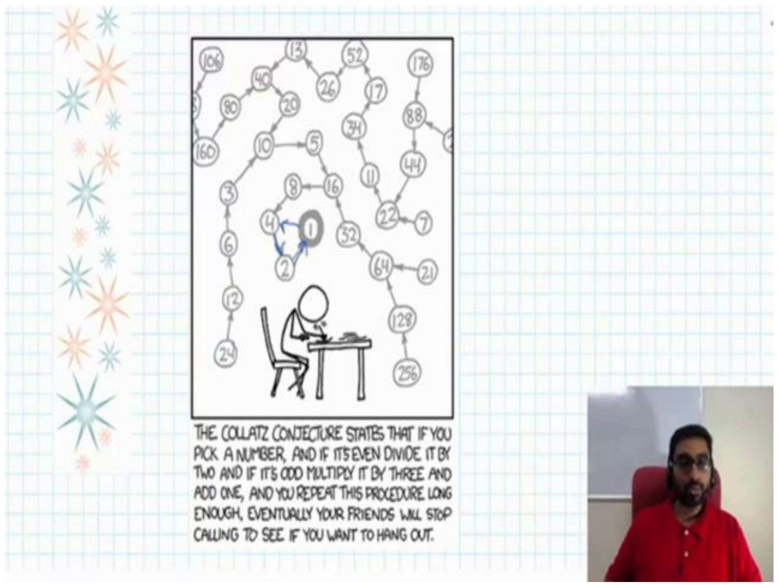
Algorithms for Big Data

Prof. John Ebenezer Augustine
Department of Computer Science and Engineering
Indian Institute of Technology, Madras

Lecture - 15
SAT Problem

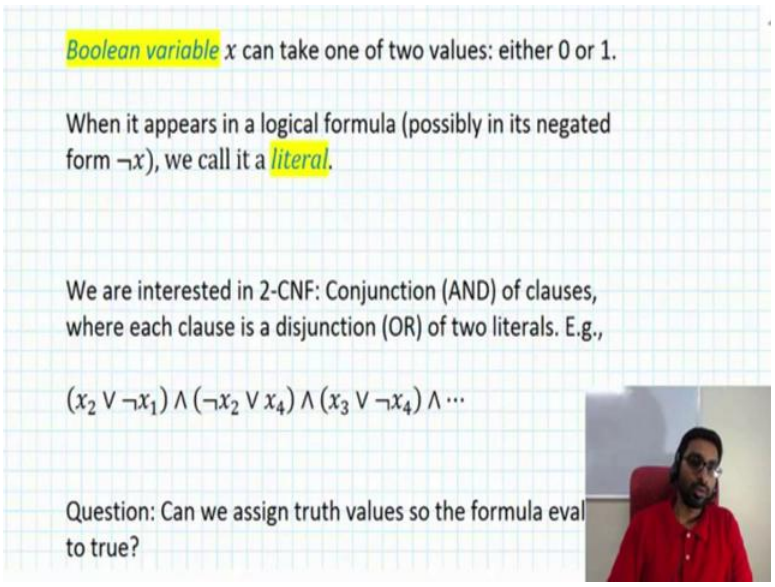
(Refer Slide Time: 00:16)


THE COLLATZ CONJECTURE STATES THAT IF YOU PICK A NUMBER, AND IF IT'S EVEN DIVIDE IT BY TWO AND IF IT'S ODD MULTIPLY IT BY THREE AND ADD ONE, AND YOU REPEAT THIS PROCEDURE LONG ENOUGH, EVENTUALLY YOUR FRIENDS WILL STOP CALLING TO SEE IF YOU WANT TO HANG OUT.

The collatz conjectures is quite different process, you start with some number. If it is even you divide by 2, if it is odd you multiply it by 3 and add 1 to it and you continue. So, this is an interesting process and the conjectures states that eventually you will end up in the ending state being 1. So, why is the ending state 1? Because well once you reach 1 it is an odd number times 3 plus 1. You are going to go back going to 4, 4 is even. So, you come back to 2 and then back to 1 and so on and so forth.

So, you have essentially started cycling this process both the conjectures states that any where you start you are eventually going to come to this cycle of 3 numbers and this is been quite interesting for a long time, but as the cartoon indicates, you probably do not want to get to access with that. In fact, this more interesting process is that we can talk about well, what could be more fun than the collatz conjecture? Well, what is the type of process there are drunk man would take while walking on a street, turns out that might have quite interesting properties in on itself. So, that is what we are going to talk about today later not. Let us get to it, but let us motivate this by well known problem.

(Refer Slide Time: 01:58)



Boolean variable x can take one of two values: either 0 or 1.

When it appears in a logical formula (possibly in its negated form $\neg x$), we call it a *literal*.

We are interested in 2-CNF: Conjunction (AND) of clauses, where each clause is a disjunction (OR) of two literals. E.g.,

$(x_2 \vee \neg x_1) \wedge (\neg x_2 \vee x_4) \wedge (x_3 \vee \neg x_4) \wedge \cdots$

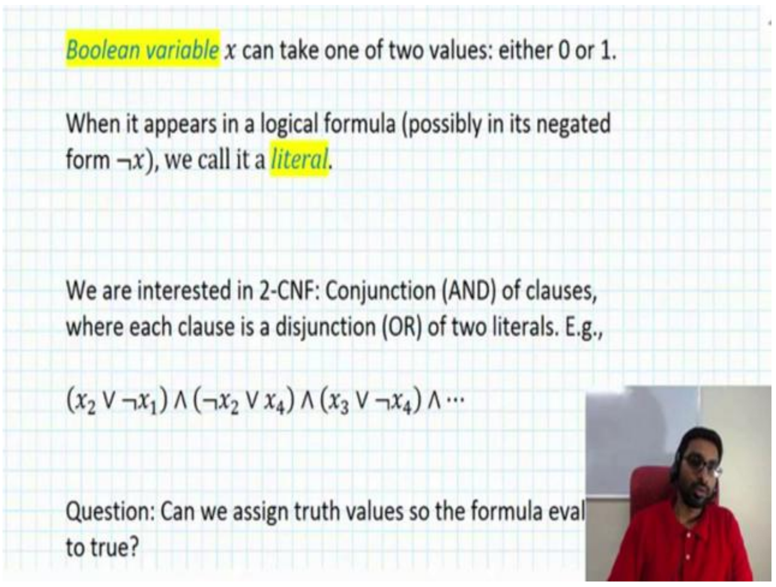
Question: Can we assign truth values so the formula eval to true?

We are going to talk about the 2 SAT problem and just talk about 2 SAT problem you know perceptual setup some definitions. So, what is a Boolean variable x it is the variable that can take 1 of the variables either 0 or 1, and we are interested in a logical formula and when the logical where the Boolean variable appears in the logical formula it is called a literal and of course, it can appear in 2 ways either the positive form or the negative form and particular in the 2 SAT or the k.

More generally in the k SAT problem we are interested in logical formulas that conformed to 3 CNF or algorithm CNF format, what is that? Well, it is a conjunction basically and of several clauses and each clause is a disjunction of literals in the context of 2 CNF each clause will have exactly 2 literals.

So, here we see an example of 2 CNF logical formulas and the SAT question is can be assigned truth values to the variable. So, that the formula will be validate to true another way of stating it is can we assign truth values. So, the formula can be satisfied.

(Refer Slide Time: 03:33)


Boolean variable x can take one of two values: either 0 or 1.

When it appears in a logical formula (possibly in its negated form $\neg x$), we call it a *literal*.

We are interested in 2-CNF: Conjunction (AND) of clauses, where each clause is a disjunction (OR) of two literals. E.g.,

$(x_2 \vee \neg x_1) \wedge (\neg x_2 \vee x_4) \wedge (x_3 \vee \neg x_4) \wedge \cdots$

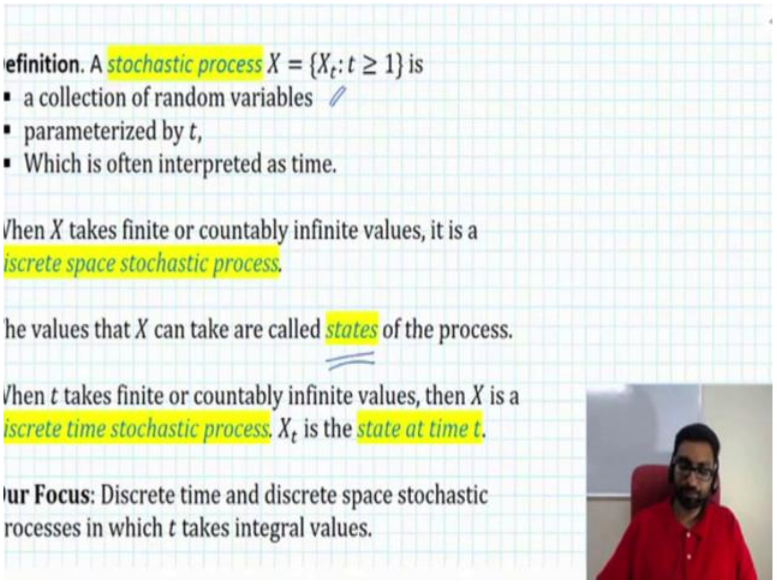
Question: Can we assign truth values so the formula eval to true?

So, here we see a very, very simple algorithm and I call it, “can you believe this works” algorithm because the first time I saw it is a little bit magical. So, here is the algorithm the input is of course, 2 CNF formula and there are n variables in it and you how do you find out whether there is a truth or satisfying truth assignment, well you start with some arbitrary truth assignment and you keep repeating this process in and you repeat up to 2 m n squared times, where m is a suitably large integer or until you find a clause is satisfied of course, if you find the clause as the satisfied you can terminate that is what we do here, but let us not jump ahead.

So, we repeat this process, in each time we repeat we choose an arbitrary unsatisfied clause. Suppose, if we have found truth assignment that satisfies the formula will be out of the loops. So, it is always going to be an unsatisfied clause if we are inside the loop and so we find one such clause and we look at the 2 variables in that clause remember this is 2 CNF.

So, there are 2 variables and which is pick one of them uniformly at random and flip it is truth value and then again check if the truth assignment is a satisfying assignment and if it is not be repeat this process and we repeat this for this many times; 2 m n squared times until we either exhaust the number of trials or we simply or we find the truth assignment and of course, we find the truth assignment we will return it, but if we do not find the truth assignment we are going to conclude that there is no satisfying truth assignment and this is where it seems a bit magical to me. Let us make some statements about this algorithm.

(Refer Slide Time: 05:59)



efinition. A stochastic process $X = \{X_t : t \geq 1\}$ is

- a collection of random variables
- parameterized by t,
- Which is often interpreted as time.

/hen X takes finite or countably infinite values, it is a iscrete space stochastic process.

he values that X can take are called states of the process.

/hen t takes finite or countably infinite values, then X is a iscrete time stochastic process. X_t is the state at time t.

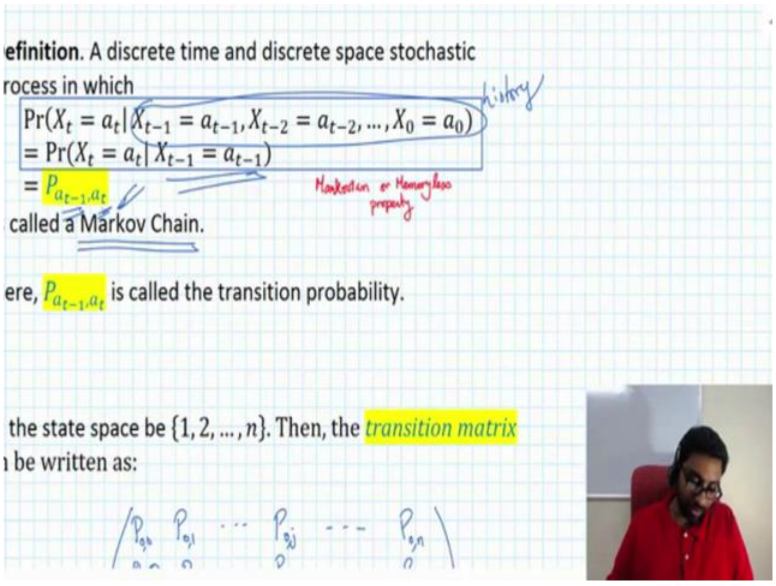
ur Focus: Discrete time and discrete space stochastic rocesses in which t takes integral values.

First of all it is quite clear that it is always correct when there is no satisfying truth assignment because in that case, we will always be exiting this loop having not found the satisfying clause and so we would be executing line four and saying that there is no truth assignment. So, that is not very surprising the surprising part comes when we have a satisfying assignment. If such satisfying assignment exists satisfying truth assignment will be reported with probability at least 1 minus 2 raise to the minus m and here m is an arbitrary numbers. So, you will notice that you can make this probability arbitrarily close to 1 and it of course approaches 1 exponentially fast in m. So, this seems surprises.

Let us actually try to figure out, why this is the case and for this we need to build a little bit of theory and this where it could help for us to understand this random process. So, let us start by defining a stochastic process is nothing, but a collection of a random variables and these random variables are parametrized by this parameter t, and the most useful way to interpret this parameter t is time. So, this variable therefore, can be thought of as the way a random variable changes over time and if this random variable x takes finite or countably infinite number of values then it is called a discrete space stochastic process.

Now, these values that x can take are often called the states of the process. So, you can think of this process as going from 1 state to another state to another state all over time, and we are most often interested in processes where the state space is discrete meaning it is finite or countably infinite and of course, the parameter t are also is often finite and or countably infinite and such processes, we call the discrete time stochastic processes and so we can interpret x t as the state at time t. Of course, our primary focus will be on discrete time and discrete space stochastic processes and t takes integral values starting from typically t equal to 0 to t to the 1 and so on.

(Refer Slide Time: 09:17)



efinition. A discrete time and discrete space stochastic rocess in which

$$\boxed{\begin{aligned}&\Pr(X_t = a_t \mid \underbrace{X_{t-1} = a_{t-1}, X_{t-2} = a_{t-2}, \ldots, X_0 = a_0}_{\text{history}})\\ &= \Pr(X_t = a_t \mid X_{t-1} = a_{t-1})\end{aligned}}$$

$= P_{a_{t-1},a_t}$

called a Markov Chain.

ere, P_{a_{t-1},a_t} is called the transition probability.

the state space be $\{1, 2, \ldots, n\}$. Then, the *transition matrix* ı be written as:

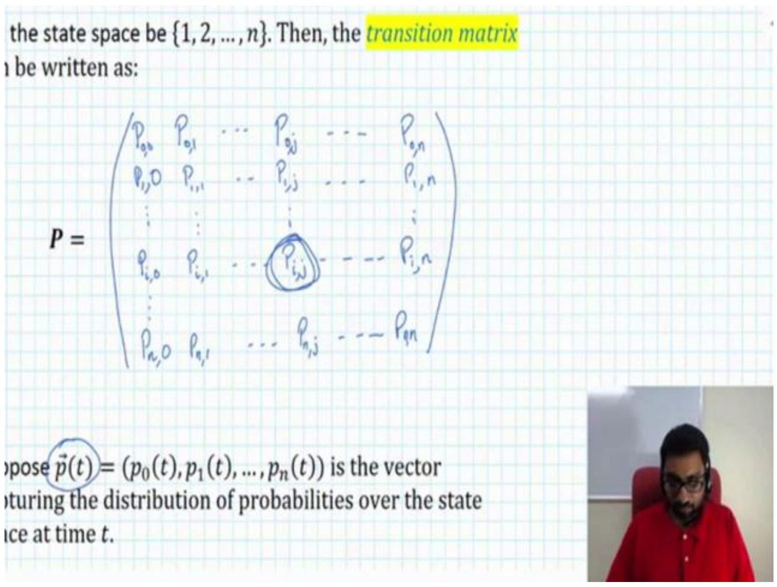
$$\begin{pmatrix} P_{0,0} & P_{0,1} & \cdots & P_{0,j} & \cdots & P_{0,n} \end{pmatrix}$$

We are interested in a discrete time and discrete space stochastic process that has an interesting memory less property and here is what that property is, let suppose we are interested in the event that the state of the process is at a t a time t. Well, how could that have reached that state well, obviously, depends on the past history and of course, if it might depend on the entire past history. So, that is this probability stated here and this is it is entire history.

Now, if this stochastic process was memory less in this probability condition on the entire history would end up being the probability that the state would be at a t th time t given only that is previous state and that is sufficient that x t minus 1 equals a t minus 1. So, the probability that the current state is a t only depends on where the process was in the previous time stamp and the history beyond that is has no bearing at all and therefore, the transition probability transitions. So, since they do not have any, since the history does not have beyond that 1 previous step does not have any bearing this probability get simply be written as P a t minus 1 to a t.

Because if you are at a t minus 1 at time t minus 1 then with this probability will be at state a t at time t and such a stochastic process that has this memory (Refer Time: 11:28) that forgets this past history beyond just the 1 previous step is called a Markov chain and these are very, very elegant and useful because they well the model a lot of real world processes and they are also elegant and easy to analyze and end up being quite useful for many algorithmic purposes. So, this probability P a t minus 1 to a t is called transition probability; basically, the probability that if you are at state a t minus 1, then you will move to the state a t with this probability.

(Refer Slide Time: 12:16)



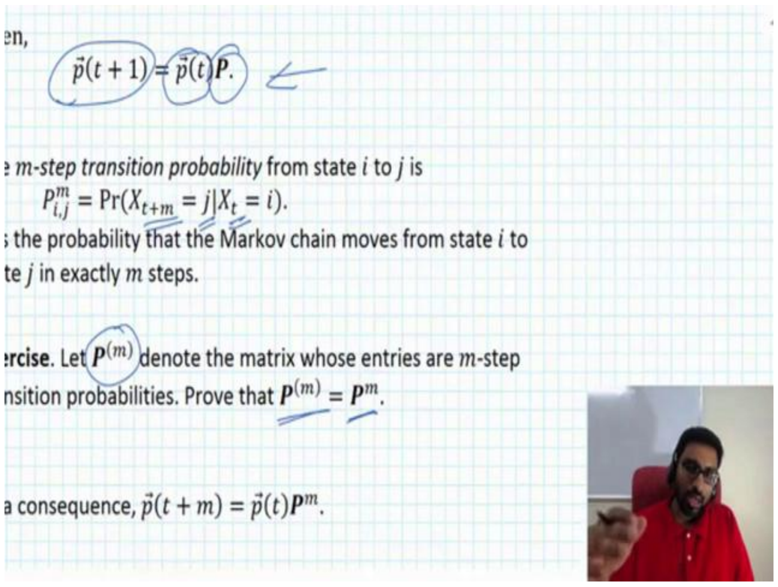
the state space be $\{1, 2, \ldots, n\}$. Then, the *transition matrix*
ı be written as:

$$P = \begin{pmatrix} P_{0,0} & P_{0,1} & \cdots & P_{0,j} & \cdots & P_{0,n} \\ P_{1,0} & P_{1,1} & \cdots & P_{1,j} & \cdots & P_{1,n} \\ \vdots & \vdots & & \vdots & & \vdots \\ P_{i,0} & P_{i,1} & \cdots & \boxed{P_{i,j}} & \cdots & P_{i,n} \\ \vdots & & & & & \\ P_{n,0} & P_{n,1} & \cdots & P_{n,j} & \cdots & P_{nn} \end{pmatrix}$$

ıpose $\boxed{\vec{p}(t)} = (p_0(t), p_1(t), \ldots, p_n(t))$ is the vector
ıturing the distribution of probabilities over the state
ıce at time t.

Let us now consider the Markov chain with a finite state space. In particular, let us say that it has n states and their number from 1 to n then it is probability it is transition probabilities can be captured by transition matrix in which is an n by n matrix. If you look at the entry i comma j it is a probability and denotes the probability that you would reach state j in 1 step given that your previous step was state i. So, let us denote the probability distribution over the state space by this vector of probabilities and here p 0 of t is the probability that the Markov chain will be at state 0 at a time step t state 1 at time step t and so on up to state n at time step t.

(Refer Slide Time: 13:26)



en,

$\vec{p}(t+1) = \vec{p}(t)\boldsymbol{P}.$

e *m-step transition probability* from state i to j is

$P^m_{i,j} = \Pr(X_{t+m} = j | X_t = i).$

; the probability that the Markov chain moves from state i to te j in exactly m steps.

ercise. Let $\boldsymbol{P}^{(m)}$ denote the matrix whose entries are m-step nsition probabilities. Prove that $\boldsymbol{P}^{(m)} = \boldsymbol{P}^m$.

a consequence, $\vec{p}(t+m) = \vec{p}(t)\boldsymbol{P}^m$.

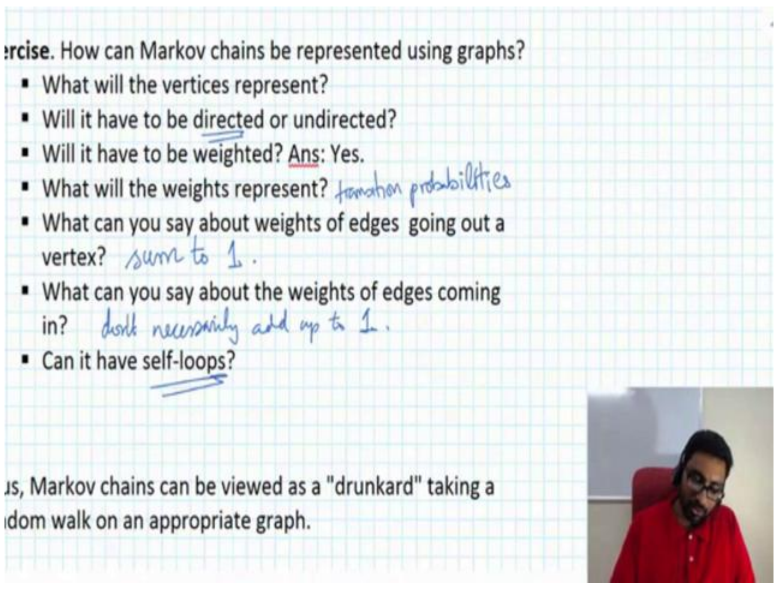
Notice that when you multiply the vector the distribution vector with the transition matrix, we are going to get the distribution vector at time t plus 1, we can extend this further. So, the m step transition probability from i to j is defined as follows. Now, suppose you are at state i at sometime t, this m step transition probability gives you the probability that at times t plus m you will be at state j. So, it is the probability that the Markov chain will move from state i to state j in exactly m steps and so this, let us consider the matrix whose entries are these m step transition probabilities and we are going to denote by p superscript m within precise.

Now, using the fact that if you take the current probability distribution and multiply it with the transition matrix, you will get the probability distribution at time t plus 1; using this formula a good exercise to be to show that this transition matrix whose entries of the m step transition probabilities can be obtained simply by taking the transition matrix and raising into the power m quite naturally as a consequence. If you take the probability distribution sometime t and multiply it with the truth transition matrix raise to the power m you will get the distribution probability distribution at time t plus m.

So, let us quickly recap we define what we call a stochastic process it is basically of course, we are interested in discrete time state space stochastic processes. So, you can interpret this as a random variable that takes various states through time and we are particularly interested in Markov chains, which are stochastic processes that exhibit the memory lessness property. So, there is a clear probability with which you transition from 1 state to another and you can also make these large m step transitions, well eventually they are just the probably transition matrix raise to the power m.

So, you can view this m step transition probabilities themselves as the sort of a Markov process. So, notice that the key thing here is in general going from 1 state to another and the transition probabilities provide you the probabilities with which you go from 1 state to another and so, quite naturally we can view this whole Markov chain, we can we can view a Markov chain as a graph.

(Refer Slide Time: 17:43)



rcise. How can Markov chains be represented using graphs?

- What will the vertices represent?
- Will it have to be directed or undirected?
- Will it have to be weighted? Ans: Yes.
- What will the weights represent? transition probabilities
- What can you say about weights of edges going out a vertex? sum to 1.
- What can you say about the weights of edges coming in? don't necessarily add up to 1.
- Can it have self-loops?

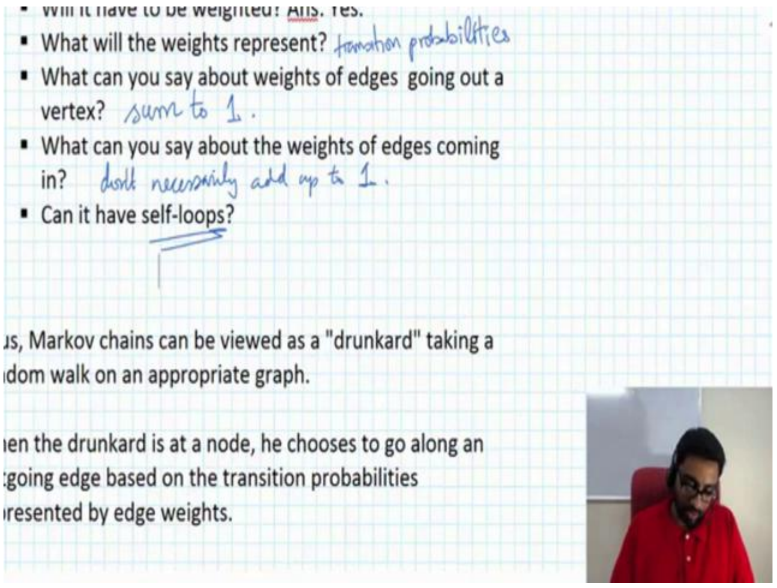
ıs, Markov chains can be viewed as a "drunkard" taking a ıdom walk on an appropriate graph.

So, quick exercise for you would be to think of Markov chains and think about how we can use graphs to represent Markov chains. So, for example, what will be the vertices of course, this is not a rocket science, it is your vertices are going to be the set of states that the Markov chain can take and, is this graph that represents the Markov chain going to be directed or undirected? Well, typically it is going to be directed because the probability of going from 1 state to another need not be the probability with which you return, come back from the other state back to the original state and therefore, you may want to distinguish the directions.

So, you probably want a directed graph and will it be weighted? Yes, because you probably want to capture the probability the transition probabilities are using weights and what can you say about the weights of edges going out of a vertex well notice that these are probabilities.

So, they will have to sum to 1, on a similar wing what can we say about the probabilities or rather the weights of edges that are entering into a vertex, will they also add up to 1? Think about it, here they do not have to add up to 1 because you could be entering into a vertex from a variety of different other states of vertices and there is no reason for the weights on the incoming edges to add up to 1 and of course, can the graph have self loops? Yes, because with there is nothing that says that the Markov chain should necessarily move to another state at each time step there you could have nonzero probability with which the Markov chain remains in it is current state and these can be normal using a self loops.

(Refer Slide Time: 20:34)



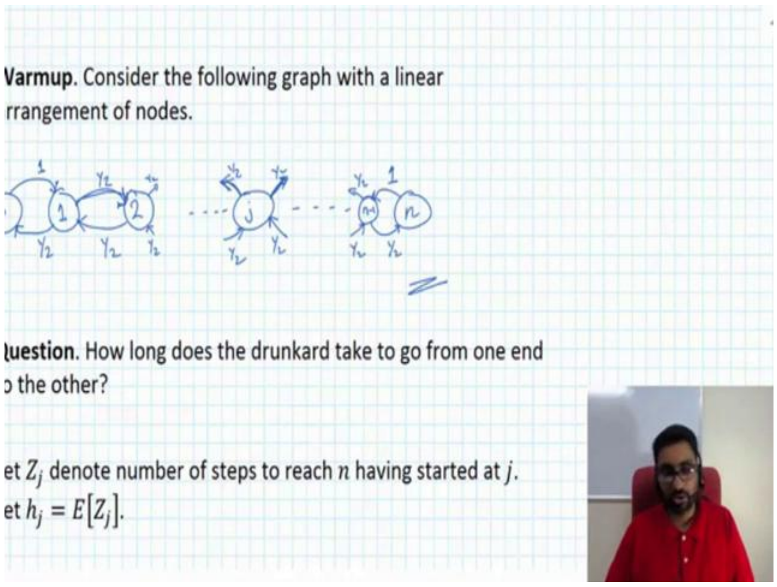
- Will it have to be weighted? Ans. Yes.
- What will the weights represent? transition probabilities
- What can you say about weights of edges going out a vertex? sum to 1.
- What can you say about the weights of edges coming in? don't necessarily add up to 1.
- Can it have self-loops?

us, Markov chains can be viewed as a "drunkard" taking a
idom walk on an appropriate graph.

ien the drunkard is at a node, he chooses to go along an
going edge based on the transition probabilities
resented by edge weights.

Essentially, this Markov chain can be viewed as drunkard taking random walk on this on this graph and if he is add a node in the graph then he will move to a neighboring node in to the probabilities the weights on the outgoing edges.

(Refer Slide Time: 20:51)


Varmup. Consider the following graph with a linear rrangement of nodes.

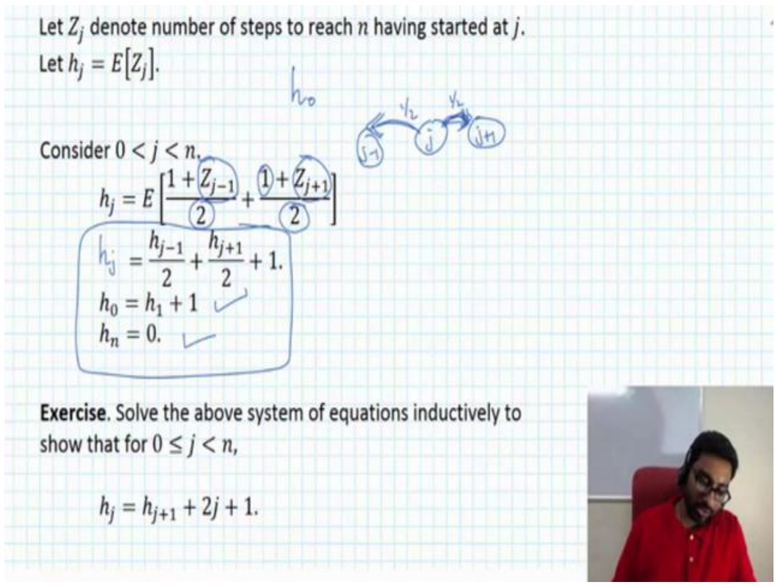
luestion. How long does the drunkard take to go from one end o the other?

et Z_j denote number of steps to reach n having started at j.
et $h_j = E[Z_j]$.

So, here is the simple warm up, let us consider a Markov chain in which the states can be arranged. So, if the state range from 0 to n and they are arranged in a linear fashions starting from 0 here to n over here, and if you notice at any intermediate state with probability have you move it forward. So, for example, if you are at state j you will go to state j plus 1 with probability half and with state and with probability half you move to state j minus 1.

So, an interesting question that we can ask is, how long does the drunkard take to go from one end of this Markov chain to the others? So, let say he starts at state 0, how long does he take to reach state n and this question turns out to be quite interesting also quite useful.

(Refer Slide Time: 22:13)



Let Z_j denote number of steps to reach n having started at j.

Let $h_j = E[Z_j]$.

h_0
$\frac{1}{2}$
$\frac{1}{2}$
$j-1$
j
$j+1$

Consider $0 < j < n$.

$$h_j = E\left[\frac{1 + Z_{j-1}}{2} + \frac{1 + Z_{j+1}}{2}\right]$$

$$\boxed{\begin{aligned} h_j &= \frac{h_{j-1}}{2} + \frac{h_{j+1}}{2} + 1. \\ h_0 &= h_1 + 1 \quad ✓ \\ h_n &= 0. \quad ✓ \end{aligned}}$$

Exercise. Solve the above system of equations inductively to show that for $0 \leq j < n$,

$$h_j = h_{j+1} + 2j + 1.$$

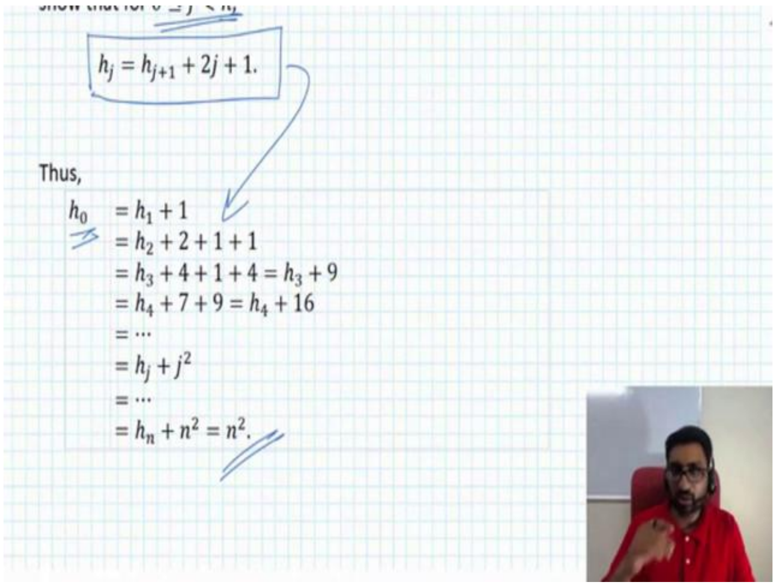
Let us use the random variables z j to denote the number of steps that the drunkard will have to take these random steps, to reach a state m having started at state j. Of course, if we since it are going to be doing random walk he could also be moving sometimes it will be moving backwards sometimes it will be moving forwards. So, we want to know how long it will take before he reaches n, and we use h j to denote the expected value of z j and we are particularly interested in h 0, which is the expected number of steps that this drunkard will take to reach n having started at the very end very, very extreme other end it starts at the state 0. So, if you consider any intermediate state j, let us look at h j, from state j there are 2 possibilities with probability half, we can go to state j plus 1.

So, with probability half we make 1 step go to state j plus 1 and then you can simply count the number of steps to go to state n from j plus 1. So, this is the probability half and with that probability half, we make that 1 set to j plus 1 not this 1 over here and then having reached j plus 1, we are going to take z j plus 1 number of steps to reach state n. So, this is you counted for half of the probability.

Now, with the other half of the probability we are going to move to state to j minus the one. So, that is the other half and we are spending that 1 step moving to j minus 1 and having move to j minus 1 will spend z j minus 1 number of steps to make our way to state n. So, h j can be written as the expectation of these 2 random variables waited in this manner. So, with the probability half you use that 1 step going to j plus 1 plus the time goes it takes to go from j plus 1 to n and the probability half you take 1 step to state j minus 1 and then the amount of time it takes to reach state m and the expectation of these 2 possibilities is going to be h j and of course, this can be expanded out to h j minus 1 divided by 2 plus h j plus 1 divided by 2 plus 1.

So, that is the quick exercise for you to figure that out and of course, if you are at state 0 you really cannot go to state negative one there is no such state in this particular Markov chain. So, the only option you have is to spend the one time step going to state 1 and from there h 1 captures the time it takes to the expected time takes to reach the state n. So, that is one of the base cases and the other base case is h n if you have already reached state n then you are already there as a 0 time steps you will reach a state n. So, h n is equal to 0. Now, what we have is a system of linear equations. So, we have h j is for intermediate values of j and h 0 and h n.

(Refer Slide Time: 27:13)



Show that for $0 \le j < n$,

$$\boxed{h_j = h_{j+1} + 2j + 1.}$$

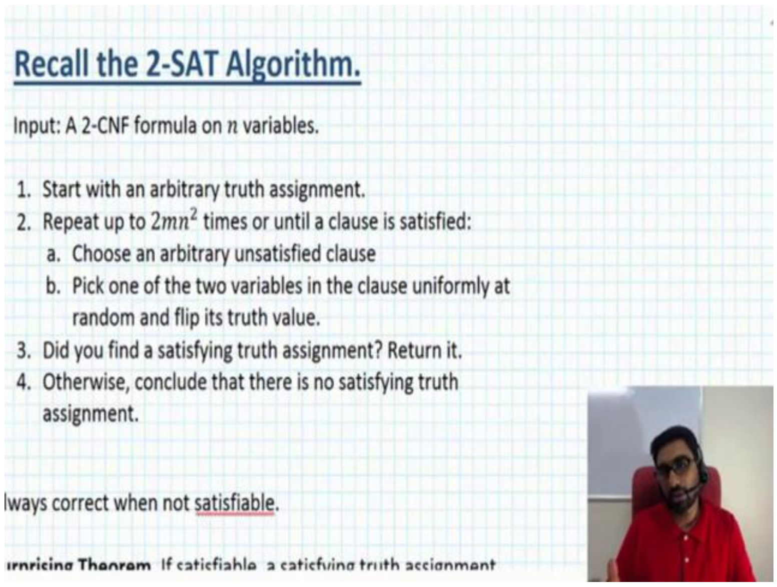
Thus,

$$\begin{aligned} h_0 &= h_1 + 1 \\ &= h_2 + 2 + 1 + 1 \\ &= h_3 + 4 + 1 + 4 = h_3 + 9 \\ &= h_4 + 7 + 9 = h_4 + 16 \\ &= \cdots \\ &= h_j + j^2 \\ &= \cdots \\ &= h_n + n^2 = n^2. \end{aligned}$$

And a simple exercise for you using induction would be to prove that h j for j ranging from 0 to up to n minus 1 equals h j plus 1 plus 2 j plus 1. So, this is basically as the solution to this equations using simple reduction and applying the base case, we can actually work out the probability or rather we can work out the expected number of steps that the drunkard will take to reach state n having started at state 0 that is our h 0, well we know that h 0 equal to h 1 plus 1, but applying this formula inductively h 1 evaluates to h 2 plus 2 plus 1 and so on.

If we work our way we end up with h 0 equaling n square. So, what this says is if you start at state 0 and make these random walk transitions, you will reach state n in an expected n squared steps. So, this is quite interesting, but why is it useful? Well, let us go back to an algorithm that we looked at for 2 SAT.

(Refer Slide Time: 29:03)



Recall the 2-SAT Algorithm.

Input: A 2-CNF formula on n variables.

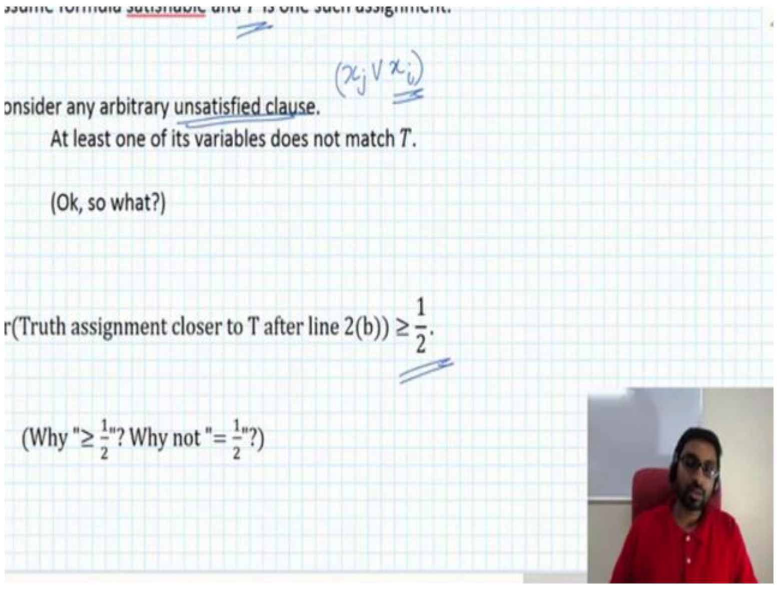
1. Start with an arbitrary truth assignment.
2. Repeat up to $2mn^2$ times or until a clause is satisfied:
   a. Choose an arbitrary unsatisfied clause
   b. Pick one of the two variables in the clause uniformly at random and flip its truth value.
3. Did you find a satisfying truth assignment? Return it.
4. Otherwise, conclude that there is no satisfying truth assignment.

lways correct when not satisfiable.

ırprising Theorem. If satisfiable, a satisfying truth assignment

So, we have the input 2 CNF formula with on n invariables. We started off with an arbitrary truth assignment and we iterated up to 2 m n squared times or until we found a satisfying truth assignment, and every time we can find truth assignment we picked an arbitrary clause and randomly pick 1 of the variables and plug it only, we kept doing this process until we either found the truth assignment within 2 m n squared time steps or we just declare that.

(Refer Slide Time: 29:50)


ssume formula satisfiable and T is one such assignment.

$(x_j \vee x_i)$

onsider any arbitrary unsatisfied clause.

At least one of its variables does not match T.

(Ok, so what?)

r(Truth assignment closer to T after line 2(b)) $\geq \frac{1}{2}$.

(Why "$\geq \frac{1}{2}$"? Why not "$= \frac{1}{2}$"?)

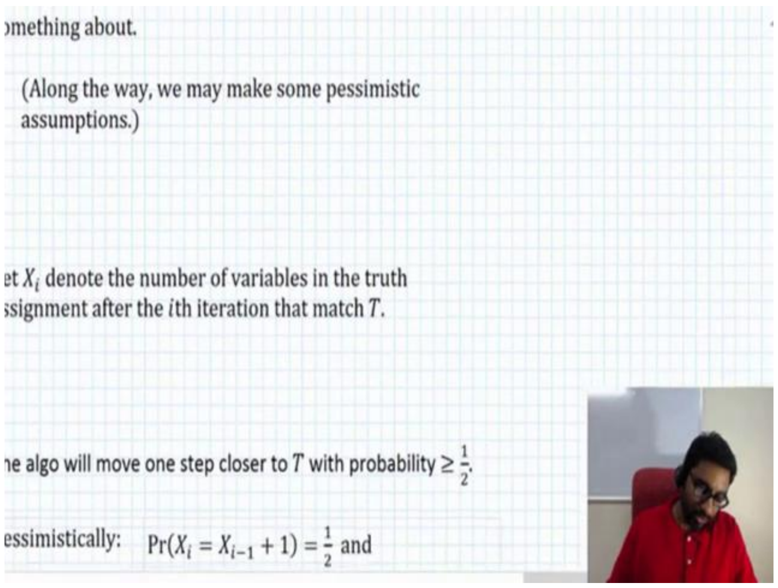
There is no satisfying truth assignment and we claimed that this algorithm, if it has the satisfying, if the input formula has a satisfying assignment will find that satisfying truth assignment with overwhelmingly large probability. So, basically probability of 1 minus 2 raises to the minus m and why is it truth? What is going to quickly sketch the proof and will see how Markov chains are useful in this process.

So, assume I mean, notice that if the formula is not satisfiable then the algorithm is always going to be correct. It is always going to say well there is no truth assignment. So, we are only interested in the case where there is a satisfying truth assignment and So, let us just pick 1 such satisfiable truth assignment let us call that t and let us look at the execution of this algorithm at some point in time, and if you consider any unsatisfied clause at least 1 of it is variables does not match t and this is quite obvious if why is that well, let us look at an unsatisfying clause of the form, let say x j or x i and if the let say both x j and x i match the truth assignments they have in t well then this clause will be a satisfying clause.

So, both of them cannot be matching their assignments in t. So, at least 1 of them will have to be different from their truth value in t well. So, this is not very surprising, but why is this important and why this is useful? Well, it is very useful because now we can say that if we choose such an unsatisfying clause which is what the algorithm does and chooses this 1 of these unsatisfying clauses and fix 1 of these variables uniformly at random and flips their value.

So, if let say x i is the variable whose truth assignment current truth assignment does not match t then it probability half, we are going to be flipping x i's value and. So, with that probability half, we are going to move the truth the current truth assignment 1 step closer to t of course, it is also possible that you know both x i and x j are different from their assignments correct assignments are different from the assignment in t in which case the this probability actually because 1 regardless of which variable we choose we are going to get closer to t and so that is why this probability to the truth assignment is closer to t after lying to be that is this line to be in the algorithm where we pick a variable uniformly at random and flip it. So, the probability that truth assignment gets closer to t after line 2 b is greater than or equal to half.

(Refer Slide Time: 33:49)



›mething about.

(Along the way, we may make some pessimistic assumptions.)

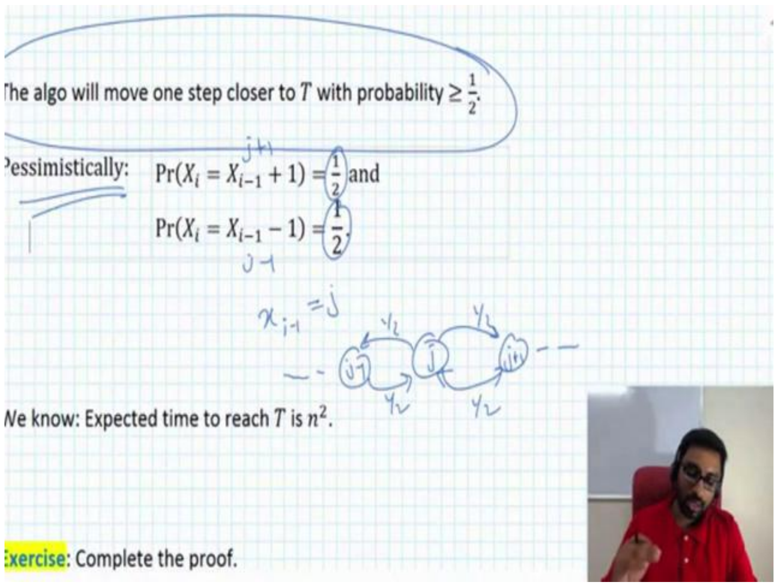
et X_i denote the number of variables in the truth ssignment after the ith iteration that match T.

ıe algo will move one step closer to T with probability $\geq \frac{1}{2}$.

essimistically: $\Pr(X_i = X_{i-1} + 1) = \frac{1}{2}$ and

Now, let us try to take insight and frame this into a notion that we know, basically a random walk in along the way what we are going to do is we are going to make some simplifying assumptions, but we are always going to make pessimistic assumptions. So, that whatever claim we have at the end is going to only be worse than the actual truth.

(Refer Slide Time: 34:24)


he algo will move one step closer to T with probability $\geq \frac{1}{2}$.

essimistically: $\Pr(X_i = X_{i-1} + 1) = \frac{1}{2}$ and

$\Pr(X_i = X_{i-1} - 1) = \frac{1}{2}$.

e know: Expected time to reach T is n^2.

xercise: Complete the proof.

So, let x i denote the total number of variables in the truth assignment after the ith the iterations that match t basically why do we need this x i because we want to be able to capture how close we are to matching t completely. So, at time step i, x i denotes the number of variables in which our current truth assignment matches the assignment t. We have already seen that the algorithm will move 1 step closer to t with probability greater than or equal to half.

So, we are going to modify this just a little bit. So, we are going to be a bit pessimistic about setting up the random walk. Let us assume that x i minus 1 equal to j. In other words, at time step i minus 1, we have j matches with the truth assignment t. Now, what is the probability that x i equals j plus 1 we are going to assume that is half and what is the probability that x i equals j minus 1 and it is also equal to half. So, from state you can pick up this as random walk in which with from j with probability half we go to j plus 1 with probability half we go to j minus 1 and this extends for all j. So, with probability half we will go from j minus 1 to j with probably half or j to j j plus 1 to j, 1 and so on. So, this exactly matches the random walk on a line that we have already seen before.

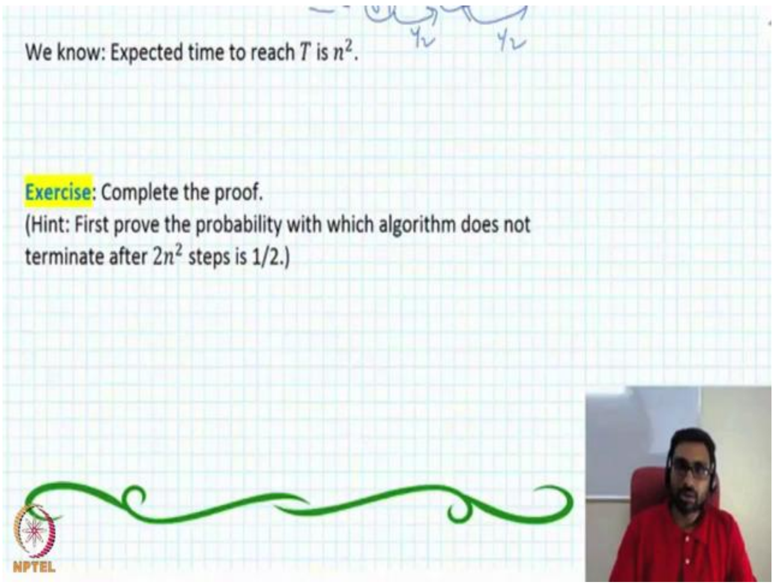
So, what we know is that this pessimistic Markov chain is actually terminates in n square steps on expectation. So, this is what we are going to use crucially to prove that the algorithm. In fact, finds the satisfying truth assignment with such overwhelming the high probability. So, that is the simple exercise for you. So, you basically have to use the

Markov inequality to prove that the probability with which the algorithm does not find the truth assignment after 2 n n square steps is at most half and then when we repeated n times we can reduce this probability down to 2 raise to the minus m which gives us the result that we need want.

So, before we conclude I would like to point out one thing, why is it that we are not using this probability? Instead we are going to choosing a more pessimistic option, well as it turns out if we view this stochastic process using the actual probabilities they are not it is not a Markov chain the probability with which we transition to a particular state with certain number of matches with the truth assignment t may depend on many historical facts. So, we do not want to; we simply do not have the tools to analyze it as elegantly as we can if we view this process as a Markov process Markov chain.

So, that is the reason why we are being a bit pessimistic, but in return we are getting the argument to form that we can analyze more elegantly.

(Refer Slide Time: 39:04)



We know: Expected time to reach T is n^2.

Exercise: Complete the proof.
(Hint: First prove the probability with which algorithm does not terminate after $2n^2$ steps is 1/2.)

So, that brings us to the end of our discussion on Markov chains at least the first segment and we will be talking little bit more about Markov chains in subsequent segments, and hopefully we will also get to the point, where we discuss a big data application that uses.