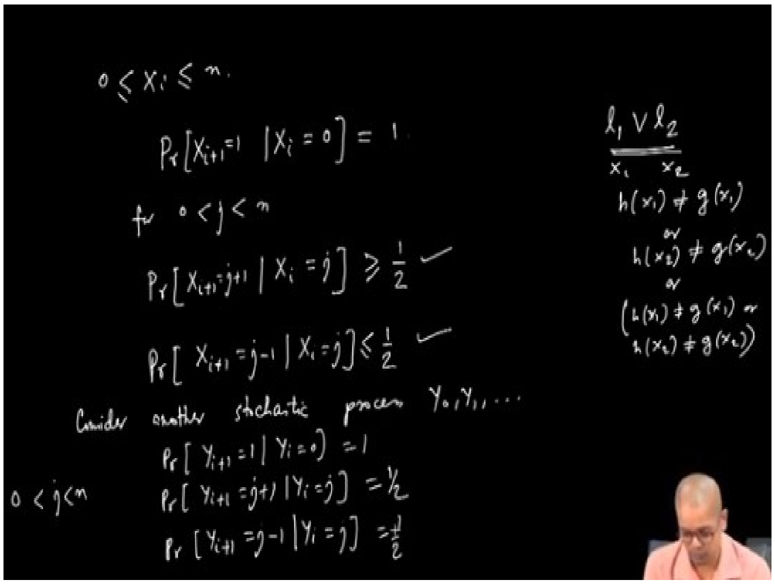
So in this case if the algorithm if the Boolean formula is not satisfiable then that means there does not exist any satisfying assignment then it does not matter how many times it tries or how it tries it can never going to find a satisfying assignment simply because it is not sat satisfiable. So if the input 2SAT formula is not satisfiable then the algorithm always outputs correctly here in seventh output is if it finds a satisfying assignment g. So it can only make an error if f is satisfiable and it has not found it.

So the only way the algorithm can make an error is that, the input formula if is satisfiable but the algorithm has failed to find satisfying assignment so that is the only way it can make an error. So let us assume so let us assume that f is satisfiable otherwise it does not make an error if is satisfiable and h be a satisfying assignment. Now we define sequence of random variable $X_0, X_1, X_2, \ldots$ and so on infinite sequence of random variables $X_0, X_1, X_2, \ldots$ and so on. Where X_i is the number of variables that h that means this assignment h and g assigns same value after iterations.

So it is like on how many variables h and g agree if h and g agree in all variables then the algorithm has found and found a satisfying assignment namely h. And what we will show is that you know in expectation if there exist an h at this algorithm will eventually find it not after too much iteration.

So if saying the same thing in other way if they are indeed exist a satisfying assignment it is unlikely that this algorithm does not find it after sufficiently minerals. Now what does sufficiently mini run mean that the analysis will tell what does that that quantification it will quantify the sufficiently may be done. But let us see first you know how does this variables are distributed.

(Refer Slide Time: 17:29)



First observe that each X_i take values between 0 and n. It can agree the assignment or g can agree with h in 0 to n variables and when does the algorithm terminate if any X_i value is n then the algorithm definitely terminates and it is successful. So the error is that you know the X_i does not take the value n after sufficiently many run. So let us see how they are distributed so what is the probability suppose you know some X_i is 0.

That means h and g are in complete disagreement whatever value h assigns g assigns exactly opposite. So in the next iteration every in every iteration the or algorithm negates 1 variable so X_{i+1} then must be 1; whichever variable it negates on that variable now this h and g agrees this is so one so if in the ith iteration or h and g does not agree on any variable then. So in i + 1 it iteration h and g agrees on exactly 1 variable namely the variable which the algorithm has negated.

For arbitrary value for j you know greater than 0 and say again less than n. Because if it is n then the if X_i is n then X_{i+1} is also n it is like the algorithm stops there. And X_{i+1} is not defined the algorithm terminates if X_i is n so what is the value of X_{i+1} if X_i is j you know it negates 1 variable so the X_{i+1} can have only 2 possible values 1 is j +1 and if X_i is j in the ith iteration if h and g agrees on j mini variables and in i iteration you know the in i + 1 is iteration the algorithm negates only one variable then after i + 1 of the titration the number of agreement could be either j +1 or j -1. But what are their probabilities?

First observe that how the algorithm does picks a variable to negate it picks a clause which is not satisfied by g. So suppose this clause is involves 2 variables X_1 and X_2. Now definitely because h and g does not agree for both X_1 and X_2. Or at no at least one there is at least one variable where h and g does not agree so $h(X_1)$ is not equal to $g(X_1)$ or $h(X_2)$ is not equal to $g(X_2)$ or both.

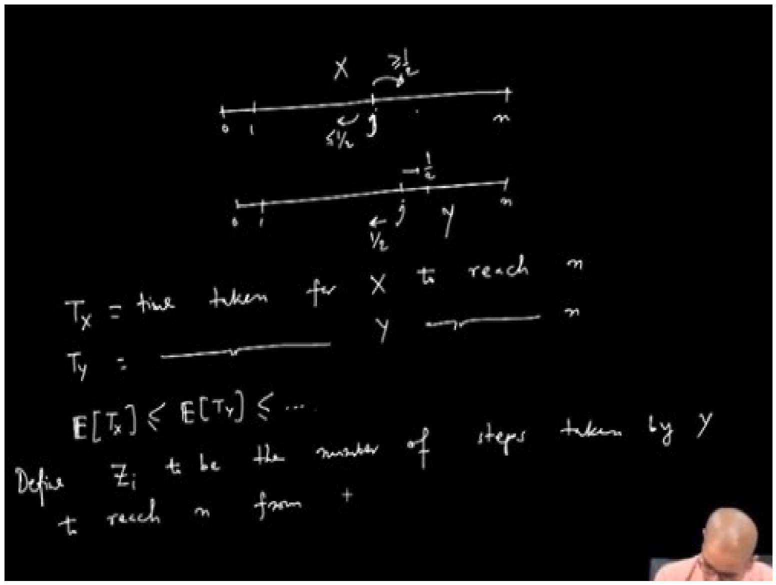
It can of course the of course disagree on both the variables but in at least one there is there exists at least 1 variable where they disagree after i iterations. Now what is the probability that in the i +1 in the iteration our algorithm picks that variable where h and g does not agree it picks 1 of the variable uniformly randomly. So that variable is split with probability half the other variable is picked with probability half.

And this is the case when there is disagreement on exactly 1 variable but it could be possible that h and g disagree on both the variables that are the third case. In this case you know it does not matter whichever variable it picks the algorithm picks it say that variable after negating it now agrees h and g agrees on that variable. So in that case this probability will be 1 so in both cases this probability X_{i+1} increases with probability at least half.

And so it decreases with probability at most half now you see that you know these variables this is bit difficult to analyze. Because you know this is we cannot exactly write what are what are this this is it depends on which clause it picks and how h assigns them so. This could be half or 1 at the top of this probability and the bottom probability could be half or 0. So what we do is that for analysis purpose it is easier to analyze something simpler stochastic process.

Which is like this that considers another stochastic price is nothing but a sequence of random variable where let us $Y_0, Y_1, \ldots$ and so on. Which is sort of a pessimistic version in some sense it is like probability of Y_{i+1} = 1 given Y_i = 0 is 1 and probability of Y_{i+1} = j +1 is for j greater than 0 less than n given Y_i = j this is exactly half and probability Y_{i+1} = j -1 given Y_i = j is half. Now what we will show is that we will analyze this stochastic process and that would be enough y.

(Refer Slide Time: 25:20)

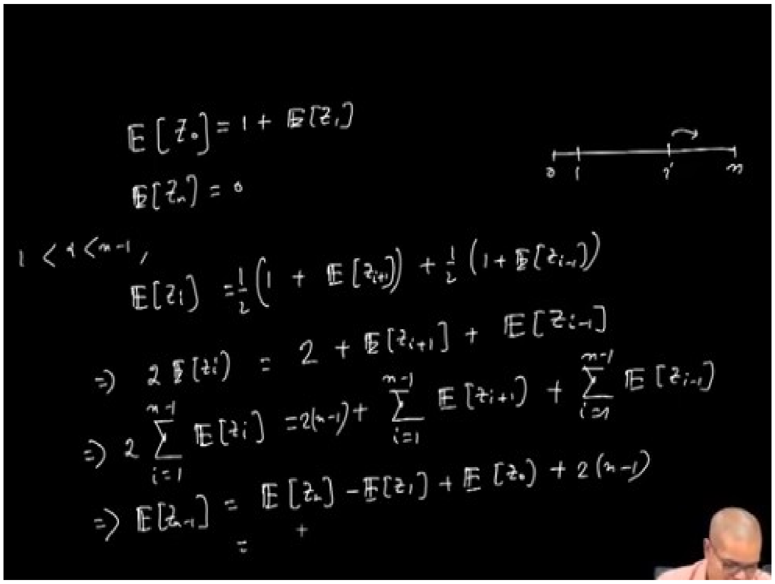


Let us understand this pictorially so this is 0, 1 these are the possible values of these random variables and here is n. Now there is a process this let us see how this random variable x behaves if X_i is say j the ith iteration this value is j then it goes to j +1 with probability greater than equal to half and it goes this way with probability less than equal to half. This is how X and there is another process everything is same but from j it goes to j +1 with probability exactly half and it goes to j -1 with probability exactly half.

So x has a tendency of going towards n more than Y X goes towards n with probability at least of where Y goes to n with probability exactly half. So which one will reach is likely to reach n first of course X so if we can show that you know bound how many after how many iterations y reaches n that that gives a bound an upper bound on what is the expected number of iterations X takes to reach n. So that is the idea so for that what we will do is that T_X is the time taken for X to reach n and T_Y is time taken for Y to reach n.

It is clear that expected expectation of T_X; X takes less time than Y is expectation of T_Y now what we will show will give an upper bound on expectation of T_Y and that way we will get an upper bound on expectation of T_X. So towards that we define another random variable define you know random variable Z_i define Z_i to be the number of steps taken by Y to reach n from i.

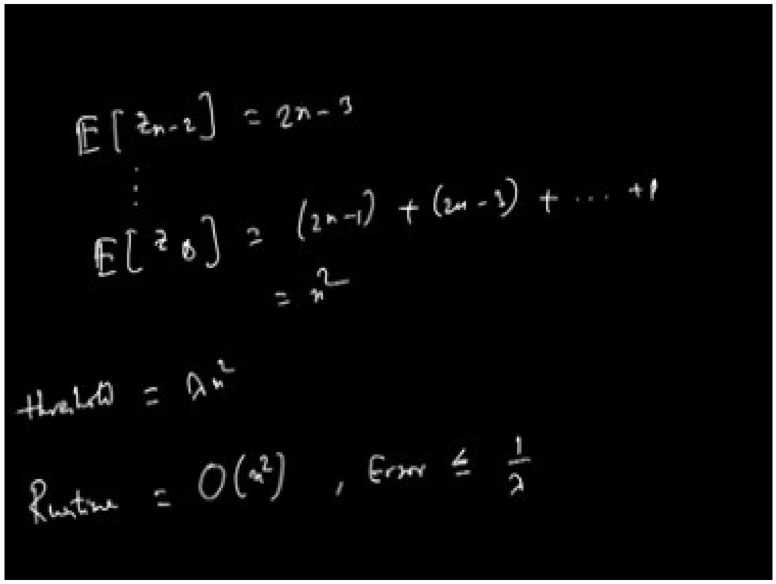
(Refer Slide Time: 28:58)



So then if it is Z_0 so if it is at zeroth position then the next iteration it will with probability one it will go to 1 position because it will negate a variable so expectation of Z_0 is 1 + expectation of Z_1 and expectation of Z_n is 0 because it has already reached. For i in between values 1 to n -1 what is expectation of Z_i if it is at i it goes to right side i +1 with probability half this is 1 + expectation of Z_i this happens with probability half and with probability half is 1 + expectation of Z_{i-1}.

So this is twice expectation of Z_i is 1 + expectation of is Z_{i+1} expectation of Z_{i-1}. These 2 we know we now add this inequalities twice some i = 1 to, n -1 expectation of Z_i is 2 + summation i = 1 to n -1. Expectation of Z_{i+1} + summation i = 1 to, n -1 expectation of Z_{i-1} and this is twice n - 1. So n - 1 equality so we are adding so what we get is expectation of Z_{n-1} is expectation of Z_n - expectation of Z_1 + expectation of Z_0 +...+ Z_{n-1}. Now expectation of $Z_0 - Z_1$ is 1 so this is an expectation of Z_n is 0 this is 2 n -1.

(Refer Slide Time: 32:39)



So now using this we now compute expectation of Z_{n-2} is like 2 n -3 and so on. So expectation of Z_0 is $2n-1+2n-3+...+1=n^2$. So even if it starts at 0 after x in expectation n^2 steps you know it reaches n so if we sat threshold to be some you know $\lambda \times n^2$ then by Markov inequality does not hit n with probability at most $\frac{1}{\lambda}$. So the runtime of the algorithm runtime is $O(n^2)$ and error is $\frac{1}{\lambda}$ less than equal to $\frac{1}{\lambda}$ so this concludes or algorithm so we will stop here today