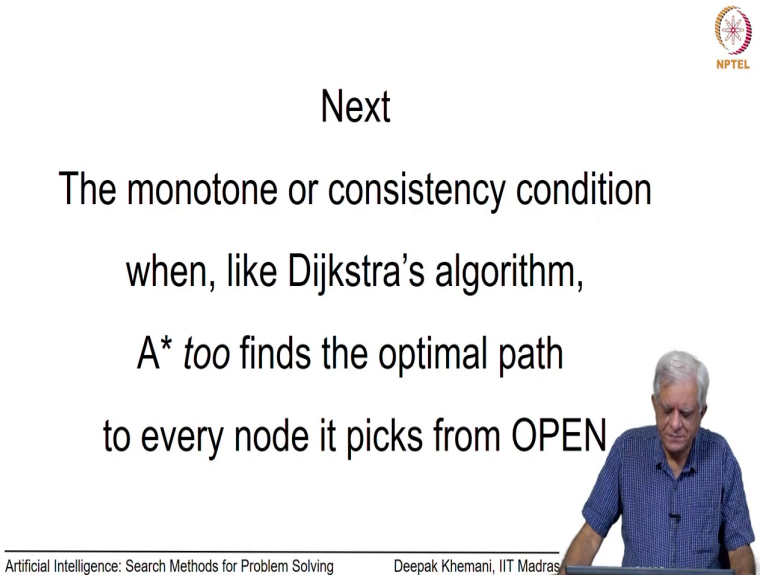
Artificial Intelligence: Search Methods for Problem Solving
Prof. Deepak Khemani
Department of Computer Science and Engineering
Indian Institute of Technology, Madras

Chapter – 05
A First Course in Artificial Intelligence
Lecture – 43
The Monotone Condition

(Refer Slide Time: 00:14)



So we have been looking at the algorithm A star, and we have done quite a detailed study of the algorithm as well as the fact that we can formally show that it is admissible under certain conditions. The conditions were that the branching factor is finite that h cos have a minimum positive value some epsilon which is a lower bound.

And most importantly that the heuristic function underestimates the distance to the goal essentially. So, under those conditions which saw that A star will always find an optimal path to the goal provided a path exists essentially.

We did not state any other conditions on the heuristic function. Our only concern was that will it find the path on optimal path to the goal or not. And for that the condition was that it underestimates the distance to the goal at every stage, so that particular node will always be in contention list.

Now, let us look at some other conditions. So, for example, what about the distance estimated distance from one node to some other node which is not the goal node, how does A star handle that? Or for example, the distance from start node to some node n essentially, we saw that unlike Dijkstra algorithm A star can pick a node from open with a certain f value which is a smallest at that point of time and add it to close essentially.

And it can turn out later that it can add pick up another node from open, and from that other node find a shorter path to the node in close that we have added which means that when it pick the node it did not necessarily find the shortest path to that node from start unlike Dijkstra's algorithm which we argued that it does.

So, and we also argued and that was because Dijkstra's algorithm work with known values which is a known cast. And at the point when it picked up it knew that there was no other possible shorter paths anywhere in the graph from the start node.

They have also observed that the space requirements of A star are more than that of best first search. And we have observed that best first search itself can often be exponential in nature and that is because the heuristic function is not always such a good function that it can guide you straight to the goal node. And A star being more conservative than best first it explored more nodes than best first, and therefore, A star also has huge space requirements.

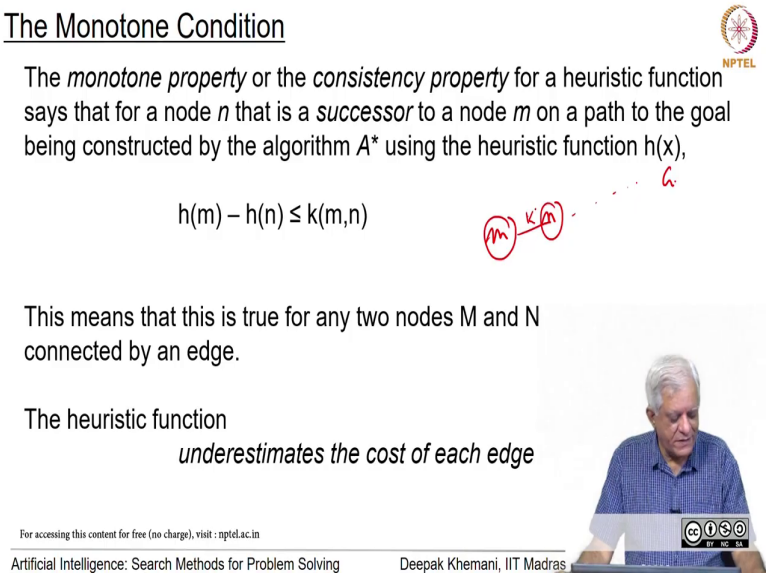
So, what we will do now is to first look at a condition on the heuristic function which will make A star behave more like Dijkstra's algorithm in the sense that if it has pick the node n from open, then at that point it has found the shortest path. And then we will look at domain which has come up in recent times which will motivate us to see if we can find admissible versions of A star which require less space essentially.

We saw already that we can use the blue A star, but the blue A star because it multiplies a heuristic estimate by a certain factor there by it makes it less than the optimal distance and there by the algorithm as we saw with an example then find paths with a longer than the optimal path.

Then we saw IDA star and a RBFS both of them require less space because both of them in some sense only maintain linears number of nodes in proportion to depth. But both of them have to repeatedly come back and try newer paths, and therefore, the time complexity was much larger.

And eventually now we will see that there would be algorithms which are closer in terms of time complexity to A star but much more efficient in terms of space. So, let us begin our study with this condition on the heuristic function which is called the monotone condition or the consistency condition although monotone property or the consistency property, and see what is the impact it has on the search algorithm which is A star algorithm.

(Refer Slide Time: 05:36)



So, the monotone condition, all the monotone property are also called as the consistency property for a heuristic function says that for a node n that is a successor to another node m on a path to the goal which is being constructed by this algorithm A star remember A star uses both g values and h values and it is using a heuristic function h essentially.

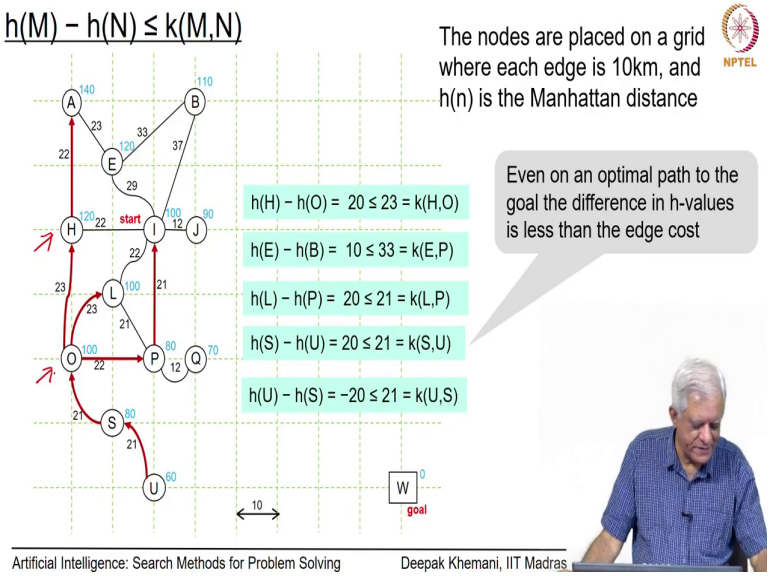
What the monotone property says that if we take the difference between h m minus h n difference between h m and h n, h m is the node which is h n is the second node is the one which is closer to the goal, and obviously, it will be less than the value of h m. So, it would be something like you have m here and then you have n here, and this is on some paths to the goal. So, the goal is somewhere.

And this cost of this edges k m, n. We say that the heuristic function is satisfies the monotone property if the difference between heuristic distance is heuristic estimates is less than the cost

of that edge which is connecting m and n essentially. We will see that this is in fact will be true if it is true it will be true for any two nodes m and n which are connected by an edge.

And in fact what we mean by the monotone property is that the heuristic function underestimates the cost of each edge. So, every edge that it has to traverse it has a estimate that the amount of distance or cost it will cover is less than it actually guess essentially.

(Refer Slide Time: 11:27)



Now, look at this graph that we have been working with for so long now, it is a partial graph. And let us say we do not know what is happening on the right side of the graph, but we know that the goal is somewhere on the bottom right corner and this is a algorithm that we had seen. And what I have shown in this melon colour or red reddish brown colour is that optimal paths which lead to this goal w which is on the bottom right essentially.

So, let us just inspect some edges here, and see what is happening with the h values. Remember that the heuristic function we are using is the Manhattan distance which is basically like a city block distance. It essentially you how many steps do you move right, and how many steps do you move down or up as the case may be and that is the estimate of heuristic distance.

Now, look at these two nodes H and O ok. So, H is here and O is here. And as depicted here the path from H to the goal passes through O essentially now the heuristic value of h is 120 the Manhattan distance; the heuristic value of O is 100. The Manhattan distance the difference is 20. And as we can see 20 is less than 23 which is the H cost from H to O.

Let us took at some more distance we have E and B here which are in no way related to paths that we have been exploring. But if you wanted to path find a pass on E to the goal or B to the goal, we would see that this particular property is true that h E minus h B h E is 120, h B is 110, the difference is 10 which is less than the actual cost of the h which is 33.

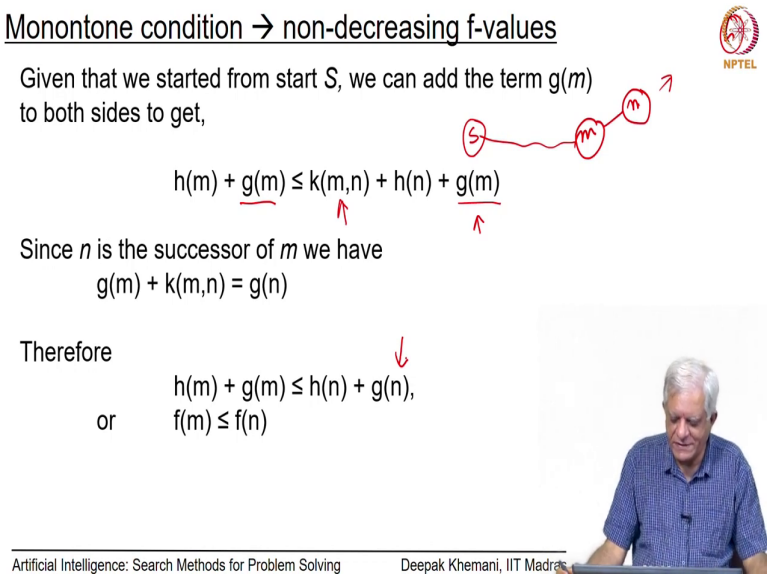
Then look at L and P. As you can see either of these lie on the optimal path to the goal, the paths through L goes through O and the paths through P also goes through O. So, neither path traverses this H. But this property is true here as well the difference 20 is less than actually difference 21.

And one more instance of path or two nodes on the optimal path and this is going to be of our interest that S U the optimal paths from S goes via U, and then it goes on to w as we saw earlier when we looked at A star in branch and bound. The difference is 20 and h cost is 21. So, this property is also satisfied essentially.

Now, if he had considered the opposite that between U and S then the difference would have been minus 20 and that, obviously, would be less than the h cost 21, because it is a negative number. So, obviously, that is a trivial case that we are not very much interested in. Our interest is going to be that as we move towards a goal every edge that we cross are we under estimating that edge or not.

And if we are then we say that the heuristic function satisfies the consistency property, not only does it in underestimates the distance to the goal, it underestimates the cost of every edge on the paths to the goal and our interest is there essentially.

(Refer Slide Time: 10:54)



So, the first thing that we can observe is it if this monotone condition or the monotone property or the consistency condition, or the consistency property is true, then as we move towards a goal we see non decreasing f values.

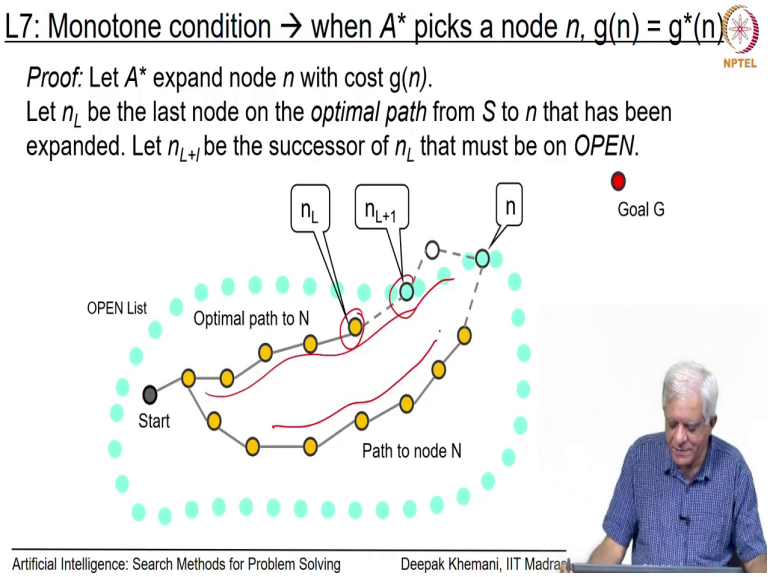
Let us see how that happens. So, given that we started from start node S, we have the values g of m which is a cost of coming from S to m. Remember again that m, m is a node its successor is n. And we have found some path from S to m, and we are going towards the goal somewhere there.

So, since g of m is a cost of coming from S to m it is just a number we can add it to both sides of the inequality that we had, and rearrange the inequality and what we will get is that h m plus g m is equal to is less than or equal to k m, n plus h n, and we have also added g m to both the side essentially. So, first removed h of n to the right, and then we added g of m to both sides, and obviously, this inequality must hold as well.

So, now we know that since n is a successor of m, we know that the cost of coming to n is the cost of coming to m plus h cost than this is something that we use in the A star algorithm to update the g values. And therefore, we can replace g of m n and g of m with the cost g of n. So, this value and this value together gives this value.

Now, obviously, this equation is simply saying that f of m is less than equal to f of n and this is the first observation we make that as we go towards the goal, the f values n to increase or strictly speaking they are non-decreasing essentially they cannot decrease as we go towards the goal.

(Refer Slide Time: 13:09)



Now, the lemma that we want to prove we have already proved 6 lemmas including the fact that A star is admissible, and including the fact that a more informed heuristic function will explore less of the space.

This is a last of the series of those lemmas that we want to produce to prove. And it says that if the monotone condition is true, then when A star picks any node n, it has found the optimal path to that node n, or in other words at the point when A star picks n g of n is equal to g star of n which is the optimum path.

One of the earlier lemmas lemma 5 had said that whenever A star picks a node its f value is less than the optimal cost. Now, we are saying that whenever A star picks a node n, its g value is equal to the optimal cost to that node n. And remember this is what Dijkstra algorithm did is that whenever it pick the node we could have found an optimal costs paths to that node essentially.

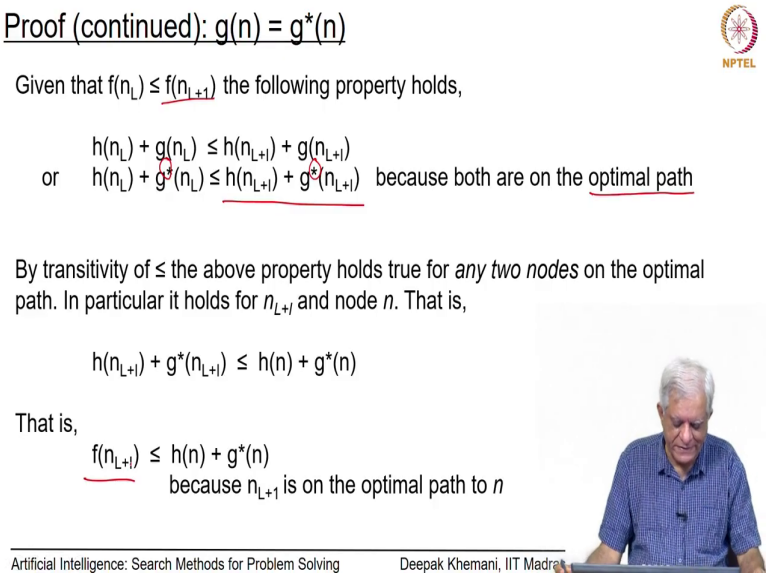
So, let us prove this thing, and we will prove it by negation, prove it by contradiction. And we will assume that let A star expand the node n with a costs g of n, and we have not made any statement about what is the cost, but we do not we are not saying that it is optimal at this stage, some costs g of n.

Now, let the optimal paths to the node n from the start contain a node n of L the last L stands for last node on the optimal paths from S to n that has been expanded by A star. So, there is some node which has been picked by A star. And let it successor be n of L plus 1 last plus 1, and that successor must therefore be on open essentially.

So, the situation looks like. This the cyan circles are the representation of the open list. The node n is the one that A star is just about to expand. The optimal paths to n passes through two nodes n last n last plus 1, n last has been expanded, but n last plus 1 is not been expanded.

So, it must be on open as well because it is a child of a node that was expanded. And somehow n A star is about to pick this node n. So, it is found some other path to n essentially. And we say that the costs of that path is g of n. Now, what you want to show is that g of n must be. So, equal to the optimal path costs to n which means that it should be equal to the other paths, so that we have drawn here.

(Refer Slide Time: 16:08)



We want to show the g of n is equal to g star of n. We have already observed that an optimal path or on a paths to the goal successive nodes have non-decreasing f values. So, it is true for f of for n the last node n L and the last plus 1 node n L plus 1 remember n L is in closed and n L plus 1 is on open this property must be true. And when we expand the we replace f n by its definition h plus g, we get the second inequality.

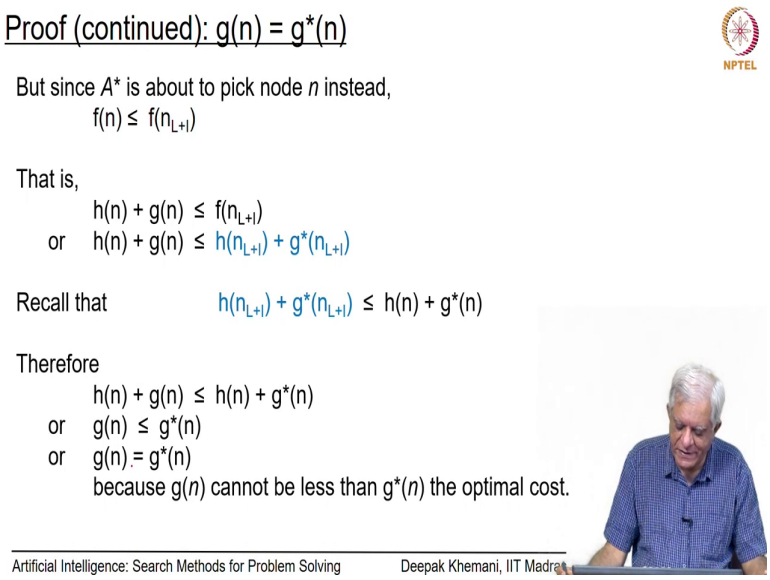
And the third inequality we get this g star here in both the places, and the reason why we can add this g star is that because they are both on the optimal path. We are saying that let the optimal paths through the node n pass through this n L and n L plus 1. And therefore, this around the optimal path and therefore, their g values must be optimal. So, this property holds.

Now, by transitivity of this less than or equal to relation, the our it will hold for L plus 2 n n of L plus 2 n of L plus 3 and so on and eventually we know that n lies on that path. So,

eventually it must also hold for the node n. So, we can also say that the relation between the L plus 1 node which is on open and the node n which is also on open is that the h plus g star values is lower for lower or equal for L plus 1 node and then as compared to the node n essentially.

So, we will use this to argue that any other path to n also must respect this inequality. And remember that we started with this value f of n plus 1, we expanded it into this. And now we are just again repeating that is the f value for the node which is A star is not picking essentially.

(Refer Slide Time: 18:20)



But because A star is not picking that node it must be true that the node that it is picking which is n must have that less than or equal value of f value essentially. Which we can now again rewrite by saying that h of n plus g of n is less than equal to the f value for the last plus

1 node. And we already know that the f value for the last plus 1 node, because it is on the optimal path is the h value plus the g star value for that node essentially.

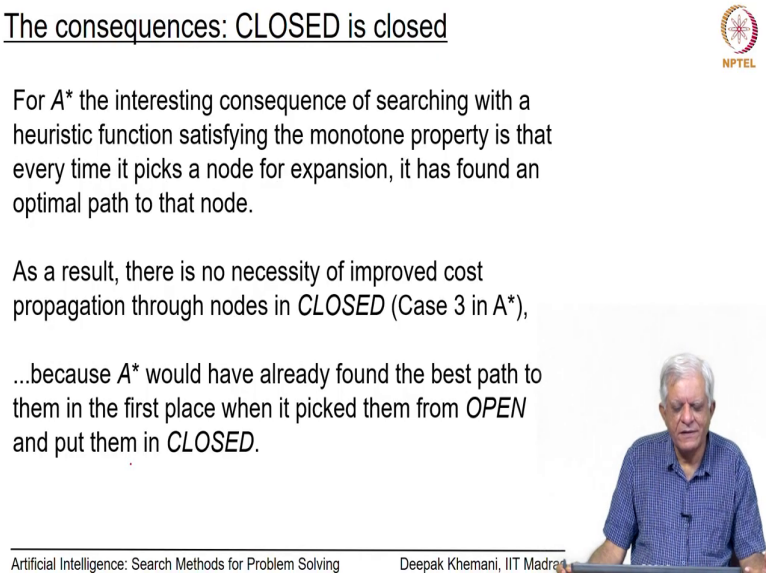
But remember that we had also observed that this expression the h value plus the g star value for last plus 1 node is less than or equal to the h value plus g star value for the nth node essentially. So, we can again now apply transitivity here.

And say that h of n plus g of n must be less than equal to why h of n plus g of n because f of because A star is about to pick that node n which the value f of n which is h of n plus g of n and which we have now shown must be less than or equal to h of n plus g star of n.

Now, h of n is common to both and we can remove it from there. So, we get g star, g star of n is greater than g of n or g of n is less than or equal to g star of n. But g of g of n can never be less than g star of n because by definition g star of n is a optimal cost up to node n.

So, we must be forced to conclude the g of n is g star of n which means that at the point when A star is pick this node n, its g value g of n is actually the optimal value which is g star of n which is what we wanted to show which is what Dijkstra algorithm does essentially.

(Refer Slide Time: 20:15)



So, what are the consequences of this property? The consequences that when you have found when you have added a node to closed, you do not have to think a with it any further essentially, you have already found the optimal paths to that node and you can let it be there. So, then interesting consequence for A star is that if the monotone property is satisfied, then every time it picks a node for expansion it has found an optimal path to that node. And remember that when it picks a node for expansion it added to closed.

And as a result, there is no need to propagate any improve costs to those nodes which are in closed which was a case for in the algorithm A star case 3 if you remember that when A star expanded a node there were three kinds of neighbours, one was new nodes which was case 1.

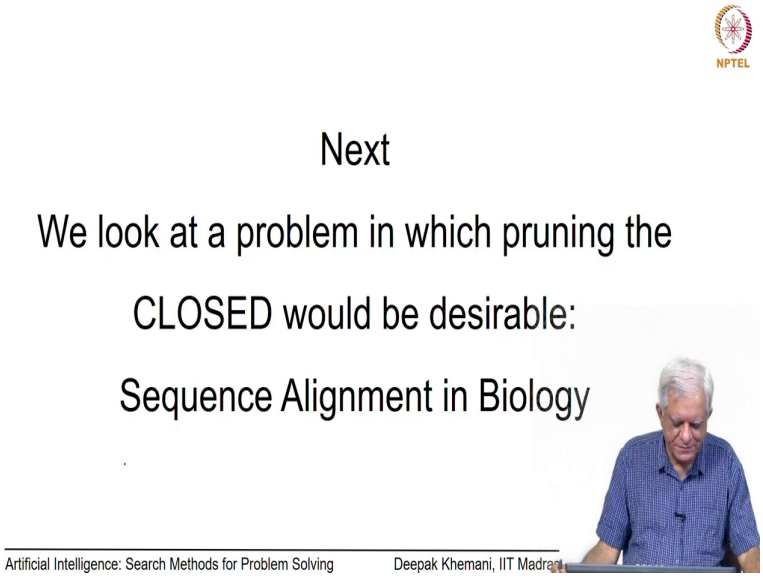
Other was nodes already an open which was case 2, and the third was nodes which are in closed which was case 3. And we said that we may need to propagate the improve cost to

those nodes, and further from children of those nodes because they have in close and they must have had neighbours which were generated.

But now we do not have to do that and that is because A star would have already founded the best path in the first place when it pick node from open and put them in closed essentially. We will use this property a little bit later to show that we can devise a algorithms which do not even need to keep the closed basically. Remember that one of the reasons why we needed to keep close was to make sure that we do not go into infinite loops.

So, we could check that every time we added nodes to open we would first check whether it is already in closed or not essentially. If we can change the algorithm, so that we do not have to use closed to this non leaking back of the search, then we do not really need to keep close at least for that purpose, but of course, close had has another other uses as well. And we will come back to that when we look at those space saving algorithms.

(Refer Slide Time: 22:34)



What we will do next is to look at a problem where pruning of closed would be a desirable feature. And this problem has come from modern times from biology. Biology has become more and more involved with computational techniques, we have bioinformatics, we have computational biology, all kinds of biology work is being done by computer scientists nowadays. And so computer scientists are taking up biology, and biologists are taking up computer science, and its quite an exciting field of research.

We will look at this problem of sequence alignment in biology. And we will see why does it make it desirable that if you can have an algorithm we does not keep closed which can solve larger problems. In general that would be the case, but for sequence and alignment problems it is particularly. So, we will do that in the next session.