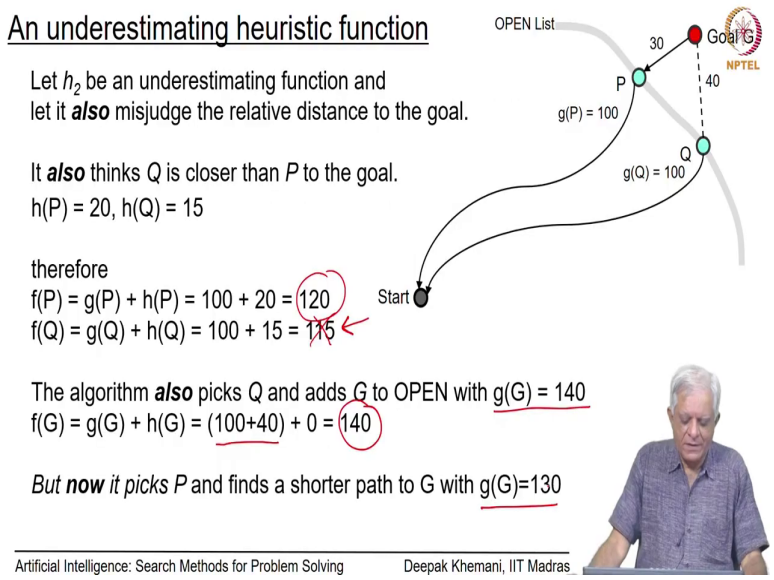
Artificial Intelligence: Search Methods for Problem Solving
Prof. Deepak Khemani
Department of Computer Science and Engineering
Indian Institute of Technology, Madras

Lecture – 39
Chapter - 05
Finding Optimal Paths: Admissibility of A Star

(Refer Slide Time: 00:13)

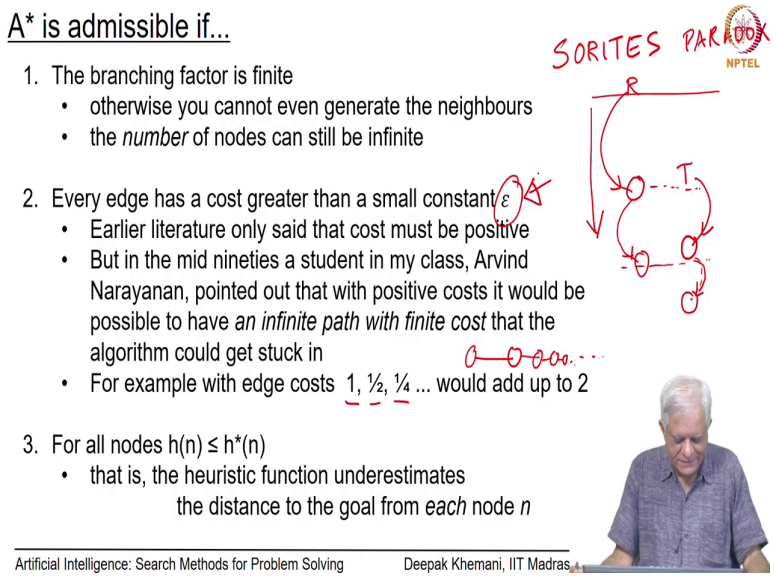


So, welcome back we have been exploring the Algorithm A star and we are looking at the conditions under which A star will always return an optimal paths to the goal node. We did little bit of a thought experiment with this slide in the last session which ended in the conclusion that it pays to underestimate the distance with the goal.

That means h of n should always be less than the actual cost of going to that node in this example the actual cost were 30 and 40 and the heuristic values were 20 and 15 respectively.

So, even though there were misjudgment they still ended up helping us find the optimal path, but these are not the only conditions that you need for a formal proof these are good for an intuitive understanding.

(Refer Slide Time: 01:17)



Let us now completely state what it takes to make sure that the algorithm A star is admissible. The first condition is that the branching factor is in finite essentially. So, this is something which we can generally not handle that if a node has infinite neighbours, then most algorithm that we are studying would not even complete generating the neighbour.

So, there is no question of the algorithm terminating or anything like that except for the case of simulated annealing where we do not generate all the neighbours we just pick a random neighbour. So, in those kind of situations some algorithm will work, but in general branching factor has to be finite because otherwise you cannot even generate the neighbours.

So, our condition is going to be on the branching factor. The branching factor needs to be finite, but we are not saying the graph should be finite the graph can still be infinite and that could be possible if the number of nodes in the graph are infinite provided that the branching factor is finite. So, that is a first condition, then the branching factor should be finite.

The second condition is that every edge in the graph has a cost greater than a small constant epsilon essentially. Now, in the literature that came earlier and some books might still say that they say that the cost must be positive they do not put this limit of a small non - zero constant lower bound which is epsilon.

They simply says the cost is positive, the idea of having a positive cost is that not to have a negative cost or 0 cost for example, if you had negative cost edges then one could simply go in cycles round and round those edges and keep decreasing your actual cost.

Now, here that is not a realistic situation. So, it makes sense to say that the cost are positive and that is what I used to teach till in the mid nineties when I was teaching this to a class at IIT Madras one of the bright students in the class, Arvind Narayanan who was little bit mathematically inclined pointed out that with positive costs it would be possible to have an infinite path which has a total finite cost and then the algorithm could possibly get stuck into that infinite path essentially.

What kind of an infinite path, it could be and he gave the example that for example, if there is a path in which the edge cost are they start with the value 1, then the next one is half, the third one is one - forth. So, we can imagine that the first edge is from this one, the second is smaller, the third is even smaller, the fourth is even smaller and so on.

But the path is infinite and you can see that if you sum up this series 1 plus half plus one - forth it would add up to 2 which means that if there were such a path in your graph. Now, I think this is not a realistic this thing, but if you want to do a mathematical proof you have to cater that into account.

You then that path would add up to 2 it is an infinite number of hops, but the total cost is 2. So, the algorithm might not even come out of such a path because the other path may be more expensive than 2, to counter this fact we decided that that the every edge cost must be greater than some small constant which we will call epsilon no matter how small that constant is.

It can never become close to 0 it will be always a small constant which means that the path cost can never tend to a constant it will always keep growing as the path becomes longer and longer. This is kind of related to an ancient paradox from the Greeks called the Sorites paradox and I hope you will go and look this up sometime.

This paradox says that once a tortoise challenged a hare or a rabbit for a race and said that if you give me a lead or a handicap, then I will always win the race that you can never catch up with me and the paradox goes as follows. Then, let us say this is a starting point and the rabbit is at the starting point.

And as agreed between them the tortoise has been given a lead, the race is going in this direction and the tortoise has a little bit of lead over the tortoise. The argument that the tortoise uses is that by the time the rabbit comes to where the tortoise is the tortoise would have moved a little bit ahead. The rabbit would need some finite amount of time to cover come to the place where the tortoise is and in that finite amount of time the tortoise would have moved up ahead.

So, now the tortoise is ahead of the rabbit. So, again the rabbit has to now come to this place where the tortoise is and again in that time period the tortoise would have moved up ahead. So, you can see that this argument will continue indefinitely that the tortoise will say that by the time you come to where I was I would have moved ahead a little bit hm.

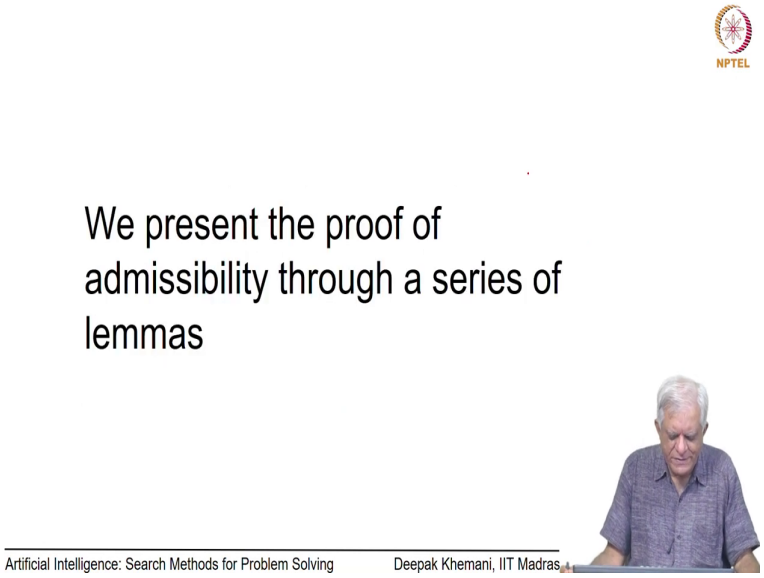
So, maybe I should think a little bit of this and try to see where is the paradox because in real life you know the tortoises cannot win races unless of course, the rabbit follows asleep and is over confident and falls asleep on the way, some interesting is happen it is a great analogy to this problem of having an infinite paths.

So, this are infinite steps that the rabbit is talking about that it will take you infinite steps to catch up with me. And likewise here we are saying that the path would have infinite edges, but the total edge cost would be constant. So, to eliminate that we have say that the edge cost must be greater than some value epsilon so that was a second condition.

The third condition is that as we have observed now that for all nodes h of n should be less than or equal to h star of n which means that heuristic function always underestimates the distance to the goal from each node n.

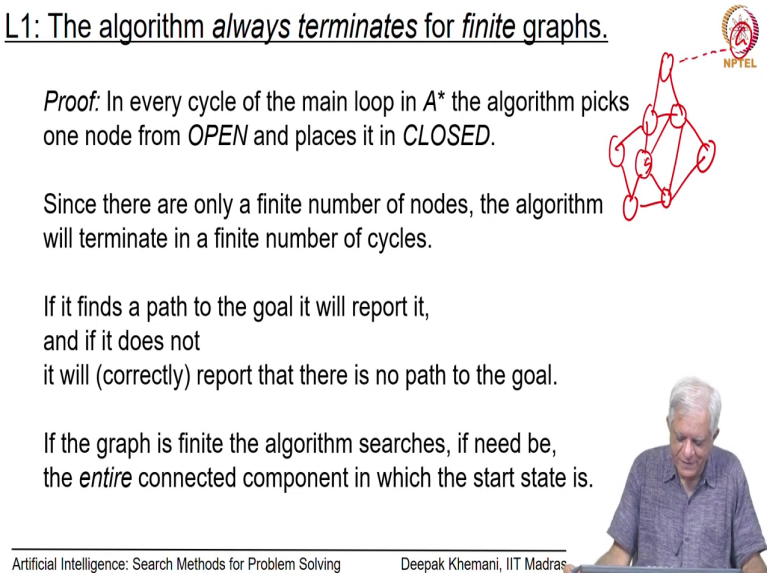
What will we do now is a formal proof of the admissibility of A star given these three conditions. So, what are the three conditions the branching factor is finite every edge cost is greater than some small value epsilon and the heuristic function is an underestimating function.

(Refer Slide Time: 08:28)



So, we will do this proof as was done originally by (Refer Time: 08:33) through a series of lemmas essentially, lemmas are as you know small theorems or small results as some people would say.

(Refer Slide Time: 08:47)



Lemma 1 says that the algorithm always terminates for finite graphs. Now, this is not just true for A star algorithm it is true for any search algorithm that, if you have a systematic or what we said was complete search algorithm it will eventually search through the entire graph and it will terminate essentially and that is what we are saying for a star. The proof is similar is that in every cycle of the loop, remember that in every cycle of the loop the algorithm picks one node from OPEN and adds it to CLOSED.

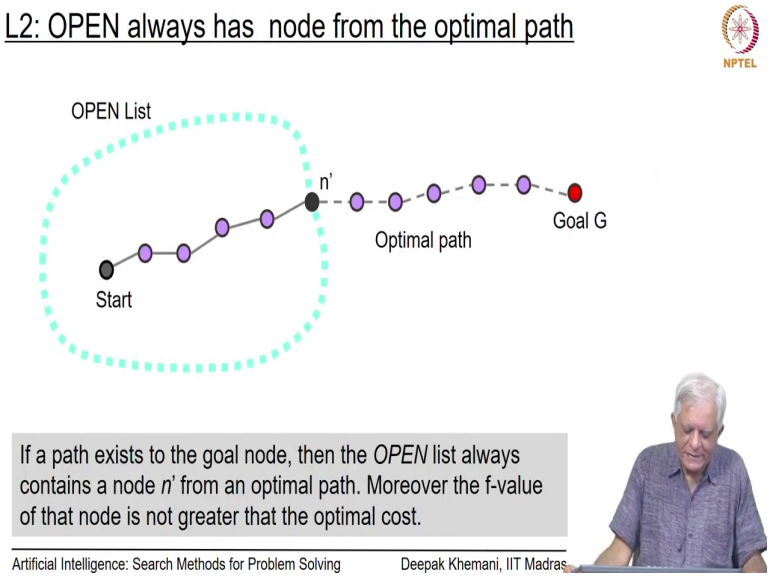
And like dash size algorithm it only keeps one copy of every node. So, it cannot have duplicate copies of nodes. So, from every cycle in every cycle it will pick one node from open and add it to closed and since there are only a finite number of nodes in the graph the graph is finite we have said the algorithm will terminate in a finite number of cycles.

There are two things which could happen if it finds a path to the goal it will report it and if it does not then it will correctly report that there is no path to the goal essentially. If the graph is finite the algorithm searches if need be the entire connected component which in which the start state is.

So, the situation is like this that suppose like this is your start state and supposing this is your goal state and we have a small graph which is finite and let us say this is the graph this is a very small graph our algorithm will search through all these nodes which are connected to the start node S and in the end it will end up with an empty opened list and it say I cannot find a path to the goal which is true for this particular example that I have drawn.

If there was on the other hand an edge going from one of these nodes to the goal node then the algorithm would have generated the goal node as we have been seeing in the examples and find a path through the goal node. So, in either case if there is a path through the goal node it will find it and report the path, if there is no path it will report that there is no path and the answer is going to be correct in both cases.

(Refer Slide Time: 11:03)



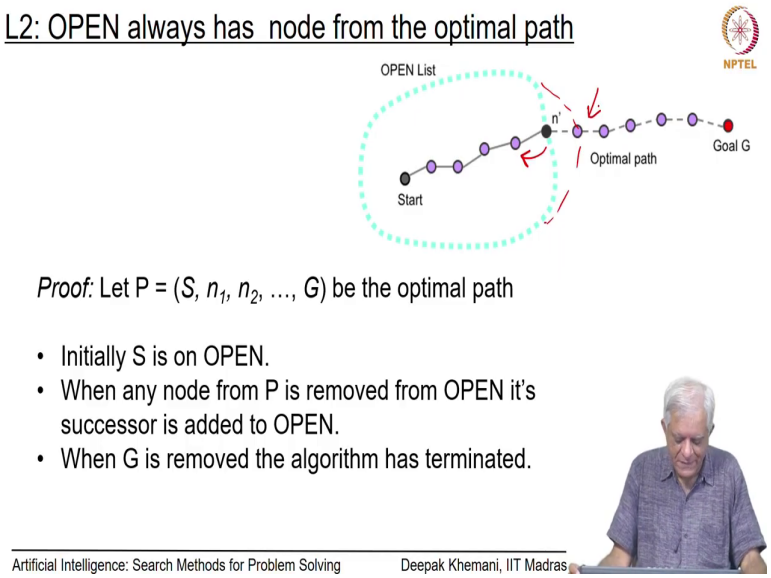
The second lemma and this is an important lemma and much of the proof based upon this lemma says that let there be an optimal path which is shown in this diagram using this nodes on purple this lemma says that if a path exists to the goal node.

And in this diagram we have drawn such a path, then the OPEN list which is been shown here in this sign open list always contains a path always contains a node n' prime which is from that optimal path. So, the optimal path will always have one node on the open list that is the first thing that we are stating in this lemma.

The second thing we are stating is that the f - value of this node this node we will call as n prime is not going to be greater than the optimal cost of the path from the start to the goal

node. So, let us prove this to two small results here first let us focus on the fact that there exist such a node essentially.

(Refer Slide Time: 12:23)



So, this is a situation you have the OPEN list and you have an optimal path and we want to state that there is always a node from the optimal path on OPEN list. So, let the path be called S followed by n 1 followed by n 2 and followed by n 3 and so on. And let G be the last node in this optimal path essentially.

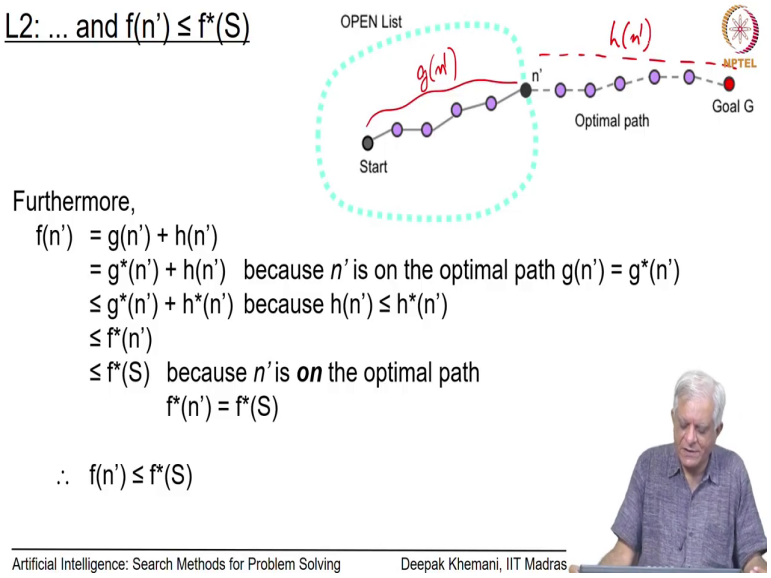
Now, initially S is added on to OPEN because that is how the algorithm begins by looking at the start node which means that we have a node from the optimal path to on OPEN and that node is S and initially. In fact, OPEN contains only S and nothing else.

At any point from this path P if you remove a node from OPEN when will we remove it when it is picked for inspection you will check whether it is a goal node or not, if it is not the goal node you will put it into closed. So, in this diagram it would mean you would put it here or it would move into CLOSED and OPEN list would expand to the next node and the next node would now come on to OPEN.

So, whenever you remove a path node n prime from on the optimal path from OPEN you will add it successor to OPEN and finally, if you are removing G from open it means you have terminated by finding a path to the goal.

So, this is saying that at all points some node from an optimal path is always in contention that you cannot say that this node was not there in open somehow and that the therefore, the algorithm can go of in some wrong direction and not find a path through the goal. This shows that a path will always be in contention because some node n prime on that path will always be on OPEN.

(Refer Slide Time: 14:34)



Now, for the second part of this the second part says that the f - value of this node n prime is going to be less than the optimal cost. So, we will use f star S, S a notation for saying that this is the optimal cost because what does f star S says it says that it is a optimal cost of a path which goes from start to goal passing via the nodes start. So, of course, it begins with the node starts so, it holds true for them.

So, f star of S is the optimal cost and we will use this this term to denote the optimal cost and what we are saying here is that the cost of this node n prime the estimated cost of this node n prime is always less than the optimal cost. Remember that f of n prime means the path the cost up to this thing which is g of n prime plus the estimated cost to the goal which is h of n prime.

We are saying that f of n prime which is the sum of these two is always less than the optimal cost. So, let us prove this very quickly we know that f of n prime is g of n prime plus h of n prime that is by definition .

Now, we can say that this is equal to g star of n prime plus h of n prime and this is because n prime is on the optimal path and therefore, g of n prime will be equal to g star of n prime, this is because it is on the optimal path. So, the cost will be optimal essentially.

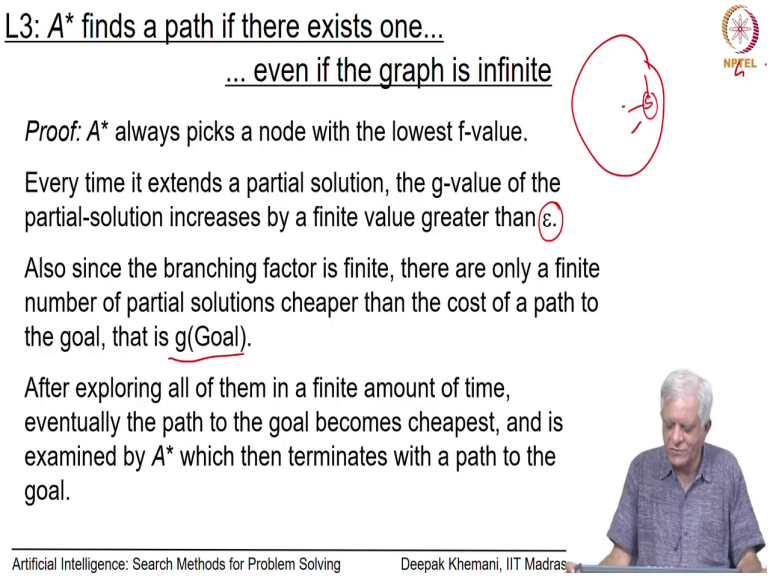
Now, we can say that this f of n prime is less than or equal to g star of n prime plus h star of n prime and that is because of one of the conditions that we have imposed upon the heuristic function is that it must always be under estimating, which means that h of n prime is less than or equal to h star of n prime.

So, we can replace h of n prime by h star of n prime and replace the equality sign by the inequality sign and now we can say that f of n prime is less than equal to g star of n prime plus h star of n prime, which means that f of n prime is less than or equal to f star of n prime by definition of f star of n prime.

But because again n prime is on the optimal path this cost f star n prime must be f star of S which I again repeat because it is on the optimal path therefore, it cost must be the optimal cost which is f star of S.

So, we can say that there always exist a node n prime which is on the optimal path which is on OPEN and whose f - value is less than the optimal cost that is very critical to our proof.

(Refer Slide Time: 17:40)



The third statement we want to make is that A star will always find a path if there exists a path to the goal node even if the graph is infinite and this is where the second condition will come into play the second condition remember said that every edge cost is epsilon where which is a small constant. So, how does the proof go?

A star as we know always picks a node with the lowest f value, what does it do after it picks a node with a picks a node? It checks whether it is a goal or not and if it is not the goal it generates its neighbour and at least for the new neighbours it adds them to open essentially. So, in that sense the graph grows as search progresses.

Now every time it extends a partial solution which is when it adds new neighbours of the node it has picked, the g - value of the partial solution increases by a finite value greater than or equal to epsilon, we have said that epsilon is a lower bound on the edge cost. So, the g -

value of all the new nodes will be the g - value of the old node or the parent plus at least epsilon essentially.

Now, since a branching factor is finite there are only a finite number of partial solutions which will be cheaper than the cost of a path to the goal node that is g of goal some path to the goal node which is a finite value and there can only be a finite number of partial paths.

So, even if you are doing the original branch and bound algorithm which kept multiple copies of nodes and found different ways of reaching those node the number of paths would still be finite, why finite because of branching factor is finite and because edge costs are have a lower bond of epsilon. So, you cannot have you will only have a finite number of partial paths which are of course, less than g of goal this is what we are interested in this g of goal.

There will be only a finite number of such paths because the branching factor is infinite and because only a finite number of paths would be less than g - value because they can we can never have as we discussed earlier, we can never have an infinite path whose cost is less than g of g goal because of the fact that we have imposed this condition of epsilon the number of paths whose total cost is less than this is going to be finite.

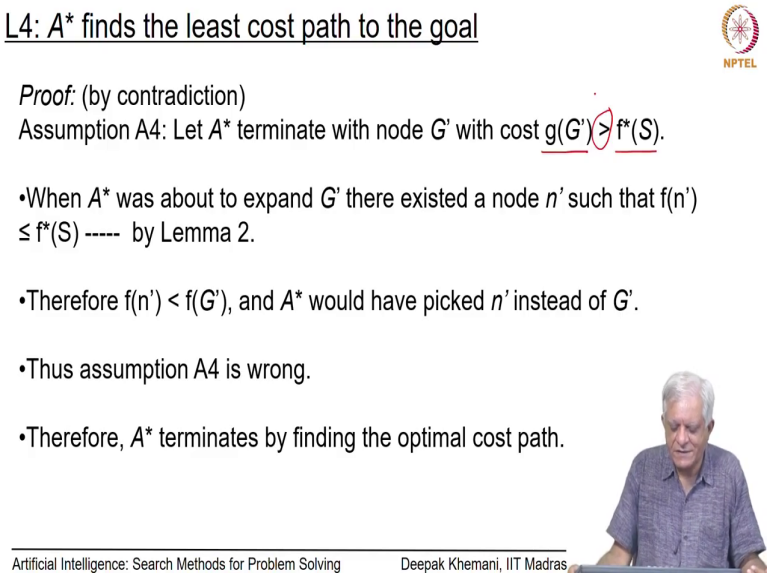
So, because of this condition after exploring all those finite paths even if the heuristic function is bad so, if you remember the example that we gave of branch and bound in last class going off into the wrong direction. So, we had a start node here and then we had many nodes here and there was a whole region that it explored before it moved towards the goal node which would not happen if the heuristic function was bad essentially.

Even if the heuristic function was bad it would explore all paths whose actual cost like this is what branch and bound does are less than the cost to the goal node g. So, at some point the goal node which will appear on open and at some point its g - value will be the smallest and then it would pick that or f - value would be smallest and then it would pick that node g for expansion. So, it will always terminate by finding a path to the goal node.

The next thing we will show lemma number 4 is that it will always terminate with an optimal paths to the goal node. So, so far we have said that the algorithm if there is a path to the goal it does not matter if the space is infinite.

The state space is infinite if there is a path to the goal the path will have a finite cost and because the algorithm always picks node with the lowest f values at some point this finite cost will become the lowest value and algorithm will pick a paths to the goal.

(Refer Slide Time: 21:59)



Now, let us talk about optimality essentially, L 4 states that A star finds the least cost path to the goal essentially hm. So, let us look at a proof for this and we will do a proof by contradiction let us make an assumption let us call it A 4, let A star let A star terminate by

picking a goal node we will call it G prime such that the cost of G prime is greater than f star of S.

Remember f star of S is the optimal path to the goal and now it is terminating with the value g of G prime and this value is strictly greater than f star of S. We want to show that this cannot happen and if you can show that this cannot happen, then by contradiction we would have shown that if it terminates it can only terminate with a cost which is not greater than the optimal cost.

Obviously it can also cannot terminate with the cost which is less than the optimal cost because by definition the optimal cost has to be the lowest. So, it can only terminate with a cost which is equal to the optimal cost. So, the proof is straightforward.

Let us say that assumption is that A star is terminating with this goal node G prime which has a larger cost and optimal cost so; that means, whenever A star was about to expand G prime by lemma 2 that we have proved there existed a node n prime such that f of n prime was less than or equal to f star of S f of n prime was less than the value of the optimal cost, this is why lemma 2 as a consequence since g of G prime is greater than f star of S.

We can see that f of n prime would have been less than strictly less than f of G prime and S r would have picked n prime instead of G prime. So, the proof is that assumption A 4 is wrong, A star could not have picked a path to the goal which is more than optimal because as we have shown in lemma 2 there would have existed a path on an optimal they would have existed a node on the optimal path whose cost is less than the optimal cost.

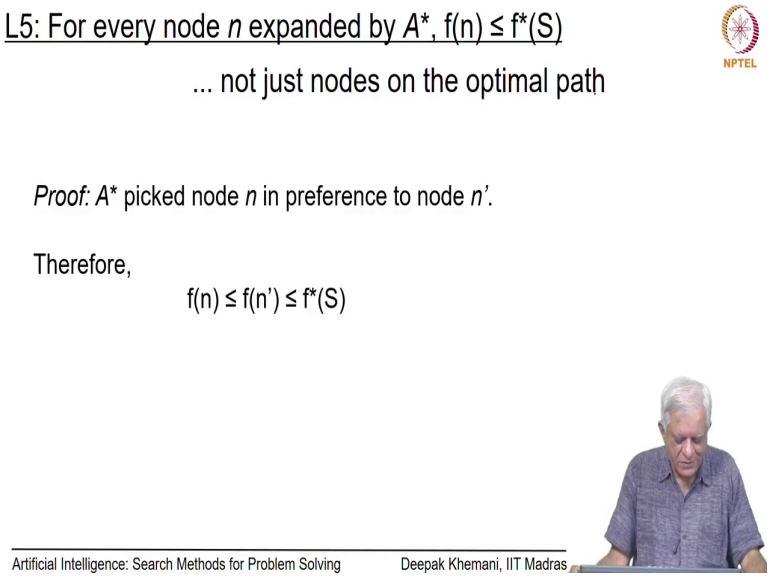
And S r would have picked f of n prime instead of G prime simply because it always picks the node with the lowest f values and we are talking of f values here we are not talking of f star values here.

So, it will always pick it will never pick a path to the goal if that path is longer than or more expensive than the optimal path and therefore, A star terminates by finding the optimal cost path.

So, this kind of completes the basic proof that we wanted to talk about which showed that A star will always find an optimal path to the goal if such a path exist, even if the graph were to be infinite provided the three conditions that we have stated that the branching factor is finite .

The h cost is greater than some small constant epsilon and the heuristic function under estimates the distance to the goal, under these three conditions A star is admissible it will always find a path to the goal. But there are certain other properties if you would want to discuss. So, let us prove a couple of more things before we start looking at variations of the algorithm.

(Refer Slide Time: 25:41)



The next lemma that we want to prove is that for every node that A star picks or every node that is expanded by A star its value f of n is going to be less than the value of the optimal cost path.

So, it is not just for the nodes which are on optimal path, but A star only picks those nodes whose estimated cost of the final solution, remember that f of n stands for path from start to goal passing through the node n. So, it will only pick such nodes such that their estimated cost through those nodes is less than the optimal cost.

In other words it gives us an inside that it will not go off in the wrong direction to too much essentially. What is the proof for this? The proof is very simple that A star has picked this node n when it could have jolly well picked the node n prime which we have shown always exist on the optimal path. So, clearly the value of n can act mostly equal to the value of n

prime because we have not stated that if two values are equal what is the criteria for choosing a node.

So, f of n must be less than or equal to f of n prime and in under such conditions f of n can be picked by the algorithm A star, but we have already shown in lemma 2 that f of n prime is less than f star of S which is the optimal cost. So, therefore, the estimated cost f of n for any node that A star picks must be less than the optimal cost.