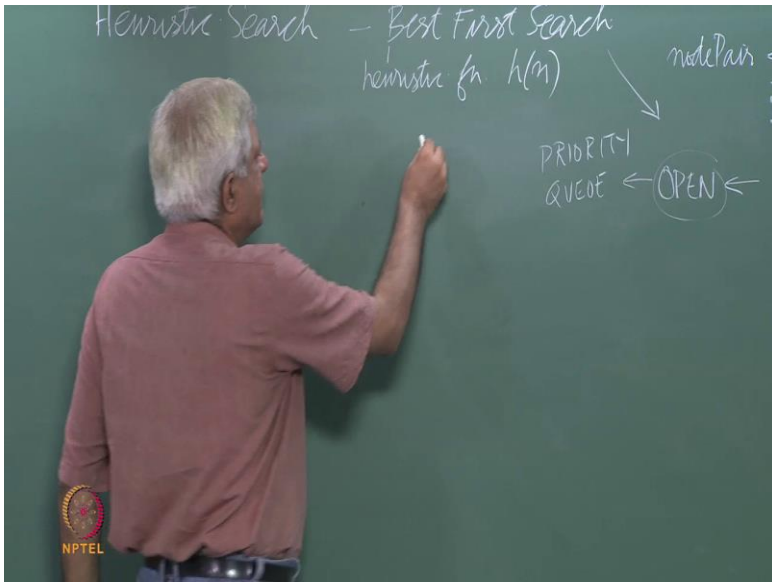
Artificial Intelligence

Prof. Deepak Khemani
Department of Computer Science and Engineering
Indian Institute of Technology, Madras

Lecture No - 10
Hill Climbing

(Refer Slide Time: 00:17)

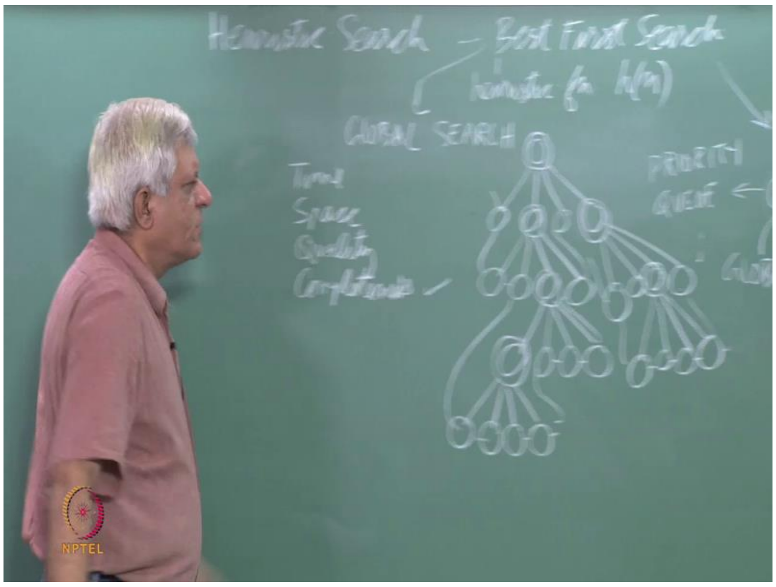


So, we were looking at heuristic search and you saw this algorithm column best first search. So, the way that this algorithm works is that it is the heuristic function h of n which gives you an estimate of the distance to the goal and what essentially best first search does is that it sorts the open list. So, let us say new is a set of nodes that is generated by Mogen and the basic difference between best first and early algorithm if we saw was that open is modified as follows.

If we do not have this sort h then you would simply have offend the new loads to the tail of open and then it is behavior would have been like depth first search, now what you are doing is that we are sorting on this heuristic function h. So, the best nodes come to the head of the lets see we always speak node that strategy is does not change we always pick the node from the head of the open. Then, start does all the processing that we do and eventually add the new nodes open in sort.

Now, when I say sort we essentially mean conceptually we have sorting it, but of course from the computational point of you it would be quite silly to sort open this every time and, but we really do is that in set we maintain open is a priority cube. This is an efficient way of maintaining a sorted list of elements and you can keep adding new elements to that, now if you look at the behavior of best first search.

(Refer Slide Time: 03:18)

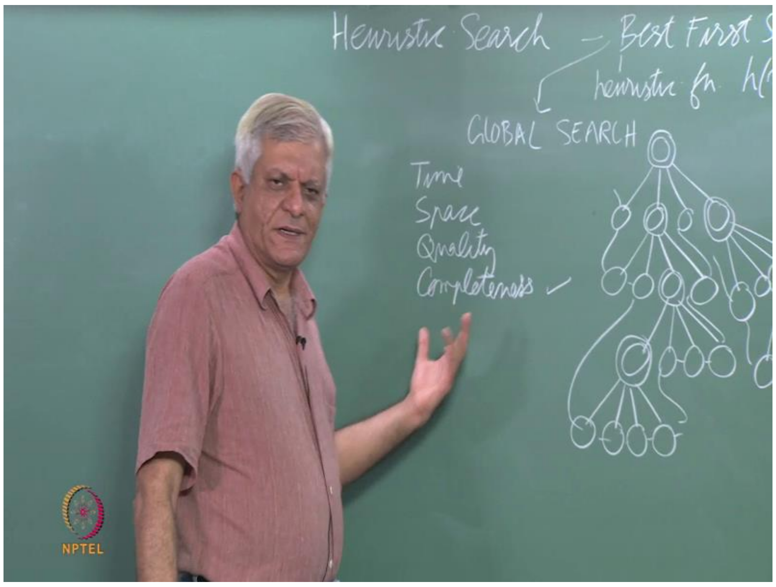


What it does is it is starts with some node then it puts it into closed and adds it is successes to the open list. So, the single circles are open and the double circles are closed then it picks one of them whichever has the best h value and expands that now everything else everything in single circles season open list.

It can take any one of them depending on the h value, so let us say that it expands this node next and I am assuming here a branching factor 4 and the search proceeds in this fashion may be this is the next one that it expands. Since it always looking at the heuristic function it not necessary that it will expand one of these four any one of them could suddenly turn out to be better than that. This is because this is my gradually become worst and worst in terms of heuristic value and is possible that after this doing this it suddenly goes off in the different direction.

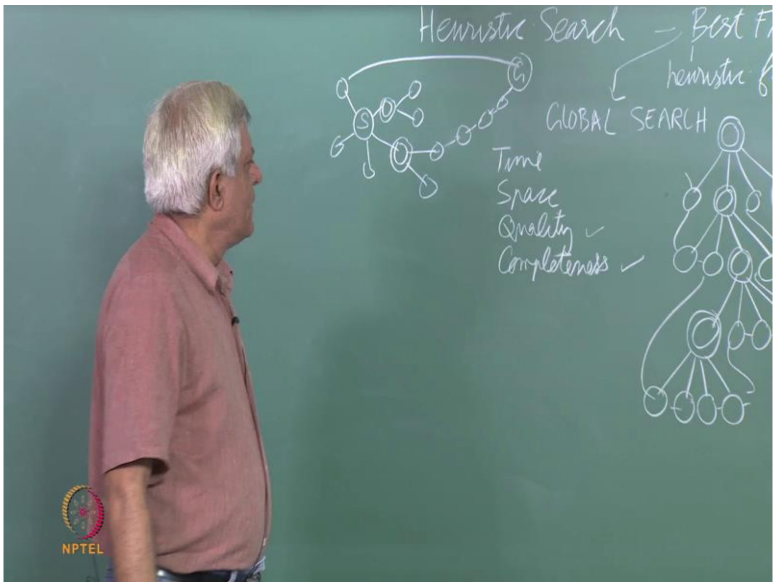
So, it expands this and then may be it expands this, so the search can, so the jump around this entire list of nodes is essentially open list if you can make this out. So, the open list is basically a list of all possible candidates that it is generated which could be inspect that some later point of time. It is a global list and best first search is a global search algorithm. So, what are the properties of this algorithm that we have looked at properties from four perspective?

(Refer Slide Time: 05:24)



This is time space quality and completeness, so if you look at this algorithm from this four qualities we can say that it is complete because the only thing it does differently from best first search. Best first search is there it is sorts opened this; otherwise it will always take one node from open and expand it and so on. So, put at least for finite spaces it is complete the quality of the solution as we saw is not necessarily an optimal path.

(Refer Slide Time: 06:11)



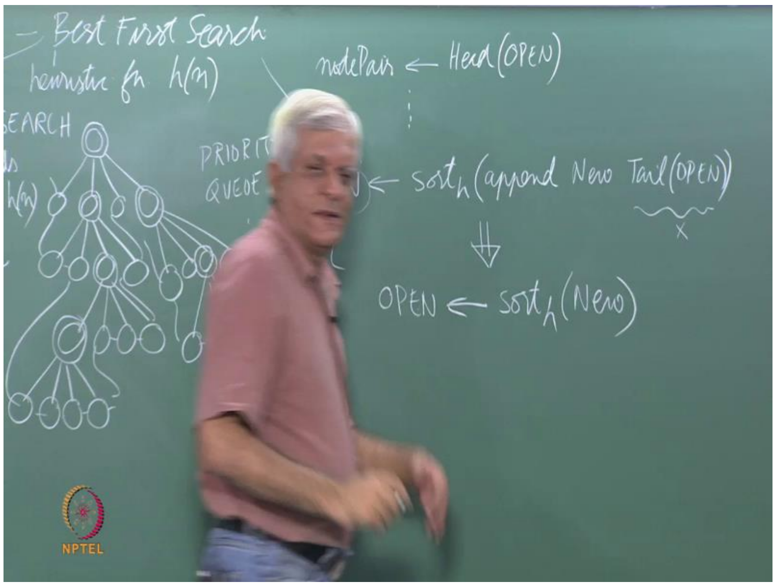
So, an example of that is if this is the start node and it has let us say some source and the goal is somewhere here then it slightly that this best of search will go along this path. So, maybe it will expand this next generated structure source may be it will expand this generated structure source and so on. So, it will find some path to the goal which will have some number of nodes where is this possible in that this one had a direct link to the goal in which case it would not have on this path of length to.

Remember that we are not counting we are not allocating any h cost we are viewing. Simply counting how many states are there in the solution first and this situation it will not find the shortest path, which is this one or it will go in this direction.

We also saw in the example that if you have if you have a city map and suddenly there is a river on the way. Then, the first best first search will drive it towards the goal and suddenly see that there is a no bridge and it will have to take a and find the longer path decision. So, quality is not guarantee we do not necessarily get an optimized path now as to time and space complexity. It really depends upon h if the heuristic function is good then it will drive the search towards this goal directly and you will find goal in linear time requiring linear space it is like a algorithm.

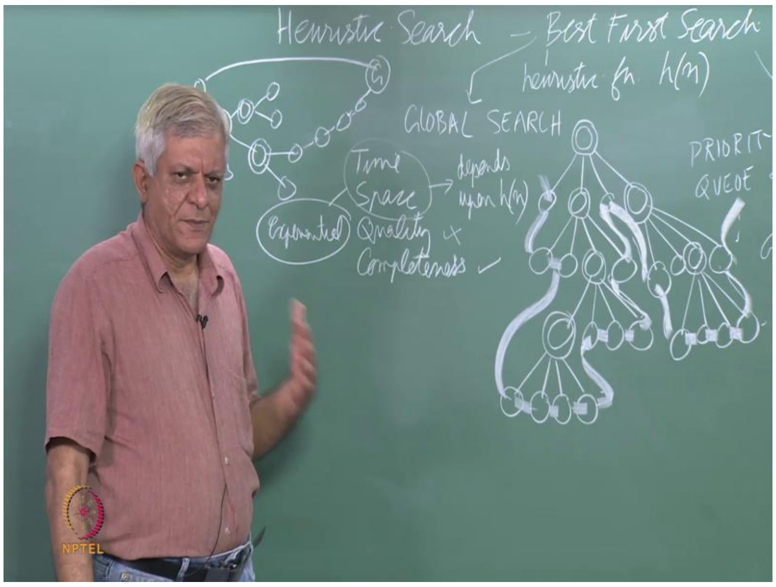
If it will directly goes straight towards the goal, but in factors of course that does not happen in factors it is very difficult to devise heuristic function which has so good and in practice it tense to be exponential. So, this is something that we want to try and that we do not want algorithm which are exponential in nature, so what can we do to devise algorithm which will require lesser space and lesser time complexity.

(Refer Slide Time: 08:43)



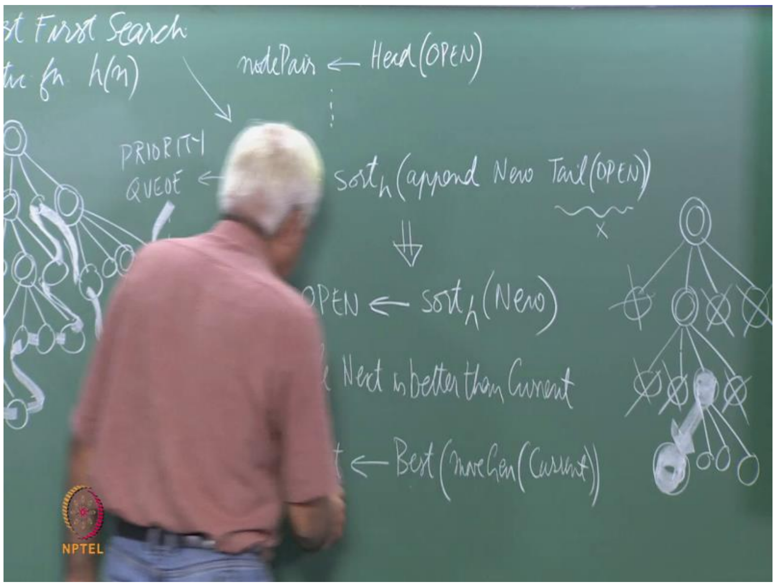
So, let us look at this variation of this algorithm in which we modify as follows we simply do this that open is the sorted version of the new nodes that we have generated. So, what have we done here we thrown away all this nodes that we have generated earlier.

(Refer Slide Time: 09:10)



So, in this case the open list is set of includes this nodes it is big list nodes that we have generated sometime in the past and nodes that we have just generated everything is included into the open list. So, this is like a search there is now what we are saying is do not do that just maintain just look at the latest note that what they would generated the new the newest note that was generated and just pick the best amongst them.

(Refer Slide Time: 09:47)


st First Search
ic fn h(n)
PRIORITY QUEUE
nodePair ← Head(OPEN)
sort_h (append New Tail(OPEN))
OPEN ← sort_h (New)
Next is better than Current
t ← Best (moveGen(Current))

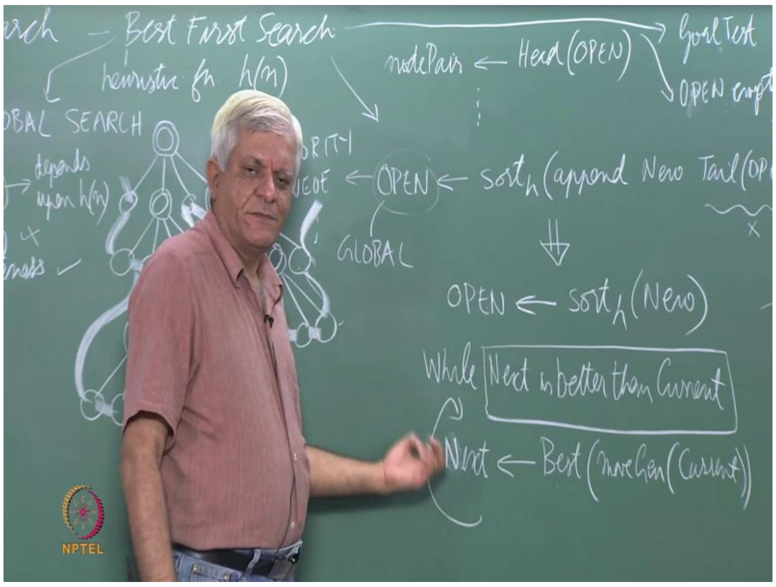
So, what is the behavior of this algorithm you start with some search some start node and let us assume that we keep that of the parent point you generate the children. You pick one of them let us say the same heuristic function we are using we take this node, but we generated children. Now, this algorithm is saying that the set of candidates is only these children of this node that we have generated and we have forgotten about those are the nodes. So, in effect we have deleted them, so those nodes no longer exist, so open as we can see is going to be a shorter list then we generate the third load form here generated children and we throw away this.

So, what is happened with this variation is that the search has access only to the latest nodes that have been generated only to the neighbors of the current node it is infect. Once we are going to do this, you do not your need to sort this thing we can simply modify the algorithm as follows that next and next is a name of a node it is simply the best of current. So, essentially what we are saying is at if you are at current node, so for example, if this is the current node then you move to the next node which is this let us say this one. So, you have a here and then you simply move to this one and that is the move that you make here.

So, that is current and this is next you see because we do not want to waste time sorting because we are not going to use those anywhere later you see once you decided this is the best of the of this current. We can directly select that and this can be done in linear time of first and we just put this in a loop while next is better than current, we just keep doing this as long as we can see a better node move to the better node. So, one thing that we have done here is that we have changed the domination criteria when we are doing best first search the domination criteria was either goal test or open empty.

This means that either we have found paths to the goal or they are no more candidates left in which case there is no path to the goal because the algorithm is a complete algorithm. You would terminate in the positive piece only when the goal test function returns true now we have these criteria while next is better than current.

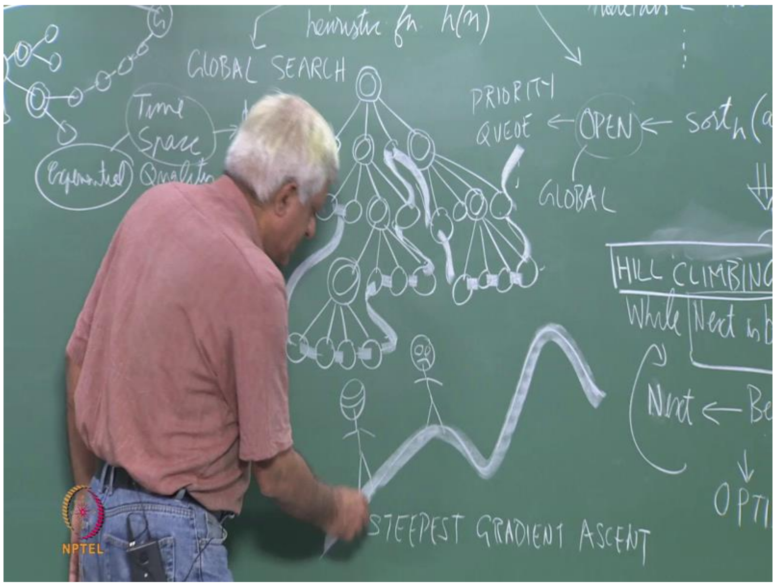
(Refer Slide Time: 13:40)



So, what we mean that better this that in the case of heuristic function that we were talking about it simply means the heuristic value of next is better than the heuristic value of current in. If the heuristic value of the goal is 0, then it should be lower than the value of the current node decision.

So, you have changed the termination criteria simply and we have converted this into an optimization problem, we are saying that optimize the value of heuristic function. So, instead of a state space search algorithm, we have now converted into an optimization problem and said find the node with a best value rich if you see and we keep moving forward till we find the better value of this.

(Refer Slide Time: 14:39)



Now, if you consider your situation, there you are blind folded and you are standing on the slop of a hill side and you have been told to go to the top of the hill then what is the algorithm that you would follow is, so this is you standing blind folded. The algorithm that you would follow is that you would pose possibly take a step in all direction well some income this set of direction. Then, move forward to that direction which seems to be going up, now this is a two dimensional world or a one dimensional world in which you can only move left or right.

Then, you can see that if you go right you will be going higher if you go left you will be going lower, so you go right, which is exactly what this algorithm is doing that it set of look exploring the neighborhood of the current state.

Then, saying that if there is a better neighbor it moves to that state next is best of this thing and you keep doing this still next is better than current actually to be more precise. You should have the check at this stage itself that you should check whether next is better and then only move to the next in principle what you are doing is you are moving along the steepest gradient. In this example, we can say that we are doing steepest gradient ascent and it is not surprising that this algorithm is actually call hill climbing. So, if I remove this version now becomes a little clear to watch, so this algorithm called hill climbing algorithm.

Essentially, it is analogous to climbing a hill blind folded and we just move in the direction of the speakers flow and hope to reach the maximum. So, this means that if you reach here and then you terminate and then you allowed to open your eyes and see if we have a local maxima then we will stopped the local maxima exactly how will that will ever reached the global maxima exactly. So, that is what I would just about to illustrate here, so this is the figure that I am trying to draw that you have done this climbing and you have reached the place where I all the neighbors are not better than the current neighbor.

So, initially of course you have a smile on your face saying that you have reach the maxima, but when you open your eyes then the smile turns into a because then you will discovered that actually this. So, precisely the problem that you have pointing out that this algorithm will take it to maxima or a minima if you are minimizing because it should be analogous. You will get start in this local maxima and that is the problem that terms because of the fact that this algorithm is a local search algorithm. It only looks in the neighborhood of a current state to decide where to go next unlike the best first search algorithm which maintains a global list of open candidates.

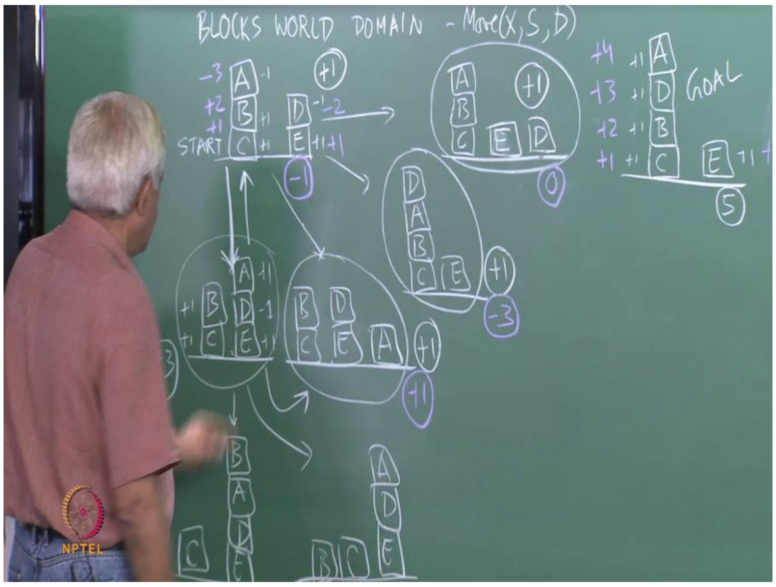
It could always move to a better candidate, but never does says must to be said about this algorithm why because of the complexity issues involved time space complexity completeness as we have just observed it is not complete. So, hill climbing cannot take you to the global optima it is not complete, it will get back of the local maxima quality of the solution.

Also, you cannot even talk of completeness, so you cannot even talk of quality again, but time and space complexity. So, what is the space complexity of this algorithm in a it is log n if we take a bindery tree see in that log n. So, similarly by space complexity we mean the size of the open disk how many nodes does it need to keep in the memory only those mean number of open it explore by this heuristic function this heuristic function is there's the best. So, next on you go on incrementing only one node at a time is present in the priority queue one plus four in this case 1 plus 4. So, it is constant space complexity, so this is the single major advantage of this algorithm is that it requires constant space as appose to best first and the entire algorithm that we saw.

We said that is general require exponential space off course well does not require exponential space, but best first search does because it could not it had no sense of direction. It would say and hill climbing also games on time because it only moves along the gradient it will stop moving once the gradient becomes negative. So, in some sense it will require linear time that it will take a if it takes n steps, it just about takes the n steps and communicates with that essential. So, the question is what is this surface appear talking about this hill that we are talking about where does this surface come from.

The answer is that this surface is defined by the heuristic function that we are using to guide such thing. So, I would want to take a couple of examples one example to illustrate that you can have two different heuristic functions and they will define different surfaces. Now, if the surface of the that you what in the space that you are searching were to be smooth if the surface was like this then you can see that this algorithm would have taken you to the global maxima that is the global maxima. So, it really depends on nature of the problem if the nature of the problem is searched that the heuristic function defines a smooth and monotonic surface then hill climbing will work, otherwise it will get struck to on a local maxima.

(Refer Slide Time: 21:30)



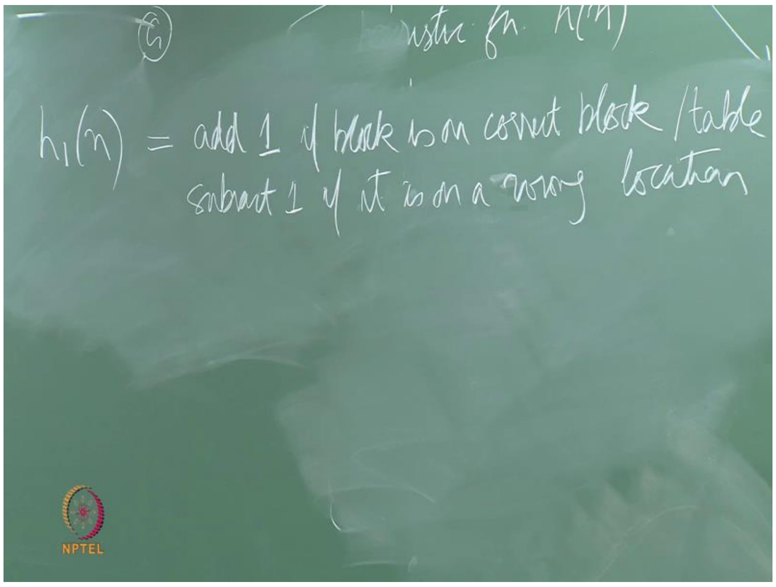
So, let us take an example from the blocks world domain which is a domain which is often used to illustrate many ideas in e i and the domain consist of a set of children blocks. So, for example, you may have a b set of blocks which are arranged like this, so you can keep only one block on another block, so here we have free blocks and filled up one top of the other. Then, there is the table and then we have two more blocks filled up like this, so this is the start state given to us and the goal state is let us say state which looks like this that you on a to sit on D and then D to sit on B and b to sit on C and let E be like this.

So, this is the goal state what are the moves available to you we have only one move which we will say is like this move block x let us call it commas source to destination and source can be a top of. So, this move can only be done if you can pick up of block, which means it must near the top of a stack all around the table and you can put it down either on the top of another block or on the table. So, the source can be only the top of a stack or the top most blocks in the stack and the destination also can be only a top most block in this stack. So, for example, in this situation we can do the following moves we can say move this to top of b.

So, you would get a situation like this that is one possible move you can then another move we can make is we can put a down. So, this B and C remain like this D is here and A is here, A third move that you can make is we can move D. So, only A and D are the two blocks we can move here because they are of the only once on the top of the stack. So, you can put D on top of a, whereas E remains here and the fourth possible move is that you can put D down. So, you will get these are the four possible moves, so this is the neighborhood of this state and you want to use hill climbing algorithm to decide which state to move.

So, now, it means to design the heuristic function, so what is what can be a heuristic function that you can use here, so I want to discuss two functions. They are as follows the first function says that if a block is sitting on a correct destination block. Then, you add one further block what do your destination block A should be sitting on D should be sitting on B and B should be sitting on C and C should be on the table and likewise for E should be on the table.

(Refer Slide Time: 25:32)



ustic fn. $h(n)$

$h_1(n)$ = add 1 if block is on correct block / table
subract 1 if it is on a wrong location

So, let us call this h one of n it says add one if block on i just use a term block, so in general we mean either block or the table and we subtract 1 if it is on, so this gives a such function which only looks at a state. So, the idea of a heuristic function in that it should be computationally cheap pieces just look at the state and get a value out of it. So, let us just give heuristic values for all this functions, so let us start with a goal, so for this we will have plus one for this plus one for this plus one for everything because everything is on the correct place. So, the heuristic value of the goal is 5 or plus 5 whatever the start state.

The start state we will have plus 1 for this plus 1 for this minus 1 for this because a should be on D, but it is on b plus 1 for this minus 1 for this because d should be on b in the goal state, but it is sitting on e. So, if we add this entire sub you will get a value of plus one now what about the other states. So, let us look at this value this is a, b, c this ends to plus one here plus one for this and minus one for this. So, just let me know if I making a mistake this state am plus one here it is minus one for this minus one for this plus one for this plus one for this. So, this cancel, so and e, so this is also plus 1 for this 1 plus 1 for this plus one for this minus 1 for this plus 1 for this minus 1 for this. So, this is also plus 1 in this case it is plus 1 for this plus 1 for this plus 1 for this plus 1 for this.

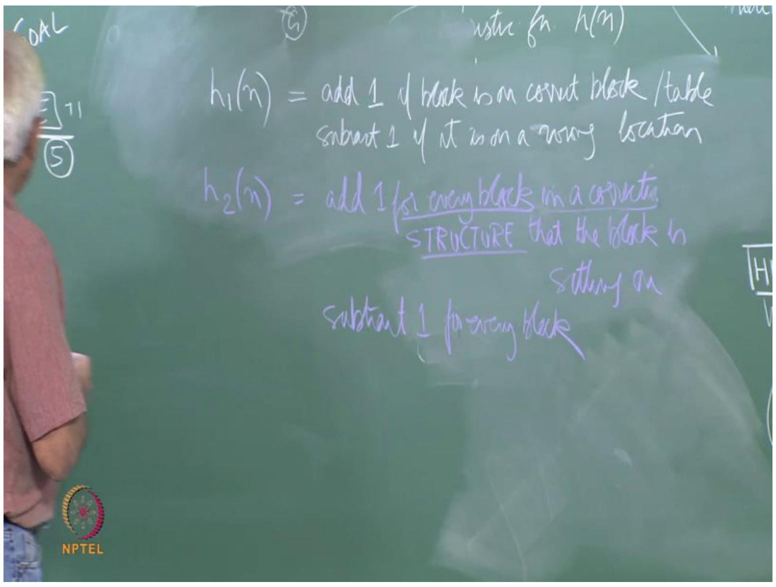
Notice because a sitting on d and in the goal you want to a to be sitting on d and minus one for this. So, this value turns out to be plus c, so these are the four states this is one state this is another state this is the third state this is the fourth state that this move gun function generates and heuristic function tells. So, the algorithm says look at the neighbors if one of the neighbors is better than if at least one neighbor is better than the current state then choose a best among the neighbors, in this example three neighbors are equal to plus 1. So, we do not consider that where we consider this one which is better than, so the first move hill climbing will make this heuristic function is this one it will complete, so from here what are the options?

Again, we do the move gun function, so you can move either d or you can move a if you move a one thing is you can go back to this state you can take a from here and put it on top of b or you can take a from here put it down here which is you will go to this state.

The other options are you can take b on top put it on a, so you will get this state and you must tell me that heuristic value of this. So, plus 1 for this minus 1 for this plus 1 for this minus 1 for this plus 1 for this, so this whole thing comes to plus 1 so that is one move from here and one more move is that you can pick up b and put it down on the table. So, you would get B C and A A D E, so this minus 1 for this plus 1 for this plus 1 plus 1 and minus 1, so this is also plus 1 is that correct.

So, here we are sitting on a state which has the heuristic value of plus 3 and it has four neighbors this has the value of plus 1 we have completed earlier this has the value of plus 1 we have completed earlier and these two also have a value of plus 1. So, you can see that this is the maxima and the algorithm will terminate here without reaching the goal state that we are interested.

(Refer Slide Time: 30:49)



$h_1(n)$ = add 1 if block is on correct block / table
subtract 1 if it is on a wrong location

$h_2(n)$ = add 1 for every block in a correct STRUCTURE that the block is sitting on
subtract 1 for every block

So, let us have a different heuristic function and this function is as follows. So, let us call it h two of n and this says add one for every block in a structure. So, we are looking at the whole structure that the block is sitting on and subtract one for every block. So, the difference between that is that you adding O W either you adding 1 or subtracting 1.

In this case, you may an a add of many things, so you are adding one for every block in the correct structure. So, if it is entire structure below the block is correct then for every element in the structure, you will add one if it is a long structure, for every element you will subtract 1. So let us together start with the goal, so will start with plus 1 here plus 2 here because it is on c and c is on the table likewise plus 3 here because d is on b and b is on c and c is on table, so three things below we factor it and plus four here and this is plus 1. So, this is 10 plus 1, 11, let us look at the start state you will add a plus 1 for c plus 2 for b minus 3 for a because it is a long structure it should have been on A D B C, it is on a b c a b c is not a correct structure.

So, three things below it along, so we have two minus three for here likewise we do minus 2 for this because it is on a wrong structure and plus one for this. So, this three and three canceled out this canceled out, so this is minus 1, the start state you can see the same force nodes are generated by the same move gun function which is that you can move this thing. So, let us see valuate the values for this four states here as before, so in this case as we have seen this adds to 0 plus 1 plus 2 minus 3, so this is 0 this is plus one and this will be minus 1. So, this will end of 0 in this case these 3 add up to minus 0 as before this will add another minus 4 to that because it is in a wrong structure which is fourth in below that.

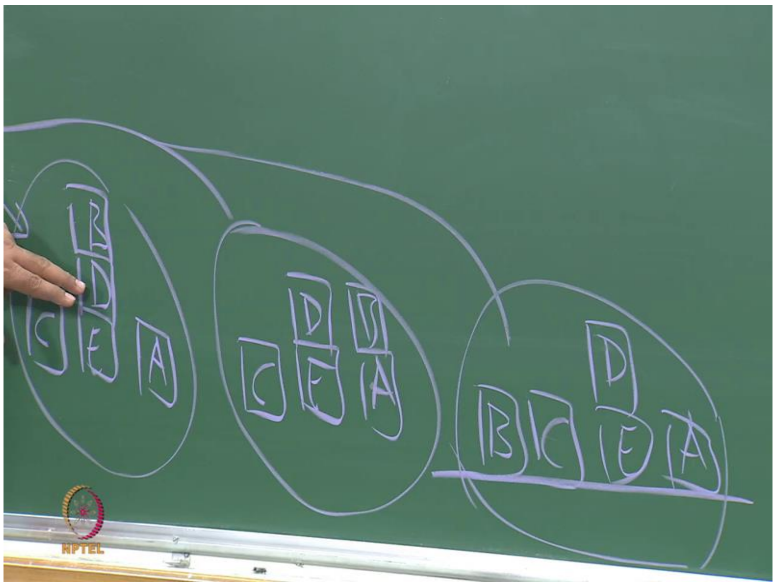
So, minus 4 here plus 1 here, so this will be minus 3 this 1 plus 1 plus 2 plus 3 for this plus 4 for this minus one for this and minus 2 for this. So, his will be plus 1 and this last one which is plus 3 for these 2 plus 1 for this 4 minus 2 for this. So, this becomes 2 and minus 3 for this, so it becomes minus 1 is that correct, so now, let us follows and look at what this heuristic function is how is it evaluating the situation, the first heuristic function took this as plus 1 and this as plus 3 and this always plus 1. So, it is thought this is the good move to make and it move make that move essentially.

Now, when you look at the problem you can see that the optimal solution is when you pick up a put it down somewhere, then pick up D, put it on B then pick up a and put it on that is optimal solution. So, the first correct move to make a s u pick up a and put it down on the table and if you look at this second heuristic function h 2 that we are looking at.

First of all you must notice that it is moved discriminative the first function had only very few set of values plus 1 here plus 3 here and plus 5, there essentially. This one has a value of minus one here plus 1, 1 for this 0 for this minus 3 for this plus 1 for this and minus one for this all values is different. So, it does not think that they are equal in states not only that is feel that this is the good state then this is the next based state and these two state it is comes as to be bad. Actually, because its values are lower essential, now using the algorithm, it has a better state to move, it was at minus 1 you can you can see plus 1.

So, it will make this moves and that you will notice is the correct move to make from here what can we do it can either pick up a and put it on b which is going back here. That you can see is the move it will not make because it is going from plus 1 to minus 1 or it can pick up A and put it on to D and that is the state that we have seen here it can make this move and that is going from plus 1 to minus 1 again. So, it will not make that move, so it is not going to pick up A and do something with it. So, A could have put on d it is going to take it to a bad state or it could have put it back one B this also in which going to take it to a back state, so what about the other possibilities, so it can either pick up B or it can pick up D.

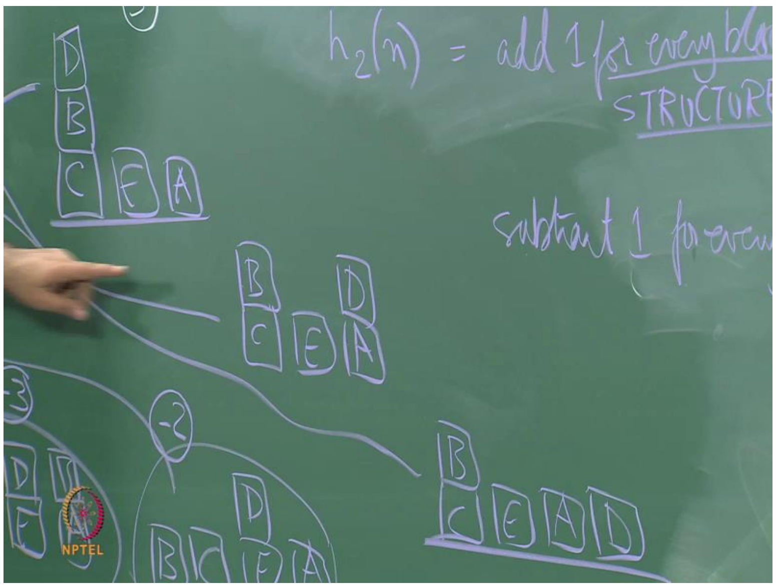
(Refer Slide Time: 36:41)



So, let us first take the B case, so it can pick up D and put it on C on D sorry B, D, E, C A, that is one possibility or it can pick up B from here and put it on a that is second possibility and the third possibility is you can it can put B down. So, these all B moves that we are looking at this the third possible state, so let us just look evaluate these values. So, we have plus one for this minus 1 minus 2 and minus 3 for this, so that is minus 6, so this value should be minus 6 is it correct, so C is plus 1 E is plus 1 E is plus one 1 2 D is minus 2, that cancels that.

So, minus 3 for this and minus 1 for this, so minus 4, so obviously, it is not going to make this move. So, it is sitting at plus one just keep that in mind it is not going to move to minus 1 here or to minus 1 here or to minus 4 here what are the states values for these 2 states. So, this is minus 3 minus 2 minus 2, so obviously, these states are worst states, so it is not going to move from it, so that leave just only the D moves, so let us look at there, so I can either pick up D and put it on B or on A or put it down.

(Refer Slide Time: 38:41)



So, there are three possibilities. So, D, B, C, E, A or it can put D on A which is B minus 1 and minus 2, 3 plus 2 is that correct for this state plus 3 for this plus one 4 for this minus 1 for this and minus 2 or this. So, it will be plus 1, so if remember it was sitting on plus 1, so it would not go to plus 1 or it won’t go it can go to plus 2, but let us look at this one. So, this is plus 1 plus 2 plus 3 that is plus 6 plus 1 plus 4 minus 1 6 plus 1 minus 1, so plus 6, so you can see that this heuristic function will actually derive the search hill climbing algorithm to make these two moves, first it will pick up d and put it. First, it will pick up A and put it on the table and then it will pick up D and put it on B and you can see that the heuristic value is going a plus 6.

If you want to take this forward you can see that this actually leads to the goal state essentially in the next step essentially it will pick up a and put it on top of D recheck value of plus 1, all other moves will be worst in that essentially. So, what we have seen here is that given a problem to solve what is the problem it is a block solve problem, you are given some initial configuration. You have some desired configuration when you are given a set of moves you want to use hill climbing both these functions are static evaluation functions.

This means, they only look at a state ant give you a value for that when I when we valued this plus 1 on this minus 3 or this plus 6 or plus 1 we are not doing any search. We are only looking at this particular state and saying this is the value for this state both is static. So, both require constant time in some sense, but one is more perceptive then the other one is more detailed one looks at the entire structure the other one only looks at what the current block. You can see that one search the first heuristic function takes it to local maxima the second one takes it to global maxima. So, what does that mean, it means that the surface that h one is defining as local maxima it is like this something like this where is the surface at this one is defining is smooth essentially.

So, one thing that you can do when you are using an algorithm like hill climbing chooses a heuristic function which will define a smooth monotonic surface like this. Then, of course you are done and you are done at a very inexpensive price the space requirement is constant. The heuristic function have takes constant time and the time complexity is linear because it will just keep taking one step in positive direction and eventually come to a stop at the goal.

So, that is why hill climbing is such an attractive algorithm because it allows you to do this thing at constant space and linear time essentially the caches can you find the heuristic function which will define the smooth surface for the searched. If you cannot, then you have this problem of having getting struck on local maxima or a local minima as the case. Maybe then, we need to look for algorithms which have variations of hill climbing which can overcome this problem of getting struck in the local maxima. So, we will look at that in the next class, so we stop here.