Artificial Intelligence
Prof. Mausam
Department of Computer Science and Engineering
Indian Institute of Technology – Delhi

Lecture 15
Informed Search: Best First Search – Part I

So the goal for today is to go beyond the uninformed search algorithms and your thinking about it. In terms of the big picture, we realized that trying to solve a new problem is an extremely important part of intelligence. We wanted to develop general purpose algorithms, which can solve any kinds of problems and we realize that if it is a general purpose algorithm, it has to have a general purpose representation for the problem.

So we came up with this search problem. We said that many problems in the world can be formulated as a search problem. In this search problem, we will have a certain states in our actions, a start state and a goal state; the idea would be to find a path from the start state to the goal state. Of course, this whole search state space will not be given to us explicitly. It will be given to us implicitly by an expansion function.

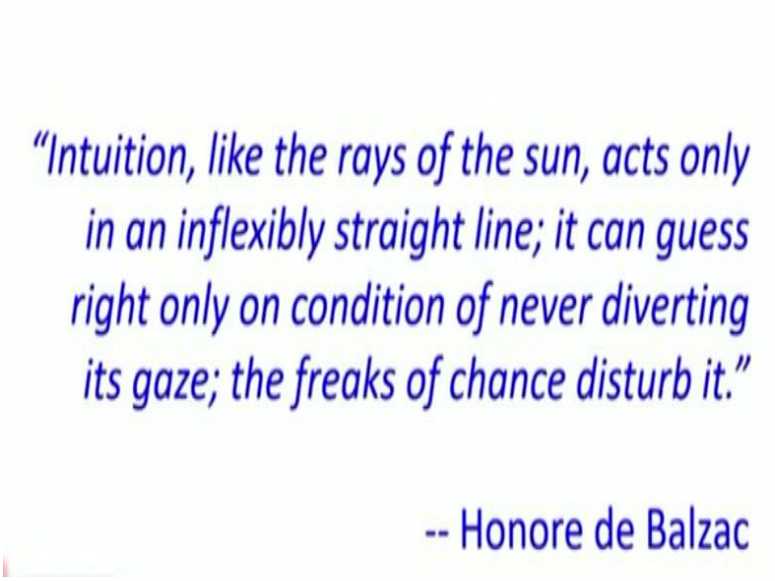
Then we said that suppose, this was the only input we were given, how will we solve it and we realized that we can solve it using a search algorithm. So in fact, if you ask a lot of traditional people; people who have been working in the AI for the last 50 years, 60 years, they will say that AI is search. Search is so important, so central to the field of AI that you can almost say that AI is search. If you can do search, right, you can solve all AI problems.

You can solve all problems of intelligence. They go as far as saying this. I am not going to go as far as saying that. We will also discuss other fields of AI where people say AI is that field and this is the mindset that different folks have and we respect all of them. But it is true that if we could come up with a general purpose search strategy, which can solve this general search problem in practice efficiently, then we would consider a lot of problems in the intelligence solved and that would be a great thing.

Now of course, the algorithms that we considered depth-first search, breadth-first search, iterative deepening search, beam search and bidirectional search. They are good algorithms, but even in the last part of the class we realized that they are not satisfying. They are looking in all directions. They are not guided towards the goal somehow or the other. We need to add this kind of guidance and this guidance that we are going to add will be what is called informed search.

That is the topic for today and we will study it in some depth, because it is an important topic in the field of AI.

(Refer Slide Time: 03:21)



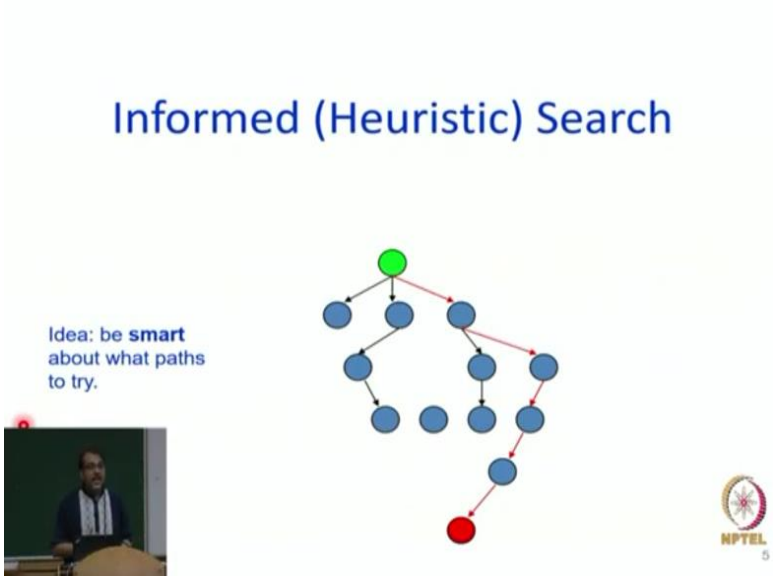
So we are going to talk about informed search algorithms and the main idea of informed search algorithms is that we need to add some intuition in our search algorithms. We do not want to be blindly looking in all directions, hoping that by mistake it reaches the goal. We would encourage it to go towards the goal. We would add intuition, but of course, intuition is like the rays of the sun that act only in inflexible straight line, as this great philosopher said, at least the person who wrote the quote said.

It can make some guesses and sometimes the guesses may be right, but sometimes, they may not be write and this you and I know from our daily experience that sometimes we have some intuition, which is really awesome, which really takes us towards the final solution that we were

looking for and sometimes our intuition confuses us. It takes us away from the final thing and so we need to take intuition as it is.

It is intuition, but not the solution and this is what we are going to somehow capture in the algorithm. So keep this at the back of your mind as we move.

(Refer Slide Time: 04:32)



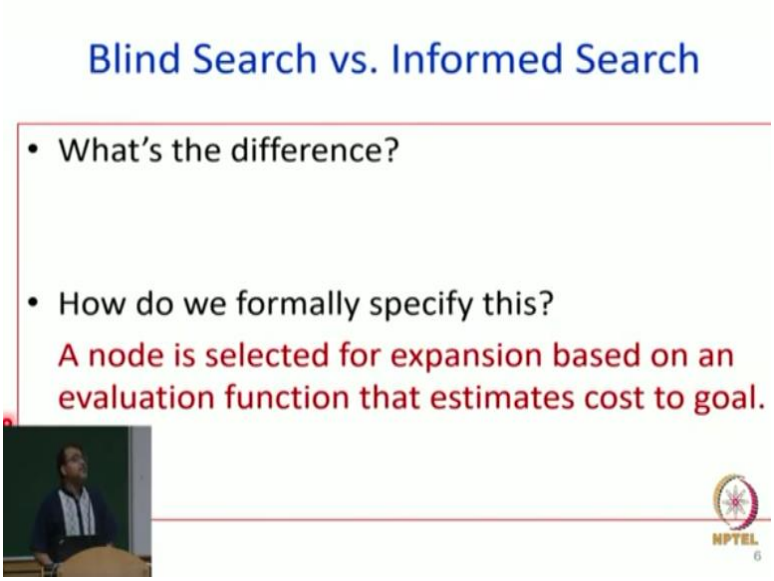
And the idea, however; as we drill down into the specific computational model is that we want to be smart about what paths we try and what paths we let go. So last time, I gave you the example of, if I want to go from where I am standing to that chair at the back, I do not want to be searching in all those noticeably or seemingly suboptimal places. I would not try to go back and try to search paths in the other direction.

I have some idea, of which direction I want to go in and I will start going in that direction and maybe it is possible that there is a new obstacle that I did not know about it, maybe that door is blocked by an invisible force and as soon as I try to go through this between these two chairs, I get a shock and I have to come back. It can happen in science fiction for example. So the intuition guided me towards the solution, but I cannot go to the next step.

So then I will have to think again, I will have to replan, I will have to go back, maybe the optimal path is to go behind it, go outside the door and maybe go somehow from the other door and what

not, it is possible, but I will not start out doing like that. I will start out trying to go towards the goal as much as possible. So I would be smart about what paths I want to try and what paths I want to avoid for now.

(Refer Slide Time: 05:55)



So as we said, the difference between blind search and informed search is that in informed search, we are giving some more information, but what information? What information could we give? That is the question. So we have a search node. The search node sort of tells you, gives you some paths to reach from the start stage, but is this a good search node or not. What would be a good search node, anybody, probability of success. Somebody says, what is your name?

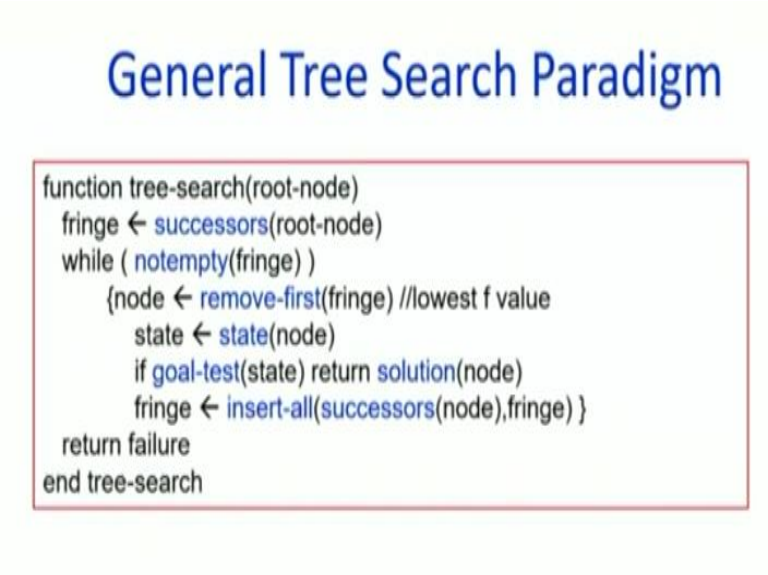
Subham says, probability of success, but there is no notion of probability in our world so far and that probability would be very hard to compute also. We would not know what are the various paths to the goal and fraction of the paths go through this particular node, etc., etc. That will be very hard. So we need something simpler. See I am not completing the probability of this node versus that node.

I am simply making an observation that this node is probably closer to the node. That is the intuition, I have. After planning, after full search, they may find it is not, that is a different story. For now, this node looks better to me than this node, because this node seemingly is closer to a goal and that node is farther and that is the intuition they want to somehow capture and we will

capture it via heuristic function and formally we will select a node to expand based on some evaluation function.

Now this is very generic concept at this point. Some evaluation function that sort of estimates the cost to the goal and if we do this, we can write a general purpose tree search paradigm.

(Refer Slide Time: 08:05)



Now this is a very general paradigm. In fact, we will shortly show that even if some of our old friends like breadth-first search is just a special case of three searches, so think about it, how? So this general tree search paradigm is, I have a fringe. Initially, the root is in the fringe. Fringe, frontier are same thing. Now while fringe is not empty, I take out some fringe node. Now it uses, remove first and in the comments, you can see it says lowest F value.

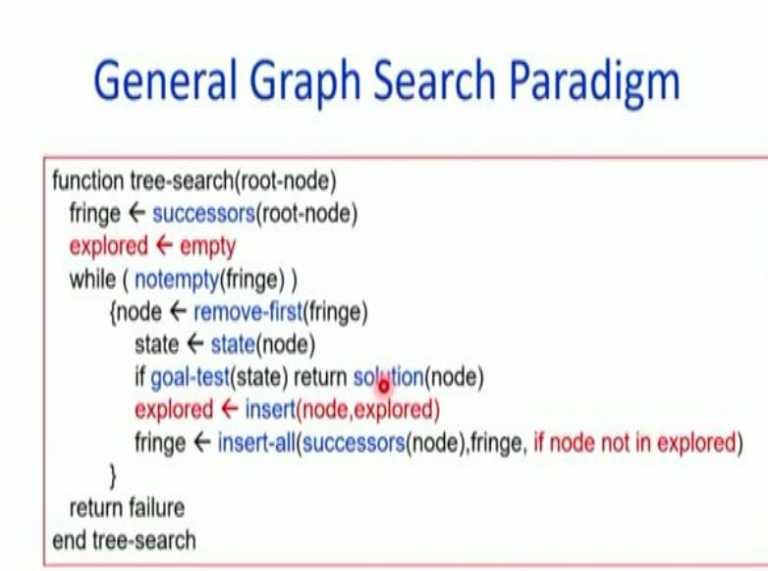
F here is my evaluation function. With every node I am associating some evaluation function and from the fringe, I am going to take out the best node that I can think about right now. The best node will be one, which has the lowest F value. Now everything else will be similar. Once I take this out, this is my current state. I am going to see if it is the goal. If it is the goal, I am done. if it is not the goal, then I will take all its successors and put it back into the fringe and repeat.

When I am adding all the successors, each successor will be a new node, because I am not doing any duplicate detection, because this is a tree search paradigm. This is a generic tree search

paradigm. Now we can also have an generic graph search paradigm, if I am doing full duplicate detection and in this graph search paradigm, there will be three main changes that are going to happen. In addition to the fringe, which is sometimes also called the open list, this is equivalent terms.

Open list, fringe, frontier, these are same things. The reason I introduced open list is because I want to introduce, guess? Yeah, if I am introducing open list, I have to also introduce a closed list. This is how we name our algorithms, name the parts of the algorithm. I work in this field for open information extraction, because rest of information extraction is closed information extraction. That is a joke, but sort of it.

(Refer Slide Time: 10:25)

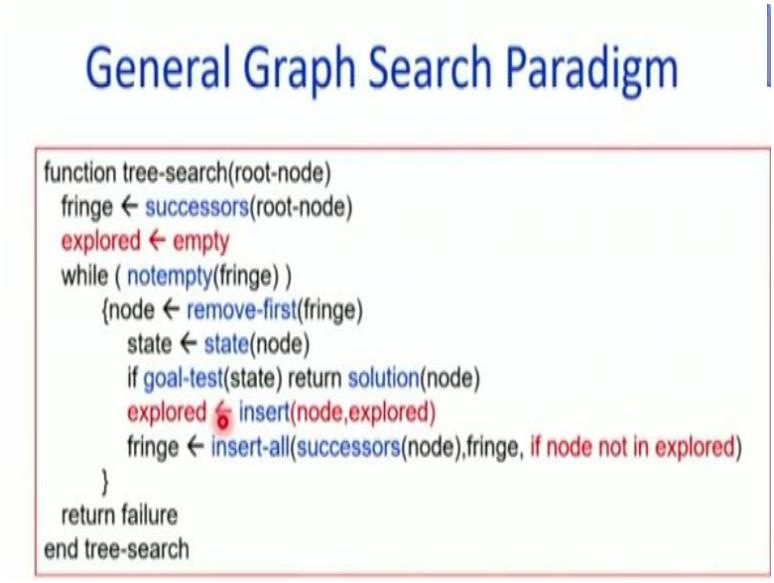


So we will have an explored list or the closed list, again explored and closed are sort of synonyms in this case. Basically, the point is that that particular node has been explored, so I have put it in my memory, so I can do duplicate check. So what will happen is, until fringe is not empty, I am going to remove the first node from the fringe. I will check whether it is a goal or not and if it is not a goal, I am going to add it in my closed list, in my explored list.

Then, when I am expanding the state to add all the new nodes in the fringe, I will double check, if the node is not already in the explored list, because if the node is already explored, then I will

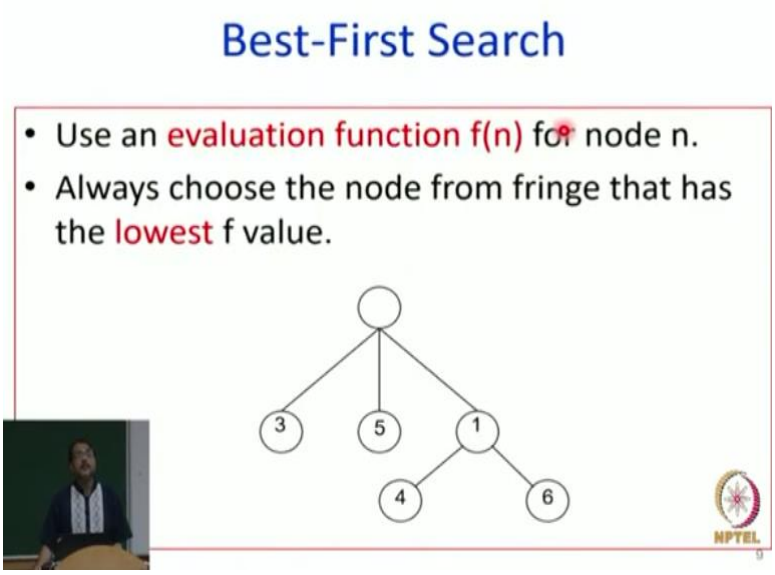
not add it. This is the full duplicate detection part and this is the generic graph search paradigm. Any questions on the tree search paradigm and the graph search paradigm?

(Refer Slide Time: 11:22)



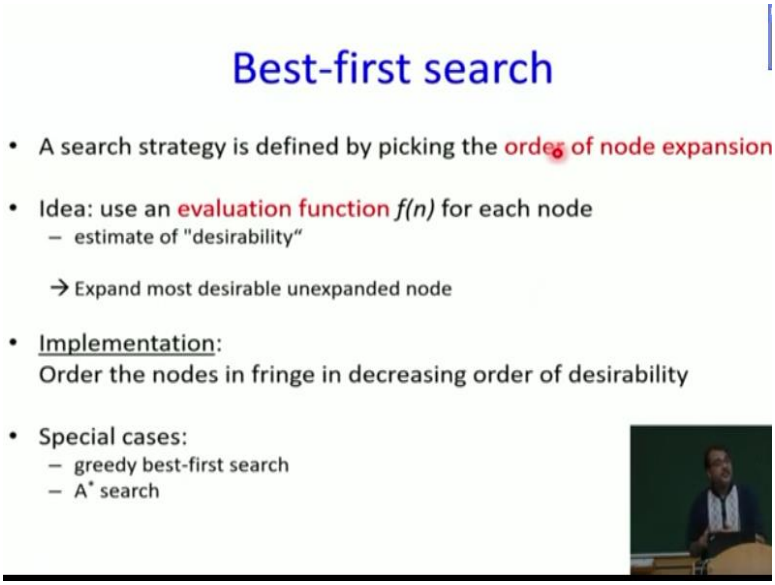
Another important thing to note and this is the mistake that a lot of students make in the first minor, is that, when do I know I have found my goal? When I add the node to the fringe or when I remove the node from the fringe? When I remove the node from the fringe. Be careful about it and will also see an example of why this is important.

(Refer Slide Time: 11:51)



So that leads us to our generic best-first search algorithm. Best-first algorithm uses an evaluation function f of n and always chooses the node from fringe that has the lowest f value. So let us think about what does this mean.

(Refer Slide Time: 12:11)



A search strategy is defined by the order of node expansion, depth-first, breadth-first, cost so far first, and that has been generalized to lowest f first. It is just a generalization, f is inverse of estimate of desirability. A low f is a highly desirable node. I have some hope that I will reach to the goal from this node and so I expand the most desirable node first, right. Now a quick implementation note.

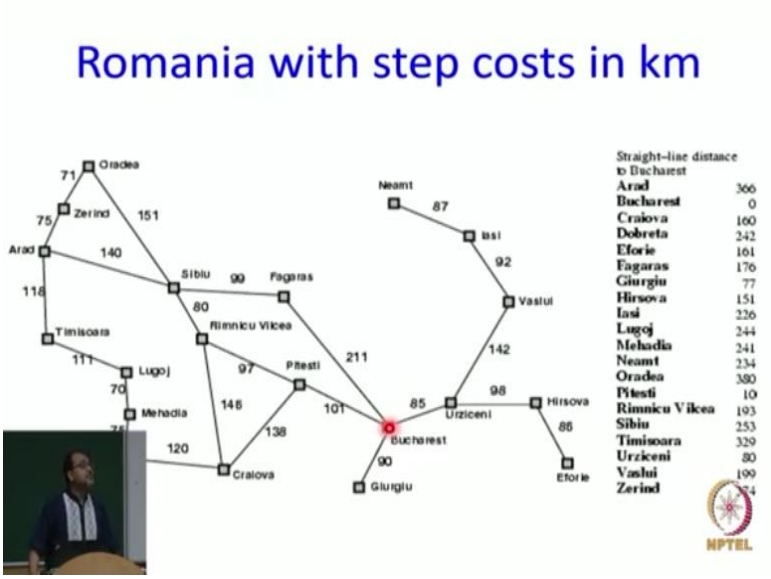
We have not been talking that much about implementations, but if you have to implement depth-first search, what data structure do we typically use to maintain all the nodes that are in consideration, stat, use a stat, why, because you have to back trap. Last in goes out first. This is something that you learn in your data structure class. If I am using breadth-first search, what data structure do I need?

I need a queue, because I am always exploring nodes and I am always adding nodes breadthwise, in other words, one depth all the nodes of the first depth are in my fringe and only then the next nodes of the second depth come in and because of which I can say, whatever came first, goes out

first. But if I want to do best-first search, with each node I have associated an f function. I have some nodes in the fringe and I want to take out the node with the lowest f value.

What data structure do I need? I need a priority queue a heap (()) (14:03) any kind of heap or priority queue is what I need, very good. So, great. Now the good news is that this best-first paradigm is one of the most generic paradigm in search, because not only are we going to study new algorithms like A star search today, but many of the older algorithms will become special cases of best-first. But before that, let us look at the running example.

(Refer Slide Time: 14:34)



Our running example is a map of Romania. We are on a vacation. We are currently in the city called Arad and our goal is to get to the city called Bucharest. I have a flight to catch in Bucharest. I am in my car. I am driving around and then I realized I have a flight to catch, so let us go back to Bucharest and catch the flight. Now this is a simpler problem, we can even use the Dijkstra's algorithm. For that part, this will be illustrative.

So what will be my state? My state would be the current city. The state would be the current city. What would be my actions? Drive between city 1 to city 2. What will be my goal test? I am in Bucharest. So far, so good. Now in addition to all these, because we are in the lecture for informed search. We have to think about some information, additional information that we can

add to my state, which tells us, how far this city might be from the goal, without actually doing the shortest path computation.

Because if I do the whole computation, then I have solved the problem. So some quick and dirty way to figure out which city is closer to Bucharest and which city is farther from Bucharest and what is the simple way to do this? Somebody has to be louder, I cannot hear. Yes, what is your name, Mohit. Yes, Mohit, no I am not looking for path, very good, but no. I am not asking you for the path and this is important.

Because to compute the path, I may end up doing a lot of work, which I am going to do after I use this heuristic function. I am only interested in a guess on the length of the path. Let us make sure that we understand this. My heuristic function is not asking me for the path. It is asking me for the guess on the path length and this guess on the path length in this case can be, yes. Let us just look at the coordinates of the two cities and take the distance between them, also known as the airplane distance or the closed flight distance or something like that.

So if you are a bird and if you wanted to go from Arad to Bucharest, the trip that you might take not accounting for any food breaks, is the distance. It is the guess and that is exactly what I use when I start moving in that direction. I am saying that this node is likely to be closer to my goal node than this node because my straight line distance seems to be better here. So that would be my heuristic function. The straight line distance to Bucharest would be my heuristic function.

Let us come back to this, but before that, I want to show you that best-first search is a general purpose algorithm. So first we will look at our old friends:

(Refer Slide Time: 18:14)

Like breadth-first uniform cost search show that they are like best first search with different f functions and then, use the heuristic function to device better algorithms. So first question for you, you just learn the best first search paradigm. Can you say that breadth first search is like best first search and if so, what is the f function? F function is the depth of the node, because if I have multiple nodes in my fringe, some of them at lower depth and some of them at higher depth, which node do I pick?

I pick the node from the lower depth and between the node from the lower depth, I actually do not have that much tie breaking, although if I use a queue, I can say first in first out, but it does not matter from the point of view of a breadth-first search. More interesting, is uniform cost search, think about this now. Is uniform cost search in instance of best first search and if so, what is the f function.

If I have many nodes in my uniform cost search, which node do we expand first? So let us say, I have start stage s, I have the goal state g and I have a node n. The node n that we pick is the node, which has the lowest cost from start to n. So uniform cost search says, find that node, which has the best or lowest sum of the edge cost from start to this node and by the way, I am going to define this as g of n. So we are going to use three functions today, f of n, g of n and h of n.

Let us make sure we understand the meaning of all three. F of n is the evaluation function that best-first search uses to figure out which fringe node to expand. g of n says, because with every node, I have a full path associated and explicitly associated and unique path. I will that a g of n is the cost from start state to this node and then I will have h of n and h of n is going to be my guess of the distance from or the cost from n to the closest road. Makes sense?

I want to make sure that you are comfortable with all three of these, because these are important. F of n says which is the best node overall cumulatively, g of n says which is the closer node from the start to right and h of n says how far is the goal from me.