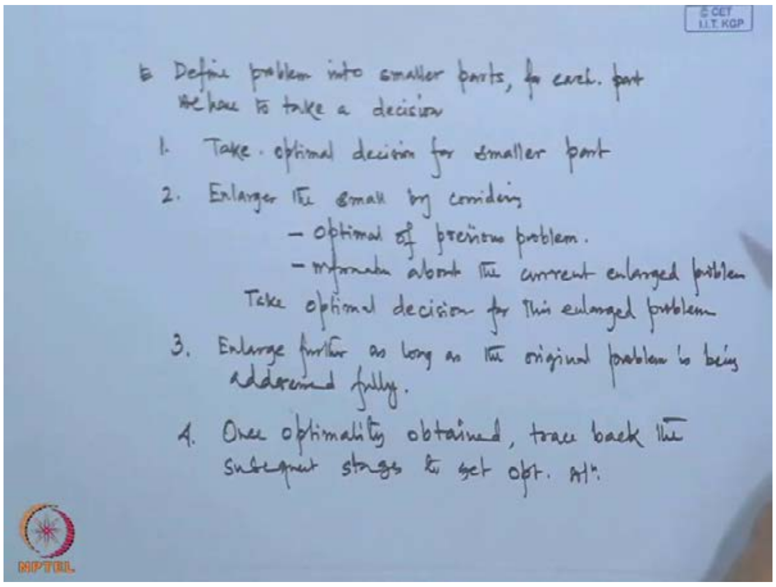
Optimization
Prof. Debjani Chakraborty
Department of Mathematics
Indian Institute of Technology, Kharagpur

Lecture - 37
Dynamic Programming Problem

Today I will discuss on Dynamic Programming Problem, in practical decision making situations, decisions have to be taken in different sequentially in different points, may be points in time, points in space rather for different subsystems. Now, dynamic programming is an optimization tool that transforms a complex problem into sequence of simpler sub problems. So, that we can solve the sub problems and we will get the optimal solutions of the sub problems sequentially to get the optimal solution for the original problem.

Now, since the decisions are being taken at different points in time or in spaces that is why, we are taking the decision in different stages that is the dynamic programming problem is also being named as the multistage decision problem.

(Refer Slide Time: 01:27)



Define problem into smaller parts, for each part we have to take a decision

1. Take optimal decision for smaller part
2. Enlarge the small by considering
   - optimal of previous problem.
   - information about the current enlarged problem

   Take optimal decision for this enlarged problem
3. Enlarge further as long as the original problem is being addressed fully.
4. Once optimality obtained, trace back the subsequent stages to get opt. sol^n.

The basic philosophy of a for solving a dynamic programming is that, the first step is that define the decision situation into smaller parts. Rather we have to decompose the whole problem into smaller parts, and for each part we have to take a decision, this is the basic approach for the dynamic programming. Now, we will take a small part of it, and we will take the optimal solution for the small part.

Since we have considered a subset of the whole program, now we have taken the optimal decision for the smaller part. Then we have to enlarge this small part, the small problem by considering optimal solution of the previous problem, we have to consider the information about the current stage, current state, rather current enlarge problem. Then we have to take a decision of this enlarged problem, and we will repeat the process, we will repeat the process means we have started from a smaller part, we have a enlarged it a little bit then we have to enlarge further.

How the enlargement is being done, how the smaller problem will be converted to a less bigger problem that I will show you with the example in the next. And that is the technique for solving the with the dynamic programming methodology, enlarge further as long as the original problem is being address fully, and we have to take the optimal decision of the last bigger problem. And once the optimality we reach, then we have to trace back, the subsequent stages to get the optimal solution as a whole.

Now, if we see the whole process the beauty of the process is that, the big problem has been divided into smaller parts. That means, we have decompose the whole problem in different stages that is why, we are sequentially reaching from stage 1 to stage n, so that stage we are fully addressing the whole problem. And whatever optimal solution we have getting at the end, and we have each step you have to optimization technique, and that would be the optimal solution of the problem.

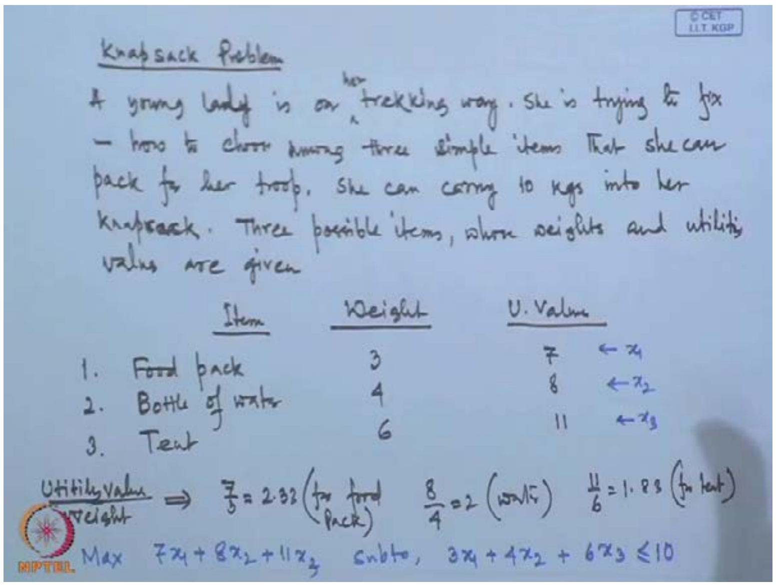
Now, this methodology this dynamic programming concept has been invented in 1950's by American mathematician Richard Billman, and dynamic programming dynamic decision. Since, we are taking decision at several stages at dynamic, and programming stands for the planning or to just set the actions in the best possible way.

And if we see the whole process you could realise that, the process we are repeating the optimization technique again and again for further large enlarge problem that is why, this is a recursive process we are adopting that is why in other way we also can say that dynamic programming as a recursive optimization thus. We can name the dynamic programming as either multistage optimization process or recursing optimization problem.

Now, the process as a whole it is scribed to you, there are few things to be discussed that how the problem can be divided into smaller parts. Rather, how the problem can be divided into smaller stages, and how to reach the optimal stage these are the basic understanding for dynamic programming problem. That is why let us start before going to the detail of dynamic programming technique, let us take a simple example several applications of dynamic programming problem, one of the most popular application is the shortest problem we are applying the dynamic programming technique.

In the knapsack optimization problem also we are applying the dynamic programming technique, otherwise we are also using for inventory management problems as well.

(Refer Slide Time: 07:42)



©CET I.I.T. KGP

Knapsack Problem

A young lady is on her trekking way. She is trying to fix – how to choose among three simple items that she can pack for her troop. She can carry 10 kgs into her knapsack. Three possible items, whose weights and utility values are given

	Item	Weight	U. Value	
1.	Food pack	3	7	$\leftarrow x_1$
2.	Bottle of water	4	8	$\leftarrow x_2$
3.	Tent	6	11	$\leftarrow x_3$

$\frac{\text{Utility value}}{\text{weight}} \Rightarrow \frac{7}{3} = 2.33$ (for food pack) $\frac{8}{4} = 2$ (water) $\frac{11}{6} = 1.83$ (for tent)

Max $7x_1 + 8x_2 + 11x_3$ subto, $3x_1 + 4x_2 + 6x_3 \leq 10$

Let us take a small knapsack problem for further discussion about the methodology of the dynamic programming problem. This is an optimization problem, let me write down the statement of this problem first, a young lady is one her trekking way she is trying to fix there is a she is having a knapsack, and there is a specific volume of that knapsack. Accordingly she is planning, she wanted to carry food pack, she wanted to carry water bottles, as well as tent for a hut troop that is why let me just put the situation in this way.

She can carry at the most in her knapsack, she can carry at the most 10 k g's we wanted to utilise the whole space of a knapsack. But, there are certain constraints certain information related to the things, which she is trying to pack up three possible items with their utility values are given, utility values are something these are being utility values are given that is from a her from her experience she gathered the values. This is certainly subjective in nature, and the utility values.

The first item is the food pack, unit food pack is having the weight 3 and the utility value 7, 2 that is the bottle of water. Individual bottle of water is having the weight 4 k g and utility value is 8, and if she wanted to carry that tent with are, the unit tent having the weight 6 k g and the utility value is 11. This is the only information is given to us, now we have to suggest to the young lady how to how she can plan, so that 10 k g will be utilised fully not only that, she can maximize hard satisfaction level.

That means, we wanted the maximize utility value for this problem, if we see from the given information. We can see that, if I just consider the ratio utility value, value divided by weight, what we can see this values are for the foot pack 7 by 3 it is; that means, 2 by 33. Similarly, for the bottle of water it is 2 and for the tent it is coming 11 by 6 1.83 that is for tent, as we could see that food pack is having better utility value, unit utility value that thus we can suggest in adhoc way that you just consider the food pack as many as you can.

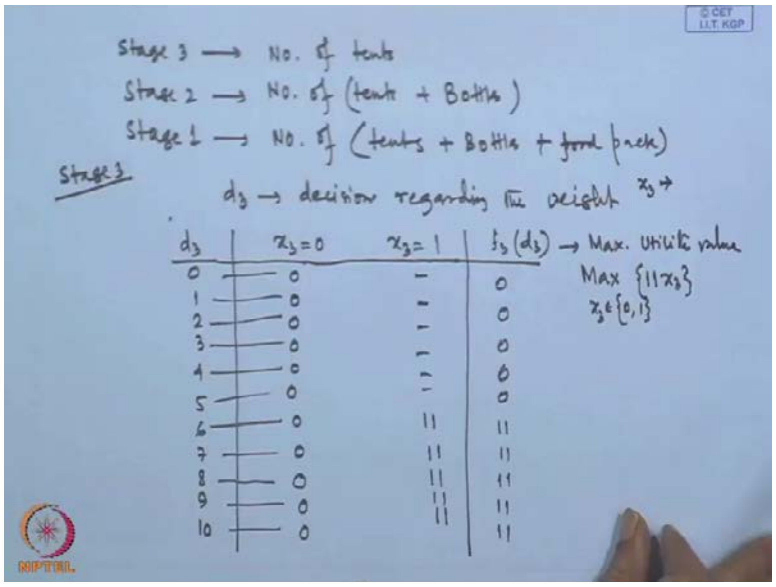
But, if we see that data set it is having the weight 3 k g s that is why, she can carry at the most 3 food packs. But, 1 k g will not be utilised that is why the whole constraint will be under utilised in this case that is whatever decision we are giving that 3 food packs that is not the very good weight, why is weight to suggest that is why let we have to take a decision in a scientific way. That is why the optimization techniques are available with us, and there is one way for solving this that is the integer programming problem, we can vary nicely just constructing integer programming problem in this way.

If we consider x 1 is the number of food packs x 2, number of food pack number of bottle of water and x 3 is the variable for the number of tents. Then we can find out the maximization of the total utility value 7 x 1 plus 8 x 2 plus 11 x 3, and there is a constraint subject to that is the total weight must be lesser than 10. And here we need to consider x 1 and x 2 and x 3 these are all the integer values, we can solve with a technique you have learnt the unbound technique in other technique of integer programming we can solve it.

But, we do not want to do it, we want to apply the dynamic programming, dynamic programming technique for solving this programme. And as I said, this is though it is a simple problem for us, but since we are doing it manually we can considering as the larger problem, we will break the problem into the smaller problems that is why we will break the problem into multi stages, and we will solve the problem. How we can divide the situation in different stages, one process could be first we will decide that if I want to utilise 10 k g's of weight.

Then first let us start from the back, we will first select how many number of tens we will consider that could be the first stage. Second stage could be the how many tents, as well as number of bottle of water will consider together all right that could be second stage. And the third stage is the tent number of bottles of water and the food packs in that way we can divide in 3 stages.

(Refer Slide Time: 15:11)



Stage 3 $\longrightarrow$ No. of tents

Stage 2 $\longrightarrow$ No. of (tents + Bottles)

Stage 1 $\longrightarrow$ No. of (tents + Bottles + food pack)

Stage 3

$d_3 \rightarrow$ decision regarding the weight $x_3 \rightarrow$

d_3	$x_3=0$	$x_3=1$	$f_3(d_3)$
0	0	–	0
1	0	–	0
2	0	–	0
3	0	–	0
4	0	–	0
5	0	–	0
6	0	11	11
7	0	11	11
8	0	11	11
9	0	11	11
10	0	11	11

$f_3(d_3) \rightarrow$ Max. Utility value

$\underset{x_3 \in \{0,1\}}{\text{Max}} \{11 x_3\}$

Thus we can say that our stage 1, stage 3 let me start with stage 3 that is the number of tents. Stage 2 number of tents plus bottles of water, and stage 1 that is the bigger problem for me that we wanted to address that is the number of tents plus bottles plus food pack. There is a constraint the total weight must be equal to 10 k g's, we wanted to utilise if it is just proceeding this way, then this process is being named formally in dynamic programming as a backward recursive process.

Because, we are starting from stage 3, then stage 2 then stage 1, we are considering the whole. We can do the reverse way as well, we can consider number of food packs cost then the number of food packs plus bottles, then number of food packs plus bottles plus number of tents in that way we can also consider. Now, these are different stages we have considered, now let us see once we are considering stage 3, what is happening in stage 3.

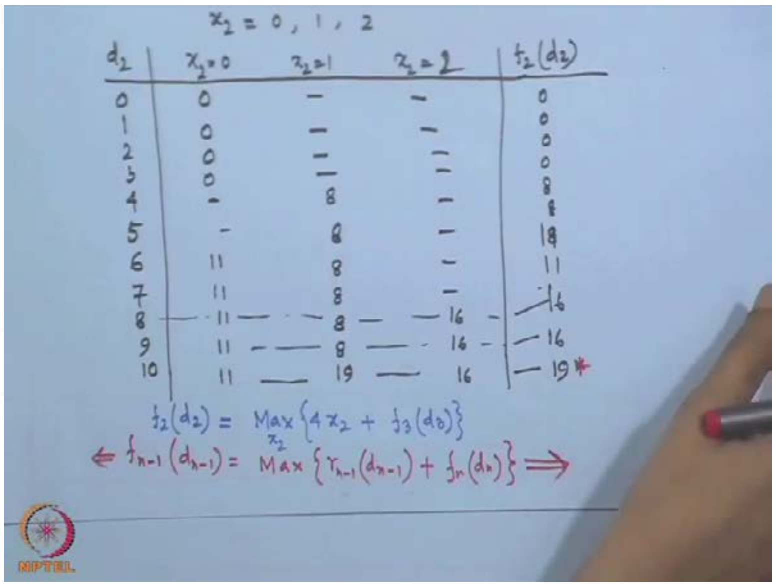
If we consider a variable as d 3 about the decision regarding the weight, in stage 3 we have consider only the tent that is why. The decision about the weight of the tent can be from 0 to 10 any value, first let me consider d 3 as 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 all right, and if we see the situation we can associate another variable say x 3, x 3 is the number of tents we wanted to suggest, this could be either 0 or 1. Because, as we know the tent weight is 6 k g that is why either x 3 can be 0 or x 3 can be 1, there is no other possibility for this.

If this is, so let us write down the utility values for the entire situation for 0 0 if the number of 10 0, then total weight 0, number of tent 1 total weight is 0 that is impossible infeasible solution that is why let me put it dash here. Number of 10 0 the total weight 1, these are all the possible, but if I just move 0, 1, 2, 3, 4, 5 if I move to 6, then what is the value for x 3 is equal to 0 that is 0, but if I consider that I we have we are suggesting one tent for here.

Then d 3 is 6 k g; that means, decision regarding the weight in the knapsack if it is 6 k g’s, then what is the utility value, utility value is 11 here. Now, go to the next 7 this is 0 the utility value 11 again, this is the simply situation and let me just complete the whole table in this way 11 and 11 here. And all other coming in feasible solution, if this is the case there are only one option in each case, but it is not happening for other stages.

Because, once we are moving to the further stages we will see that situation will be much more complicated, in that case we have to select the higher utility value that is our objective. Thus we can say that f 3 is an is the objective in the stage 3, which is the maximization of the utility value, if we just write in this way otherwise we can write mathematically maximize utility value is 11 x 3, x 3 could be either 0 or 1. Mathematically we can write it, then we can see the utility values are coming this way 11, 11 this is simple situation that is for stage 3.

(Refer Slide Time: 20:04)


$x_2 = 0, 1, 2$

d_2	$x_2 = 0$	$x_2 = 1$	$x_2 = 2$	$f_2(d_2)$
0	0	–	–	0
1	0	–	–	0
2	0	–	–	0
3	0	–	–	0
4	–	8	–	8
5	–	8	–	18
6	11	8	–	11
7	11	8	–	16
8	11	8	16	16
9	11	8	16	16
10	11	19	16	19*

$$f_2(d_2) = \underset{x_2}{\text{Max}}\{4x_2 + f_3(d_3)\}$$

$$\Leftarrow f_{n-1}(d_{n-1}) = \text{Max}\{r_{n-1}(d_{n-1}) + f_n(d_n)\} \Longrightarrow$$

Let us move to the next stage that stage 2, what is happening just see here also we are considering d 2 as the decision regarding the weight of tent, as well as bottle total weight. Already we have considered weights of tents as d 3, and what about weights about bottles if I consider x 2 is the variable regarding the number of bottles, we wanted to consider then d 2 must be is equal to d 3 plus 4 x 2.

Because, 4 is the unit weight for bottle of water if this is, so and then d 2 is the variable, and there is a name for this d 2 variable I will name it formally later on I will tell that for stage there is a set of set values, there is associate state variable that part I will discuss later on. We can see the d 2 is equal to d 3 plus 4 x 2, and what else we can say since the weight of the bottle is 4 k g, then either we can consider x 2 is equal to 0 or when or 2 at the most not more than that all right.

Let me complete the table as we did for the other set x 2 is equal to 0, x 2 is equal to 1 and x 2 is equal to 2 1 we will write on the utility value for each and every cases. Here also the same d 2 can be from 0 to 10, if d 2 is 0 then this is 0 if d 2 is 1 0, d 2 is 2 3, but once it coming through the weight d 2 equal to 4. That means, we are considering one set of bottle, we cannot consider the tent here d 3 is 0 rather here, then what should be the utility value just now we have to considered in the table that the utility value of the bottle of water is 8.

Thus we can say for this is 0, this is 8 let me consider 5 if it is 5, again there is only one possibility x 2 must be is equal to 1, there is no possibility if I consider 6. Then there is a possibility that, it could be 1 tent tats why the utility value is 11, we can consider as well 1 bottle even, still the knapsack is underutilised still we may consider. Because, we wanted to see each and every alternatives for this different decisions all right, if it is 7 here also the same situation.

But, once it is coming to 8, either we can consider 1 tent or we can consider 1 bottle, we may consider 2 bottles as well that is why possible. Because, total k g is 8 k g and our maximum limit is 10 therefore, it is quite possible for 8, we can have three alternatives, and if it is 9 we can consider 1 tent or we can consider 1 bottle of water or we can consider 2 bottles of water all right. If it is 10 the beauty here comes, we can consider one tent x 2 is equal to 0 x 2 equal to 1; that means, we are considering 1 bottle, but there is a provision to have 1 tent as well.

Because, 1 tent 1 bottle will be total will 10 k g that is why what is the total utility value here 11 plus 8 that would be 19 here. And here if we consider 2 bottles there is no space for tent that is why 16, let us see the utility values for the individual cases, when d 2 is equal to 0 the utility value is 0 if I just move in this way up to 3 this is 0 up for 4 it is 8 again for 6 its 11, for 7 it is 11, for 8 maximum value is 16, this is for 8, this is for 9 maximum value is 16, but see for 10 the maximum value is 19.

Thus if we without knowing even the much complicated dynamic programming technique, here also we can decide that, that could be the optimal solution up to this stage. If I consider this problem as an as individual optimization problem, then we can say that this is the optimal solution, and here you see we have just considered the value of the previous stage. And utility value and the utility value of this stage, without considering further else.

If I go to say stage 1, just see what is happening that is why this optimal solution for stage 2. Let us go to stage 3 and also we can say many thing here just see, we can develop a relationship, we can say that f 2 d 2 equal to maximization of 4 x 2 all right plus f 3 d 3 clear and over x 2, x 2 could be 0 or 1 or maximum value coming 8 plus f 3 d 3 just now we have calculated that was 11. That is why we can say that, this could be the relationship between the previous utility value, and previous objective function, previous stage objective function the current stage objective function.

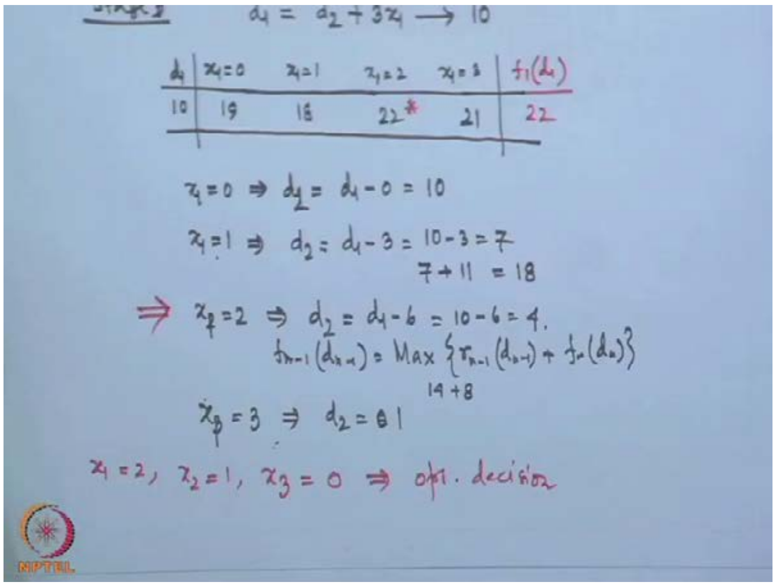
Here, we will see later on that we can develop a nice result from here, we can write down the result as f for the backward recursive relation f n minus 1 d n minus 1. What is your d 2, d 2 is in this stage the d 2 can take any value d 2 is the decision we wanted to take it could be 0, it could be 10 even, it could be in between any value. Thus here also if we consider the n minus 1'th stage for a general problem we will see that, that would be maximization of this is the notation.

Let me use it now I will describe later onwards the meaning of r n minus 1 d n minus 1 this means that, immediate return we are getting after selecting d n minus 1. If I see here for d 2 is equal to 9, d 2 equal to 9 what is the immediate return, immediate return for x 2 equal to 1 that is 8 that is why we have considered in this way plus f n d n, you can understand this relation later on. This is a very important relation in our dynamic programming technique, I will discuss later on this one.

At least we can say that, the optimal objective functional value of the next stage is very much dependent on the objective functional value of the previous stage. Thus I said if you remember I just started my lecture with that thing, that we have to first take the smaller part of the problem. Then we have to enlarge the problem little bit by considering the optimal value of the previous problem, and we have to consider the return value, rather the information regarding the decision at this stage in detail.

And both considering we have to take the optimal decision for the next stage, and we have to proceed further and further as long as we can average the whole problem totally.

(Refer Slide Time: 28:46)



$d_1 = d_2 + 3x_1 \longrightarrow 10$

d_1	$x_1=0$	$x_1=1$	$x_1=2$	$x_1=3$	$f_1(d_1)$
10	19	18	22*	21	22

$x_1=0 \Rightarrow d_2 = d_1 - 0 = 10$

$x_1=1 \Rightarrow d_2 = d_1 - 3 = 10 - 3 = 7$

$7 + 11 = 18$

$\Rightarrow x_1=2 \Rightarrow d_2 = d_1 - 6 = 10 - 6 = 4.$

$f_{n-1}(d_{n-1}) = \text{Max}\{r_{n-1}(d_{n-1}) + f_n(d_n)\}$

$14 + 8$

$x_1 = 3 \Rightarrow d_2 = 1$

$x_1 = 2,\ x_2 = 1,\ x_3 = 0 \Rightarrow$ opt. decision

Now, let us move to the next stage that is stage 3, stage 3 means what, we wanted to consider tents, we wanted to consider as well as the number of bottles of water, as well as the number of food packs, this is stage one I started with stage 3. The naming convention is up to the person up to the decision making that is why I can say that d 1, that is the number of that is the capacity that is the total weight. In this stage is d 1 if it is, so this must be is equal to the weight of coming from the previous stage plus weight of the current stage that weight is 3 x 1.

Because, if I consider x 1 number of food packs, then total number of total weight would be 3 x 1, and d 2 is the weight optimal weight just now I consider from the previous stage that is for tents as well as for bottles. If this is the case we wanted that d 1 must be 10 because, considering d 1 considering we have fully, now we are fully addressing the whole problem that is why it is not better to consider d 1 as 10. If we consider d 1 starting from 0, 1, 2, 3, 4, 5 that these not very much good way to think it, that is why better to consider d 1 as the wanted to utilise the full space that is why d 1 is 10 here.

Now, once d 1 is 10 then what about the value for x 1, x 1 k g's are 3 k g individual in unit k g that is why x 1 could be 0, x 1 could be 1, x 1 could be 2 and x 1 could be 3 even. Because, if I consider number of food packs 3 at the most it would be 9 k g's, now we wanted to utilise let us see the whole situation in this stage, if we consider x 1 is equal to 0 what is it mean, x 1 equal to 0 means from here d 1 is equal to rather d 2 let me consider d 2, d 2 is equal to d 1 minus d 0 that is equal to 10 only.

If I go back to 10, the previous stage if d 2 is equal to 10 the optimal corresponding objective functional value is 19. And here the return is 0 for this case that is why 0 plus 19 it would be 19 only all right, if we consider x 1 is equal to 1 it means that d 2 must be is equal to d 1 minus 3. That means, 10 minus 3 that equal to 7 go back to the previous table, for the 7 the corresponding value objective functional value is 11, if I just substitute here, return from this state we have considered 1 that is why return will come as 6 no 7.

If I consider 1 food pack the return is 7, and here the return is for 7 the return is 11 that is why we will consider 7 plus 11 that is equal to 18 thus this is 18. Let us consider x 2 is equal to 2, which means d 2 is equal to d 1 minus 6; that means, 10 minus 6 that is equal to 4, go back to the previous table. The 4 the corresponding utility value optimal utility value is coming 8 that is why if I go to the formula f n minus 1 d n minus 1 is equal to maximization of r n minus 1 d n minus 1 plus f n d n.

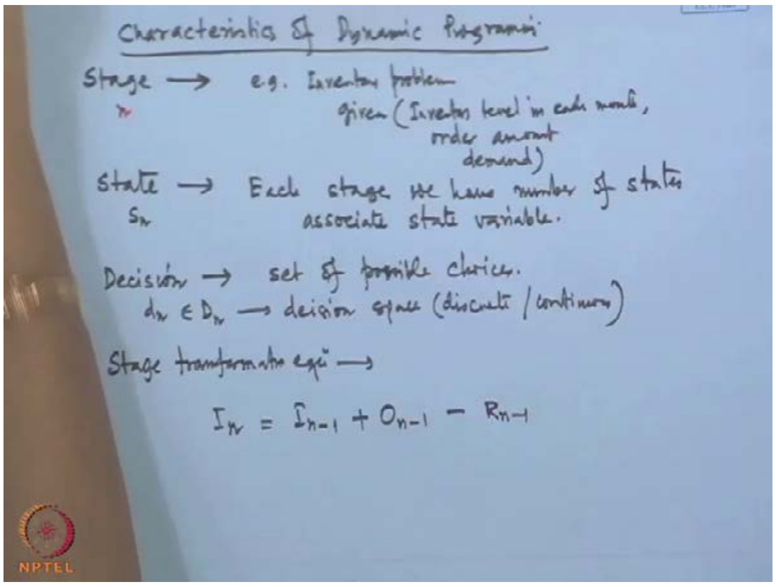
Just now we have considered f n d n is equal to 8, and what about r n minus 1 d n minus 1, we are considering 2 food packs that is why total utility value will be 14 it is coming 14 plus from here 8 that is coming 22 clear. If I consider x 1 why I have written x 2 this is x 1, x 1 equal to if it is 3 then certainly d 2 must be equal to 0, d 2 must be not 0 d 2 must be is equal to 1. That means, we cannot consider any tent at all the corresponding value is 0 that is why the utility value is coming only for 3 food packs that is coming 3 into 7 21.

If it is a c this stage, there are something to be decided that, the maximum utility value is coming for this option only that is why this is the optimal solution for us. And if I consider f 1 d 1 that would be is equal to 22 here, and this is the one thing and here we can conclude that how many food thing, how many food packs etcetera, how many items we will consider one by one. For this case if I go back to this case what we have seen, we have seen that x 1 is equal to 2 and corresponding d 2 equal to 4 go back to the previous page d 2 equal to 4 means we have considered x 2 equal to 1 all right.

And we have considered x 3 is equal to 0 that is the optimal decision, and I just explain the process in detail. Because, just we wanted to say few things up to this that, in every stage if I just consented on this stage the last stage, in every stage the optimal objective functional value is dependent on the optimal objective functional value in the previous stag. Without looking at that how we have reached to that position that is very important for dynamic programming technique.

Thus the bellman win he invented these methodology, and there is a principle of optimality by bellman. And it has been really said that very nice optimality theory I will just formally introduce that part right now.

(Refer Slide Time: 35:17)



Characteristics of Dynamic Programming

Stage $\rightarrow$ e.g. Inventory problem
n given (Inventory level in each month, order amount demand)

State $\rightarrow$ Each stage we have number of states
S_n associate state variable.

Decision $\rightarrow$ set of possible choices.
$d_n \in D_n \rightarrow$ decision space (discrete / continuous)

Stage transformation eqn $\rightarrow$

$$I_n = I_{n-1} + O_{n-1} - R_{n-1}$$

The characteristics of dynamic programming, since the problem I have explained hopefully it is clear to all of you how we have considered the stages etcetera. That is why for applying dynamic programming technique, first thing is that we have to consider the stage, we have to decompose the whole problem into smaller parts that is called the stage, and this is essential feature of dynamic programming problem. And stages are being computed, stage we have to get the optimal solutions of individual stages sequentially.

Now, the stage can be dependent on time it may not dependent on time, and the problem I just consider there is no dependence of time. But, if I consider simple example that is for the inventory problem, we wanted to take the decision that about the quantity to be ordered in every month. That means, the decisions are being dependent on very much on time, if we are in we have a dataset from there the experience you just wanted to design that in which month, how much to order.

For that thing, we need few information with us it is given that inventory level in each month. We have the order amount in each month, not only that we have the demand rate as well in each month, from there we have to take the decision how much to order in each month, this is the data for previous year it could be. Now, I just wanted to say that the bigger inventory problem can be subdivided into smaller problems by considering in each part there is a small optimization problem as if we have.

One we are defining the stage, the next critical part is to define the stage, generally stage is being termed as n, n can start from the n to 0 or we can start from 0 to n. Depending on that we as I have just now developed the backward relationship from stage 3 to stage 1, similarly we can go from stage 1 to stage 3. And we can develop the forward recursive relation for the problem that is the thing about.

Now, regarding the stage for each stage there is a stage, since we have different states in stage that is why we will just introduce we will associate some state variable. For the previous problems sate variable could be for stage 2 it was x 2, for stage 3 it was x 3 like that, thus we can have different values for different state for individual state variables, these are these correspond to different states. And there is no state rule, how we will define the state.

As we have considered the state variables like by considering the stages with first tent, then tent plus bottle of water plus food pack. Instead of that one may consider tent bottle of water and food pack I individual stages etcetera that is also quite possible there is no set rules from the experienced, you have to do it that is why it is very critical to find out the states. Generally states we are just writing with the variable S n, there we have written x n in the previous problem.

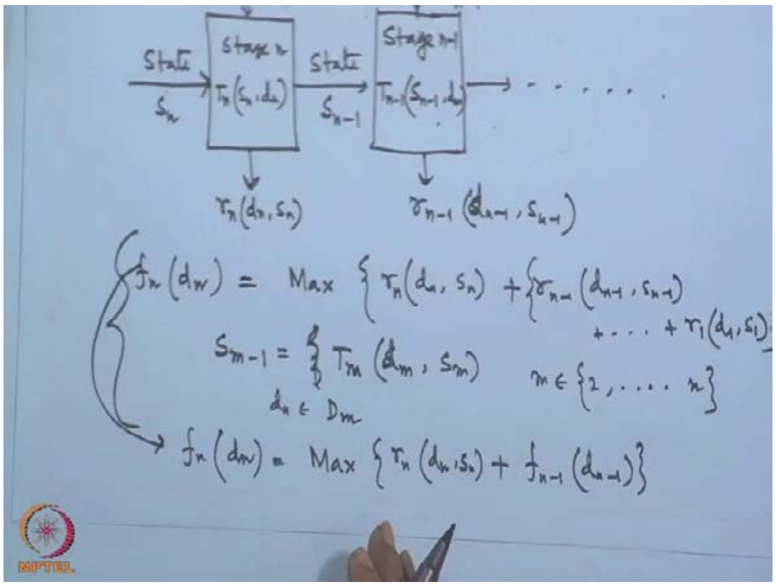
There is another part is there is a decision variable as we have done there d 1, d 2, d 3 these are the decision variable. There is a set of possible values of the decision variables; that means, there is a domain of decision variables, and that domain is being named if I consider decision variable as d n, domain may be considered as capital D n, D n is the decision space.

And regarding the decision space, it could be discrete it could be continuous that is why, once you are learning the dynamic programming technique you have to know the detail of the discrete that decision space how to handle as well as the continuous decision space how to handle that both the things you have to learn together that is why it could be discrete, it could be continuous.

And second very important thing is that, that is called stage transformation equation or transformation equation also it says. And this equation tells you that, how one set of the previous stage is being transformed to the next state of the next stage; that means, one stage in stage n how it is being converted to the next state of stage n minus 1, whose responsible this stage transformation equation is responsible.

For example, as I have taken the inventory problem, in the inventory problem I can consider, if I consider the inventory level as at n'th stage. This is very much dependent on the inventory level of the previous state plus ordered in the previous month minus demand that is the consumption in the previous month, if I consider it as R if it is, so this is the stage transformation equation for the inventory problem because, as you could see that I n is function of I n minus 1 that is the beauty of it.

(Refer Slide Time: 42:08)



$$f_n(d_n) = \text{Max}\left\{ r_n(d_n, S_n) + \{ r_{n-1}(d_{n-1}, S_{n-1}) + \dots + r_1(d_1, S_1) \right.$$

$$S_{m-1} = \{ T_m(d_m, S_m) \quad m \in \{2, \dots n\}$$

$$d_n \in D_m$$

$$f_n(d_n) = \text{Max}\{ r_n(d_n, S_n) + f_{n-1}(d_{n-1}) \}$$

Now, this is the next part, now if I just draw that figure of a multistage decision making process. Then we would see that state S n and the decisions d n, these are the inputs for stage n, then we will have the stage transformation equation S n, d n and we will move to the next state that is S n minus 1. That means, we will move to the next stage, where we have different states stage n minus 1 all right, and if these are the inputs we are moving to the next, there is a for individual.

As we have seen for individual stages, we have considered the optimal return value that is why we may consider the optimal return value as r n. This is again the function of d n and S n, here also the stage transformation equation is there for each stage we have the stage transformation equation. And again we have the optimal return in each stage, and in this way we will just move to the next state or next stage, again and again to reach to the whole problem.

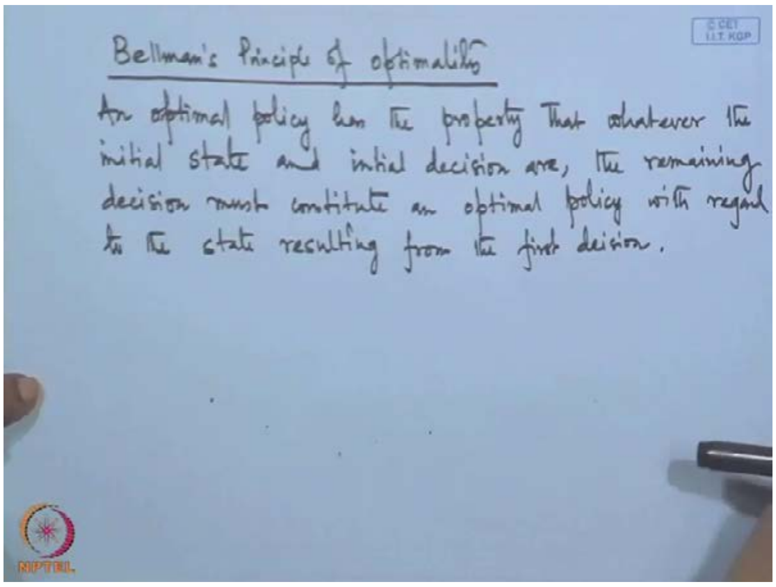
And if I just write in a mathematical rotation this fact, whatever I have done in this figure to same thing, if I just want to write in mathematical rotation that, then we have to write down that is the decision that is the objective functional value in the n'th stage is equal to maximization of return of individual stages plus if I consider this stage, here is a input decision about d n minus 1. Then at this stage, if I want to consider the decision this is dependent on the return of this value plus return of the previous one.

In this way we will just proceed to r 1, d 1, S 1, where we have considered the stages S has the through the stage transformation equation. Because, as we have seen this T m function is responsible for this, where m is running between 2 to n because, if I consider it as n, n minus 1 S n minus 1 is the function we are getting T n, d n, S n in this way and it will end to m equal to 2. Then only we will reach to the last stage S 1 and it is coming from then the 3 all right.

And here the decisions are in the decision space, just wanted to write down in a mathematical way this thing. In other way also we can write it just see, you will get the you have seen the same pattern previously that f n d n is equal to maximize r n d n, we can write S n as well if I just consider the whole part from here to here, this is the f n minus 1 d n minus 1 thus the n'th stage value is dependent on the n minus 1'th stage for the forward recursive relation.

In the backward recursive relation the situation is totally different from here, the next the formal statement of optimality has been reached.

(Refer Slide Time: 46:25)



Bellman's Principle of optimality

An optimal policy has the property that whatever the initial state and initial decision are, the remaining decision must constitute an optimal policy with regard to the state resulting from the first decision.

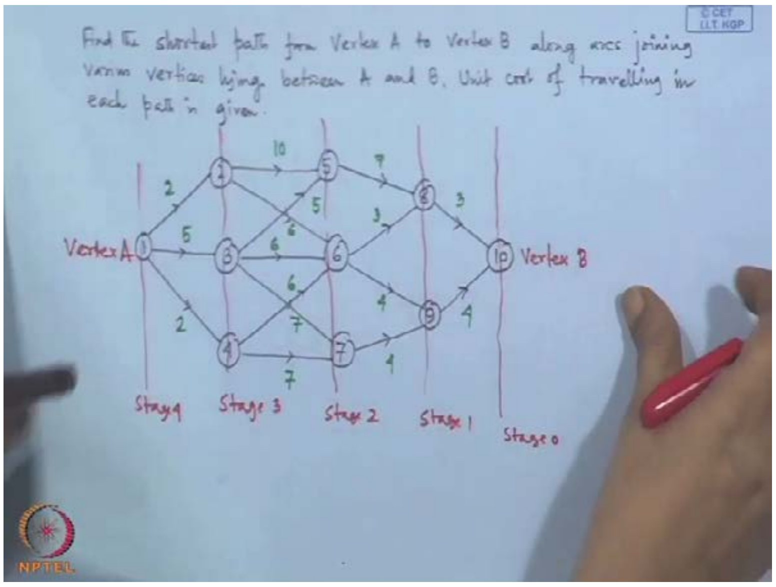
And let me write down that optimal bellman’s principle of optimality, if I write down formally the bellman’s principle of optimality it says that an optimal policy has the property that whatever the initial state is I have just copying the bellman’s principle of optimality. Because, that is very important you have to learn the thing in a formal way, I will explain it further constitute, and the remaining decision one must continue constitute and optimal policy which regard to the state first decision.

Now, what is the meaning of it means that for a sequence of the decision in multistage optimization problem, in dynamic programming problem regardless the current decision d n, and the current state S n all the subsequent decisions must be optimal. Now, the problem we have solved before you must have seen that, if I am in one stage n'th stage I am just using the optimal solution of the previous stage. And what how we have reached that point that is not more important for us, only important is up to reaching that point what is the optimal value.

And according to that we have taken the decision, that has been said in the bellman’s principle of optimality that is very important. And if we just see the that is the shortest of the problem, there we will see that if I just see the shortest path you know the definition of shortest path problem. And I will just explain the thing in a better way in the next, and we will see that, if I consider a particular stage at that stage it is not important for me how we have reached to the state stage from the initial point it is more important for me.

In the previous stage, what was the optimal value, how much distance I have covered, how much minimum distance I have covered, how much minimum cost I have to incurred for it. That information only needed because, that was the optimal solution of the previous stage, that is why how we have reached to that that is not important for taking decision at the current state. That's the bellman's principle of optimality, and that can be very nicely explained with the shortest path problem just look at the shortest path problem.

(Refer Slide Time: 50:08)


Find the shortest path from Vertex A to Vertex B along arcs joining various vertices lying between A and B. Unit cost of travelling in each path is given.

This is the problem we have to move from vertex A to vertex B, and there are different path in between different cities are there, we wanted to travel from city 1 to city 10 say these are the different alternatives in front of us, we have to take a decision that in which way shall we move. So, that the cost is minimum, and for travelling from one node to the another node individual costs are given with the green colour pen, and we have to take the decision accordingly how shall we approach for this.

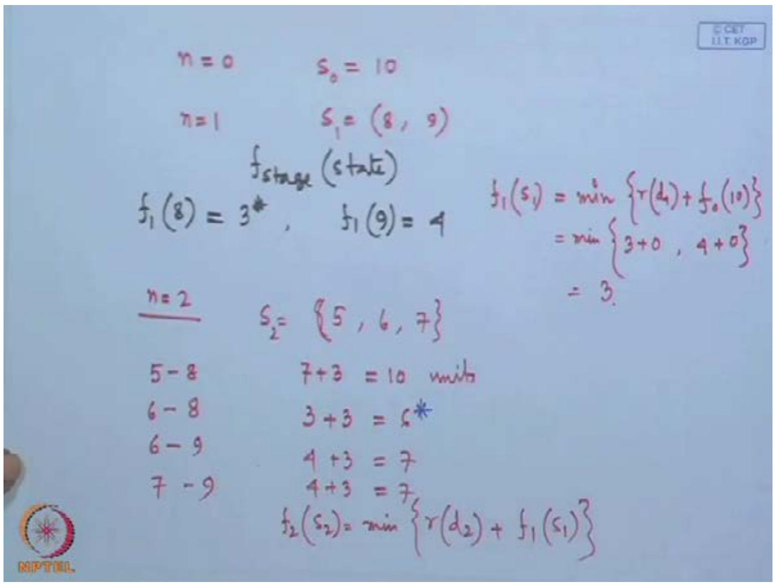
For approaching to this problem, first of all we have to divide the whole problem into different parts. That is why we have to divide the problem into different stages, we can go forward we can come backward as well, we can start our journey from these points vertex A we can start our journey from vertex B as well. If I just move from vertex A to vertex B with the iterative process as I explained before, with the bellman's principle of optimality at individual stage we will take the decision, and we will reach to the ultimate stage to get to address the whole problem.

Now, we that recursive relation is the that processes are forward recursive process, we can go backward as well that is starting from vertex B to vertex A. That is why let me first divide the problem into different stage, see we can name these stages as stage 0, the next stage where only the nodes 8 and 9 are there, stage 1, this is stage 2, this is stage 3, this is stage 4 just. If I just divide the whole problem into these different stages, then first there is nothing to decide at stage 0, you will just jump we will just move from 10 to the 8 or 9.

Because, we wanted to go backward, in stage 1 we have to just take a decision whether to chose 8 or 9 from the figure it is very clear that is 3, that is why we will select the 0.8. Once we have reached to 8 we need not consider anything about the previous stage, we are happy with this decision, we have this optimal value is only needed for the next. We will move to the next stage, stage 2 if I move to next stage 2 if you see if I just associate a state variable as I say that for each state there is a state variable. That is why state variable either can be 5 or 6 or 7.

Thus the state variable can take any value of these three, but which one to select for that thing we have to select the written from the previous stage only, that part let me just elaborate in the next.

(Refer Slide Time: 53:23)



$n = 0 \qquad s_0 = 10$

$n = 1 \qquad s_1 = (8, 9)$

$f_{stage}(state)$

$f_1(8) = 3^*, \quad f_1(9) = 4$

$f_1(s_1) = \min\{r(d_1) + f_0(10)\}$
$= \min\{3+0, 4+0\}$
$= 3.$

$\underline{n = 2} \qquad s_2 = \{5, 6, 7\}$

$5-8 \qquad 7+3 = 10$ units

$6-8 \qquad 3+3 = 6^*$

$6-9 \qquad 4+3 = 7$

$7-9 \qquad 4+3 = 7$

$f_2(s_2) = \min\{r(d_2) + f_1(s_1)\}$

Once we are in the stage 0; that means, we are at the vertex 10 here, let me put n is equal to 0. And the state variable s is equal to 10, when n is equal to 1 then state variable could be either 8 or 9 as we have seen, and let us see the corresponding objective functional value at different alternative. Because, we are moving in a discrete decision space, here we have only two alternatives with us either 8 or 9.

That's why we can take a decision here, we can formulate the this is the naming convention, we can formulate this objective functional value. That means, we have been consider f 1 8 that is from 10 to 8 this is coming 8, and f 1 9 that is the root 9 to 10 this is 4. And from here we will certainly select 3 that is my optimal value if I move to the next; that means, we are moving through n is equal to 2, how many alternatives we had. We had 5, 6 or 7 these are the alternatives.

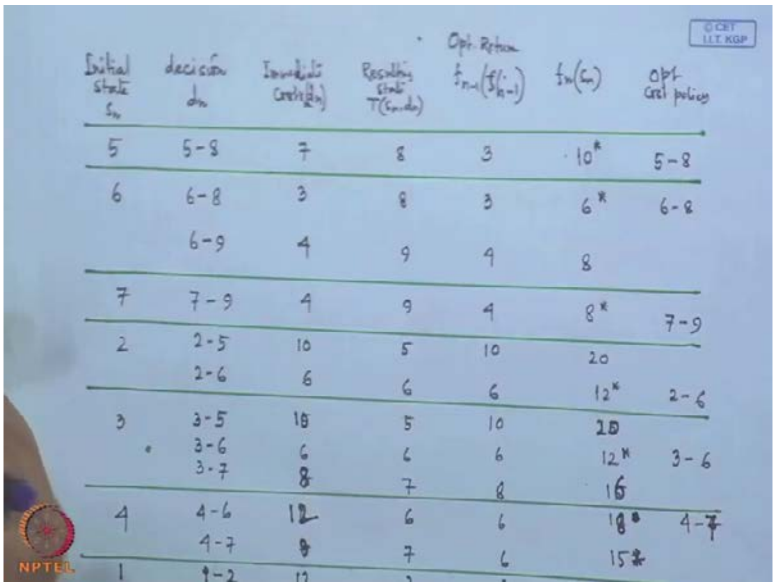
Here I just wanted to mention one thing that for taking decision I have used the transformation equation like that f 1. Let me put S 1 this is 0, this is S 2, f 1 S 1 is equal to this is the minimization problem, we have to we wanted to minimize the objective function f 1 s 1 is equal to minimum of r d 1 plus previous stage that is f 0 10. Rather S 1, S 0 rather all right if this is, so then we have two alternatives with us, one is a d 1 is 3 plus 0, this is 0 always and this is 4 plus 0 that is why we have selected the three value here.

Here let us see, how many alternatives we are having, we are having only 4 alternatives 5 to 8, 6 to 8, 6 to 9, 7 to 9 all right. If we have this alternatives with us, let us formulate the function here as well 5 to 8 means, we have considered once we are reaching 2.8 the value was 3 and at 5, it was 7 that's why it is 7 plus 3 which is equal to 10 units say 10 rupees or 10 units of rupees we have to consider.

If I move to 6 to 8 again I have to come if I just move 6 to 8, the immediate cost I have to incur 3. And what about the previous one that is again coming 3 because, at 0.8 it was 3 that is why we are getting equal to 3 plus 3 6, for 6 to 9 previous stage the optimal value is 3, always it is if I just write down the equation f 2 S 2 is equal to minimization of r d 2. That means, the decision if I consider either this or d 2 can take either this value, this or the 4 alternatives it has f 1 S 1, f 1 S 1 we have already got 3 that is why we have these are the values for us, and this is 4 plus 3 is equal to 7.

Then certainly this is optimal for us, 6 is the optimal at this stage, thus if I just look back the table. Once we are reaching to stage 2 optimal solution we got that is 6, we did not once we are moving to stage 2 to stage 3, we need to consider that how we have reached to stage 2. Where the optimality occurs it is not important for us, only important thing is that how much optimality we have achieved here that is why if I write down the whole detail in the next.

(Refer Slide Time: 57:55)



Initial State s_n	decision d_n	Immediate Cost(d_n)	Resulting State $T(s_n, d_n)$	Opt. Return $f_{n-1}(S_{n-1})$	$f_n(s_n)$	Opt Cost policy
5	5-8	7	8	3	10*	5-8
6	6-8	3	8	3	6*	6-8
	6-9	4	9	4	8	
7	7-9	4	9	4	8*	7-9
2	2-5	10	5	10	20	
	2-6	6	6	6	12*	2-6
3	3-5	10	5	10	20	
	3-6	6	6	6	12*	3-6
	3-7	8	7	8	16	
4	4-6	12	6	6	18*	4-7
	4-7	8	7	6	15*	
1	1-2	12	[illegible]	[illegible]	[illegible]	

Then I can write down in a table in this manner, just let me reads this table properly, I have written the alternatives 5, 6, 7 that is for stage two here. These are the headings for it, initial state the decisions immediate cost we have to incur for that decision, and what is the resulting state. What was the optimal return in the previous stage, and the functional value accordingly we will take the decision that is the thing at 0.5 5 8 whatever I said same thing I have written here.

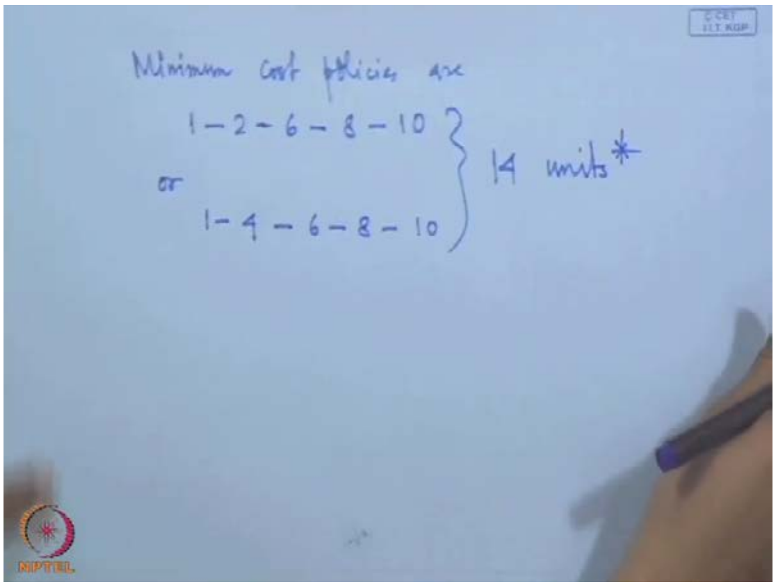
And if I consider the optimization problem up to this point, then this could be the individual optimal solution for each state values. Let me move to the next, where the another three 2, 3, 4 that is stage 3 is there 2, 3, 4, if I just write down the same kind of information here. We could see that from two there are only two alternatives 2 to 5, 2 to 6 this is the immediate cause, this is the resulting stage, this is the optimal solution for the previous stage.

Because, just now we have considered for the 5 it is 10 that is why we are writing 10 and therefore, the total cost 10 plus 10 20, 3 to 6. Once we are reaching to 6 how much I have to immediate cost I have to incur that is 6 and for 6 this is the 6 the optimal solution we have written 6, how we have reached to 6 that is not important for us, as I said again and again and this 2 to 7. For 7 the optimal is 8 in that way we have written 8 plus 8 is equal to 6 10. And among these 3 12 is the minimum 1 that is why this is the optimal solution at this stage.

Let me move to the next, and if I juts reach to 0.4 then this is again the same thing we have written the at 0.4 the optimal value is 15. But, if I consider the whole together 2, 3, 4 that is stage 3 together, then the optimal value is 12 either coming from this point or this point any one of these two all right. Let me move to the next that is stage 4, only one that is the whole problem, there are three alternatives 1 to 2, 1 to 3 and 1 to 4, for 1 to 2 we are incurring 14 units.

Similarly for 1 to 4 also we will incur the same 14 units that is we can say the optimal solution is for this problem the optimal solution, rather the minimum cost policy we can write down this way.

(Refer Slide Time: 1:00:57)



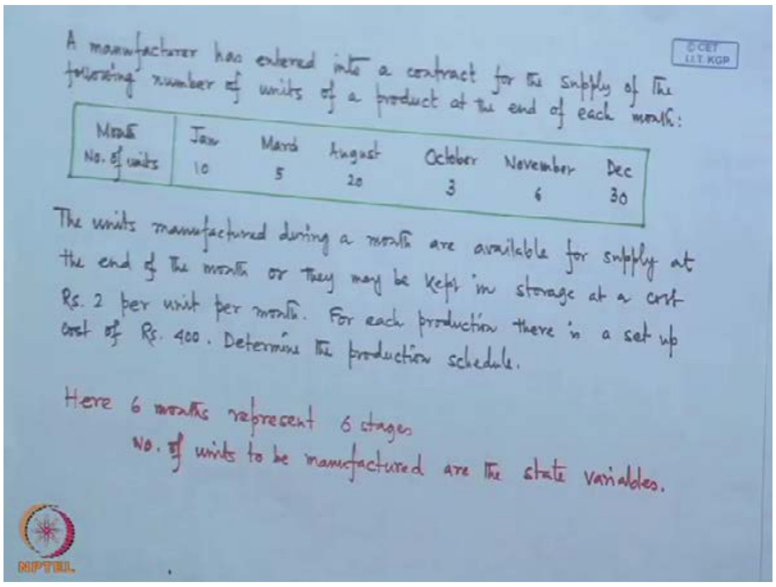
Minimum cost policies are

$$\left.\begin{array}{l} 1-2-6-8-10 \\ \text{or} \\ 1-4-6-8-10 \end{array}\right\} 14 \text{ units}^*$$

Either we have done this is one alternative with 14 units, we have to just bear the cost and this is coming 1 to 2 14, if we go back then it was 12 and this is the 12 is the this is 2 this is 12 this is 2 12. Now, this 12 we are getting from 2 if I am go to 2 this 12 we are getting here; that means, go to 6, for 6 we are getting 6 to 8, in this way we can get the whole path. And this is the path for us 1 to 6 to 8 to 10 there is another alternative path is also there, we can move 1 to 4 to 6 to 8 to 10 for both the cases we need to 14 units.

And that is the optimal solution as a whole for this problem, now that is all about the dynamic programming problem, now I can give you another problem for practice.

(Refer Slide Time: 1:02:18)



A manufacturer has entered into a contract for the supply of the following number of units of a product at the end of each month:

Month	Jan	March	August	October	November	Dec
No. of units	10	5	20	3	6	30

The units manufactured during a month are available for supply at the end of the month or they may be kept in storage at a cost Rs. 2 per unit per month. For each production there is a set up cost of Rs. 400. Determine the production schedule.

Here 6 months represent 6 stages
No. of units to be manufactured are the state variables.

This is a nice problem we can work on it, and there is a hint also how to consider the state variable and state variable. We have considered 6 stages, and there are few state variables we have considered, and this is the problem we can work it and you can get the optimal solution in the similar manner. And this is the multi stage decision process we have considered the decisions space as discrete one, that is why there is a need to discuss further.

How to consider the uncertainty in the whole decisions process this is one part we should consider in the multi stage thing, the multistage decision process that is dynamic programming. And the another topic we have to address that is the dynamic programming problem, where the state variables will move in the continues space that is all for today.

Thank you very much.