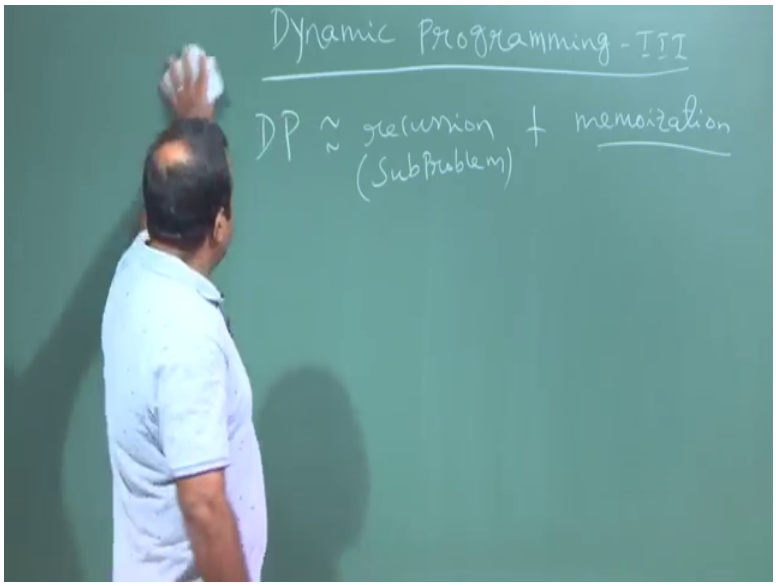
An Introduction to Algorithms
Prof. Sourav Mukhopadhyay
Department of Mathematics
Indian Institute of Technology, Kharagpur

Lecture – 58
More on Dynamic Programming (Contd.)

So, we are talking about dynamic programming technique it is a very powerful to tool to solve for most of the; I mean many optimization problem it is very much useful for optimization problem. So, in the last we have seen the Fibonacci; how we can get the; how we can use the dynamic programming technique to find the nth Fibonacci number. So, basically we have given a recurrence. So, we just try to write any formula in the recursive form if we can write that then we can apply the dynamic programming technique. So, basically dynamic programming is basically.

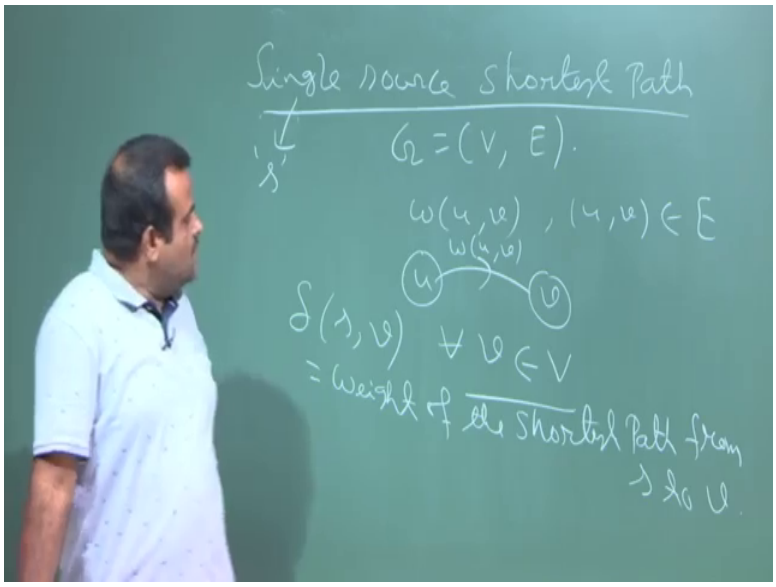
(Refer Slide Time: 01:02)



So, we call this as DP in short, it is basically the recursion. So, recursion means sub problem. So, we have a problem. So, we reduce into your sub problems and then and then we use this solution of the sub problem reuse; reuse means we have to memoize. So, there are 2 version of DP, we have seen in the last class one is memo memoization. So, we memorize the value once it is computed then we use that reuse that value. So, that is the memoization.

So, this is the basically. So, we memoize the; we stored this into some dictionary then we use it. So, in this lecture we will talk about another problem which is basically the shortest path problem single source shortest path problem.

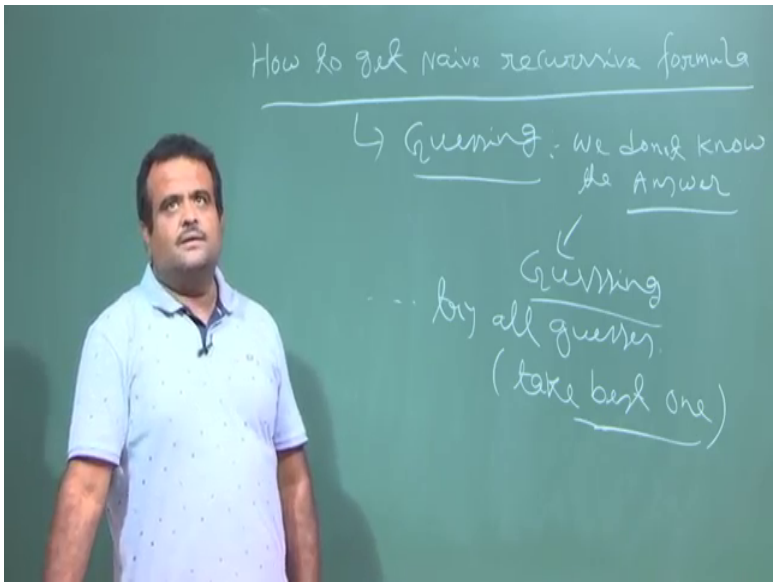
(Refer Slide Time: 02:05)



Single source shortest path; so, what is the problem? Problem is you have given a directed graph and we have some age weight on the graph. So, that is basically w u v. So, this is the weight function where uv is an h. So, u and v are 2 vertices in the graph. So, each edge is associated with a weight function wuv. So, then you have to find it. So, there is a single source there is a source s which is also another input. So, we have to find basically delta of s comma v for all v belongs to v. So, this is basically the weight of the shortest path of the shortest path from s to v. So, this is basically we consider the path from S to V all path. So, among them which is the minimum weight. So, that is the weight of the shortest path. So, we have seen the algorithm like distress algorithm which will run for positive weights I positive weight h and then we have seen the bellman ford algorithm.

So, now we will look at. So, these are all the greedy approach well one for now we will look at how we can solve this using the help of dynamic programming technique. So, for that; so, you want to write this in a naïve recursive way. So, the question is how we can get the recursive formula for this how we can get the recursive naïve recursive algorithm to solve this problem.

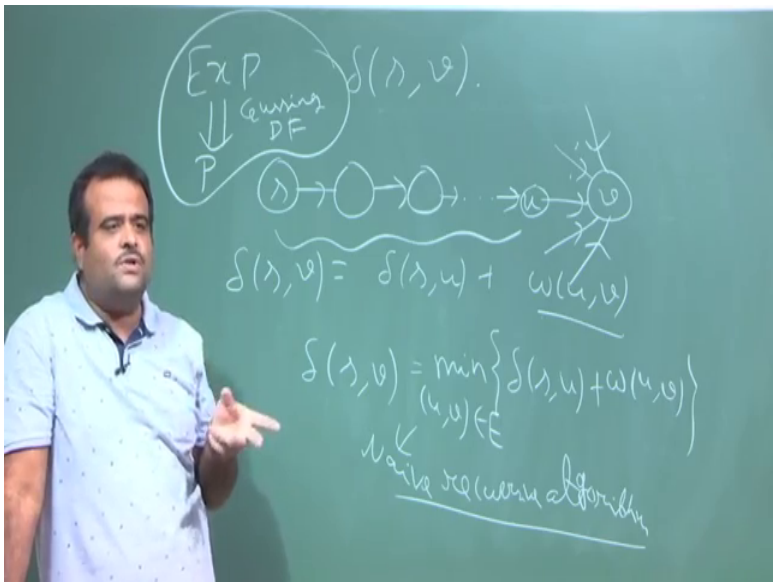
(Refer Slide Time: 04:29)



So, let us try to get that then I we can then we can think to apply the dynamic programming technique. So, question is how to get naïve recursive algorithm or recursive formula. So, the answer we do not know how to guess. So, answer is. So, the guessing; so, answer is guessing we want to guess. So, this is very powerful tool guessing we do not know the solution we do not know the answer. So, guessing means we do not know the answer. So, then what is the solution? Solution is guess just guess guessing not only guessing, but try all guess all possible guesses. So, try all guesses this is sort of exhaustive search and this is the beginning of the dynamic programming technique. So, we have exhaustive search, then we will take the best one I mean we will try for all possible guessing then take the best one with the help of with the help of DP technique. So, that is the idea.

So, let us just try to; so, how we can get a naive recursive formula or naive algorithm for the single source shortest path. So, we want to find delta of s comma v. So, basically we have vertex source vertex s and we have another vertex V.

(Refer Slide Time: 06:30)



So, we have some intermediate nodes. So, there are some vertices which are incoming vertex of v. So, these are the edges which are direct edge form which are direct edge going to v there. So, these are the incoming edges which are going to v. So, dot dot dot this.

So, now we want to find the shortest path from s to v. Now we want to write this in a recursive form. So, suppose hypothetically suppose somehow we know some shortest path from s to u just one note before v. So, these are all direct edge now there are many direct edge, we do not know which will give us the minimum one. So, what we do we try for all and then we try for all and then we guess and that try will be doing by guessing. So, we tried for all and then we choose the I mean that so. So, now, the question is by clever way or by v, we have to choose the best guess. So, that the DP will give us; so, then this plus w uv, this is the one of the vertex direct edge form to v, but we do not know which one. So, we have to explore all positives is sort of exhaustive search. So, this is the exhaust, but we do not know which one. So, suppose we have a this is the sub problems. So, we reduce the problem into sub this the sub problems.

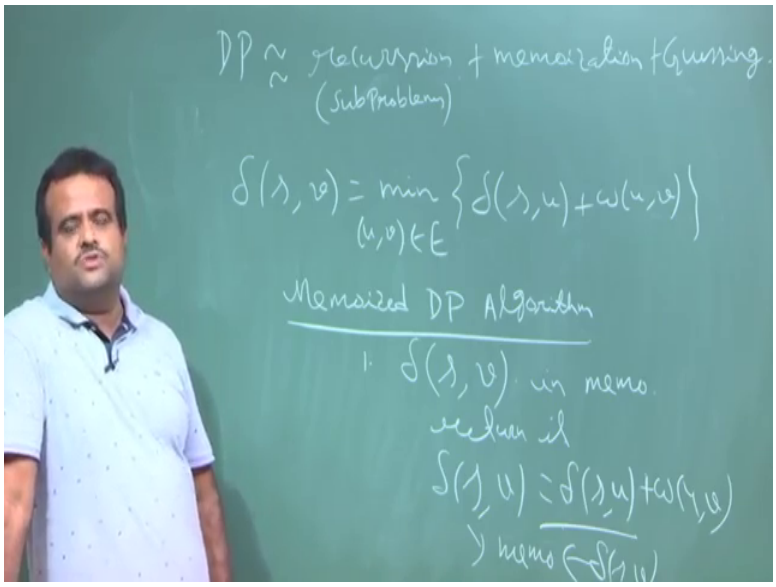
Suppose hypothetically we have the solution of this sub problem then we have this direct edge. Now we want to choose one of this will give this shortest path, but which one we do not know. So, we have to try for all possibilities these are the. So, we have to guess which one. So, this is the guessing technique. So, so if the; our guess is correct then this

will give us delta of s comma v, but we have to do this exhaustive search. So, basically delta of s comma v, this is the recursive formula minimum among delta of s comma u plus w of u v where u v is the direct edge form to v. So, these are all possible edges from e.

So, this is the naive recursive algorithm recursive algorithm. So, this is the naive recursive algorithm. So, is this good? So, so this is the exhaustive search. So, this is basically this is the exhaustive search we are doing. So, this is the algorithm is good. Now, this is not good, this is huge. I mean this is exponential. So, this algorithm is exponential algorithm.

So, now we want to reduce this to be a polynomial with the help of guessing and that will be done by DP method. So, that is the goal of the DP method. So, goal of the DP technique is we have a exponential we have exhaustive search it is a careful brute force. So, carefully we have to do the guess. So, that is the idea. So, this is the exponential time algorithm. So, we cannot go for all possibilities we have to guess which edge will give the correct one we will give the minimum one that you have to guess. So, that is the goal of the DP. So, you have to guess that. So, that is basically that we are going to do. So, that is the repeat techniques. So, that will do by memoization so; that means, we will once we; so, we will use the same algorithm what we have done earlier. So, let us just write this. So, basically the DP technique is now guessing is added there.

(Refer Slide Time: 11:22)



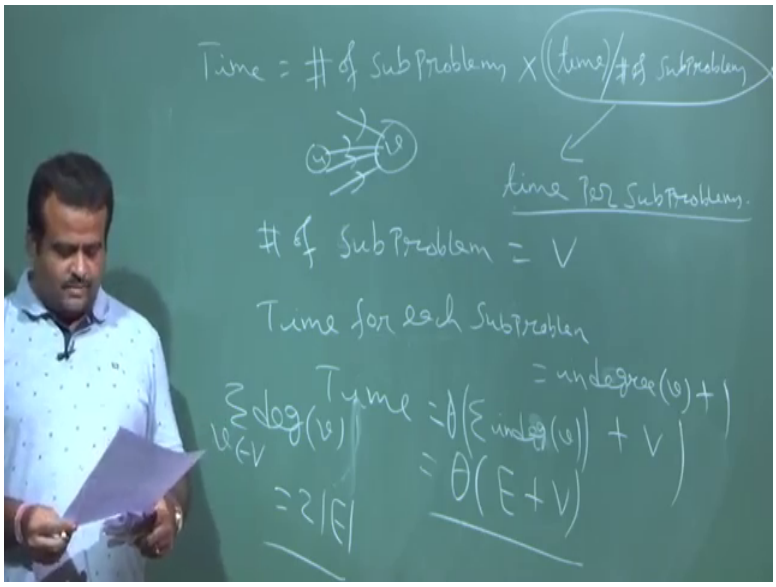
So, DP is basically we have we have sub problems the recursion we have this is basically the sub problems sub problems and then what we have we have we have to do the memoization and one thing is added now is guessing we want to guess the correct path.

So, now, what is the delta of s comma v delta of s comma v is the minimum of delta of this is the recursive formula ys comma u plus w of u comma v. So, this u v belongs to e. So, this is the basically now these formula. So, this is the Naive approach this is the exponential time at approach. So, we have to use we want to use the memoized DP algorithm DP algorithm for this. So, that will reduce this from exponential to polynomial. So, what is that algorithm? Algorithm is basically.

So, we compute this deltas now we compute delta of s comma v and we put it into a table. So, again if we want to compute it; so, we first look into the table. So, if it is in the table if it is in memo then we return it otherwise what we do otherwise we compute delta of delta of s comma v is equal to delta of s comma u plus wuv and this is the sub problem. So, this is the sub problems now once we compute this then we put it into the memo. So, that next time if we need this value we can reuse that. So, that is the idea.

So, this is the memoization algorithm for this version now what is the time complexity. So, we know the memoization.

(Refer Slide Time: 14:03)

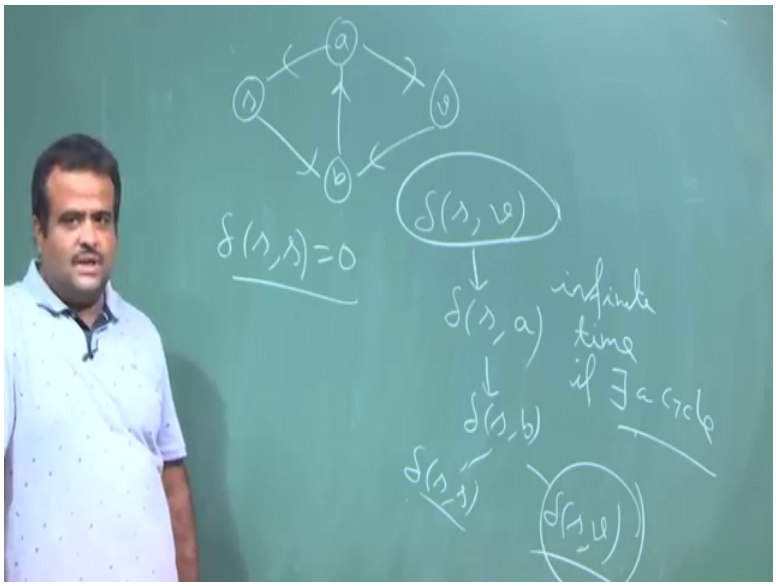


So, this we know the time for this type of DP is basically time is equal to number of sub problems into the time required parts of problems. So, that is basically into the time required parts of problem. So, this is basically time total time divided by number of sub problems. So, this whole thing is specifically rate of executing the sub problem. So, this is basically time per problems time per sub problems.

So, now what is the time then? So, time is basically. So, time is basically. So, how many sub problems we have we have free sub problems. So, number of sub problems is equal to order of v now what is the time to solve each sub problems how many guesses are there the exhaustive search we are doing on the. So, the basically the time for each sub problems is basically. So, this is v. So, this is the basically possibilities. So, we are. So, this is basically in degree of in degree of v plus 1 now if you take the sum. So, total time is basically summation of. So, we can put a theta over here summation of in degree in degree of v in degree of v plus v for this one. So, in degree of v is basically order of e order of e plus order of v this is coming from of hand handshaking lemma. So, summation we know the summation of degree of v is basically I mean order of e this is the handshaking lemma this is coming from handshaking lemma.

So, now is this algorithm work for any graph no. So, the unfortunately no; so, otherwise this technique is quite because this is this will not work for this is work for only dag direct acyclic graph if there is no cycle, but problem is if there is a cycle then this method will fail.

(Refer Slide Time: 17:29)



So, let us look at that. So, let us look at that. So, suppose we have a graph like this s if v. So, a b and these are the vertices I these are the edges.

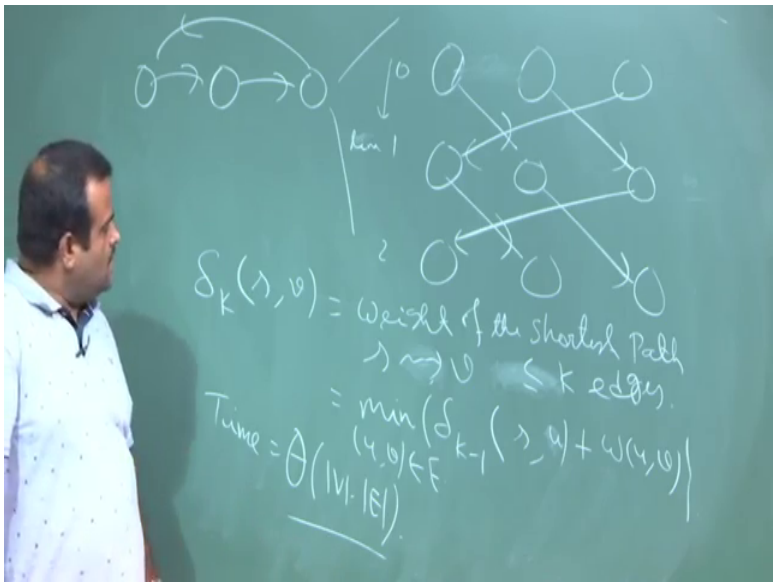
So, now suppose, we want to apply this memoize algorithm the guessing algorithm and they want to memoize, suppose, we want to compute s this s to v. Now, to compute this what are the edges going to free in degree edges a. So, to compute this you have to compute delta s comma a because there is only one edge going for this now to compute this delta of s comma a we need to look at number of edges vertices edges which are in degree to this. So, this is one only. So, for this we need to compute delta of s comma b, now there are 2 edge incoming. So, to compute this we need to compute delta of s comma s delta of s comma v.

Now, delta of s comma s this is the base case this we assume zero, but this again is same as this one now how to get this value we haven computed see this is the we are storing in the dictionary after getting the value, but this value is yet to get we haven't completed. So, we stuck here we cannot proceed further because this is again aloof. So, this is the infinite time if there is a cycle. So, this will take infinite time this is a; this we stuck in finite time if there is a cycle in the graph that is the problem.

So, this method will not work this way directly on the on the graph which is having cycle because here we stop because how to get this value we haven't computed. So, we will look at the table, but it is not yet computed because we have to compute we are

computing this and then we stuck in this way. So, then this is the problem so, but to. So, how to overcome this problem? So, we know a technique. So, that is suppose you have a graph with cycle. Now how we can view this how we can remove the cycle from the graph.

(Refer Slide Time: 20:18)

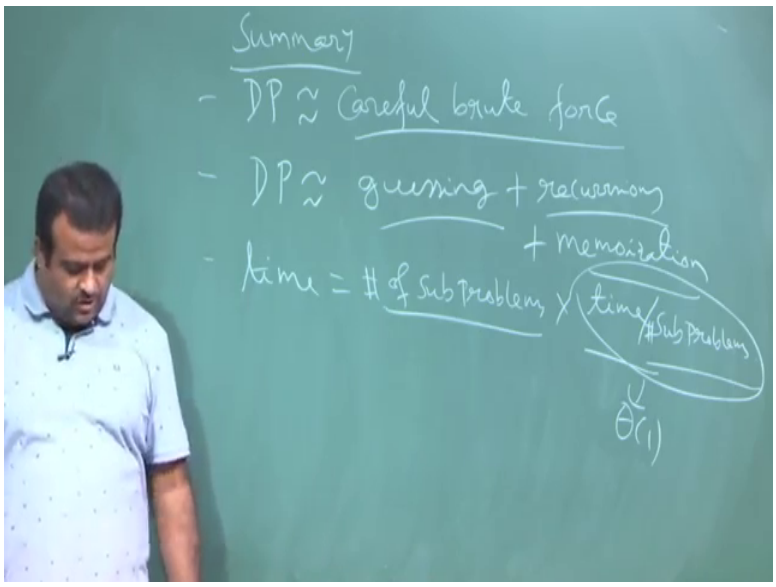


Suppose we have a graph like this and there is a cycle now we want to have a; many version of this graph. So, this is the way how this is the time. So, this is the; this is connection. So, this is these are the connection this is going here this is going here. So, this is the; this we are we are just writing the copy of the same graph many times like this. So, this is the zeroth copy one time.

Now, once we move from here to here we just move from here to here like this. So, this way we can just avoid the cycle so, but the thing is we are just increasing that. So, we will try to this is a cyclic graph. Now we will try to find the; we will apply this DP memoize DP to get the shortest path on this graph, but there. So, what is the problem with this approach problem with this approach is it will in. So, it is increasing the number of vertices. So, if you write this delta k is s comma v is equal to weight of the shortest path from s to v which use yeah which uses which use less than k edges. So, number of edges in that path is less than k. So, this is basically we know the formula there minimum of delta of k minus 1 is comma v plus w of uv. So, this is the recursive formula we used.

So, this is sort of its relaxation step if you remember the bellman ford algorithm or distress algorithm we have a relaxation step this is sort of the relaxation step. So, now, here what is the number of sub problem number of sub problems has increased by the order of v square. So, the time complexity here is for this graph is basically order of v into e. So, this is the same time complexity of bellman ford algorithm, but this is the way we just usually avoid the we just avoid the cycle in a graph. Now we will just quickly summarize the DP technique in general there are basically three easy step like sorry five steps in any DP approach.

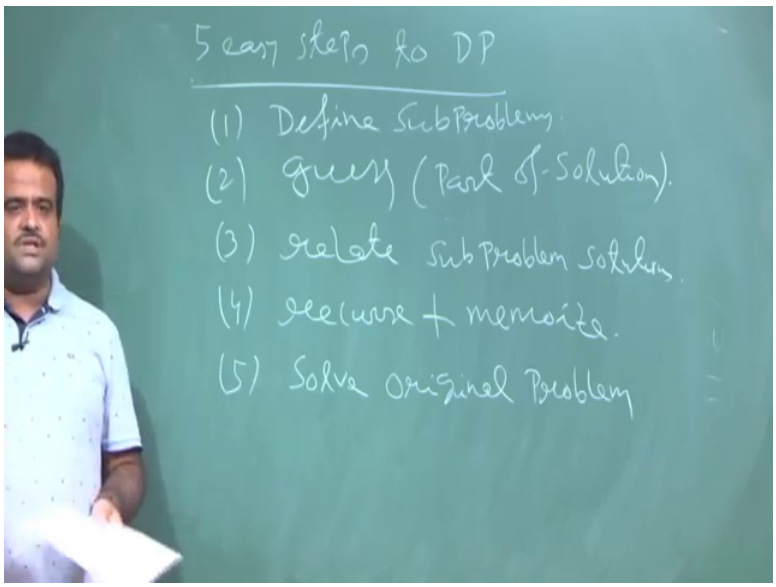
(Refer Slide Time: 23:41)



So, let us summarize. So, we have seen a powerful technique which is dynamic programming technique which is mostly used to solve any optimization problem because it is basically reduce the space form we have a exponential, it is basically careful good force. So, it is careful exhaustive search. So, we have exhaustive search we have exponential space now from there we are going to reduce this to a polynomial space carefully. So, basically it is we for this technique we need to have a recursive formula or recursive algorithm which is naive approach which is recursive algorithm which for which we can use the memoization version. So, this is basically guessing plus recursion basically we need to have the sub problems plus the memoization. So, we will we will just store the value into the dictionary so that when the next time we need that value we can reuse that.

So, the idea is to divide the problem into reasonably number of sub problems and once we compute the value of the sub problems we store it, we memoize that and then we will reuse that and the time complexity is basically we know this is basically number of sub problems into time per sub problems this is the sub problems this is the time per sub problems and this is the total number of sub problems. So, this is basically our. So, usually this time is theta one because we just mostly we will do the table lookup. So, the total time is if this is order of n then the total time will be n. So, this again this analysis is amortized analysis. So, because we are doing the average case analysis I mean on an average what is the time. So, this is the amortized sense. So, then we write the five easy step for any DP method.

(Refer Slide Time: 26:27)

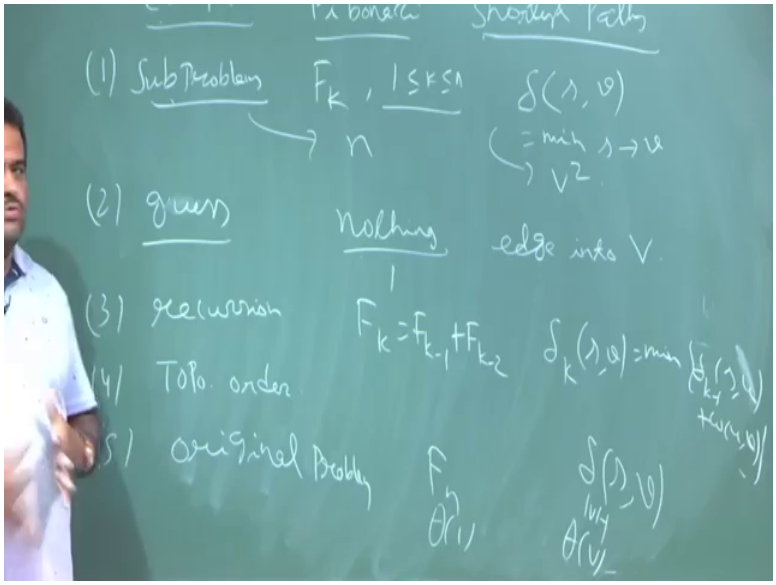


So, the five steps to dynamic programming technique one is first one is we have to define the recurrence we have to define the sub problem. So, we should have that naive recursive formula define the sub problems this is the first step second step is we need to guess guessing part of solution we need to guess. So, that way we reduce this exhaustive exponential space to polynomial space now we relate the solution of sub problems relate sub problem solutions and then we just recurs plus reuse plus memoize or inset of memoize version we can have a bottom up version. So, that also can be done.

Then we solve the original problem which is a combination of solution of the sub problems. So, let us take quick example we have seen 2 problems in the today's lecture

and last lecture one is Fibonacci number finding n th Fibonacci number and today we have talk about the shortest path problem.

(Refer Slide Time: 28:33)



So, let us just write what are the values of what are the steps for these 2 problem. So, this is an example. So, we have. So, this is basically we have seen the Fibonacci problem Fibonacci and then we have seen the shortest path problem shortest path problem.

So, number one step number one step is basically sub problems. So, what are the sub problems basically we need to find Fk for this for one less than k less than n, this is the sub problems for this and what is the sub problems for shortest path we need to find delta of s comma v. So, this is basically. So, minimum from s to v which is used at most k number of edges. So, this is somehow hypothetically we know the answer up to the sub problem. So, that is the things. So, what is the number of sub problems here this is n, but this is here is v square.

Now, the guessing step guessing step is here we no need to do any guessing nothing. So, the choice is one and here we need to guess because there are how many know how many vertices are coming from other how many in degree. So, this is basically in degree plus 1. So, this is basically h into v. So, this is basically in degree. So, this is in degree plus 1 and then we have a recursion. So, for here we know the recursion Fk is equal to Fk minus 1 plus Fk minus 2 and for here we know the recursion delta of k s comma v is the minimum of delta of k minus 1 s comma u plus w of u comma v, then the then we need

to check the topological order; for that topological ordering and then we have we have the solution for the original problem which is going to use the; so, this is basically Fn and here it is delta of s comma v this is basically delta of Vk minus 1 s comma v. So, this we will take order of one time and this will take order of v times and there are ok.

So, this is the 5 basic step for any DP problem. So, so this is the this is we have jot down for 2 x 2 problem like Fibonacci number and the shortest path problem.

Thank you.