

**Wheeled Mobile Robots**  
**Prof. Asokan Thondiyath**  
**Department of Engineering design**  
**Indian Institute of Technology, Madras**


**Graph Search Methods**  
**Lecture - 6.3**  
**Graph Search Methods and Algorithms**

Hello everyone, welcome back. So, in the last few classes we talked about the path planning methods and we saw that there are two stages one is basically you create a graph within the given environment and then you go for the graph search method. So, today we look at the Graph Search Methods and Algorithms because last class we discussed about few methods of creating graph.

Now, the question is that once you have multiple paths available how do you search for the best path or the most optimal path. So, that is basically the graph search method. So, there are few methods and then some existing algorithms. So, we will go through some of those commonly used algorithms, but there are many more such algorithms existing in the literature.

(Refer Slide Time: 01:08)

Deterministic Graph Search

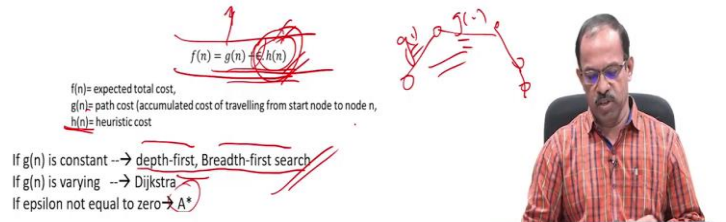


The environment map has been converted into a connectivity graph using one of the graph generation methods presented earlier. Whatever map representation is chosen, the goal of path planning is to find the best path in the map's connectivity graph between the start and the goal, where best refers to the selected optimization criteria (e.g., the shortest path).

$f(n) = g(n) + h(n)$

$f(n)$  = expected total cost,  
 $g(n)$  = path cost (accumulated cost of travelling from start node to node  $n$ ),  
 $h(n)$  = heuristic cost

If  $g(n)$  is constant  $\rightarrow$  depth-first, Breadth-first search  
If  $g(n)$  is varying  $\rightarrow$  Dijkstra  
If epsilon not equal to zero  $\rightarrow$  A\*



So, this one I yesterday in the last class I mentioned about this deterministic graph search methods where we use a cost function or a expected total cost and then find out the paths

where the cost will be the minimal. So, that is the graph search method in the deterministic method we go by this.

And we use this function or the total cost function as  $f(n) = g(n) + \epsilon \times h(n)$ , where  $g(n)$  is the path cost or the accumulated cost of travelling from the start node to the end node assuming that the cost between the nodes are known. So,  $g(n)$  is the cost between nodes that is  $g(n)$ .

And then in not only the cost of travelling we can actually add something else also into the cost function and that one we calls as the heuristic cost  $h(n)$ . So, the total cost will be sum of node cost plus a heuristic cost. The this heuristic cost is designed by the programmer or the designer who actually decides the search methods.

So, that can actually be decided based on the strategy we adopt and if there is no heuristic cost then we will put this  $\epsilon = 0$  and if there is an  $\epsilon$  heuristic cost, we will put this  $\epsilon = 1$  and then we will have  $g(n) + \epsilon \times h(n)$  as the cost function.

Now, if  $g(n)$  is constant between the nodes. So, you have travel from start to goal and it passes through many nodes. So, these are the cost of travelling and then each one will be having a; this is  $g_1, g_2$  etcetera will be the cost. Now, we if we assume that these costs are equal, the travel cost between each node is equal then we call this as a particular method.

And there are methods called depth first and breadth-first search for such cases where the travel cost is equal or the node cost is equal, then we have methods called depth first, breadth-first etcetera. And then if  $g(n)$  is not constant if this  $g_1 \neq g_2$ , then we have an algorithm called Dijkstra or a Dijkstra method or Dijkstra algorithm is there which takes care of that kind of a situation.

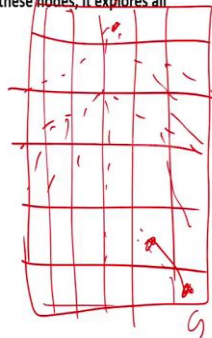
And then if there are cases where actually the when you use a heuristics a heuristic cost, then there are many methods and then one of the method is A star algorithm. So, we will go through these methods and algorithms though there are many such algorithms existing, we do not go through all we just explained these algorithms only ok.

(Refer Slide Time: 04:07)

## Breadth-first search



This graph-search algorithm begins with the start node and explores all of its neighboring nodes. Then, for each of these nodes, it explores all their unexplored neighbors and so on.



So, the breadth-first search is basically a search algorithm where you have a grid. So, you have already created a cell or a grid. So, you divide this into multiple cells and then you identified all these cells. Now, you start from a start node and you want to reach the goal. Suppose, this is the goal and this is the start.

So, the breadth-first approach says that you search along the breadth direction and then finish all those possible options and then go to the next one and then go along this ok. So, that is basically the breadth search. So, you search along the breadth first and then slowly go move towards and then see which nodes finally reaches the which path reaches the goal. So, that is basically known as a breadth-first approach.

So, it start with the start node and explores all of its neighboring nodes and then for each of these nodes it explores all their unexplored neighbors and so on. So, it keeps on exploring its neighbors in the breadth direction and then once that is all the neighbors are explored, it will go to the next level and then explore keep on doing this. This is basically known as the breadth-first search algorithm.

(Refer Slide Time: 05:26)

## Breadth-first search



This graph-search algorithm begins with the start node and explores all of its neighboring nodes. Then, for each of these nodes, it explores all their unexplored neighbors and so on.

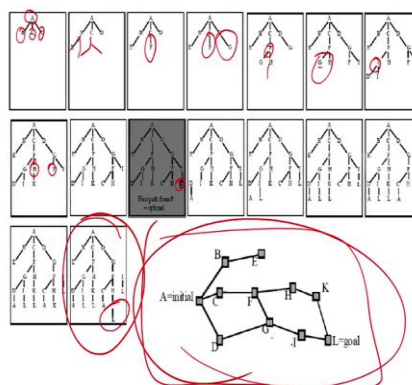
In breadth-first search, nodes are expanded in order of proximity to the start node with proximity defined as the shortest number of edge transitions. The algorithm proceeds until it reaches the goal node where it terminates.



So, they are expanded in the order of proximity to the start nodes. So, those are close to the start node or explores continuously till all the nodes are explored in that neighborhood, then it will go to the next level keep on doing this ok. And the algorithm proceeds until it reaches the goal node where it terminates. So, once it reaches the goal node, it actually terminates and then say that ok, these are all the paths that is possible to reach the goal nodes.

(Refer Slide Time: 06:01)

## Breadth-first search



So, this is the way how it will process progress. So, it start with A, suppose this A is the start node. And then you have a L as the goal node. So, assume that you have a goal node of L, this is the node. So, it is actually divided into multiple grids multiple cells and then

you will be have a one cell as the target and the other one as a start. So, what will do? It will go to the next neighboring cells B, C and D.

So, these are the neighboring one and then this B will expand and then see all its neighbors whatever the neighbors are available and then if it is not possible to expand further then it will go to C and then expand further. Like this it will keep expanding to the next one then D will expand to the next one that is a neighboring node.

And when then this is finished then it will go to the F (Refer Time: 06:52) and again expand it. Keep on expanding like this then G and H; G will expand first then your H will expand then F will expand. So, like this it will go in the breadth direction and then keep expanding till it reaches the first path which reaching the L.

So, when if it reaches the first one it will not stop completely, it will again try to expand other nodes to see whether there are alternate paths to reach L. So, it will keep on expanding to all and then finally, it will be having a path all the paths identified to reach L. So, that is basically the breadth-first approach.

And then it will try to identify a route map like this or a road map will be created where it will see that A B E then A C F H K L A D's G L G I L like this it will create the paths. So, this is basically the breadth-first search algorithm. So, it will keep searching for the paths.

(Refer Slide Time: 07:57)

## Depth-first search

Depth-first search expands each node up to the deepest level of the graph (until the node has no more successors). The search backtracks by expanding the next neighboring node of the start node until its deepest level and so on.

An inconvenience of this algorithm is that it may revisit previously visited nodes.

A significant advantage of depth-first over breadth-first is space complexity. In fact, depth-first needs to store only a single path from the start node to the goal node along with all the remaining unexpanded neighboring nodes for each node on the path.



And then there is another one called depth-first approach. So, the breadth-first which actually goes in the breadth wise direction, next one is actually goes in the depth first approach, where it will go straight to the I mean in the depth direction and then see whether it is able to reach the goal.

If not it will go to the next starting point and the next adjacent side to the start node and then expand that node and keep expanding towards the depth direction and then see whether it is able to reach the L ok. So, the depth first search expands each node up to the deepest level of the graph until the node has no more successors.

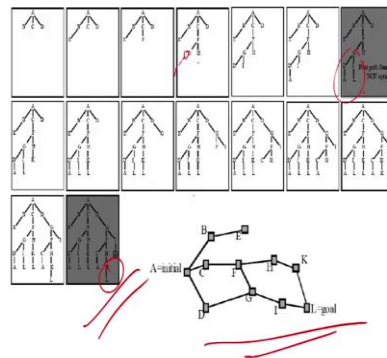
And then it backtracks by expanding the next neighboring node of the start node until its deepest level and so on. So, that is the way how it will expand. So, compared to the breadth-first, depth first will keep on expanding in the same in the depth wise direction or in the deepest go into the deepest level and then go back and then again expand from the beginning. So, that is the way this happens.

So, it may have to revisit previously visited node many times because of the depth-first approach, but the advantage is that its space in complexity. So, you do not need to have store all the information as the search progresses, it will need to only if it reaches the L it will it only if it reaches the goal node it will record otherwise it need not record all the information.

So, it needs to store only a single path from the start node to the goal node along with all the remaining unexpanded neighboring nodes for each node on the path. So, this is the depth first approach.

(Refer Slide Time: 09:39)

## Depth-first search



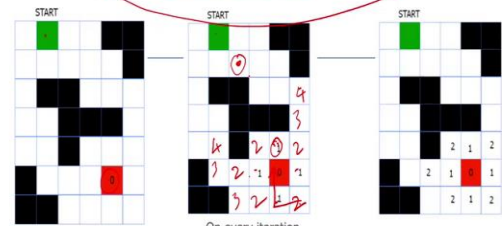
So, it actually explains here. So, you can see that it will keep on expanding towards the depth. So, first we will go along the depth up to E and then there is no more node to expand. So, it will start expanding the C, then F, then H, then H will go to one of this G and then expand it further.

It will reaches no more expansion possible then it will try to expand the other one. Keep on doing this and as you can see here first it will reach one node here. The goal node will be reached, but it will still continue with the other nodes also and keep on expanding till all the nodes are expanded and all the routes are identified.

So, this is the way how the depth first approach works. So, both will give you the paths which actually connects to the goal and that will only the way it is exploring is different. One is the breadth-first and the other one is the depth first. So, this one will give you all the connectivity between the nodes and now you need to calculate the cost of this expand this route. So, that is the next place, where next step where you try to calculate the cost of travel that is the using the cost function ok.

(Refer Slide Time: 11:02)

Grassfire Algorithm (Breadth-first search) // for fixed size cell arrays



Begin by marking the destination node with a distance value of 0

On every iteration find all the unmarked nodes adjacent to marked nodes and mark them with that distance value + 1.

$g(n)$



So, the algorithm one of the algorithm which is very common is known as the Grassfire algorithm which is the breadth-first approach and it is for fixed size cell arrays and assume the cost is equal for travel between each cell the cost is equal. So, the  $g(n)$  only will be there, so,  $g(n)$ . So,  $g(n)$  will be the cost and that will be that is equal between all the cells. So, this is the algorithm which actually as it search for the path, it will actually calculate the cost also.

So, in the breadth-first and depth first as it expand the nodes, it will try to calculate the cost and then store the cost of travel in each to each node. So, the grassfire algorithm, in the breadth-first what actually it does is same thing it will keep on searching along the in the breadth direction and then it will calculate the cost of travel to each nodes by an accumulated cost will be calculated.

So, how is it works? So, you have a start nodes and you have the goal nodes. So, the goal node is marked as a red one and start is the green one. And this algorithm what does it do is to begin by marking the destination node with a distance value of 0 or we can say the cost value of 0. That means, to reach this goal I mean from that particular cell to reach that goal the cost is 0. It means the robot is already in that node. So, there is no more cost involved in reaching the goal.

Then it will try to expand the neighboring nodes to see, how much will be the cost of travel or cost of the cost what will be the cost of travel from the neighboring nodes to this goal node? And then it will give that value as 1. So, you can see from this one, from



this cell to reach here is cost is 1, similarly from here to this is 1, from here to this is 1, from here to this is 1.

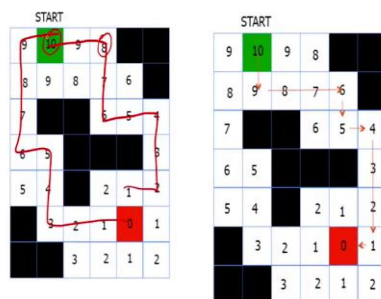
Assuming that it cannot travel the diagonal way, it can only go in the horizontal or I mean the lateral direction only; forward or lateral direction only not in the angular direction. So, we will expand these four nodes and find out what is the cost; so, 1. Now, it will expand each one of these will expand this neighboring nodes.

Now, this node will be expanded. So, it will say this will be 2 because it will be 2 cost of 2 to travel from here to the thing. So, we have to travel like this. So, the cost will be 2. Similarly, this will be 2, this will be 2, this will be 2 and this also will be 2. So, all the neighboring nodes will be expanded with cost values and this will continue.

You will keep on expanding this, then you will do, this as 3 and this as 3 and this as 3 like this because all the nodes of 2 will be expanded then the all the nodes of 3 will be expanded this will be 4, it keep on expanding like this and then till it reaches this point it will keep expanding the nodes in the breadth direction.

And as that expands it will know how much will be the cost of travel from this node to goal. Similarly, if you want to know the what is the cost of travel from this point to this goal. As you expand you will get find a value here and that will be the cost of travel. So, this is basically known as the grassfire algorithm.

(Refer Slide Time: 14:40)



So, you can see it keeps expanding like this and then it keeps going expanding like this and finally, all the nodes will be explored and you will see the cost of travel from those nodes to the goal. So, you can see that if you are in this if this is the start point the cost of travel will be 8. So, any path you can take where the total cost is 8 will be the a feasible path.

I mean in the start point now this is 10; that means, the minimum cost to travel from start to goal will be 10. So, any path which gives you that cost it would be a feasible path. Now you will see that there can be many paths. So, you can actually take 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 that will be a path or you can go like this ok, if you can start from here.

So, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10. So, there will be many paths which the cost will be the same. The 10 as long as the minimum cost is 10, you will be able to the maximum cost that you can have for the feasible path is 10 only. So, that is the minimum value of travel cost. So, this is the grassfire algorithm. So, you will see that there can be many paths to travel to the goal from the start point.

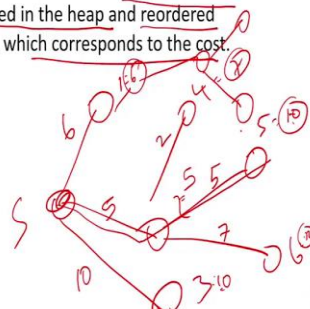
So, as the once you have the graph and then it the graph search algorithm will start exploring either through a breadth-first or a depth first approach and get the cost and that cost can be used to find out the most optimal path for travel. So, that is the grassfire algorithm.

(Refer Slide Time: 16:20)

### Dijkstra's algorithm

This algorithm is similar to breadth-first search, except that edge costs may assume any positive value and the search still guarantees solution optimality.

This algorithm expands nodes starting from the start similar to breadth-first search, except that the neighbors of the expanded node are placed in the heap and reordered according to their value, which corresponds to the cost.



NPTEL



So, in this grassfire algorithm what we used was that  $g$  was constant that is we assumed that the travel cost from one node to the other node is equal for all the nodes irrespective of how far they are and what were there may the condition, we assume that the cost is equal to one cost one node to another node is equal and therefore, we assume that  $g_n$  is constant.

Now, suppose  $g_n$  is not constant then we cannot use that method because there will be too much of variation. So, we go for something called a Dijkstra algorithm to do this, where  $g$  is not constant ok. So, it is similar to the breadth-first approach, it will follow in the same procedure of breadth-first, but it will the cost may be assume any positive value and the search still guarantees the solution optimality.

So, the method of search will be breadth first, but since the costs are different it will take a different approach in expanding the nodes further because when the costs are equal, you can actually keep expanding for all the nodes, but in this case you do not need to expand the based on that we will actually look at the cost value and then expand it further, ok.

So, the expands the algorithm expands nodes starting from the start similar to the breadth-first, except that the neighbors of the expanded node are placed in the heap and recorded according to their value which corresponds to the cost. So, it will start with a node and then identify all the next node cost ok. So, there will be cost involved in travelling to the all the neighboring nodes.


So, suppose the cost is 1, 2, 3 this is a start and these are the nodes, it will try to find out what is the cost of travel from start to the 3rd one. So, maybe suppose this is 10, this is 5 and this is 6. So, it will put all these things into a group and then find out which one is the smallest one. So, it will say ok to reach here it is cost is 5, then it will try to expand this one further ok.

So, this is 4, 5, 6 are the nodes and then suppose the cost is 2, the cost is 5 and the cost is 7. Now, we know that if we come from here to 5 and then go over here it will be 5 plus 5 this will be 10 cost this will be 7 and this will be  $5 + 7$ ; 12. So, the cost will be 12, 10 and 7. Then it will look at all those unexplored nodes. So, this is 1 this is 6, now this is 6, this is 10.

So, which one is the smallest one it will out of 10 12 10 7 6? So, it will see that this is still the smallest one. So, it will start exploring this smallest one and then go expand all the nodes. And continue to do this till all the nodes are explored and then it will try to find out which one gives you the minimal cost to the goal and that will be the shortest path. So, that is the way how the Dijkstra algorithm will proceed in a breadth-first approach.

(Refer Slide Time: 19:47)


Dijkstra's algorithm



This algorithm is similar to breadth-first search, except that edge costs may assume any positive value and the search still guarantees solution optimality.

This algorithm expands nodes starting from the start similar to breadth-first search, except that the neighbors of the expanded node are placed in the heap and reordered according to their value, which corresponds to the cost.

Subsequently, the cheapest state on the heap (the top element after reordering) is extracted and expanded. This process continues until the goal node is expanded, or no more nodes remain on the heap. A solution can then be backtracked from the goal to the start. Due to reorder operations on the heap, the time complexity rises



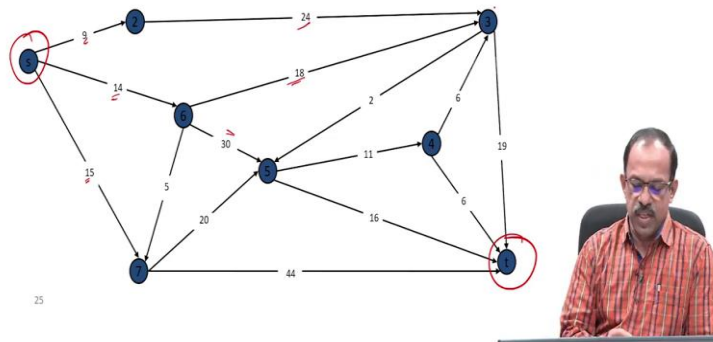
I will show you an example ok. So, the cheapest state on the heap is extracted and expanded further as I mentioned. So, that you look at the cheapest or the lowest cost and then expand it further and it continues until the goal node is expanded or no more nodes remain on the heap. A solution can then be backtracked from the goal to the start. So, you can actually find out go back from the goal and then check which path is the one which gives the lowest cost. So, that is basically the Dijkstra algorithm ok.

(Refer Slide Time: 20:23)

# Dijkstra's Shortest Path Algorithm



- Find shortest path from s to t.



So, let me skip here. Let us take this as an example for explaining the Dijkstra's shortest path algorithm. So, now, we have a start goal I mean start point and we have a target t. So, this is the target, these are the nodes, you have 2, 7, 6, 5, 3, 4. So, these are all the nodes through which the robot can pass and the cost of travel from each one is given as.

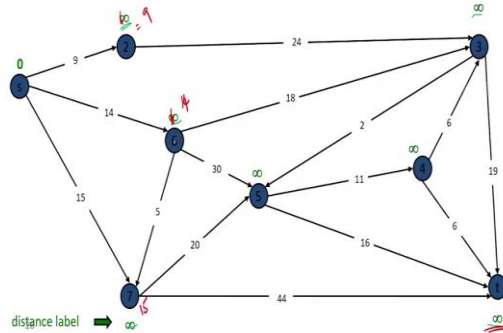
So, from s to 2 the cost is 9, and this is 14, and this is 15 and similarly from 2 to 3 this cost is 24, 6 to 3 the cost is 18, and 6 to 5 the cost is 30 like that given and the arrow shows that the direction which can travel. So, it for example, this shows that from 5 you cannot go to 3 only 3 to 5, then 5 to 4 or 4 to 3 is possible. So, all those costs are given. Now, how the Dijkstra algorithm works is that it will try to find out. So, initially it will assume that the cost is infinite to all the nodes.

(Refer Slide Time: 21:34)



## Dijkstra's Shortest Path Algorithm

$S = \{ \}$   
 $PQ = \{s, 2, 3, 4, 5, 6, 7, t\}$



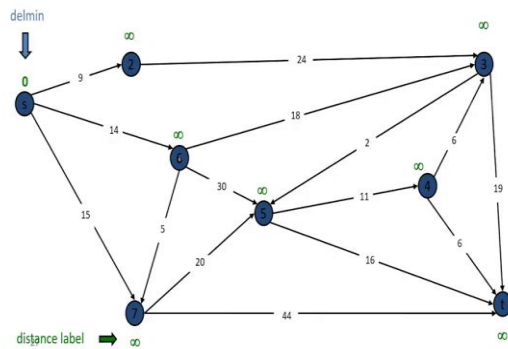
All the node costs are where total cost is infinite initially and this is 0 and here it is infinite. So, it will assume this is the initial values for all the nodes. And then what it will do? It will try to expand the neighboring nodes. So, you have the neighboring nodes as 2, 6 and 7 and then it will find out what will be the cost of travel from s to 9, s to 2, s to 6 and s to 7, it will write this cost as 9, 14 and 15 ok. So, instead of this it will put the new cost as 9, 14 and 15. So, that is the next stage ok.

(Refer Slide Time: 22:22)



## Dijkstra's Shortest Path Algorithm

$S = \{ \}$   
 $PQ = \{s, 2, 3, 4, 5, 6, 7, t\}$



(Refer Slide Time: 22:23)

### Dijkstra's Shortest Path Algorithm

$S = \{s\}$   
 $PQ = \{2, 3, 4, 5, 6, 7, 1\}$

decrease key

distance label  $\rightarrow$  15

NPTEL

So, it will decrease the (Refer Time: 22:26) 9, 14 and 15 as the cost. And then if you look at in this heap that or the set of cost which one is the lowest. So, it will see that 9 is the lowest and therefore, it will expand the 9 to the next neighboring node. It will look at what is the neighboring node of this 2 and how much will be the cost of travel from 2 to the neighboring one is 3. So, it will see that ok this is 9 plus 24. So, this will become 33 as the next node explored.

(Refer Slide Time: 23:00)

### Dijkstra's Shortest Path Algorithm

$S = \{s\}$   
 $PQ = \{2, 3, 4, 5, 6, 7, 1\}$

delimin

distance label  $\rightarrow$  15

NPTEL

(Refer Slide Time: 23:02)

### Dijkstra's Shortest Path Algorithm

$S = \{s, 2\}$   
 $PQ = \{3, 4, 5, 6, 7, t\}$

31

NPTEL

(Refer Slide Time: 23:05)

### Dijkstra's Shortest Path Algorithm

$S = \{s, 2, 6\}$   
 $PQ = \{3, 4, 5, 6, 7, t\}$

32

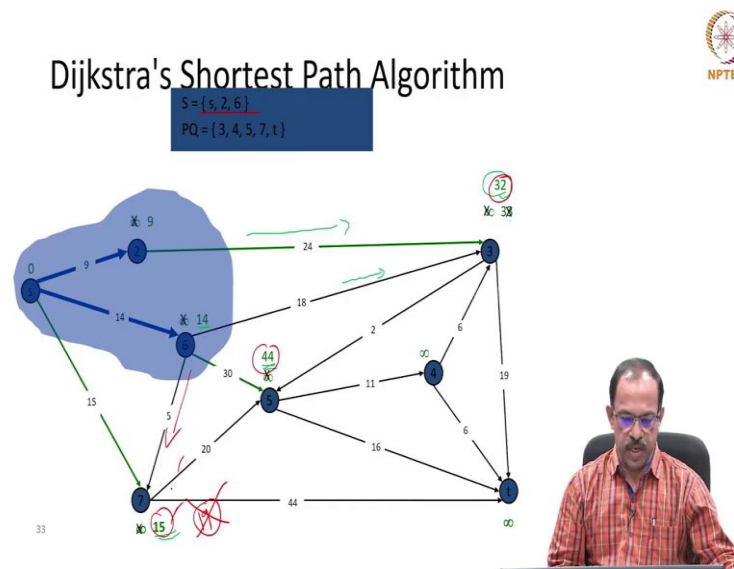
NPTEL

So, it will explore the next one. So, it will look at this one. So,  $s$  and  $2$  will be the one the all these are all unexplored one and then it will expand this and then make this as  $33$ . So, now, it is  $s, 2$  and  $3$  will be explored that is all it has become  $33$ . Now, it will look at out of all this unexplored one which one has the lowest one.

So, this is  $33$ , this is  $14$  and this is  $15$  and so, this is the minimal one. So, it will look at this as the minimal one and therefore, it will include this  $6$  in this one. So, it will become  $s, 2, 6$  because that will be added and then we will try to expand it further ok.



(Refer Slide Time: 23:46)



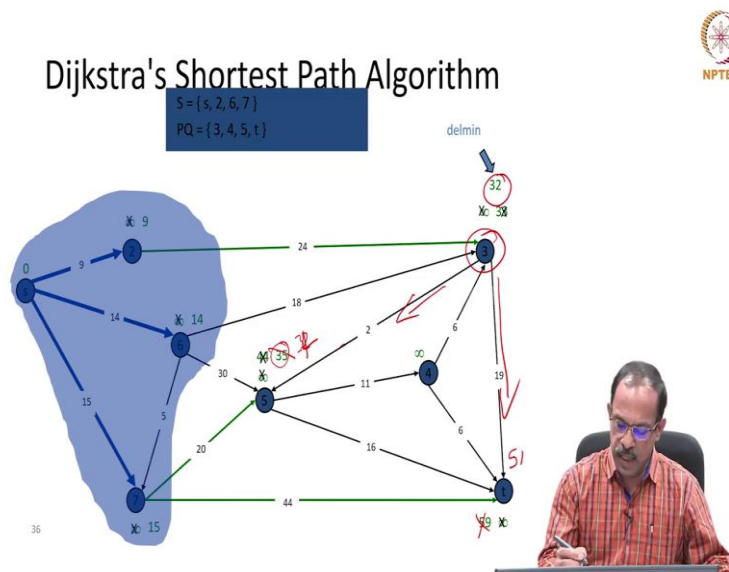
So, that will be expanded further. So, you can see now it has become set as s, 2, 6 and it is explored and then it will try to calculate the new cost. So, from here, so, this is 14. So, this is this cost will be the and it will explore all the other all the neighboring nodes. So, from here this is the neighboring node, this is the neighboring node, then this is the neighboring nodes. So, it will try to explore all these cost.

So, it will actually what it will do? Suppose, it from 14. So, this cost is 14 and it has to travel here then it become 19 ok, but then you already have a cost of 15. So, lowest cost of travel from s, 2, 7 is 15 is already there, but if you take this path it will be 19 and therefore, it will not consider this as the cost, it will still keep the 15 as the minimal cost for that node 7.

Similarly, it will look at from 6 to 3, it will be 14 plus 18 it will be 32 and then from 6 to 5 it is 44, 30 plus 14; 44. So, it has expanded all the neighboring nodes and then found out the cost. And if the cost is more than the already existing cost to that node that will not be updated otherwise it will be updated. So, it will be still remains as 15 and now it will look which one is the shortest smallest.

This is 32 and this is 44 and this is 15. So, 15 is the shortest, I mean smallest value. So, it will try to expand this node now. So, 7 will be expanded. So, 7 will be added to this and then it will be expanded ok.

(Refer Slide Time: 25:25)



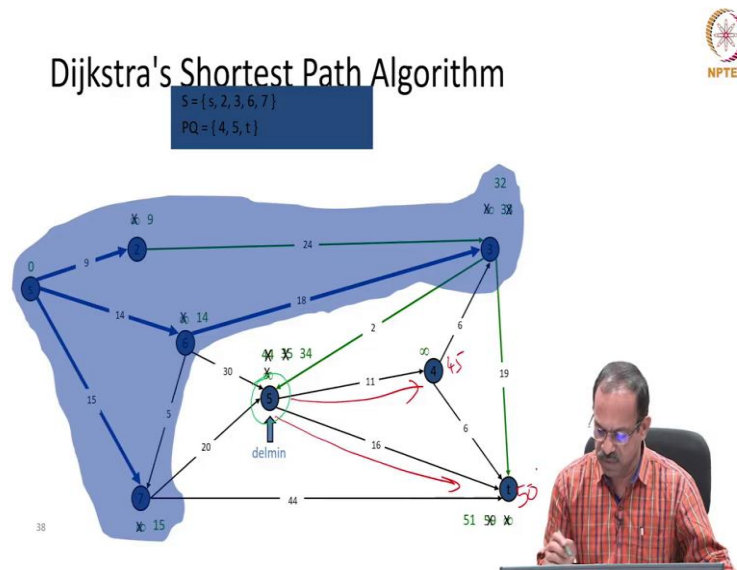
So, 7 is added. Here that it become s, 2, 6, 7. So, 2, 6 and 7 all explodes and now you expand 7. So, 7 can actually two neighboring nodes; one is 5, other one is t. So, this cost is 44 and this is 15. So, it becomes 59 to reach. So, the target can be reached with a cost of 59 if you take from come from 7.

We do not stop there. We need to find out is there any other path which actually gives you lower cost. So, what it will do? It expanded this. So, this has become 15 plus 20, it is 35 which is smaller than 44. So, it will take 35 as the new value. And then it is already 32 here.

So, this one there is no route from here. So, it will not do, it will not update this. The next one it will be now it will look whichever is the, which one is the short smallest one sorry, yeah 32 is the smallest one because it is out of 35 is there, 32 is there and 59 is there. So, unexplored this is 32. So, 32 will be expanded further because this is 35. So, 32 will be expanded.

So, this node will be expanded and this node has got one route here and one here also. So, it will try to find out 32 plus 19, the value will be the travel cost to this t. So, that will be 51. So, instead of 59 you will get a 51 now. Similarly, this is 32 and 32 plus 2 this will be 34. Now, if you want to reach 5, if you take this route then from 3 and if you come to 5 it will be 32. So, it will become 32 sorry 34; 32 plus 2, 34. That is the way how it will be expanded ok.

(Refer Slide Time: 27:31)



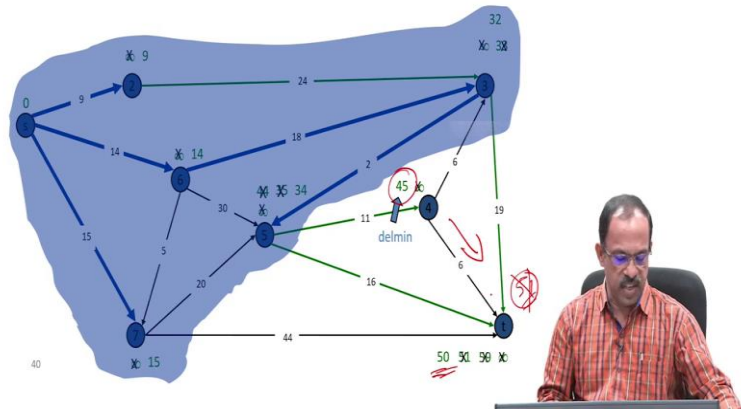
So, we will can expand this 3 now. So, if you expand this 3, you will find that this has become 34 and this has become 51, from 3 this has become 3 to 5. So, 34 cost and then 3 to t their cost will be 51. So, you can see that the total cost to the reach t has reduced from 59 to 51. Again, we look at which one is the smallest in this case. So, you have a 34 and then you have a; you have that one, only 31 is the what at currently.

So, we 34 is the one, you expand that one further ok. So, you expand this 34 and then you can see that 34 to here is a route, here is a route ok. So, 34 plus 11 this will become 45, 34 plus 16 this will become 50, ok. So, that will be the new cost for these nodes ok.

(Refer Slide Time: 28:27)

## Dijkstra's Shortest Path Algorithm

$S = \{s, 2, 3, 5, 6, 7\}$   
 $PQ = \{4, t\}$

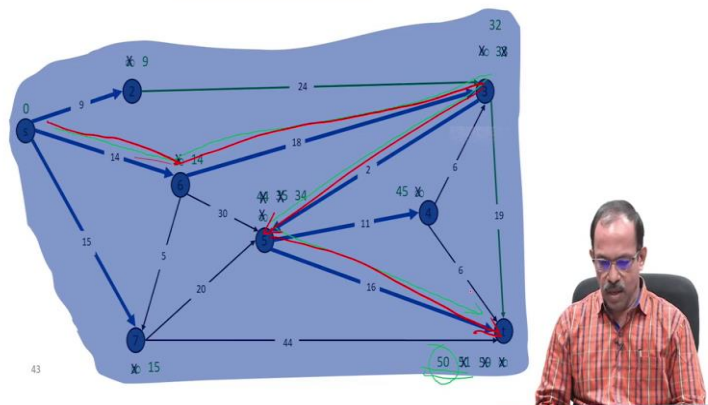


So, it has become 45 and 50. And out of this, so, 45 is the minimum. So, the no more further nodes are there. So, 45 is the minimum. So, we expand this further and then find out what is the cost of travelling from 4 to t. You will see that it is  $45 + 6$ , it is 51. So, we do not add that too. We (Refer Time: 28:48) consider that because 50 already, it is 50 is there. So, 51 is more than that. Therefore, we do not consider that upgradation, we just leave it as a 50 only ok.

(Refer Slide Time: 28:57)

## Dijkstra's Shortest Path Algorithm

$S = \{s, 2, 3, 4, 5, 6, 7, t\}$   
 $PQ = \{\}$



So, we have explore all the nodes and found out the cost of travelling from all these nodes to the target and we found that the minimal cost that you can have is 50. So, any

path that gives you the cost of 50 will be the most feasible path. So, this is the way how the Dijkstra's algorithm works.

Now, you can look at the path from here which one is giving you the 50, any path which gives you the 50 will be the feasible path and you can work back and then find out which path is the one which gives you the 50, ok. So, that will be the final path that you can see here.

Now, you can see there can be multiple ways you can reach here, but there will be only one path. There can be there can be more than one path, but in this case you will have only one path which gives you the cost of 50, that is you have this  $14 + 18, 32 + 24, 34 + 16, 50$ .

So, this becomes the feasible most economical path 6 to 1 start to 6, 6 to 3, 3 to 5 and then 5 to the target. So, that will be the most economical or the most optimal path for the robot to travel. So, this is basically the Dijkstra's shortest path algorithm. I hope you understood the principle.

Only thing here is that the cost of travel are different from between the nodes and therefore, we need to find out the shortest path based on the total cost of travel. And the Dijkstra's algorithm go by the breadth-first approach and then try to calculate the cost of travel to each node and then expand the lowest cost node further and continue this process till all the nodes are explored ok. So, I will stop here. We will continue the discussion in the next class. We will look at the A star algorithm in the next class.