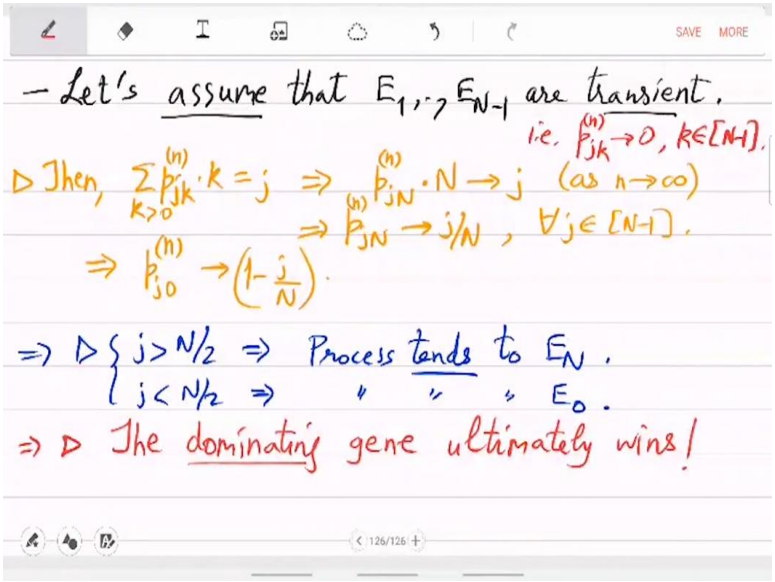
Probability for Computer Science
Prof. Nitin Saxena
Department of Computer Science and Engineering
Indian Institute of Technology - Kanpur

Module - 6
Lecture - 24
Random Sampling

I do not know how to give a general argument, but let us simplify and try to use the equations of this martingale Markov chain. So, let us assume that E 0 and E n are the only two closed sets, because they are absorbing states; all the others, E 1 to E N - 1, they are transient. So, their probabilities in the end, probability of reaching to transient is 0. So, what happens if you assume that?

(Refer Slide Time: 00:47)



So, let us assume; it is an assumption that E 1 to E N - 1 are transient, which will immediately tell you that P jk tends to 0 for k 1 to N - 1. So, if this happens, if P jk tends to 0 for all these states, then, using this martingale property, Sigma P jk times k = j, what you can deduce is, Sigma P jk times k = j implies that; and notice that k here is positive; k = 0 does not contribute; so, it is really k positive.

So, this implies that, and also after a long time, right? So, this means that, when you look at P jN times N, this tends to the whole thing, as small n tends to infinity. It is in the limit. The only term that is contributing here is P jN. So, P jN - 1 ... P j1 and also P j0, they are not

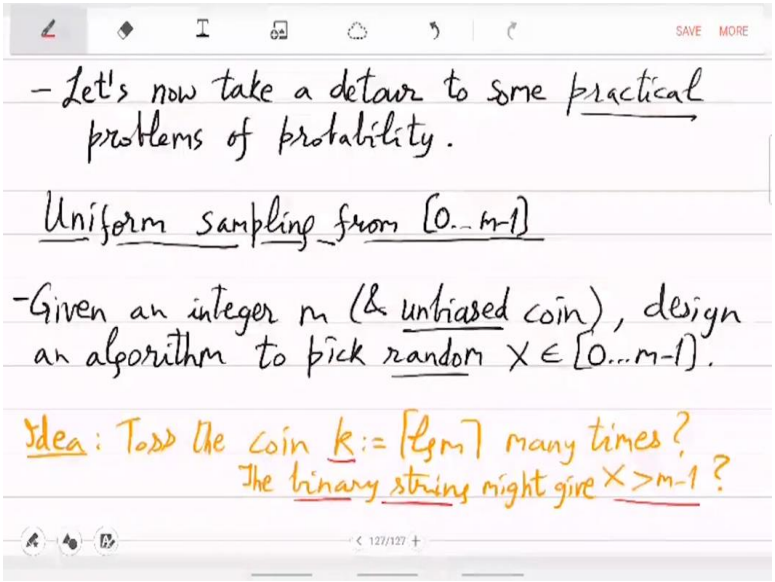
contributing anything in this expectation. So, you get this from which you deduce that P jN is tending to j by n for all j, 1 to N - 1.

So, if you are in a transient state, if you start from a transient state j, then the chance of reaching E N, E big N after a long time is exactly j by big N. And also p, if you start it from 0, then it is 1 - j by N. So, we have, these two probabilities are positive, the other probabilities; sorry, it is not like this; j0, going from j to 0 versus going j to big N; we deduce that. And what this means is that, if j is greater than n by 2, then you are reaching E N, the process tends to E N higher, because this j by N is greater than half.

So, it is actually, with more than half probability, you will be reaching E N. On the other hand, if j is less than N by 2, then with probability more than half, it will be tending to E 0. So, if A-types are more, then A-types will dominate and they will become, all the particles will become A-type. If A-types are less, then in the end, all the particles will become B-type. So, what you learn is that, the dominating gene ultimately wins.

Either everything becomes A-type or everything becomes B-type. That is what we have learnt. So, this is assuming that this simplification that other states are transient. So, you do not get stuck in them. In the end, you are either in E 0 or E N. So, that finishes this outstanding example of Markov chain in cell genetics, and you can study many such examples. This is a simplified model of evolution.

(Refer Slide Time: 06:02)

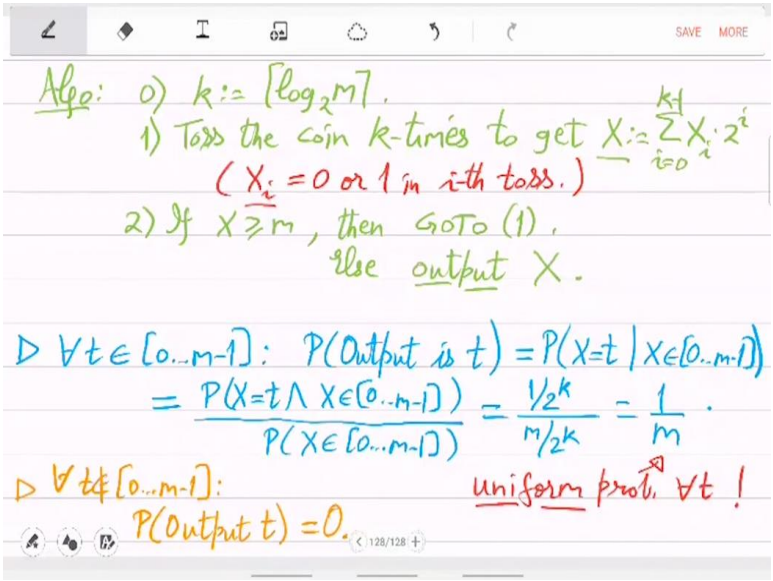


So, let us see some other practical examples, which are not really based on Markov chains. So, let us now take a detour to some practical questions of probability. So, in practice, how do you create a probabilistic process? So, the first example is from uniform sampling, from 0 to m - 1. So, when you say that in an algorithm you want to pick a random element less than m, which is a non-negative integer, how do you do that?

In an algorithm, how do you actually pick a random element? So, you can always assume that there is a coin to be flipped; that is there. But using coin flipping, how do you achieve this? That is a practical question. So, given an integer m and a unbiased coin, design an algorithm to pick a random number X, in this range? How do you do this by using unbiased coin flips? So, the idea here is that you toss the coin log of m to base 2, many times.

That is the idea, but the problem is that this will give you k bits, but this is more information, this might be more information than you are asked. So, if you look at these k bits and look at the number which this binary string represents, that may be a number more than m. So, you are not in this range. So, how do you get in this range by so many coin flips? The problem is that, this might be extra, this X might be more than m - 1. So, the binary string might give this X > m - 1. So, what will you do then? How do you come back, how do you come back to this range? So, let us see that algorithm.

(Refer Slide Time: 10:05)



So, first you compute k to be log of m to base 2. Then you toss the coin k times to get X which is the number out of this binary string. So, head is 1, tail is 0; and that gives you 1 and 0, and that gives you a k bit string. So, the bits are X i's. X i is 0 or 1 in the ith toss. So, you

have gotten this number X, but as I said, this number may be more than m; so, it does not help. So, what you do is that, in that case, if X is exceeding m - 1, then you throw it away; repeat the experiment; so, go to 1; else, you output this X.

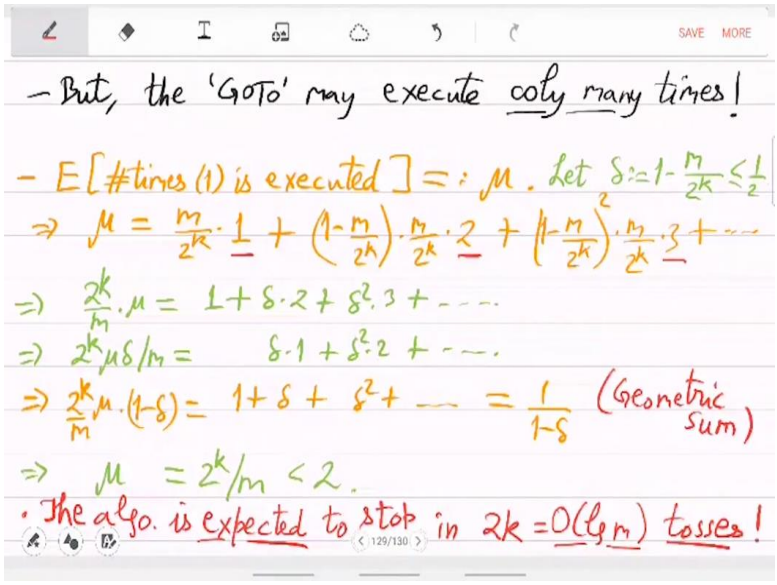
So, if it is in the range, output it; if it is outside the range, then throw it away, repeat the experiment. That is a simple algorithm. Why does it work? So, let us see for t in the range 0 to m - 1; let us see what is the probability of output is t. So, what is the probability that the algorithm outputs t? So, this probability is that X comes out to be t, which is subject to X in the range 0 to m - 1.

So, given that the algorithm is outputting, what is the chance that it is t? That exactly is the probability you are interested in; because, if X is not in 0 to m - 1, then there will not be an output. And that comes out to be; so, this is equal to probability X is t and X is in 0 to m - 1 divided by the probability that X is in 0 to m - 1. So, the first event is just that probability is X equal to t, which is, what is the chance that this number is picked?

That is, there are 2 raised to k possibilities out of which only 1 is favourable. So, it is 1 over 2 raised to k. And X falling in this 0 to m - 1 range, that is m possibilities out of 2 raised to k. So, you get 1 by m. So, this is uniform probability. So, this process will give you a t, any t in the range, with equal probability. There is no difference; that is the point. By changing t, you are not increasing the probability or decreasing it.

And it is also true that, for all t not in this range, the probability of output being t is 0. So, if t is in the range, probability is 1 by N; if t is not in the range, then the probability of outputting is 0; because, then this step 2 will, then will be followed and it will go to, it will repeat step 1, and not output. But this does not seem to be a good practical algorithm, because there is a GoTo, it is an infinite loop.

(Refer Slide Time: 15:36)



So, but the GoTo may execute infinitely many times. That is very bad; it is an infinite loop. Maybe every time you do the experiment in step 1, it is outside the range, and then the algorithm is in infinite loop, the answer never comes. So, here, that can happen, but let us try to see then expected number of steps, expected number of GoTo. So, let us do that. So, expected number of times 1 is executed, let us call it mu.

So, you toss the first time, and next time, it will be, you have to do it again if the value was more than m - 1. So, in case it is in the range, which is, the chance is m by 2 raised to k. Out of 2 raised to k possibilities, X came out to be in the range which is only m possibilities; so, then, it is executed only once. Or, the first time it did not happen, it was outside the range; second time, it was in the range.

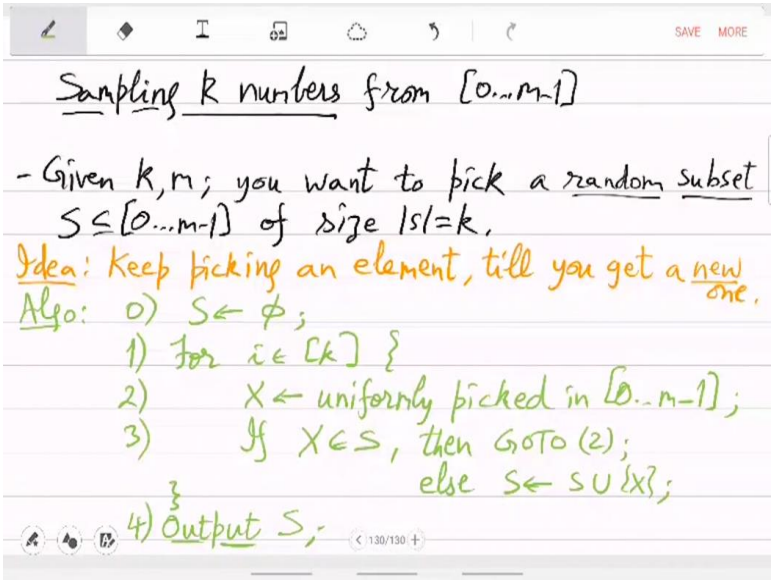
So, then it is 2 times, or the first two are bad, third time is good. It is an infinite sum; so, this will need some work. So, 1 time, 2 time, 3 time and then the respective probability. So, let us do this work. What do you get? So, let me define delta here. Let delta be 1 - m over 2 to the k. Now, the way we have defined k, m over 2 to the k will always be at least half. So, this is at most half. So, let me write this down.

So, I can write it as 2 raised to k by m mu is equal to 1 plus delta times 2 plus delta square times 3 and so on. And I can multiply by delta, this whole equation. So, I get 2 raised to k mu delta by m is equal to delta plus delta square 2 and so on. And then I take the difference of these two. So, if I take the difference, what do I get? 2 raised to k mu by m times 1 minus delta is 1 plus delta plus delta square. Delta is a fraction, so you do the geometric sum.

So, this gives you what? So, 1 minus delta is m over 2 raised to k. So, you get; on the left-hand side, you are left with mu, and on the right-hand side, you have 2 raised to k by m. And 2 raised to k by m is at most 2. So, that is what you get. That is the expectation. So, the GoTo or the, or step 1 in the algorithm, that is expected to run only 2 times. So, let us note that down. The algorithm is expected to stop in only 2 times, right? So, 2k coin tosses.

So, it is not the case that you expect this to run infinitely many times, in an infinite loop. It will actually, this step 1 will happen only twice; and in that, it will output X. So, this is only log n many tosses that you are doing. So, in log n many steps, you are able to sample in the range 0 to m - 1. So, this is the most basic algorithm that you will need in probabilistic algorithms.

(Refer Slide Time: 21:46)



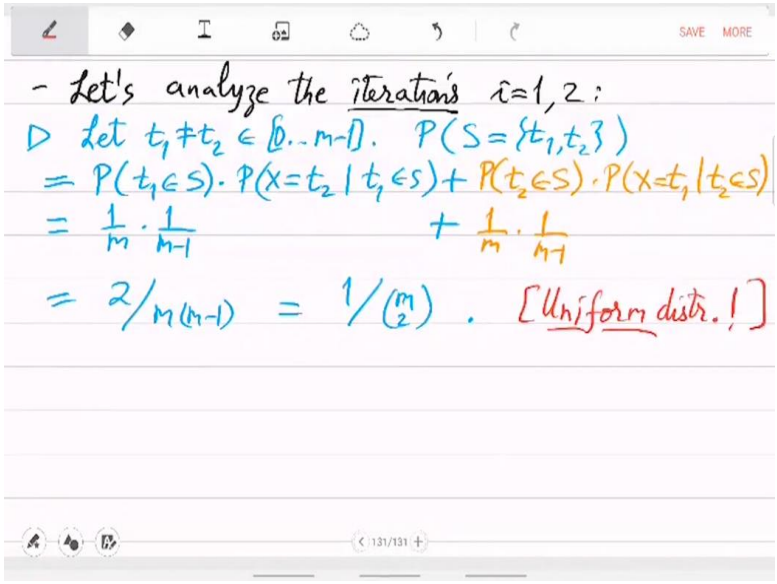
Let us now extend this a bit. What if you need k distinct numbers? What will you do then? Not just 1 number. So, sampling k numbers from the same range. You have seen a solution for k equal to 1, right? But now you want, say 2 numbers which are different. And you want these numbers to come out in a uniformly distributed way. So, given k and m, you want to pick a random subset S of 0 to M - 1 of size k. What is the idea now?

See, idea is, do something like you did in step 1. So, if the answer is not good, if the choice is not good, you repeat. So, when you are picking the second element, if it is the same as first element, you repeat. And third element, if it is the same as first or second, you repeat. So, keep picking an element till you get a new one. So, let us see this algorithm. It is very similar. So, you start with a empty set, and we will grow it slowly.

So, for the ith element, what you will do is; so, uniformly picked element in the range. So, use the previous algorithm. So, that will quickly give you a random element in the range; but that is only 1; so, you add this in S. Well, not so fast. First, you have to check. In general, you have to check whether it is a new element or not. So, if this element is already in S, then you go to 2.

If this is already in S, then it is not of any help; you have to repeat the uniform picking. Otherwise, it is a good element, it is a new element; so, you will add. And then you finish the for loop. In the end, you have k distinct elements in S, and you output this S. That is the simple algorithm to now sample k distinct numbers. So, what is the probability? And how many steps are required? Let us check that quickly.

(Refer Slide Time: 25:58)



So, let us analyse the iterations i = 1, i = 2, the first two iterations. So, consider 2 elements t 1, t 2 in the range. Now, let us look at the probability that set formed in the first two iterations has t 1, t 2; it is equal to t 1, t 2. What is that probability? So, this probability is equal to, in the first iteration, t 1 was found; and then t 2 was found; plus the symmetric one. So, probability that in the first iteration t 1 was found, since you uniformly picked in the range 0 to M - 1, that is 1 by M.

And then, after picking this t 1, the X will come out to be t 2, correct X; it will come out to be t 2 only when the choice is made from the M - 1 things. So, this is 1 over M - 1; otherwise, the key, I mean, this X will not be correct X; plus the same thing here. So, this becomes 2 over M M - 1, which is 1 over M choose 2. So, that is a uniform probability. So, for distinct t

1, t 2, this set coming out to be obtained in 2 iterations. It is actually, exactly what you wanted. We will finish this next time.