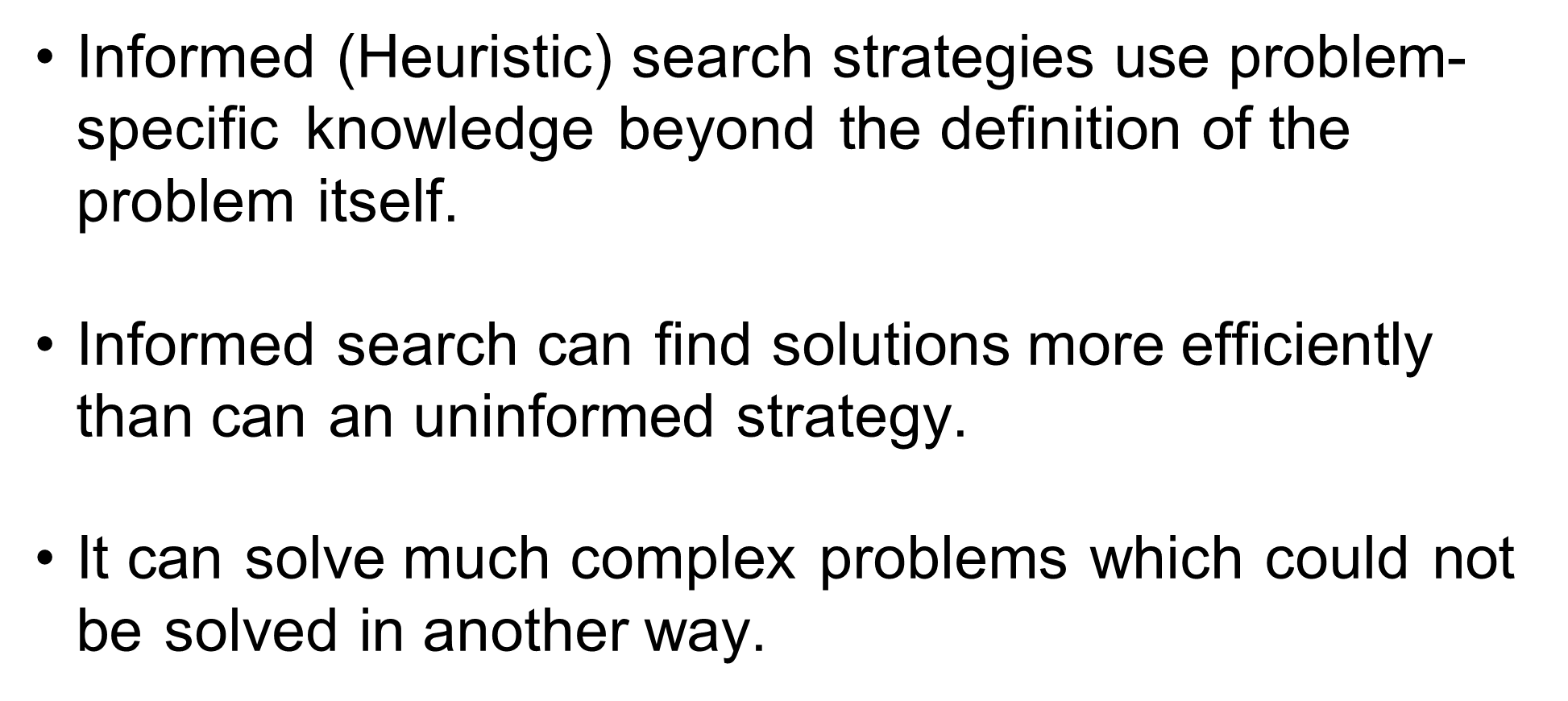
**LECTURE 4**

**SOLVE PROBLEMS BY SEARCHING INFORMED SEARCH**

**WHAT IS INFORMED SEARCH(HEURISTIC SEARCH)?**



**TYPES OF INFORMED SEARCH**

1)Best first search

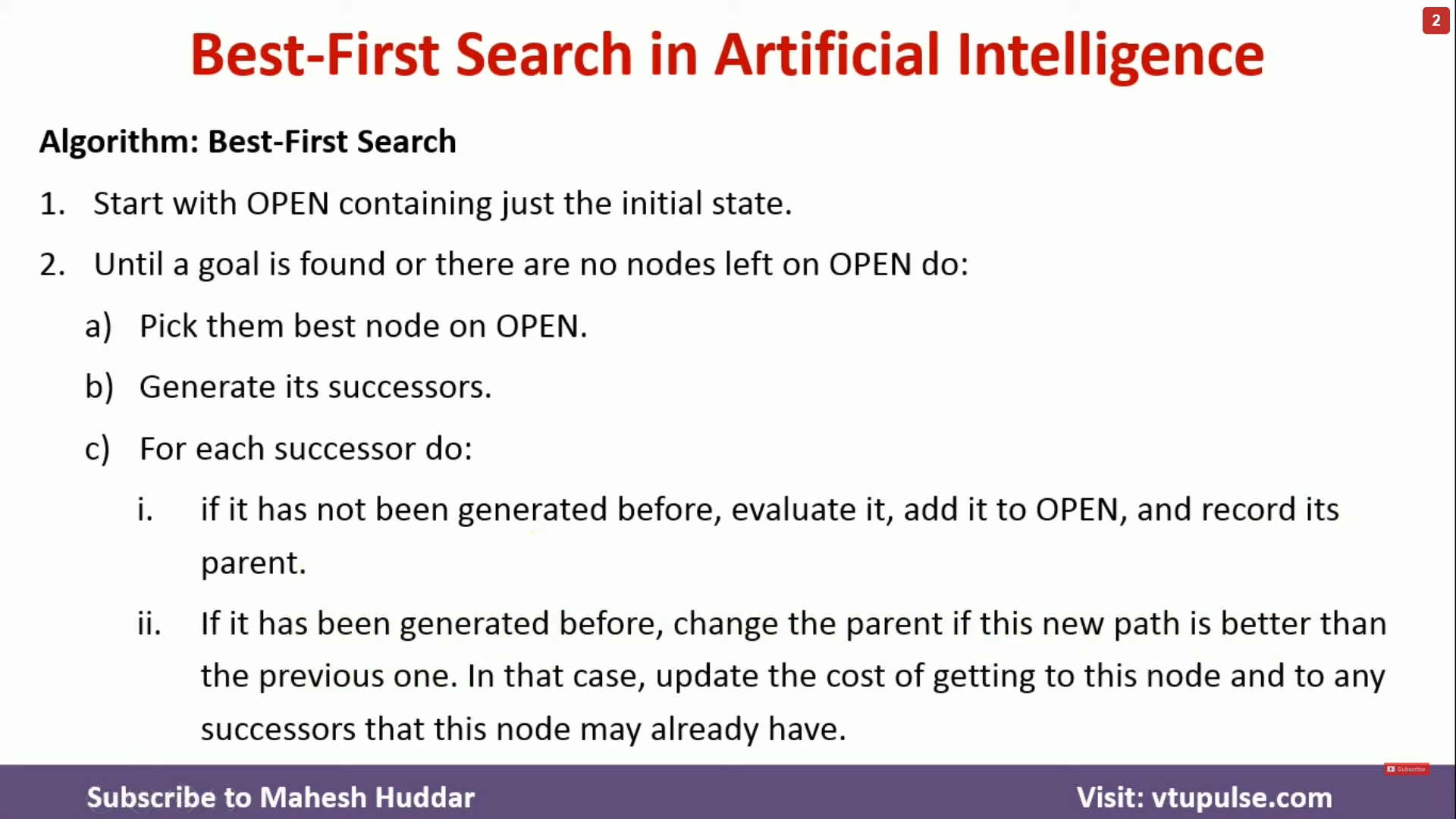
2) A\* Search

**BEST FIRST SEARCH**

* Best first search combines the advantages of both breadth first search and depth first search into a single method.
* It always selects the path which is best at that moment.
* At each step of the BFS search process, we select the most promising of the nodes generated so far.
* This is done by applying an appropriate heuristic function to each of them.
* We then expand the chosen node by using the rules to generate its successors.
* The greedy first algorithm is implemented by priority queue.
* It always selects the path which appears best at the moment.
* It is a combination of Depth First Search (DFS) and Breadth First Search (BFS).
* It uses the heuristic function h(n) <= h\*(n).
  + h(n) = Heuristic Cost
  + h\*(n) = Cost
  + If n is a goal node, h(n) =0

To implement such a graph procedure we will need two lists of nodes:

1. OPEN – nodes that have been generated and have had the heuristic function applied to them, but which have not yet been examined,(had there successors generated) .
2. COSE – nodes that have already been examined. We need to keep these nodes in the memory if we want to search a graph rather than a tree, whenever a new node is generated, we need to check whether it has been generated before



1)Place the starting node into the open list

2)If the open list is empty, stop and return failure.

3)Remove the node from the open list which has the lowest h(n) and places it in the closed list.

4)Expand the node n and generate the successors of node n

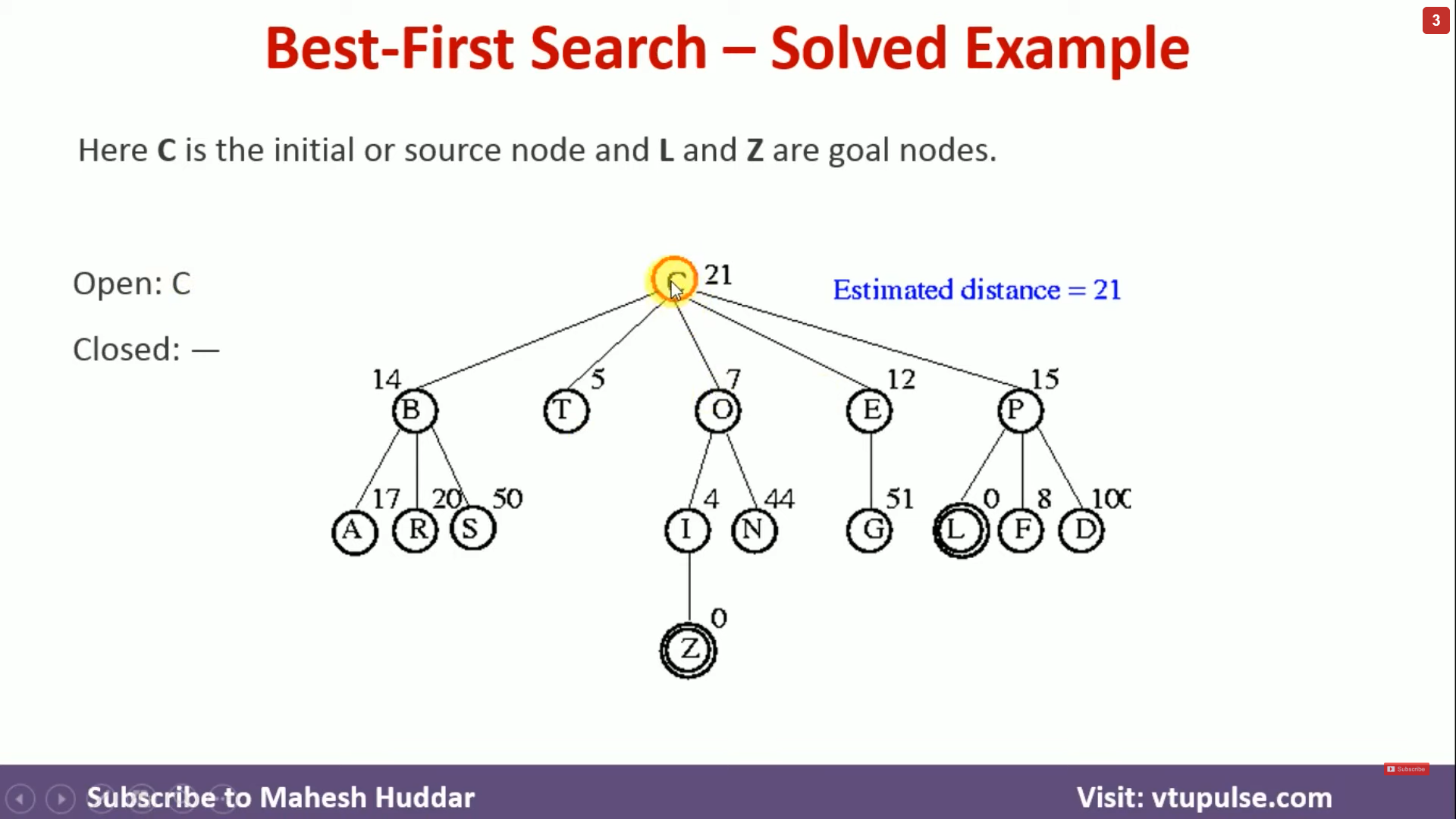
5)Checks each successors of node n and find whether the node is a goal node or not. If any success node is a goal node then return success and terminate the search .

6)For each successor node, algorithm checks for evaluation function f(n) and checks If node has been either open or closed list. If the node has not been in both list add It to open list

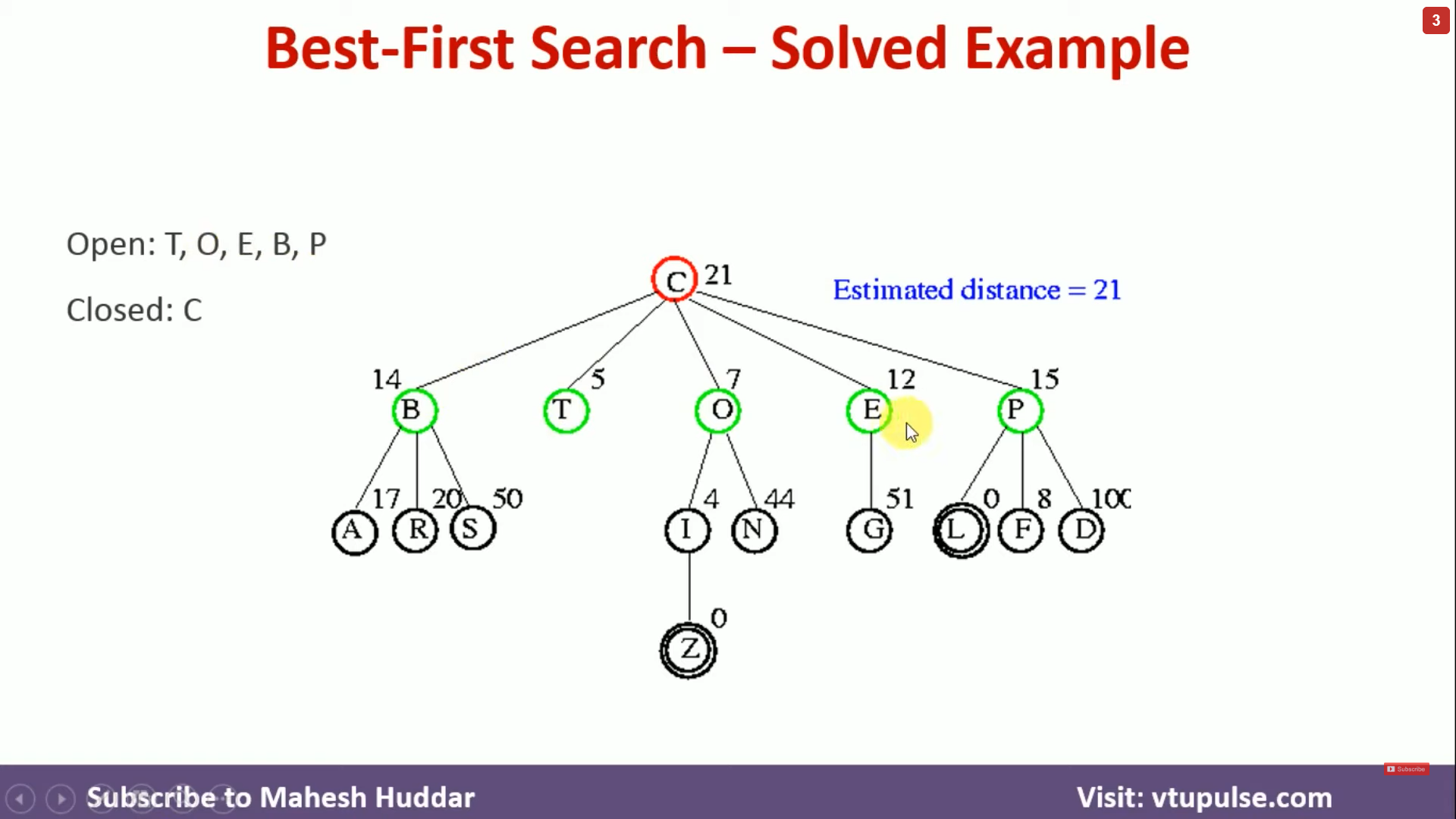
7) Return to step 2

**EXAMPLE**

**STEP 1**

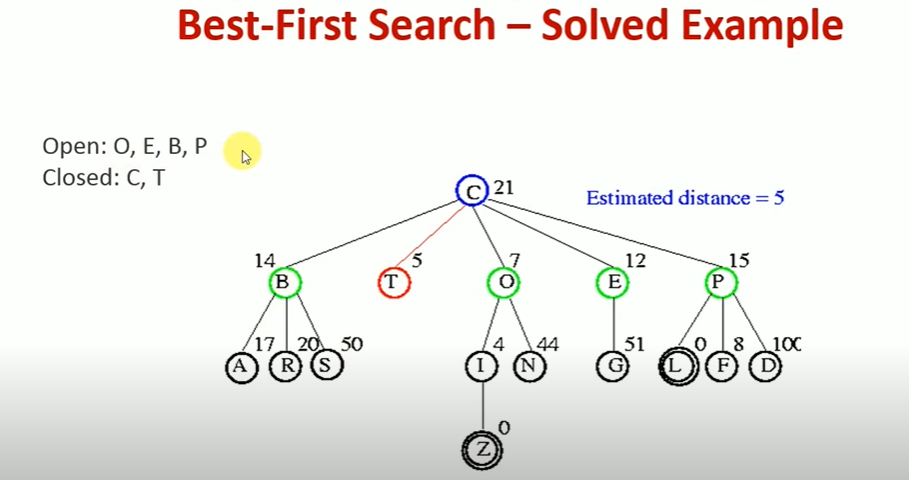


**STEP 2**



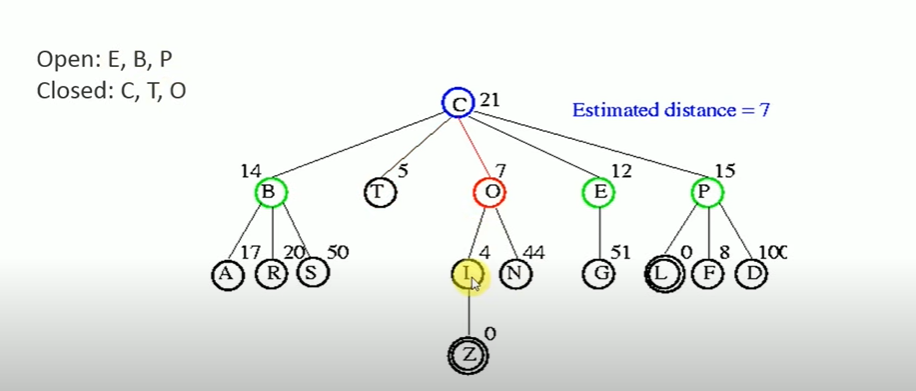
All the successors of C are B,T,O,E,P so C is sent to close and C are B,T,O,E,P are open

**STEP 3**

****

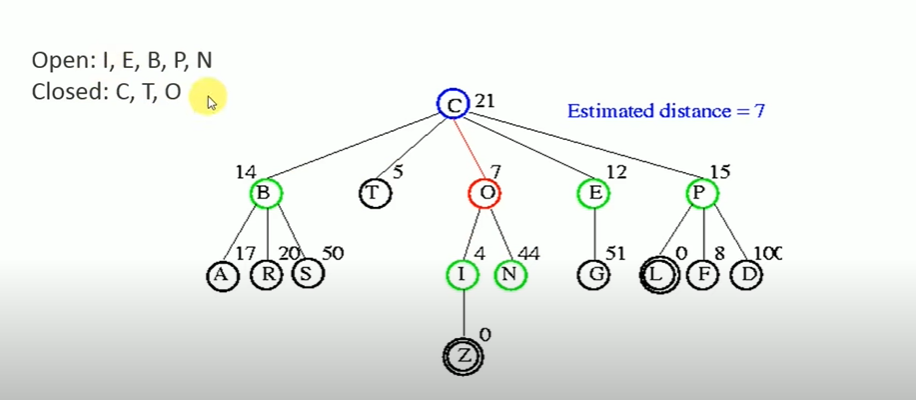
Next in the open list T is considered and checks for the successors T has no successors so there will be no successors to be added to Open

**STEP 4**



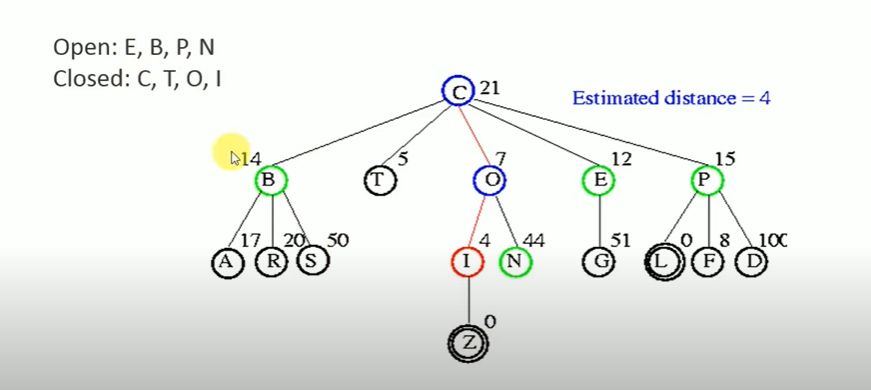
After T is O so O is added to close

**STEP 5**



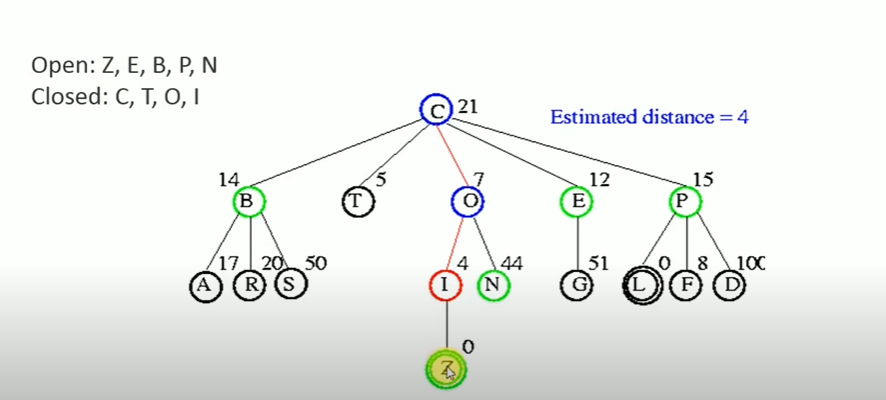
Checks for its successors it has 2 successors L and N but L has a minimum distance of 5 so its added to the beginning and N has a distance of 44 so its added to the end of the list.

**STEP 6**



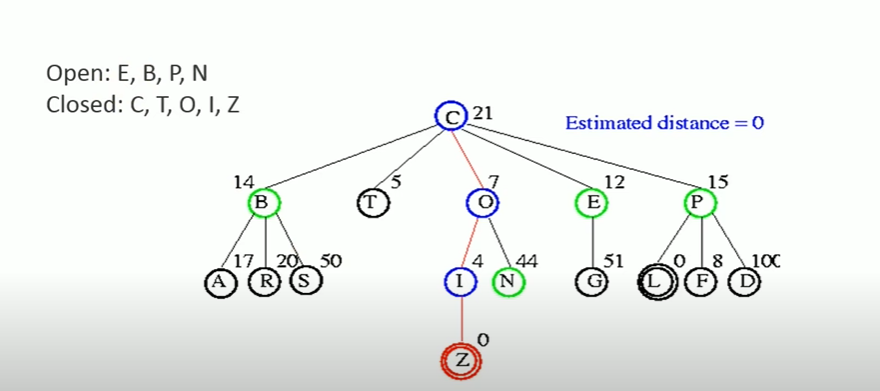
In the open list I is the first one so its added to close and checks for successors.

**STEP 7**



I has one successor which is Z so its added to the beginning of the open list as it has a distance of 0.

**STEP 8**

****

Then Z is considered and added to the close list since Z Is the goal node, the algorithm is stopped.

FINAL PATH IS C – O – I – Z

**ADVANTAGES**

Best first search can switch between both BFS and DFS by gaining advantages of both algorithms

The algorithm is more efficient than BFS and DFS.

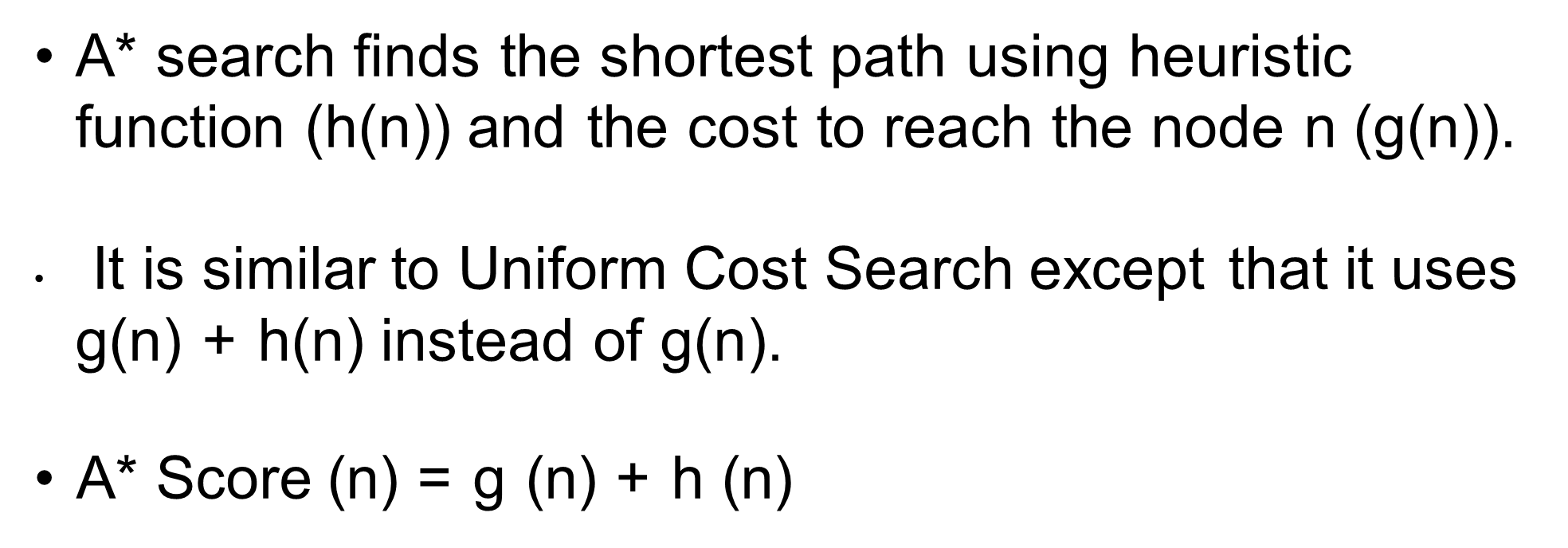
**DISADVANTAGES**

It can behave as an unguided depth list search in the worst case scenario.

It can get stuck In a loop as DFS

This algorithm is not optimal

**A\* SEARCH**



* A\* search finds the shortest path through search space using the heuristic function.
* It uses h(n) and cost to reach the node n from the start stage that is g(n)
* This algorithm expands less search tree and provides optimal results faster.
* It is similar to UCS except that it uses g(n) + h(n) instead of g(n).
* A\* use search heuristic search as well as the cost to search the node. Hence we combine both costs as

F(n) = g(n) = h(n) {fitness number}

F(n) = estimated cost of the cheapest solution

g(n) = cost to search node n from start state

h(n) = cost to search from node n to goal node.

ALGORITHM OF A\* SEARCH  
1) Place the starting node in the open list.

2) Checks if the open list is empty or not, if the list is empty then returns failure and stop.

3) Select the node from the open list which has the smallest value of evaluation (g + n), if node is a goal node, it returns success and stops otherwise

4) expands node n and generate all of its successors, and put n in the closed list.

* For each successors n check whether n is already in the open and closed list.
* If not then complete evaluation function for n and place into open list.

5) Else if node n is already in open and closed list then should be attached to the back pointer which reflects the lowest g(n) value

6) Return to step 2

**ADVANTAGES**

It is the best algorithm than other search algorithm.

It is optimal and complete.

It can solve complex problems

**DISADVANTAGES**

It does not always produce shortest path.

It is not practical for various large scale problems.

**Applications of A \* Search**

**Real world applications**:

* It can be used as a path finding algorithm for map based applications.
* String searching applications can also use this by determining the goal state.NLP uses this to check any parsing errors.
* A lot of games use this algorithm for its positioning system.

