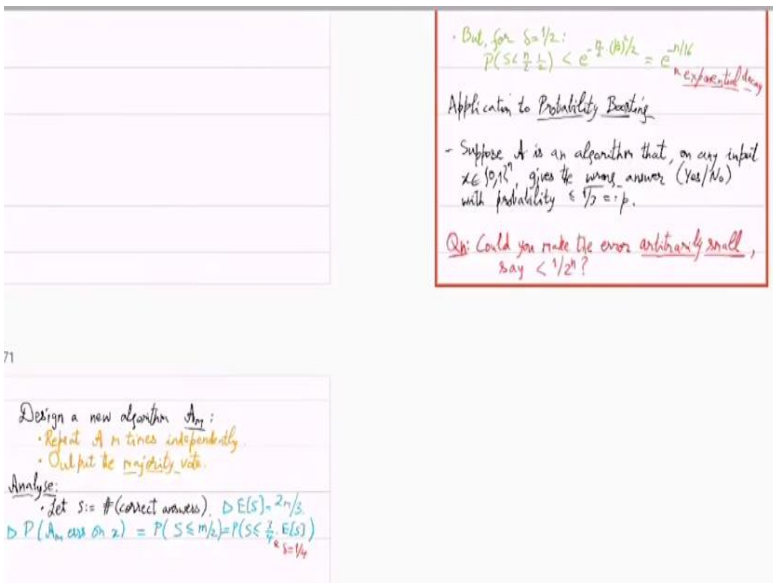
Probability for Computer Science

Prof. Nitin Saxena
Department of Computer Science and Engineering
Indian Institute of Technology – Kanpur

Lecture - 17
Union and Factorial Estimates

Last time, we were almost finishing this probability boosting application.

(Refer Slide Time: 00:18)



· But, for $\delta = 1/2$:
$P(S < \frac{n}{2} \cdot \frac{1}{2}) < e^{-\frac{n}{2} \cdot (\frac{1}{2})^2 / 2} = e^{-n/16}$ exponential decay

Application to Probability Boosting

- Suppose A is an algorithm that, on any input $x \in \{0,1\}^n$, gives the wrong answer (Yes/No) with probability $\leq 1/3 =: p$.

Qn: Could you make the error arbitrarily small, say $< 1/2^n$?

71

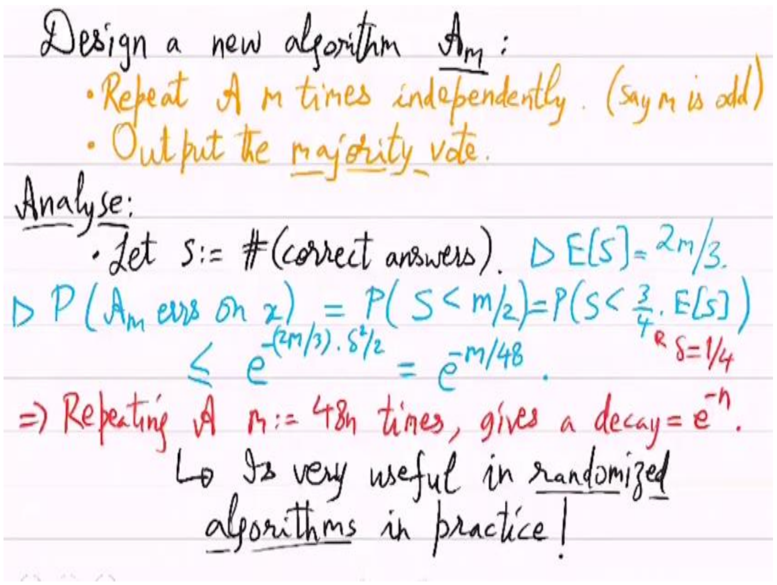
Design a new algorithm A_m:
· Repeat A m times independently
· Output the majority vote.

Analyse:
· Let $S :=$ #(correct answers). $\triangleright E[S] = 2m/3$.
$\triangleright P(A_m \text{ errs on } x) = P(S \leq m/2) = P(S \leq \frac{3}{4} \cdot E[S])$ $\delta = 1/4$

And before that we finished concentration inequalities. So, we did Markov, Chebyshev and then finally Chernoff. Chernoff has the strongest assumption and then the strongest result exponential decay of the probability. So, using that suppose, there is an algorithm A which on any input gives the wrong answer with chance less than one third. So, how can you make this error probability extremely small let us say 1 over 2 raise to n?

So, the success probability will be like 99.9999%. How do you do that? How do you go from 67% to 99.99%? So, we do it by repetition.

(Refer Slide Time: 01:14)



Design a new algorithm A_m:

- Repeat A m times independently. (say m is odd)
- Output the majority vote.

Analyse:

- Let $S :=$ #(correct answers). $\triangleright E[S] = 2m/3$.

$\triangleright$ $P(A_m \text{ errs on } x) = P(S < m/2) = P(S < \frac{3}{4} \cdot E[S])$

$\leq e^{-(2m/3) \cdot \delta^2/2} = e^{-m/48}$. & $\delta = 1/4$

$\Rightarrow$ Repeating A $m := 48n$ times, gives a decay $= e^{-n}$.

$\hookrightarrow$ Is very useful in randomized algorithms in practice!

Repeat the algorithm independently m times. m is yet to be fixed and then you output the majority vote. So, you have m yeses and noes. And you take the majority. So, here you could assume that m is odd. So, there will be some either yes or no in the majority. So, whatever it is you output that. So, now let us understand or let us analyze the error probability. So, S is the random variable which counts the number of correct answers.

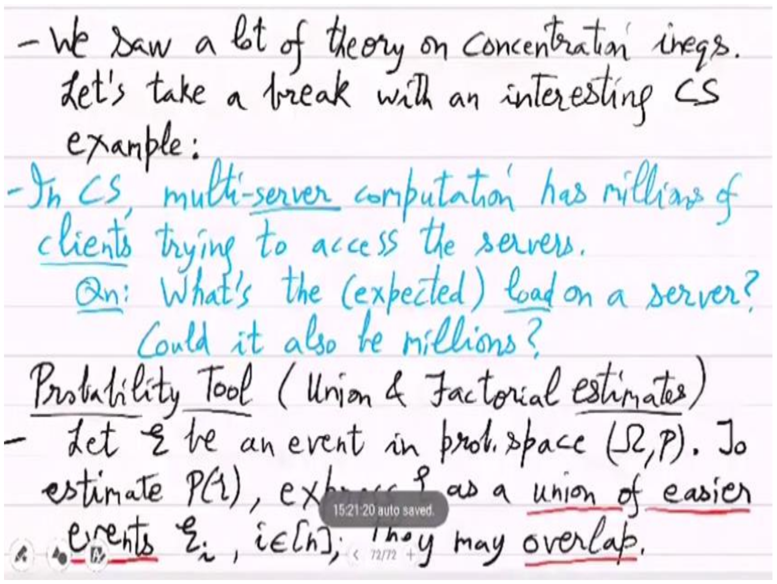
If the answer correct answer is yes, then how many yeses were obtained? So, that expectation is 2m by 3. This is simply by linearity of expectation. Every experiment 2 third you will get the correct answer. So, you get 2m by 3. And now, this new algorithm A m will be wrong if the number of correct answers is strictly smaller than half. It is in minority. So, the correct answer has gone in minority. What is the chance of that?

So, m by 2 means actually three fourth of the expectation which means that delta is 1 by 4, 1 minus 3 by 4. And this probability by Chernoff is e raised to the expectation which is minus 2m by 3 times delta square by 2, which is 1 by 32. So, you get e raised to minus m by 48. So, what should m be? Remember you wanted error to be 2 raised to minus n. So, you pick m to be 48n.

So, this means that repeating A m equal to 48n times gives a decay of e raised to minus n. So, this is a huge decay. So, if you are looking at input of size 100 then this is the decay of 2 raised to minus 100 or even close to 3 raised to minus 100. So, this is a negligible probability now for the error. So, that is why majority vote works very well. And independent experiment so you can apply Chernoff bound.

So, needless to say, this trick of probability boosting is very useful in randomized algorithms in practice. So, it is not just in theory but actually you can get very almost precise almost exact algorithms by this probability boosting. Of course, there will still be some chance of error but it will be extremely small. You can count on the result basically. Fine, so, till now, you we did a lot of theory on concentration inequalities.

(Refer Slide Time: 05:19)



- We saw a lot of theory on concentration ineqs. Let's take a break with an interesting CS example:
- In CS, multi-server computation has millions of clients trying to access the servers.
  Qn: What's the (expected) load on a server? Could it also be millions?

Probability Tool (Union & Factorial estimates)
- Let $\mathcal{E}$ be an event in prob. space (Ω, P). To estimate $P(\mathcal{E})$, express $\mathcal{E}$ as a union of easier events $\mathcal{E}_i$, $i \in [n]$; they may overlap.

15:21:20 auto saved.

And you can apply it in various ways. So, we applied it to (()) (05:45) this law of large numbers. You learnt about variance and you learnt about probability boosting. So, let us take a break now with a nice example with an interesting computer science example to do with servers and clients. So, in CS applications, so, multi-server, there are scenarios where there are many servers.

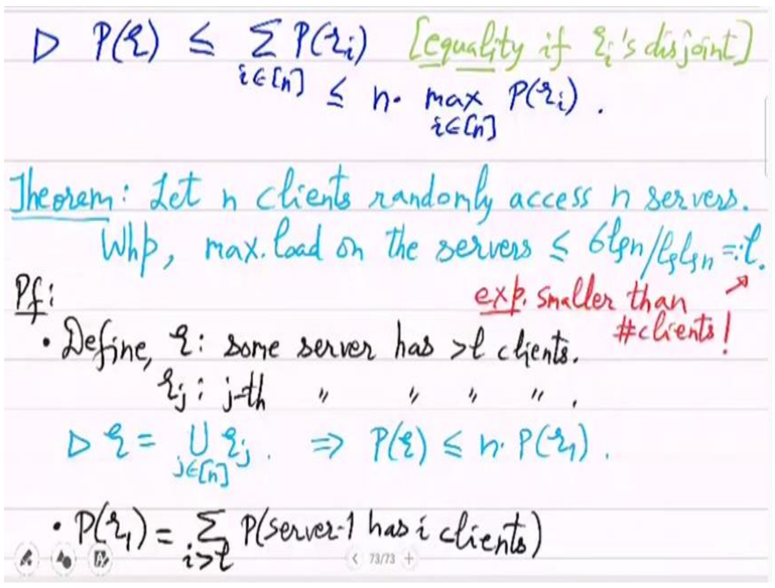
And many clients are trying to access them has millions of clients trying to access the servers. So, there are many servers and there are many clients. Clients number is huge. So, now the question is what is the load on a server? By which, of course, we mean expected load. Could it also be millions or could it be much smaller. So, we will show that surprisingly this load on one server.

Or, if you look at the max load on any server, that will be actually significantly smaller than the number of clients. So, if clients behave in a random way then the actually load will be distributed across the servers. So, let us do this. So, this will make you learn 2 important tools union and factorial estimates. So, you will see 2 estimates which are quite important in many applications.

Let us first prove in fact just note down that if you are interested in an event E in probability space. Omega is the sample space. P is the probability distribution function. So, of course, you want to estimate probability of E. So, what you can try is to break it up into many events that are simple. But there the main condition you want is their union should be E. So, express E as a union of easier events, E i.

Let us say n of them. And they may not be disjoint. So, union of easier events that is what you want to do. And they may also be overlapping but the point main point is that their union is equal to E. It is not a partition. (()) (10:42) this is different from partition formula. So, then what happens is probability of E is smaller than sum of the probability of E i's.

(Refer Slide Time: 10:51)



▷ $P(\mathcal{E}) \leq \sum_{i\in[n]} P(\mathcal{E}_i)$ [equality if $\mathcal{E}_i$'s disjoint]

$\leq n \cdot \max_{i\in[n]} P(\mathcal{E}_i)$.

Theorem: Let n clients randomly access n servers. Whp, max. load on the servers $\leq 6\lg n/\lg\lg n =: \ell$.

exp. smaller than #clients!

Pf: • Define, $\mathcal{E}$: some server has $> \ell$ clients.

$\mathcal{E}_j$: j-th " " " " .

▷ $\mathcal{E} = \bigcup_{j\in[n]} \mathcal{E}_j$. $\Rightarrow P(\mathcal{E}) \leq n\cdot P(\mathcal{E}_1)$.

• $P(\mathcal{E}_1) = \sum_{i>\ell} P(\text{server-1 has } i \text{ clients})$

Now, this will be an equality. So, equality if and only if E i's are disjoint. Otherwise, it will be an inequality because if E i's overlap then you have to subtract something to get to the correct event E, this. So, that was our inclusion exclusion principle. Let me not say if and only if let me just say if. So, if E i's are disjoint then you will have equality. Otherwise, you have this upper bound. But this can be good. Even upper bound can be enough.

You do not want exact probability. You just want to see the decay which is further smaller than n times the max probability here. So, as long as you understand all these events E i, you know the maximum so that n times the maximum is your probability of E upper bound. Fine, so, what we will show is this theorem. So, let n clients randomly access n servers. So, with high probability, max load on the servers is 6 log of n by log log n.

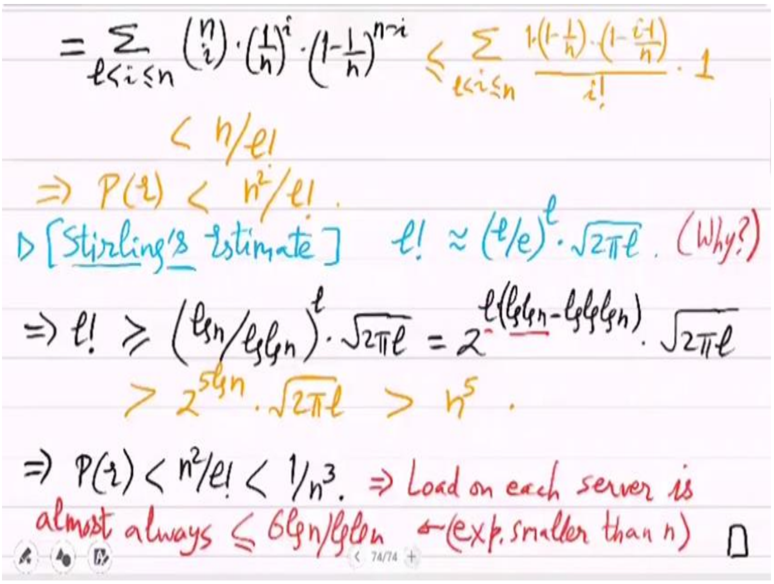
Let us call this expression l. So, this is essentially a logarithm of n. The number of clients, clients are n but with very high probability actually very few of them will come to a server. They the randomness will actually try to distribute them. And we are giving this precise quantitative estimate which is log n further smaller by log log n factor. So, this can be for n million. This can be extremely small.

So, the load gets highly distributed. So, let us try to prove this. This should be a bit surprising. The point here is that this is exponentially smaller than number of clients. So, that is why it is surprising that why is there a log. Why not something even smaller or something bigger? So, proof will be actually quite nice and simple using these 2 estimates. So, let us define E to be the event that some server has greater than l clients.

And E j be the event that this happens with jth server. Now, obviously goes without saying that E is equal to the union of E j. But this is not a disjoint union because many of the E j's may have actually simultaneously more than l clients. But at least you know from the union principle that probability of E will be less than equal to n times the probability of E 1. These probabilities will actually all be equal.

Clients will not we are assuming that the clients will not know about the server. So, they will not prefer one over the other. So, these probabilities are all equal. And hence the probability of some server having more than l clients is just n times what happens on the first server. So, let us calculate that or at least estimate it. So, probability of E 1 is the probability that Server 1 has i clients for i greater than l. These are disjoint events. So, you can sum it up by partition formula. So, let us evaluate this.

(Refer Slide Time: 16:41)



$$= \sum_{\ell < i \le n} \binom{n}{i} \cdot \left(\frac{1}{n}\right)^i \cdot \left(1 - \frac{1}{n}\right)^{n-i} \leq \sum_{\ell < i \le n} \frac{1 \cdot \left(1-\frac{1}{n}\right) \cdots \left(1 - \frac{i-1}{n}\right)}{i!} \cdot 1$$

$$< n/\ell!$$

$$\Rightarrow P(E) < n^2/\ell! .$$

▷ [Stirling's Estimate] $\ell! \approx (\ell/e)^{\ell} \cdot \sqrt{2\pi\ell}$. (Why?)

$$\Rightarrow \ell! \gtrsim \left(\lg n / \lg\lg n\right)^{\ell} \cdot \sqrt{2\pi\ell} = 2^{\ell(\lg\lg n - \lg\lg\lg n)} \cdot \sqrt{2\pi\ell}$$

$$> 2^{5\lg n} \cdot \sqrt{2\pi\ell} > n^5 .$$

$\Rightarrow P(E) < n^2/\ell! < 1/n^3$. $\Rightarrow$ Load on each server is almost always $\leq 6\lg n/\lg\lg n$ $\leftarrow$ (exp. smaller than n) $\square$

So, that is l less than i and i less than equal to n. The first server can have at most n clients. So, you want probability of it having i clients. So, which i clients, that is n choose i ways times. These have to choose Server 1. So, that is probability 1 by n i times. And the remaining have to exclude Server 1. So, that is 1 minus 1 by n to the n minus i. They have to go somewhere else, the other n minus i.

So, this expression is less than equal to let us write n choose i as 1 1 minus 1 by n 1 minus i minus 1 by n divided by i factorial. And let us write 1 minus 1 by n just upper bounded by 1 which is smaller than I mean this numerator is still a fraction. 1 minus 1 by n times these things is a fraction. So, it is less than 1. So, if you have essentially 1 over l factorial then 1 over l plus 1 factorial and so on. So, this is smaller than n by l factorial.

That is the bound you get and which means that probability of E. So, E was this bad event that saw that max load is more than l. The probability of bad event is then smaller than n square by l factorial. That is where we are. So, we have to now compute l factorial. How do we do that? So, here comes Stirling's formula or Stirling's estimate. So, this says that l factorial is very close to l by E to the l where E is this base of log.

It is 2.78 close to 3. So, think of E s close to 3. It is like l to the l times square root of 2 pi l. This is an amazing formula because it is it involves both E and pi and gives you estimates ul factorial. So, if you want to see why this is true, you can look at a standard reference for Stirling's estimate. It is very famous. So, now, going back to factorial for this particular l, so, l remember was 6 log n by log log n.

So, what you get is l factorial is around or we only want to prove that it is large. So, l factorial is large at least it is log n by log log n. l by E, E is around 3 or even smaller. So, 5 by 3 we ignore and we raise it to l times square root of 2 pi l which is 2 raised to l times log of log n minus log log n times square root of 2 pi l. So, we take the base to the exponent and apply a log and then take difference. So, numerator is essentially l times log log n.

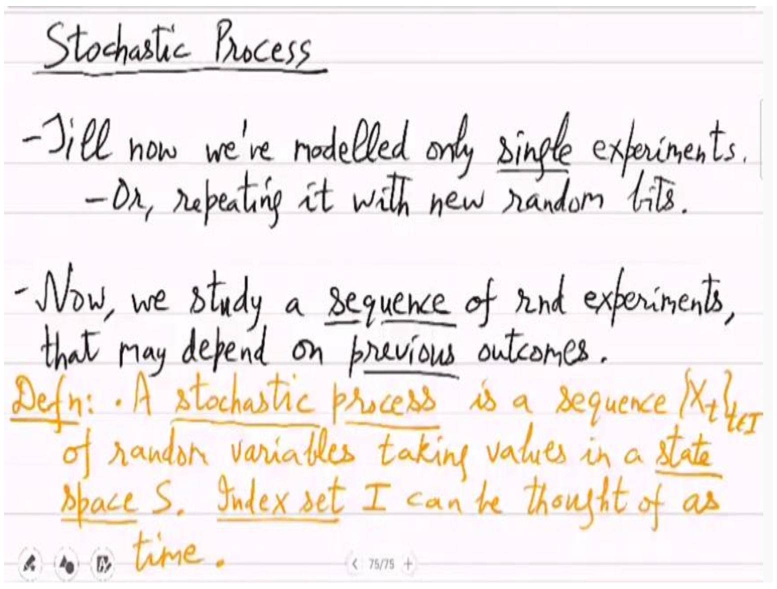
This is what you have to understand. This is the main term. This is where the action is happening times l. So, what do you get is multiply l with log log n. So, you are left with 6 log n. We can reduce it. Make it 5 log n to be safe. So, you get 2 raise to 5 log n times square root of 2 pi l which is at least n to the 5. That is what we have learnt. In fact, it is more than this. But we will be satisfied then to the 5.

Because what you get is probability of E is less than 1 over n cube. So, what you have learnt is load on each server is almost always because 1 over n cube is an extremely small probability. If n was even 1,000 you are looking at 1 over 10 to the 9. This is an extremely low probability. So, in with extremely high probability this will not happen and thus max load will be only l or below l.

So, load on each server is almost always smaller than 6 log n by log log n, so, exponentially smaller which I already said. So, it is exponentially smaller than n. So, this was a nice application of probability. And you learned some new tools which help in estimation also in estimating binomial numbers like n choose i. You may also play with this by changing the number of clients.

So, you can say there are m clients and n servers. Servers may be fewer. And then what happens? How much is the load? How does, so, does the load balance? And, how much is the max load? You can look at those questions. So, let us now shift gears and start a new theory which will be stochastic processes.

(Refer Slide Time: 24:23)


Stochastic Process

- Till now we've modelled only single experiments.
  - Or, repeating it with new random bits.
- Now, we study a sequence of rnd experiments, that may depend on previous outcomes.

Defn: • A stochastic process is a sequence $\{X_t\}_{t\in I}$ of random variables taking values in a state space S. Index set I can be thought of as time.

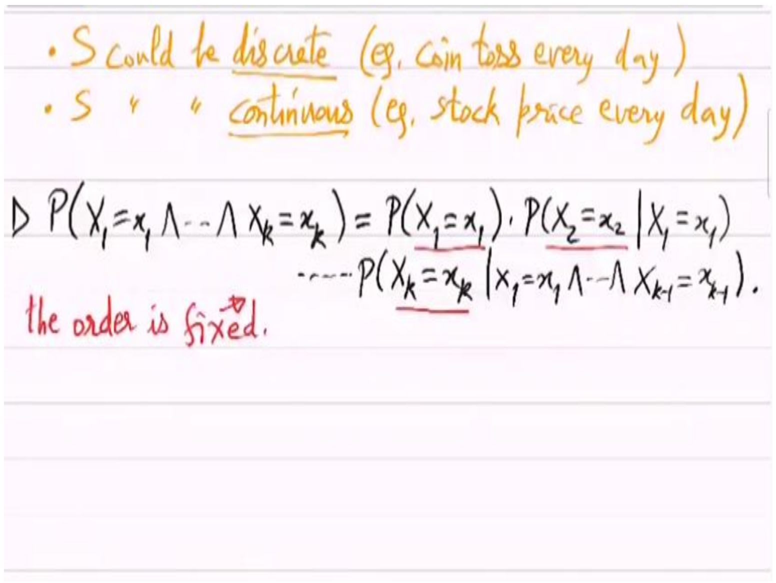
So, till now, we have focused only on one experiment. And then we may want to repeat that experiment. So, in but, so, correspondingly or analogously, we focus on one (()) (24:48) variable. So, instead of this, suppose you want to look at a sequence of experiments which are not independent. One depends on the previous one in time. Till now, we have modeled only single experiments or repeating it with new random bits.

So, we have one experiment in mind let us say tossing a coin. You do it once then you do it twice. And the randomness is I mean these are you think of them as independent events. So, that we have modeled. But now, what we want to do? We want to study sequences, a sequence of random experiments that may depend on previous outcomes. So, dependent but then we have to study what are the types of dependencies.

So, let us formalize this as a stochastic process. So, a stochastic process, why process, because it is not an experiment. Now, we are looking at a bigger experiment that contains many small experiments. This is what we are doing. We are looking at a sequence of experiments. So, we call it a process. A stochastic process is a sequence X t, t from some index set I. It may also be infinite of random variables taking values in a state space.

So, the values that these X t's take, we call that universe state space. So, they are taking values from the same space. But, these may be completely different experiments and they may be dependent. So, sequence of experiments we are calling stochastic process. And finally, this index set I can be thought of as time so which is why I say present or past, current or the previous. There is this ordering. So, this will be a good way to model many physical phenomena.

(Refer Slide Time: 29:10)



- S could be discrete (eg. Coin toss every day)
- S " " continuous (eg. stock price every day)

$$\triangleright\ P(X_1=x_1 \wedge \cdots \wedge X_k=x_k) = P(\underline{X_1=x_1}) \cdot P(\underline{X_2=x_2} \mid X_1=x_1) \cdots P(\underline{X_k=x_k} \mid X_1=x_1 \wedge \cdots \wedge X_{k-1}=x_{k-1}).$$

the order is fixed.

So, S may be discrete. Example, you toss a coin every day. So, every I mean the sample space or the state space will remain head tail. But the experiment is you are doing it one after the other and many of these. And you want to study how this evolution is happening. Or, it could be continuous. Example, stock price. So, again the index set is days. Say, from now on, count the days. Every day you look at the stock price of some stock.

Say some software company which you like you which you are following. So, now the stock price is actually a positive real. It may not be an even an integer which is why we call it continuous in this case. So, that so, state space remains the same. But as days progress what happens it may be related to the previous day or it may be related to all the previous days or it may be completely independent. That will be clear from the process.

So, what can you now immediately say in a stochastic process. Notice that this is not a big change of definitions. These kind of things you have already done when you studied conditional probability. So, from that you already know that probability that big X 1 is small x 1 dot dot big X k is small x k. How this probability behaves? So, in this process, the first k values are fixed to be this. What is the chance?

So, since there is a notion of time here in this process or in this experiment, so, you first ask what was the chance of small x 1 coming? And based on that what was the chance that small x2 came? And then based on the previous k minus 1 values, what is the chance that came? So, note this order. And you cannot change this order. You cannot ask little x 2 or big X 2 has happened and then what is the probability that big X 1 will happen. That is not allowed because big X 1 is happened actually first.