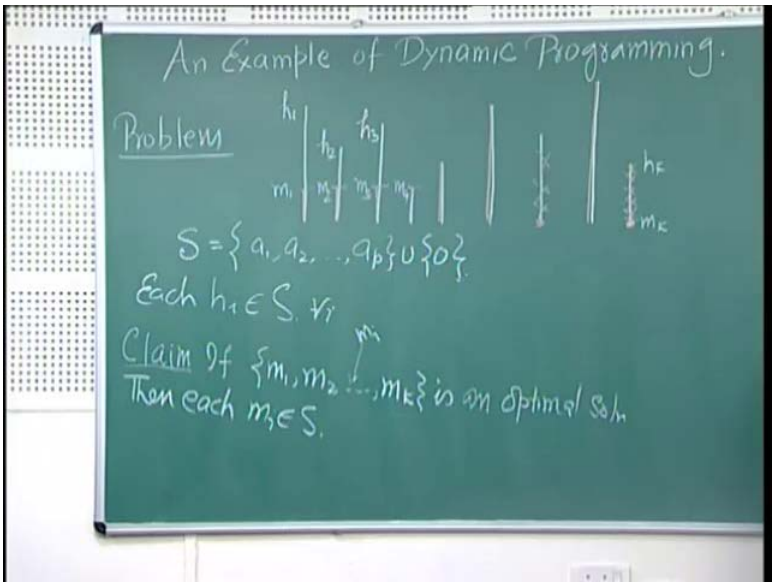
Computer Algorithms –2
Prof. Dr. Shashank K. Metha
Department of Computer Science and Engineering
Indian Institute of Technology, Kanpur

Lecture - 35
General: Dynamic Programming

Today, I will describe an algorithm based on Dynamic Programming. The problem that I am going to take for describing this paradigm is suggested by a colleague of mine professor Arnab Bhattacharya.

(Refer Slide Time: 00:34)



This problem is as follows, if suppose that we have a series of wooden poles or trees if you want of various heights planted along a straight line, say various heights are h 1, h 2, h 3 and so on. Our goal is to cut these trees such that the final result is monotonically non decreasing which means that the height as you proceed either is same or greater, but we want to do this in such a way that we cut the least amount of wood.

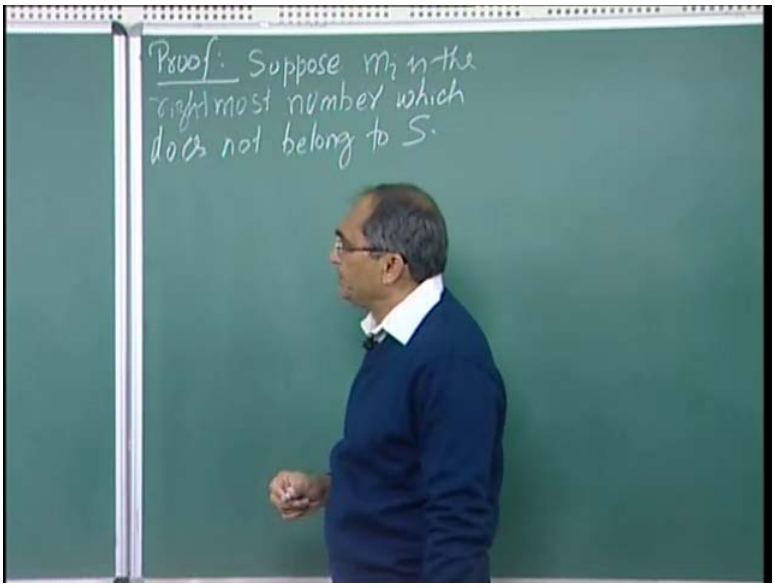
So, for example, in this case a possible solution could be that we cut this here, so we keep this much, we keep this as it is, then we may keep this whole thing. And let me just to make the point, let me make it slightly longer sequence. So, here what we do is we completely cut it off, completely remove this we keep this entire tree and then we completely remove this.

So, what has happened is that of the remaining trees, of the remaining part of the trees, these are the lengths. And that although there is 1 tree in between which is completely eliminated that is not a problem, but what remains whatever the tree is that remain they must be in monotonically non decreasing. So, this is the goal and what we want to minimize is the total amount that is we have cut that is these this, plus this and plus this, this we want to minimize which is equivalent to saying that whatever lengths are remaining their sum should be maximized.

So, let us suppose these are the lengths of the remaining trees and so on. So, this would be since this is reduced to nothing for this case it is m k this was h k and so on. So, we have k trees and their lengths, if you write down as a set can be say a 1, a 2, a p and we include a 0 in this set, so let S denote the set of various lengths that have been presented to begin with and 0. So, each of the h i that is each h i belongs to S, that is what we have, so clearly the number of elements in S can be at most k plus 1 because we have added 0 in it.

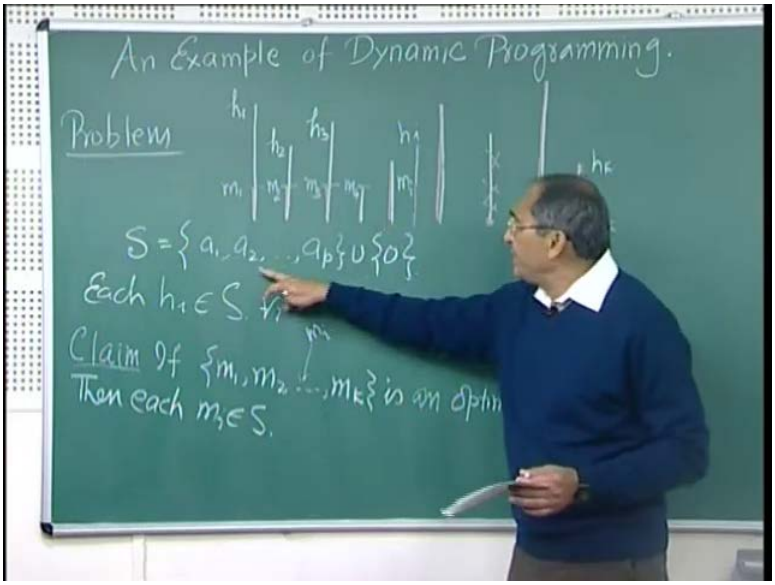
So, first claim that I want to make is that, if claim that is if m 1, m 2 through m k is an optimum solution then each m i belongs to S, this we will prove first. The purpose of this result is at that reduces the said space of the solution, now we know that only possible heights can be from this set, so let us try to establish this claim. Let us suppose that m i somewhere here is the right most number in the sequence, which does not belong to S; so everything to the right of m i belongs to set S.

(Refer Slide Time: 06:43)



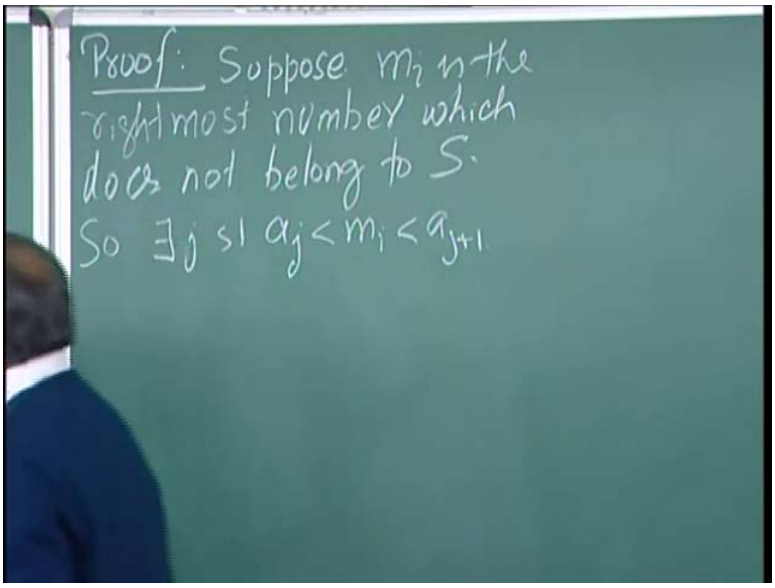
So suppose, m i is the right most a number, which does not belong to S alright.

(Refer Slide Time: 07:25)



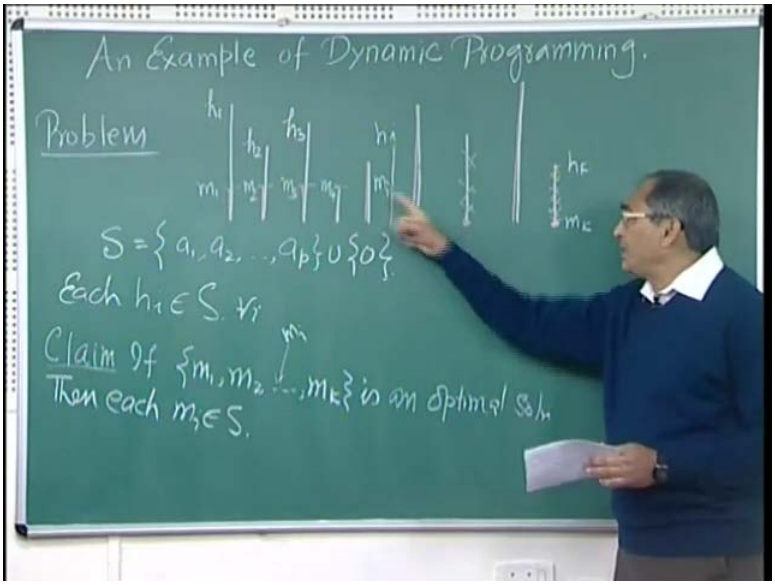
So, it does not belong to S it has to be, so it is the result, so m i let us look at this suppose this is somewhere here. Let us say somewhere here we have m i, this is m i this was h i and m i, m i is the result of cutting a part of h i clearly, we have definitely cut something in it. Hence, there must be some j such that m i is between a j and a j plus 1, because it does not belong to S.

(Refer Slide Time: 08:16)



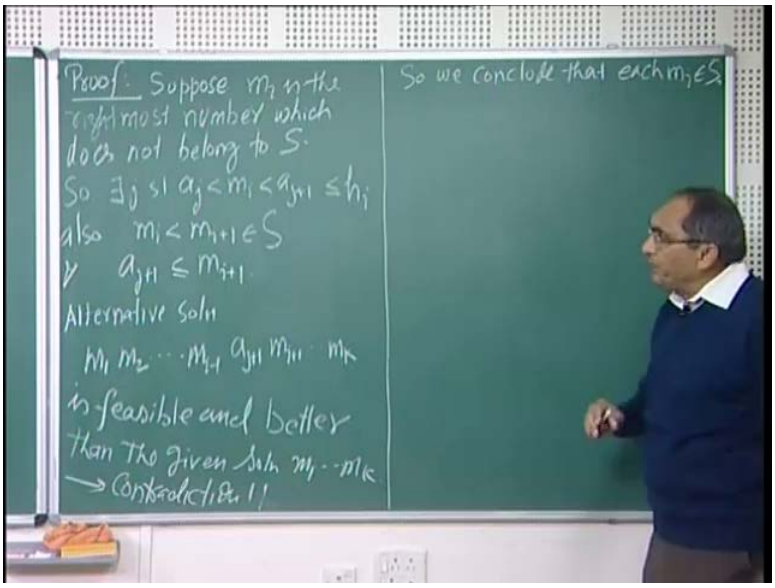
So, there exists a j such that a j is less than m i is less than a j plus 1.

(Refer Slide Time: 08:36)



Besides let us take a look at m i plus 1, since we know that m i plus 1 belongs to the set S and that is greater than equal to m i, but m i does not belong to S.

(Refer Slide Time: 08:58)



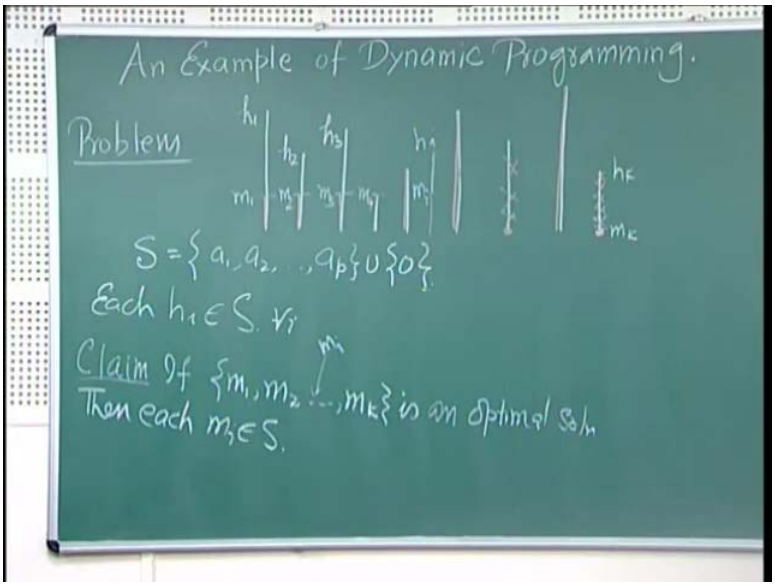
So, this inequality is strict also m i is strictly less than m i plus 1. Now, m i plus 1 which is actually this belongs to S, m i plus 1 belongs to S, a j plus 1 is a member of S. So, we can also say that a j plus 1 is less than or equal to m i plus 1, because this is also member of S this is a member of S this is the nearest member of S to m i. So, what we have noticed is that there is an alternative solution, which is m 1 m 2 m i minus 1, instead of m i we replace it by a j plus 1 m i plus 1 all the way up to m k.

From this inequalities what we notice is this thing is less than or equal to this and this thing is in fact, this is strictly less than this and this is less than equal to this. So, monotonously is held and 1 more thing we have to also establish that a j plus 1 is possible. That is to say the original height of ith tree is greater than equal to this, but that is also obvious note that this has to be less than equal to h i. The reason is this belongs to S, this belongs to S this is the nearest member to m i.

Hence, this has to be greater than equal to this, so this is a feasible solution, but what is important to notice is that in this solution we are cutting less wood, because this is longer than the original height of the solution that we had suggested m 1 through m i. So, this has to be better solution, but that original solution was given to be optimal, so this should not be possible. This solution is feasible and better than the given solution than the given solution m 1 through m k, and this is a contradiction because that is optimal . So, we

conclude that every number m i belongs to set S. So, we conclude that each m i belongs to S, so this is done.

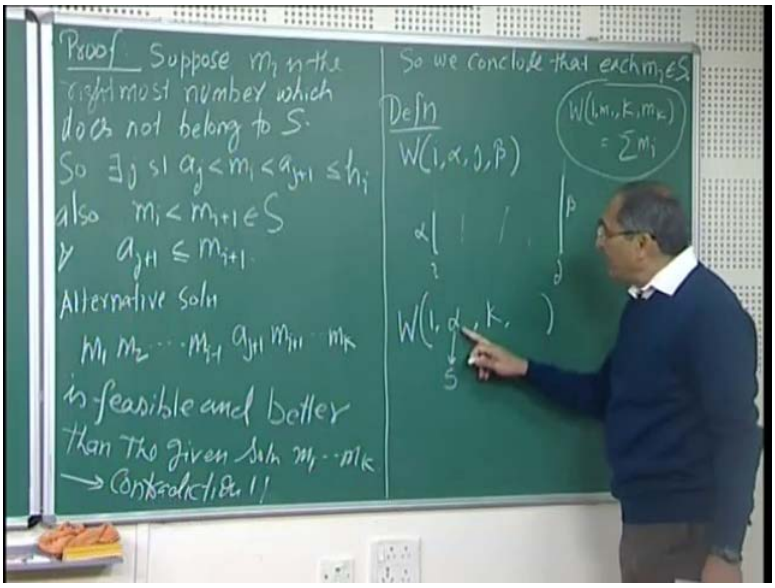
(Refer Slide Time: 12:55)



Now, let us try to review what dynamic program is about in dynamic programming we propose a family of problems, such that the given problem is a member of that family, the size of this family that the number of problems in this family must be polynomial in the parameters of the problem.

The smaller problems in the family must be easy to solve, and we should be able to construct the solutions of the upper or higher problems from the solutions of the lower problems. So, this is how we build bottom up we proceed bottom up. And finally, that the original given problem is either one of the problems of a family or it is constructed the solution that is constructed from the top level members of the family.

(Refer Slide Time: 14:05)



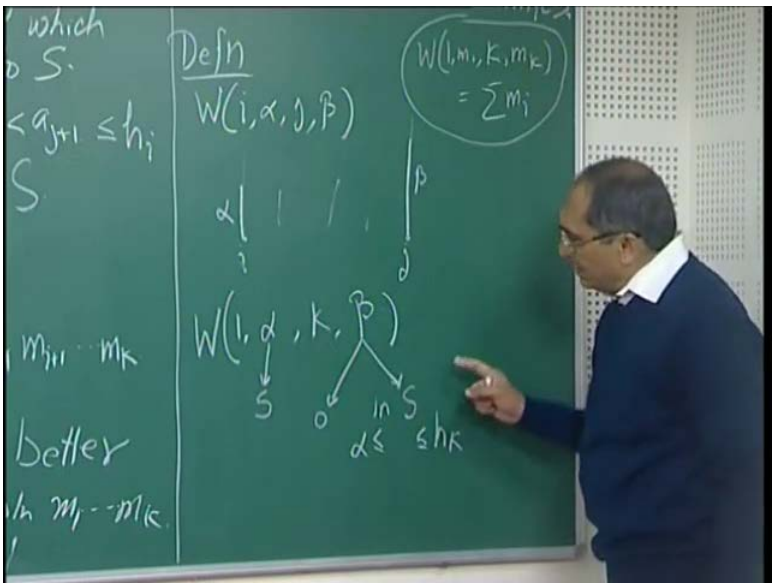
So, this is how we will try to solve this problem and the key step in this is the definition that I am giving, let me define W i alpha j beta. This is a number which indicates the that if a problem is if the input problem is starting from ith pole to jth pole, from ith to jth pole. We are given the same trees various heights etcetera, we have given those and this number denotes the optimum solution, the maximum remaining the sum of the heights of the remaining trees.

We are maximizing the sum of the heights of the final trees after cutting. This is the optimum value of that subject to the condition, that this is in the solution this is cut of alpha and this is being cut to height beta. We insist that these are fixed, these heights must be what are specified here alpha and beta. Everything, else can be whatever it is and we want to denote the optimum solution to be this. For our previous notation, if m 1 through m k is an optimum solution, then if that is actually nothing but W 1 m 1 k m k.

This is sum m i, this is what this notation means, but in indicating this or defining such a notation I have basically introduced the whole idea of dynamic program. Now, this function with 4 parameters for each set of parameters we have a problem, whenever we have certain i certain j alpha and beta we have a problem. And in case i is equal to j; obviously, alpha and beta should be equal that answer is trivial, we have to do nothing the answer will be alpha in that case.

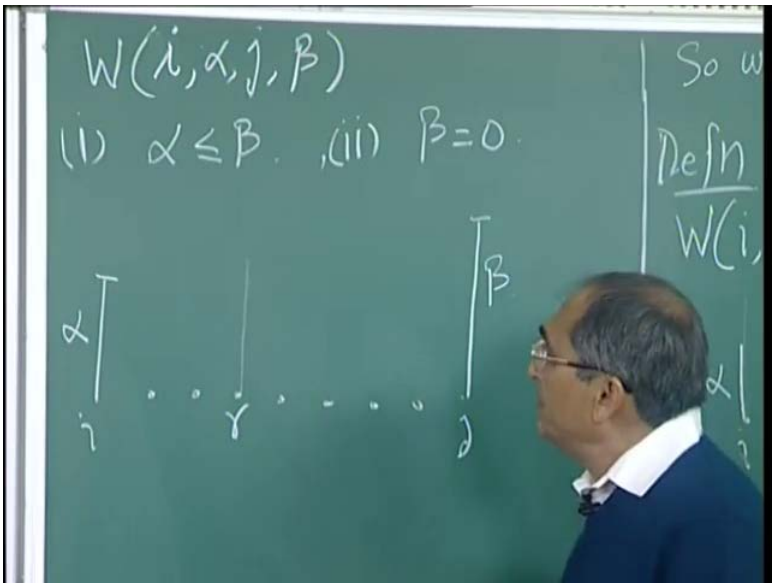
So, you notice that if the displacement between i and j is 0, then that is a trivial case. And what we will do is we build a solution by cutting this splitting this range i to j and taking the solutions of 2 parts and then stitching them. So, that is how we will build the values of W i alpha j betas for various values of the parameters, now our goal is to compute what is the relationship of this with our problem. Now, in our problem the first and the third parameters we understand should be 1 and k. What is the possible value of alpha. So, the possible value of alpha can be this alpha can be any number in set S which is less than or equal to h 1, alpha is any numbers in set S less than equal to h 1.

(Refer Slide Time: 18:32)



And beta is any number in S with a two possibilities, one either beta can be 0 it can be 0, because in monotonously 0 does not matter or it should be a number in S which is greater than equal to alpha and less than equal to h of k. Because if it is not 0 then clearly it has to be at least alpha and it can never exceed the initial height h k. So, this is the possible these are the various numbers and we will pick the largest of those, whichever is the largest among all these possibilities we will pick that as the final solution of our problem.

(Refer Slide Time: 19:35)



So, now let us see how do I compute a one particular instance of W sorry to it is i j alpha beta, so this is i alpha j and beta. There are two possibilities in this case, the first case is that alpha is less than or equal to beta and the second case would be that beta is 0 irrespective of what alpha is there are two possibilities. So, let us say we have our ith pole and jth pole that those have been cut to alpha and beta for the moment I am taking this case.

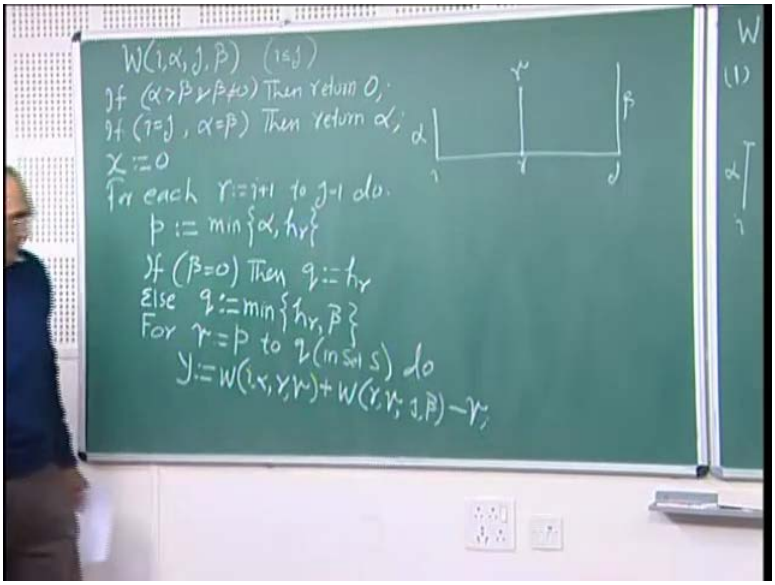
Now, in the inter mediate posts in these spaces there are two possibilities, but in the optimum solution there is at least one of these remains. That is not all of them have been wiped out completely that is one possibility. There is a possibility that maybe this tree has some length less than equal to alpha less than equal to beta in the final solution or the second possibilities that all of them that have been reduced to 0 which is also valid.

So, in the later case the solution is alpha plus beta, because this is the only remaining length. So, our final solution cannot be less than alpha plus beta, this is one of the contender in the solution, but the other possibilities we do not know where we will have a tree which is non 0 in the final solution. So, we will have to search we will pick a position r and we will actually set r to anywhere from i 1 i plus 1 to j minus 1. We will pick an r and what we know is the height of this three in the final solution has to be from set S, set S is the set of values that we were originally given it has to be one of those. So, we will try out all possible values from alpha to beta, but what whichever is all these

values are allowed as long as there are less than equal to h r, the original height because you cannot increase the height. So, let us with this picture what we will do is we will try a we will take a particular height of a r or which is valid.

And pick the optimum solution of this that is this height is fixed this side is fixed take the optimum solution of this optimum solution of this. Now, i and r and r and j these are closer than i and j, so we assume that we have already solved this problem we have already solved this problem. And then we combine them, now each of this option will give me a possible solution and we will pick the largest of this.

(Refer Slide Time: 23:40)



So, let us write down the algorithm W i alpha j beta, here we assume that i is less than equal to j, this is assumed that is how we will consider those sub problems. So, let us first of all describe a few possibilities, if alpha is larger than beta and beta is not 0 then return 0, because this is not going to happen we know that optimum solution we will have length at least alpha plus beta, so this will not be considered. The second possibility is if the base case i is equal to j and alpha is equal to beta, then there is nothing to be done then return alpha, this is the best solution you can have.

Now, we are going to consider first of all those solutions in which at least one intermediate tree is not completely cut down, but we do not know which one, so we have search everything. We will initialize this x which stands for the total the sum of all the lengths of the cut down trees after cutting down, so initialize 0. And then for each now,

we have to select an r value from i plus 1 to j minus 1, so for each r equal to i plus 1 to j minus 1, let us do this.

Now, if you want to split this problem into two independent problems, what we have decided is we will assume a fixed specific height for the tree in the added position. And we will search for various heights, that we will cover every part of the ground. What we have to keep in mind is in this search we will not consider 0 height at position r, because at least one of the intermediate tree is going to be non 0. And if all of them are 0, that case is very special we will consider that at the end.

So, the height gamma at r, this height has to be starting from alpha, it has to go to beta with the condition that it does not exceed the original height h r. So, I am going to define two parameters p which is the minimum of alpha and h r, because we will have to start with this height. While, we search we begin with this height and we will go up to a height q, let me express that because there are couple of conditions to observe. So, if beta is 0, so now, it is possible that beta is 0, hence this does not impose any condition on gamma the height that we will be considering at position r.

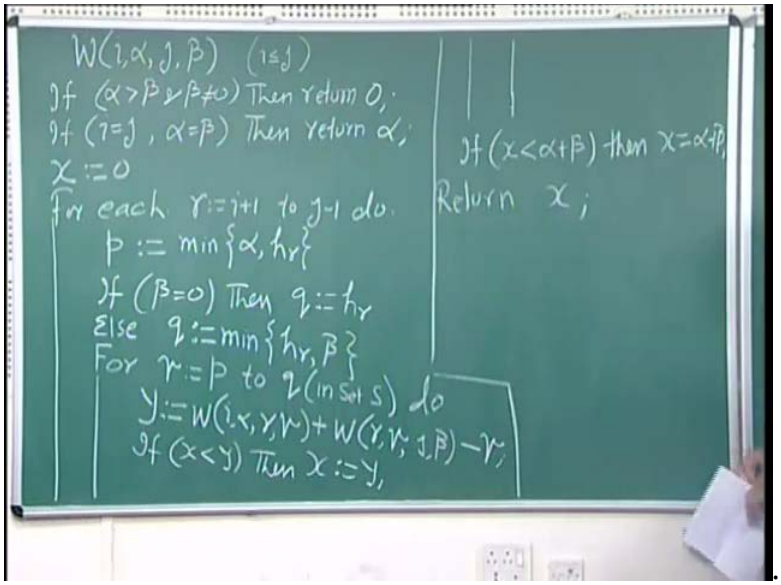
Then we will take q to be equal to h r the full height, the full height is also valid else when beta is not 0. In that situation we will take q the upper limit for gamma will be minimum of h r and beta we can go only up to that. So, now we are going to consider various values of gamma, for gamma equal to p to q. Now, here I want to point out that these are all members of S and here I am referring to various values inside set S ranging from p to q, not anything else only the numbers inside set S.

So, I will say in set S do that way I am considering all the possibilities provided h r is sorry the m r the final value at r is non 0, then we definitely are considering all the possibilities. Well, let me now see what has happened, what we have here at ith position a fixed length alpha at jth position fixed length beta, and at rth position we have a fixed length gamma. And we want to now subject to these three values fixed I want to compute the optimum, then it is very trivial, this and this have no business with each other, because monotonously ensures that this is the largest among all and smallest among all of these.

So, we can pick this best solution of this, pick the best solution of this and build the by solution of i j, i alpha j beta case. Well, the best solution we know already exists, so let

us do build a number out of it. Let us say that we will be W i alpha r gamma plus W r gamma and j beta, so this the solution of this is plus solution of this anything that it has been counted twice from subtracting gamma. So, this is a very easy to construct a solution, we have got the solutions of the two sub problems. Once, we have this, what we are going to do is check if the present value of x, if it is already better than y then there is no point in taking this.

(Refer Slide Time: 32:33)



So, we say if x is not as good as y then x is a sign y, so we have built the solution and updated the value of x. So, now this search, where we are picking every possible value of r in the range i plus 1 to j minus 1, and every possible value of gamma from p to q. We should get the best solution at the end of the day, where at least one tree is not completely wiped out between i and j.

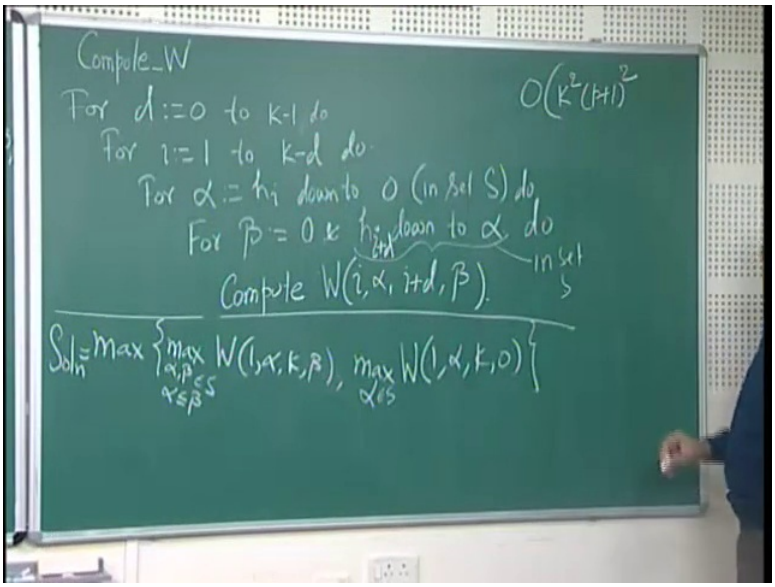
So, let us say we have over here once we come out of this 4 loop, now we are going to consider the last option namely if x is less than alpha plus beta. This is the option due to the fact that all of the values in between are 0, then x equal to alpha plus beta. So, now we are convinced that we have taken care of all possible cases, and indeed the value of h must be now at this point the optimum value of the sub problem which ranges from i to j and where the end values are fixed at alpha and beta.

So, now we can simply return x, now the thing is this solution was possible and where the assumption that these values were already available to us. The important thing is in

these two cases the difference between i and r here, here r and j these differences are less than the difference between i and j. So, if we proceed computing this W values from minimum difference to maximum difference, we can certainly compute this and we can compute this in a polynomial time.

Actually, the time is since this whole step takes p to q which is order S order the cardinality of S and this is order k in the worst case, so into k times S we will we will be able to compute this. So, now, let us try to organize this computation in such a way that all the lower values are available, when we try to compute the larger values. And now let us put together a set of loops we will compute W all values of W.

(Refer Slide Time: 36:32)



So, we begin with for d equal to 0, to k minus 1, the d denotes the difference between the 2 end points of the sub problem namely i and j. For i equal to 1 to k minus d, so what we are defining here is the range will be i to i plus d, this guarantees since we are starting from d equal to 0 to higher values. We are guarantying that the sub problems being computed are increasingly having larger and larger displacement.

Then we have to see what that we are ensuring all possible values of alpha and beta, so we are having alpha. So, the possible value of alpha can be h i that is the ith position the value of the tree can be h i, down to 0 in set S all these values which belong to set S. And the value of a beta, for beta well again it can be h j and down S to alpha starting from h j, we go down to alpha, but we are also allowed 0 value.

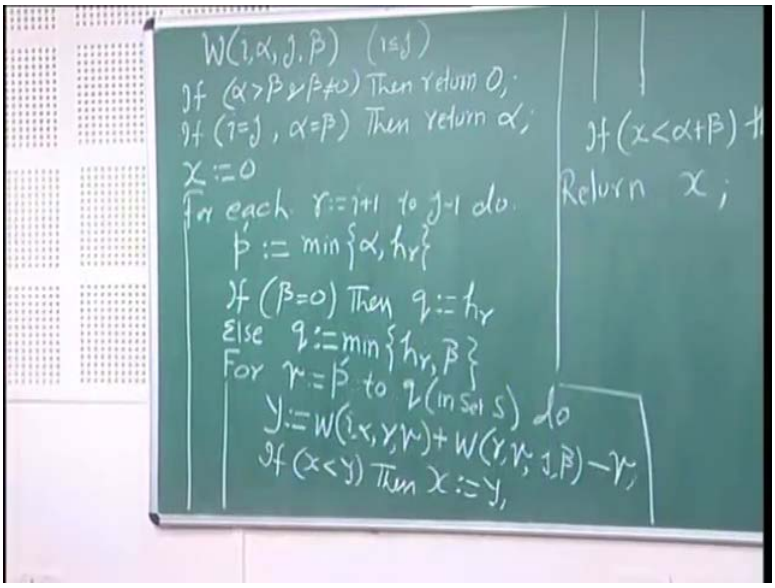
So, this will be 0 and the other sets of the values are h j down to alpha, I think we do not have j, it is actually i plus d the j values i plus d. So, i plus d is what we will write, so all values starting from h i plus d down to alpha, if alpha is already greater than beta of course, it is not of any interest to us. And special case is 0 and then we can compute W i comma alpha i plus d comma beta, this is the sub root that we have already discussed and that will work now.

So, with this we have now completed, the competition of every feasible W, so every valid set of parameters for those W value will be computed correctly. Now, the cost of the competition, there is one more thing I should actually explicitly write down what is the final solution. So, let me write down that after computing the values of the stable W, that is values for all them, meaning full set of parameters of W, we have got this. We will compute now, max of the following numbers, max of 1 comma alpha comma k comma beta, for all values of alpha and beta in set S.

I will sorry it is W of that, so the W of 1 comma alpha k comma beta such that, alpha is less than equal to beta we will consider those and of course, max of you could combine the two. That I am just explicitly writing it down, 1 comma alpha comma k comma 0 for various values of alpha in S. So, these are the finally, the numbers from which we have to pick the largest and return that is the solution of our problem, so my solution is this, this is my final solution.

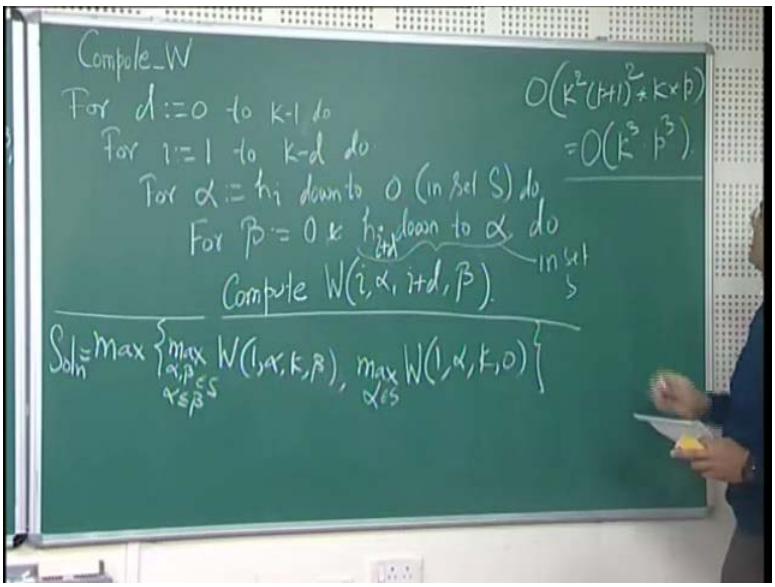
Now, let us take a look at the time complexity of the algorithm, clearly what we notice here is these loops introduce the total cost of k square and p plus 1 square in the worst case, because alpha and beta are well this also this p is also in set S. So, these two values are running inside S, which it has got p plus 1 elements, and these values run at most k times.

(Refer Slide Time: 44:15)



And now computing the value of W, if you look at the algorithm here, this runs at most k times, this one runs at most p times. So, my mistake this p is different from the cardinality of S, so ignore that we just say p prime avoid the confusion.

(Refer Slide Time: 44:48)



So, this entire thing will take order k times p times. So, let me write down this as k into p well, so this is order k cube p cube. This is the total time complexity of the algorithm which is probably not very good, but at least it is a good example of a dynamic programming based algorithm, that is all.