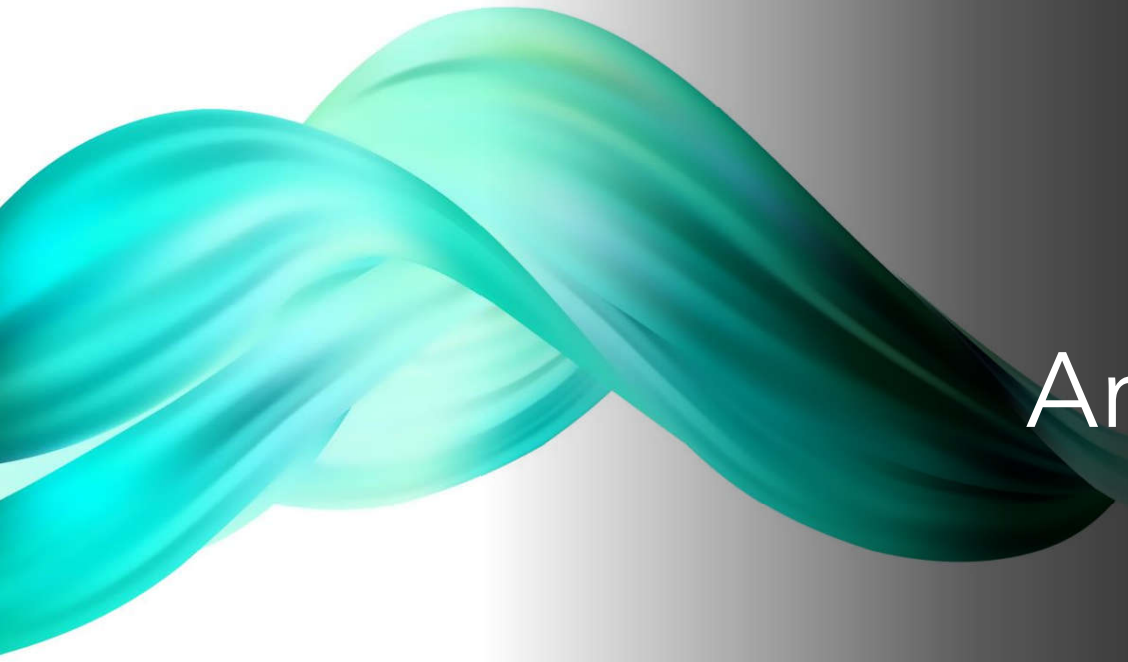


Artificial Intelligence

Dr. Mubashir Ahmad (Ph.D.)

Introduction To The Heuristic Function In AI

- Heuristics is a method of problem-solving where the goal is to come up with a workable solution in a feasible amount of time. Heuristic techniques strive for a rapid solution that stays within an appropriate accuracy range rather than a perfect solution.
- When it seems impossible to tackle a specific problem with a step-by-step approach, heuristics are utilized in AI (artificial intelligence) and ML (machine learning). Heuristic functions in AI prioritize speed above accuracy; hence they are frequently paired with optimization techniques to provide better outcomes.

The Heuristic Function In AI

- If there are no specific answers to a problem or the time required to find one is too great, a heuristic function is used to solve the problem.
- The aim is to find a quicker or more approximate answer, even if it is not ideal.
- A heuristic is a function that determines how near a state is to the desired state.
- The majority of AI problems revolve around a large amount of information, data, and constraints, and the task is to find a way to reach the goal state.
- The heuristics function in this situation informs us of the proximity to the desired condition.

The Heuristic Function In AI

- The distance formula is an excellent option if one needed a heuristic function to assess how close a location in a two-dimensional space was to the objective point.

Properties of a Heuristic Search Algorithm

- Admissible Condition: If an algorithm produces an optimal result, it is considered admissible.
- Completeness: If an algorithm ends with a solution, it is considered complete.
- Optimality Property: If an algorithm is thorough, allowable, and dominates the other algorithms, that'll be the optimal one and will unquestionably produce an optimal result.

Direct Heuristic Search Techniques

- Direct heuristic search techniques may also be called blind control strategy, blind search, and uninformed search.
- They utilize an arbitrary sequencing of operations and look for a solution throughout the entire state space. These include Depth First Search (DFS) and Breadth First Search (BFS).

Weak Heuristic Techniques

- Weak heuristic techniques are known as a Heuristic control strategy, informed search, and Heuristic search. These are successful when used effectively on the appropriate tasks and typically require domain-specific knowledge.
- To explore and expand, users require additional information to compute preferences across child nodes. A heuristic function is connected to each node.
- A* Search
- Best-first search

Best Fist Search Algorithm

- Best first search (BFS) is a search algorithm that functions at a particular rule and uses a priority queue and heuristic search. It is ideal for computers to evaluate the appropriate and shortest path through a maze of possibilities. Suppose you get stuck in a big maze and do not know how and where to exit quickly. Here, the **best first search in AI** aids your system program to evaluate and choose the right path at every succeeding step to reach the goal as quickly as possible.

A* Search Algorithm

1. Start with OPEN containing only the initial node. Set that node's **g** value to **0**, its **h'** value to whatever it is, and its **f'** value to **h' + 0**, or **h'**. Set CLOSED to the empty list.
2. Until a goal node is found, repeat the following procedure: If there are no nodes on OPEN. report failure. Otherwise, pick the node on OPEN with the lowest **f'** value. Call it BESTNODE. Remove it from OPEN. Place it on CLOSED. See if BESTNODE is a goal node. If so, exit and report a solution. Otherwise, generate the successors of BESTNODE. For each such SUCCESSOR, do the following:
   a) Set SUCCESSOR to point back to BESTNODE. These backwards links will make it possible to recover the path once a solution is found.
   b) Compute g(SUCCESSOR) = g(BESTNODE) + the cost of getting from BESTNODE to SUCCESSOR.
   c) Compute f'(SUCCESSOR) = g(SUCCESSOR) + h'(SUCCESSOR)
   d) if SUCCESSOR was not already on either OPEN or CLOSED, then put it on OPEN, and add it to the list of BESTNODE's successors.

Best Fist Search Algorithm

- While using the best first search, your system always seeks possible nodes or paths that can be taken. Then, it picks the most promising or best node or path that is eligible to traverse the shortest distance node or path to reach the goal and exit the maze.

Greedy Best First Search

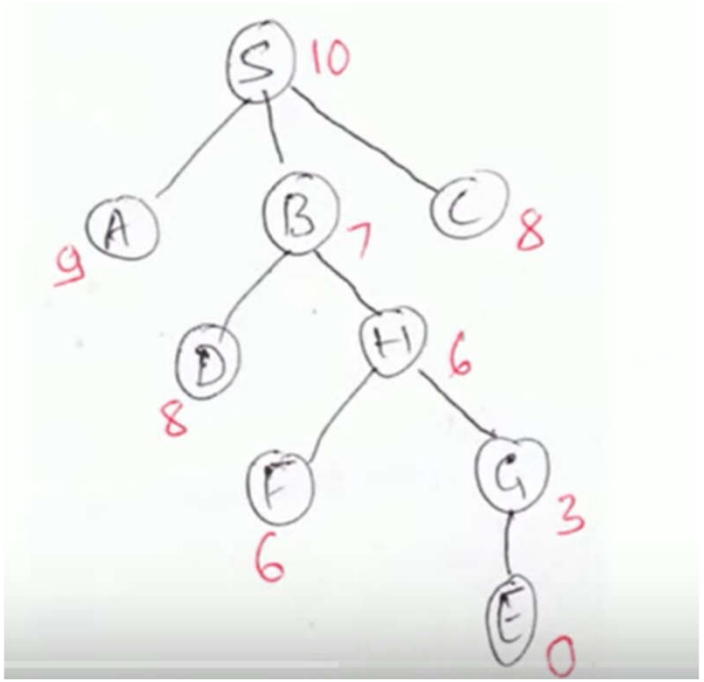
- Going with the name, this search algorithm is greedy and hence chooses the best path available at the moment. It uses a heuristic function and search, which is combined with depth and breadth-first search algorithms and combines the two algorithms where the most promising node is chosen while expanding the node present in proximity to the goal node.

Best First Search Algorithm

- Create 2 empty lists: OPEN and CLOSED
- Start from the initial node (say N) and put it in the 'ordered' OPEN list
- Repeat the next steps until the GOAL node is reached OR OPEN list is empty
  1. Select the best node (say N) in the OPEN list and move it to the CLOSED list. Also, capture the information of the parent node
  2. If N is a GOAL node. then exit the loop returning 'True'. The solution can be found by
  3. If N is not the GOAL node, expand node N to generate the 'immediate' next nodes linked to node N and add all those to the OPEN list
  4. Reorder the nodes in the OPEN list in ascending order

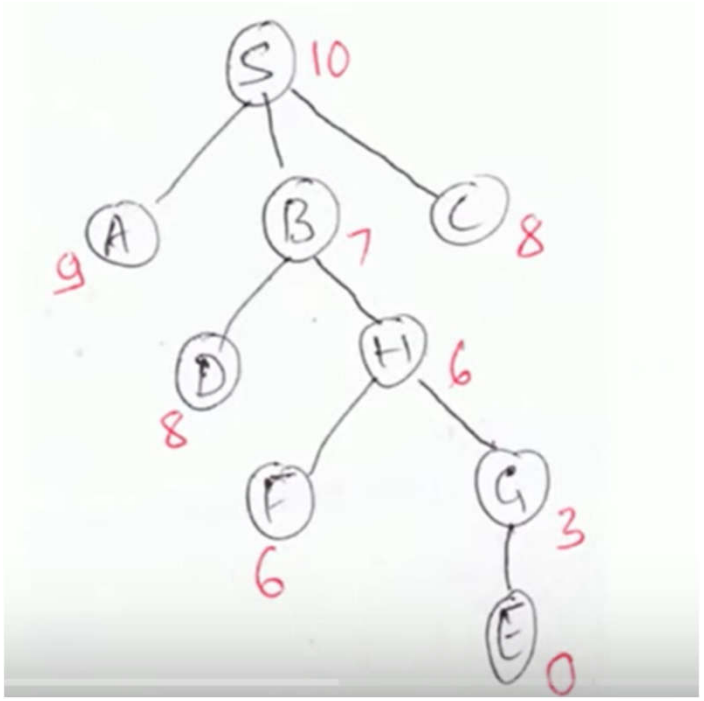
Best First Search Algorithm Solved Example

Open			Closed	
Node	H(n)		Node	Parent
S	10			



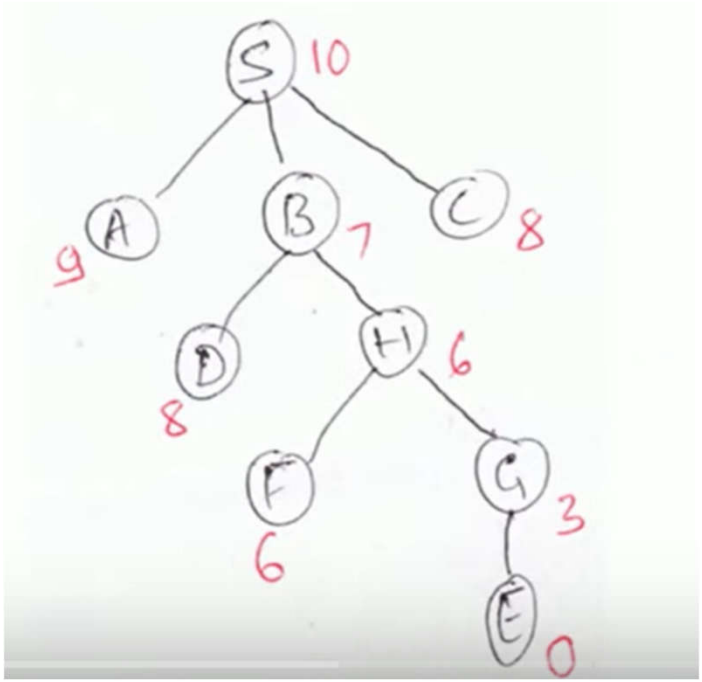
Best First Search Algorithm Solved Example

Open			Closed	
Node	H(n)		Node	Parent
A	9		S	
B	7			
C	8			



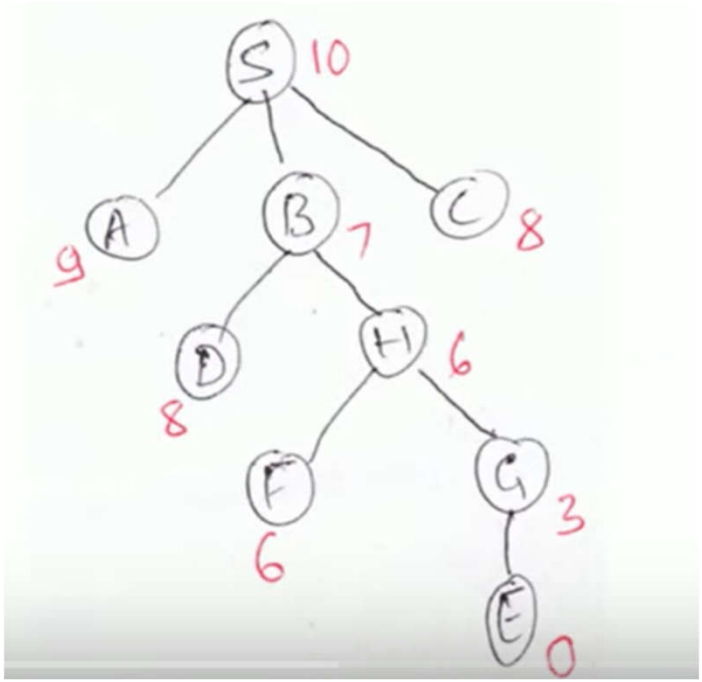
Best First Search Algorithm Solved Example

Open			Closed	
Node	H(n)		Node	Parent
B	7		S	
C	8			
A	9			



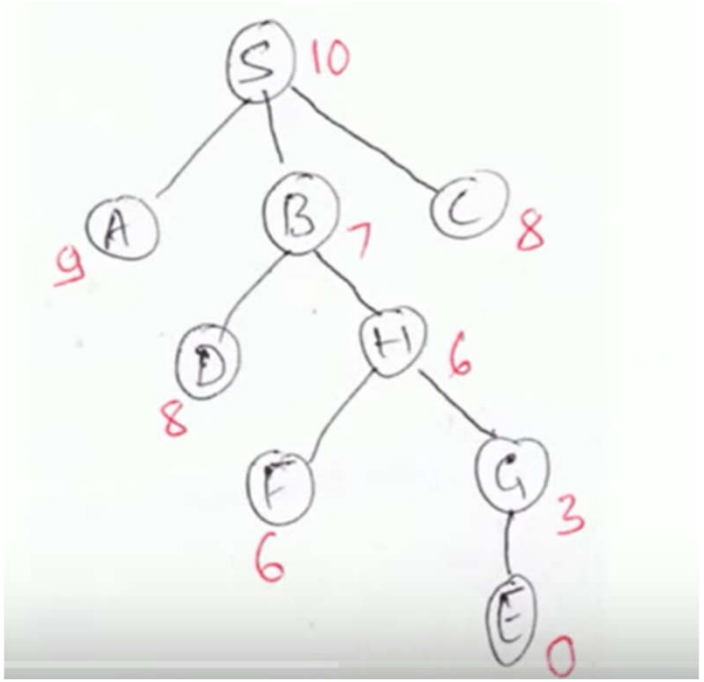
Best First Search Algorithm Solved Example

Open			Closed	
Node	H(n)		Node	Parent
C	8		S	
A	9		B	S
D	8			
H	6			



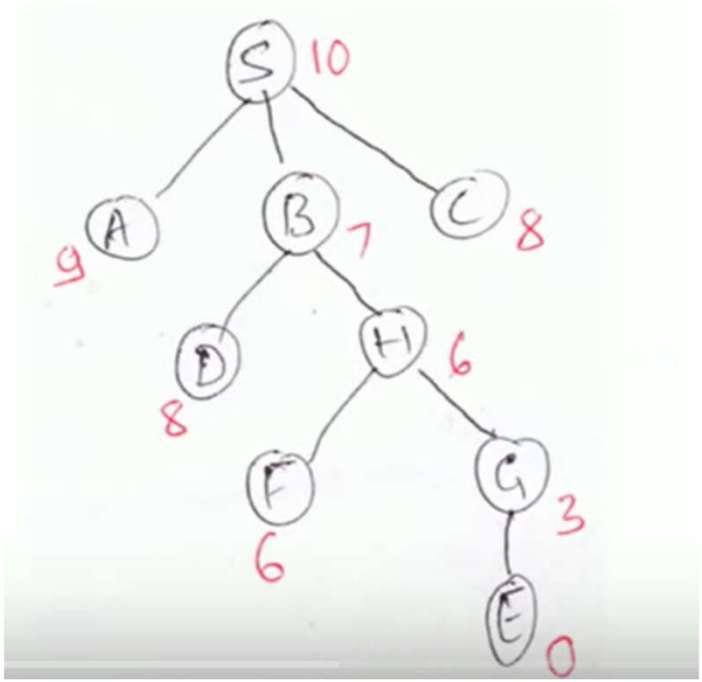
Best First Search Algorithm Solved Example

Open			Closed	
Node	H(n)		Node	Parent
H	6		S	
C	8		B	S
D	8			
A	9			



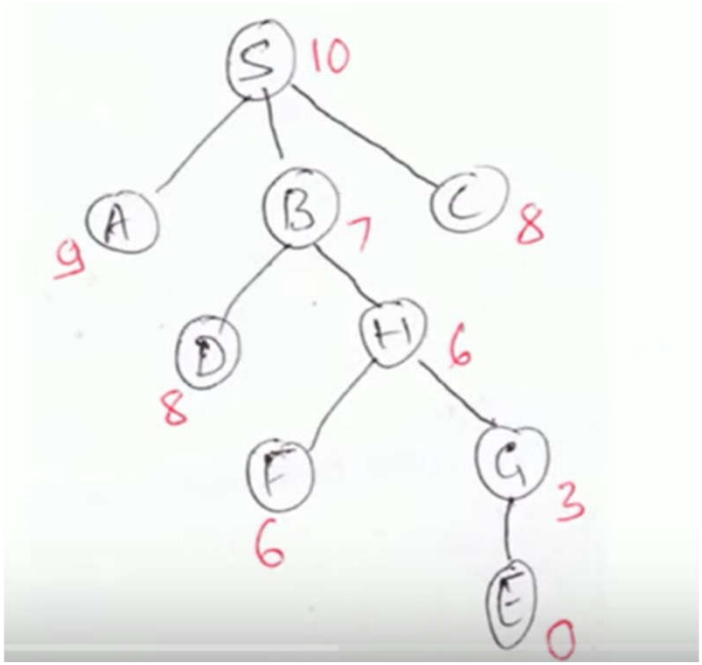
Best First Search Algorithm Solved Example

Open			Closed	
Node	H(n)		Node	Parent
C	8		S	
D	8		B	S
A	9		H	B
F	6			
G	3			



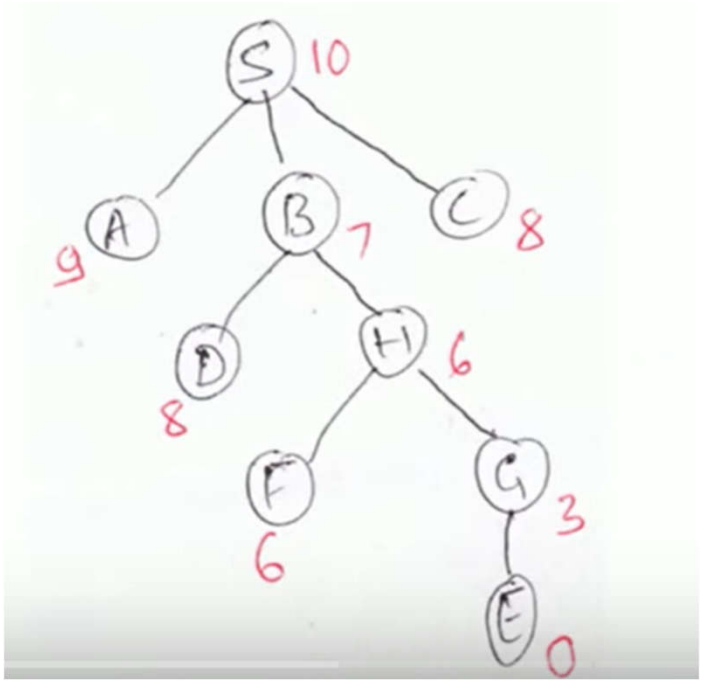
Best First Search Algorithm Solved Example

Open			Closed	
Node	H(n)		Node	Parent
G	3		S	
F	6		B	S
C	8		H	B
D	8			
A	9			



Best First Search Algorithm Solved Example

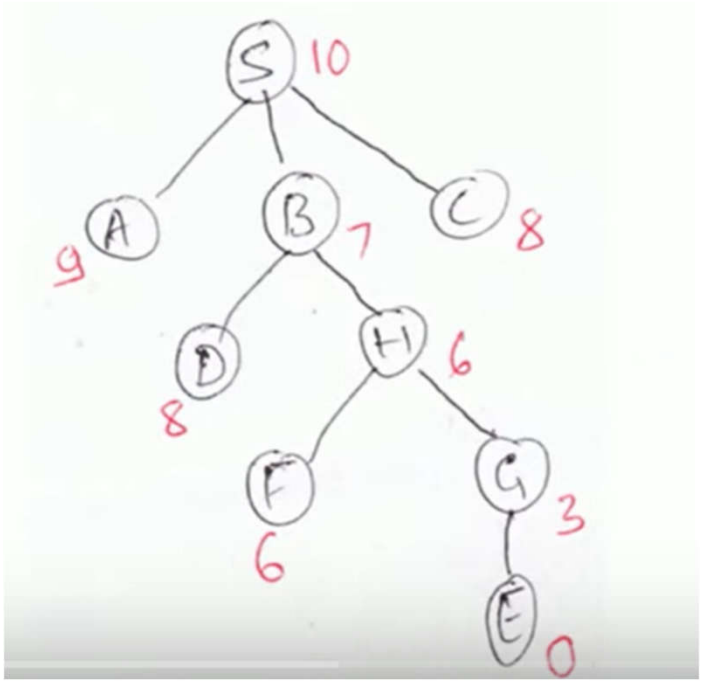
Open			Closed	
Node	H(n)		Node	Parent
F	6		S	
C	8		B	S
D	8		H	B
A	9		G	H
E	0			



Best First Search Algorithm Solved Example

Open			Closed	
Node	H(n)		Node	Parent
F	6		S	
C	8		B	S
D	8		H	B
A	9		G	H
			E	G

S → B → H → G → E

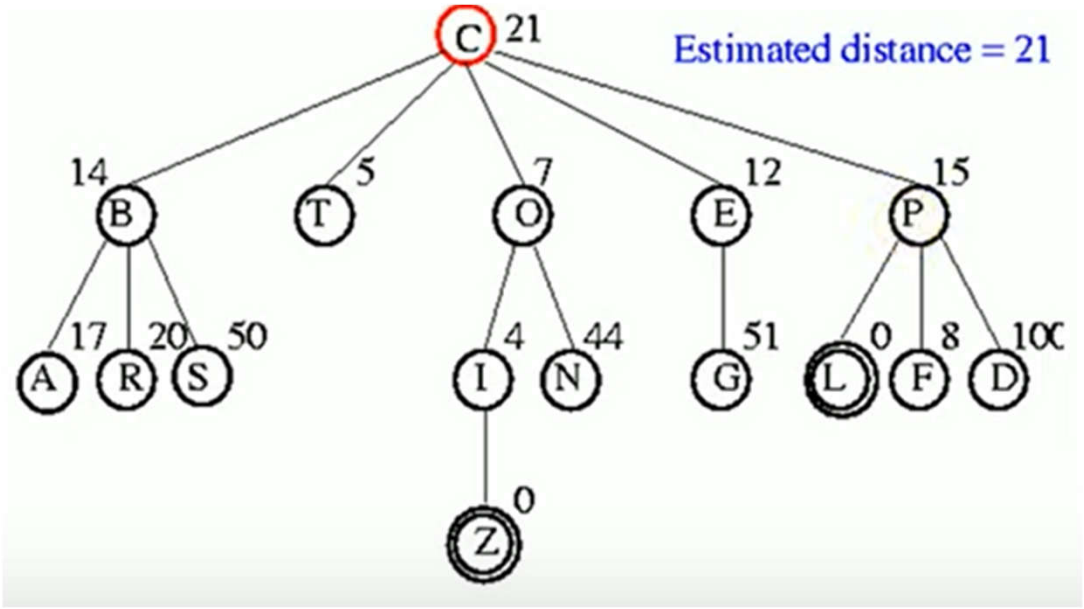


Best First Search Algorithm Solved Example

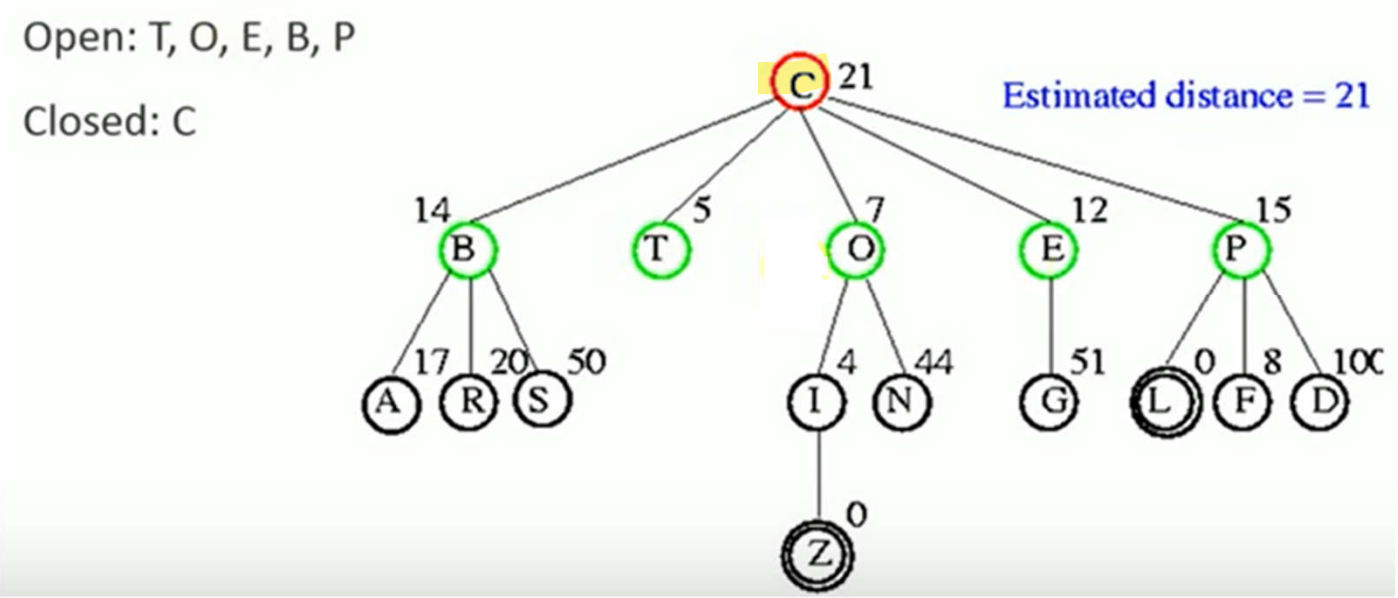
- Best First Search algorithm to find an optimal path from Source Node to Goal Node.

Best First Search Algorithm Example 2

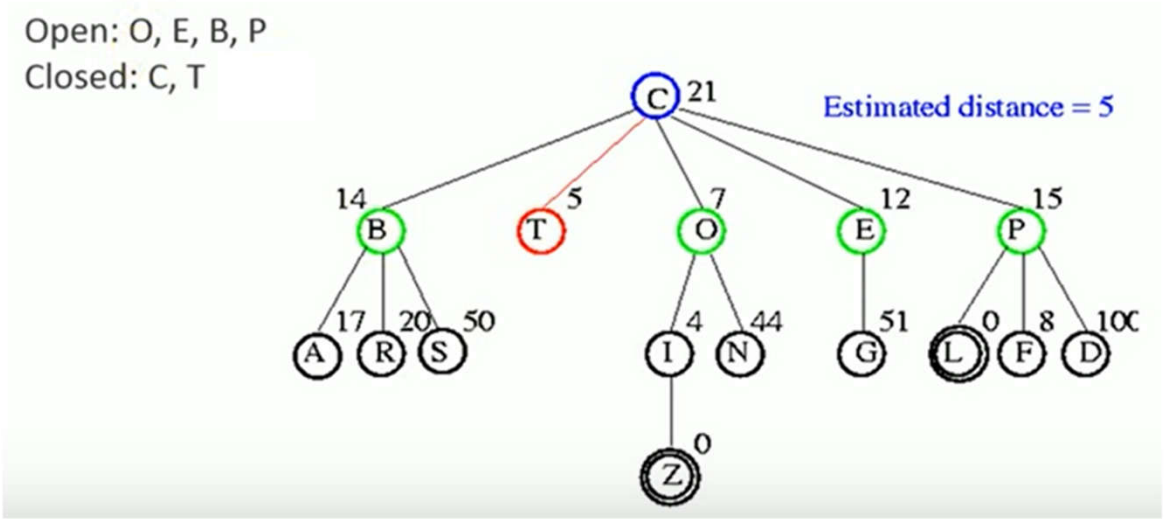
Here **C** is the initial or source node and **L** and **Z** are goal nodes.



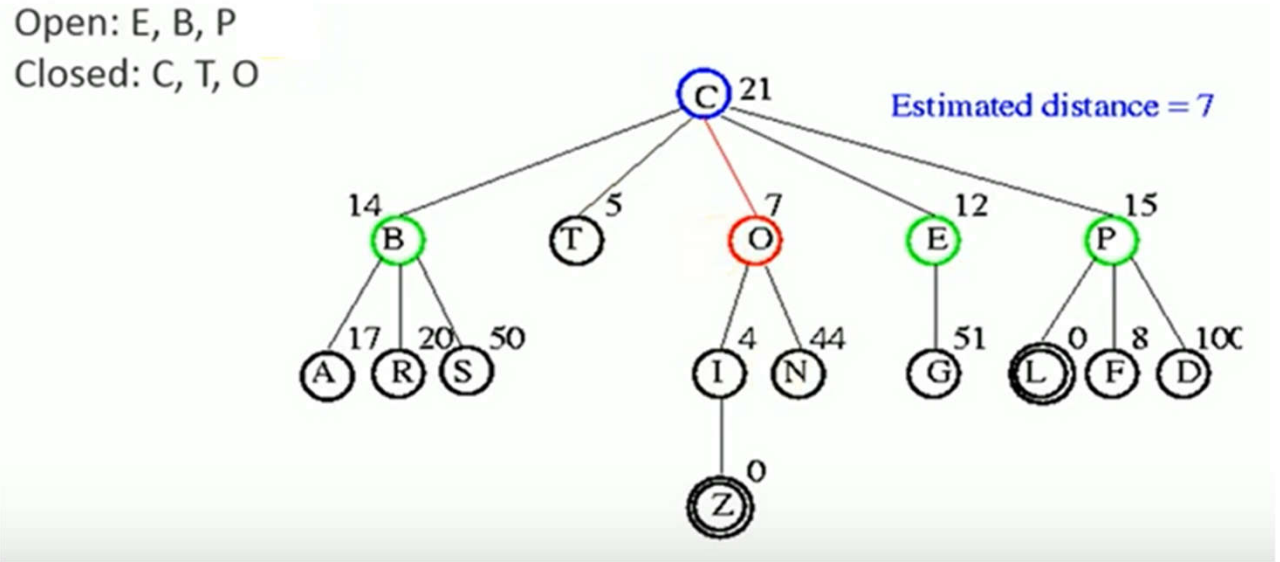
Best First Search Algorithm Example 2



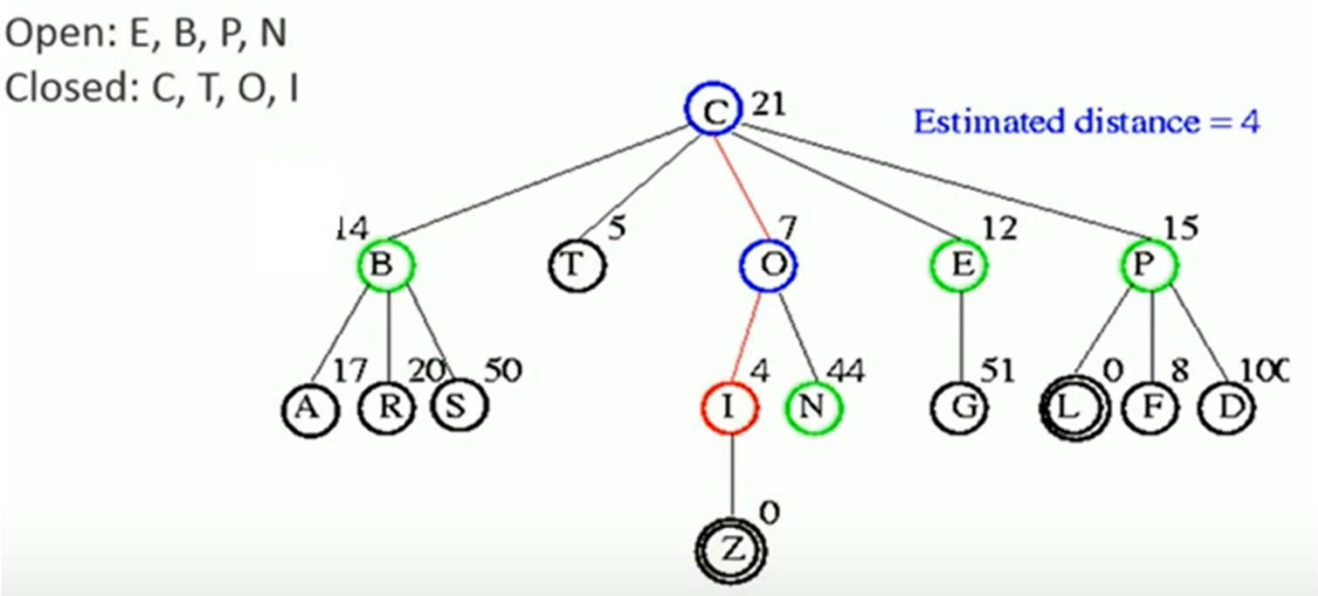
Best First Search Algorithm Example 2



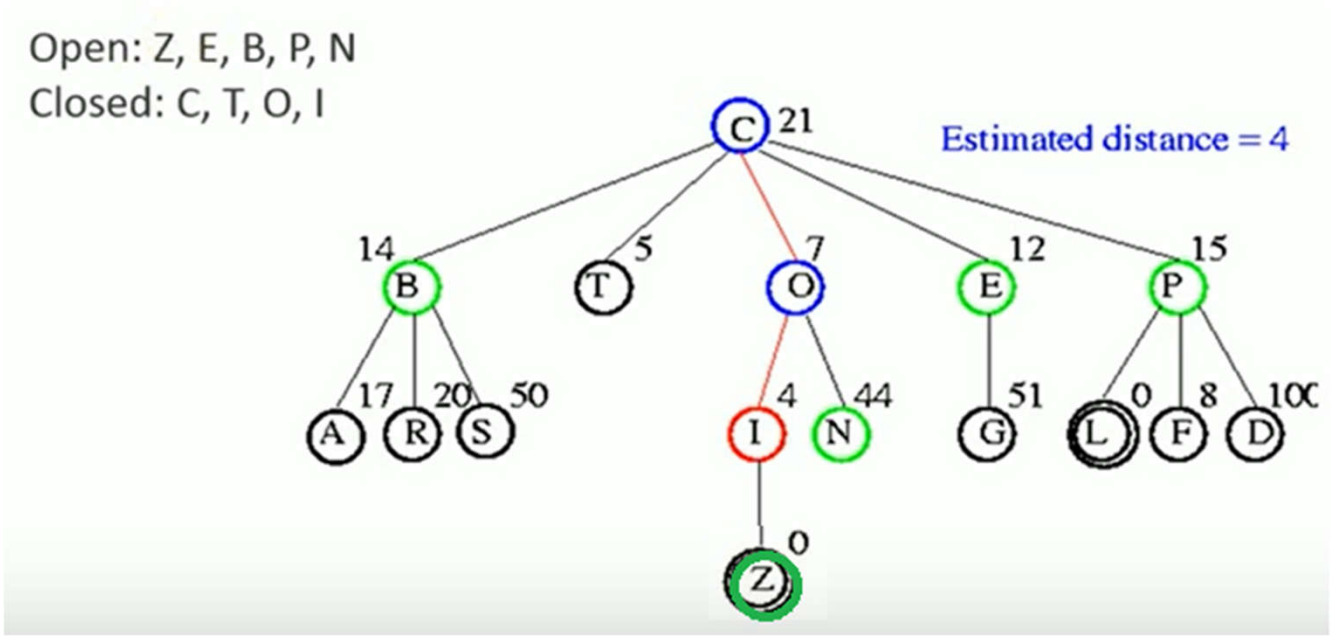
Best First Search Algorithm Example 2



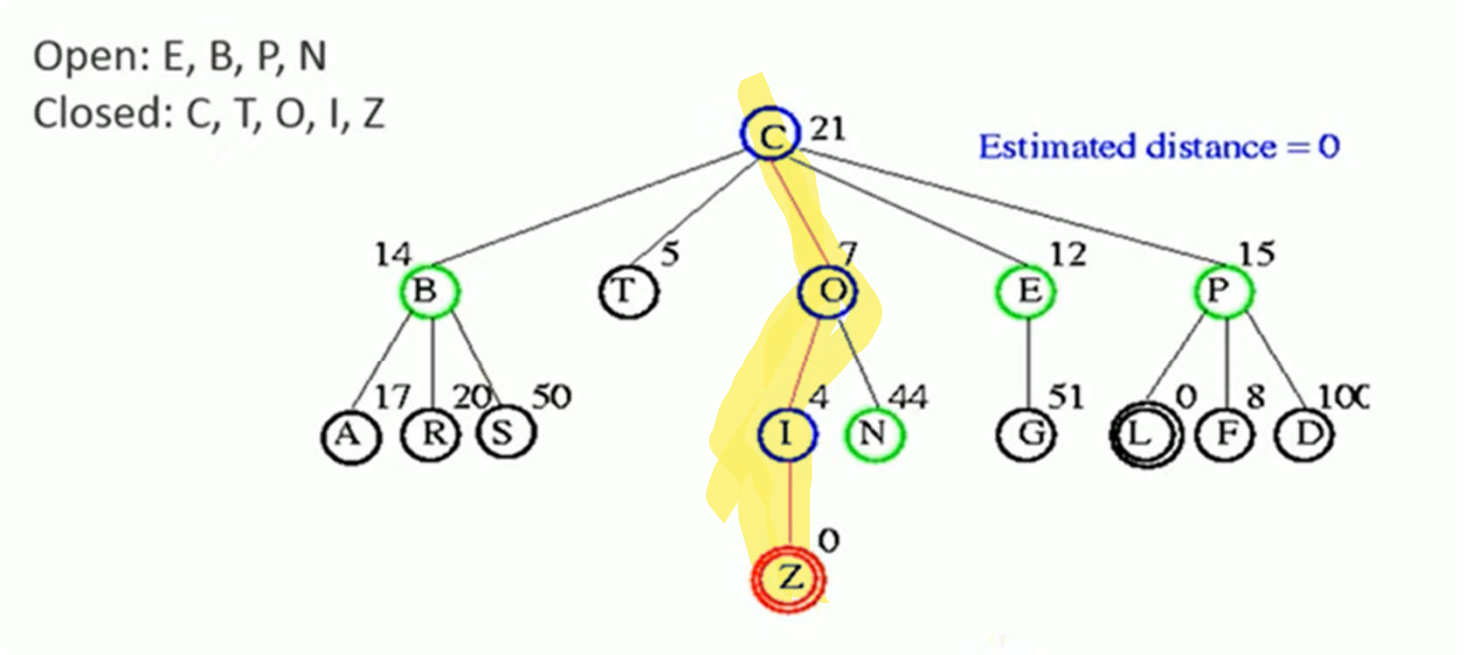
Best First Search Algorithm Example 2



Best First Search Algorithm Example 2



Best First Search Algorithm Example 2



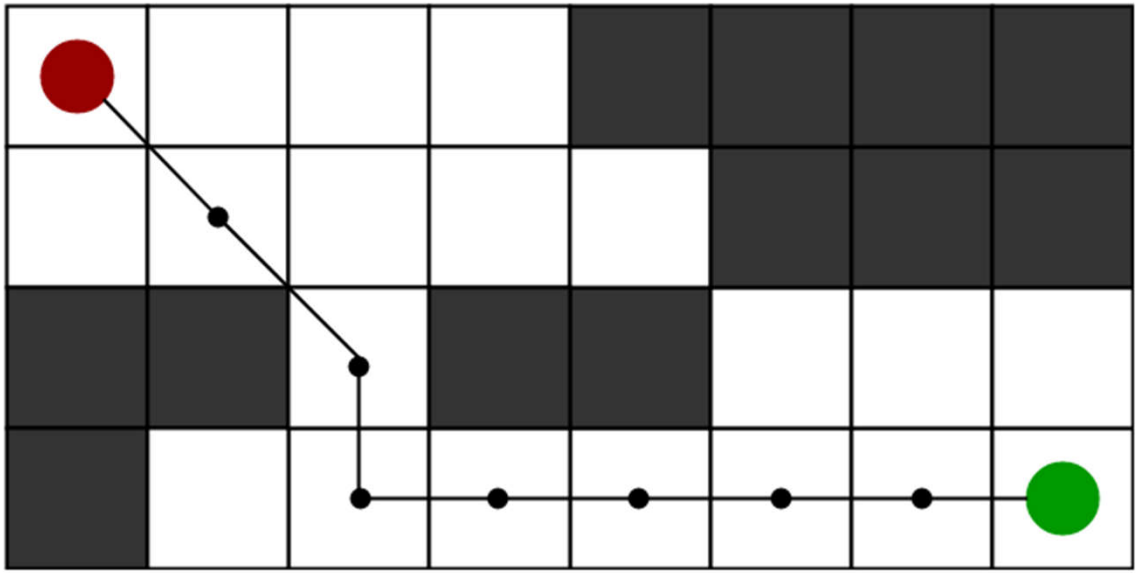
The Goal is found. The final path is C – O – I – Z

Best Fist Search Algorithm

- The **best first search in artificial intelligence** is an informed search that utilizes an evaluation function to opt for the promising node among the numerous available nodes before switching (transverse) to the next node. The **best first search algorithm in AI** utilizes two lists of monitoring the transversal while searching for graph space, i.e., Open and CLOSED list. An Open list monitors the immediate nodes available to transverse at the moment. In contrast, the CLOSED list monitors the nodes that are being transferred already.

A* Search Algorithm

- To approximate the shortest path in real-life situations, like- in maps, games where there can be many hindrances.
- We can consider a 2D Grid having several obstacles and we start from a source cell (colored red below) to reach towards a goal cell (colored green below).



A* Search Algorithm

- A* Search algorithm is one of the best and popular technique used in path-finding and graph traversals.
- A* Search algorithms, unlike other traversal techniques, it has “brains”. What it means is that it is really a smart algorithm which separates it from the other conventional algorithms.

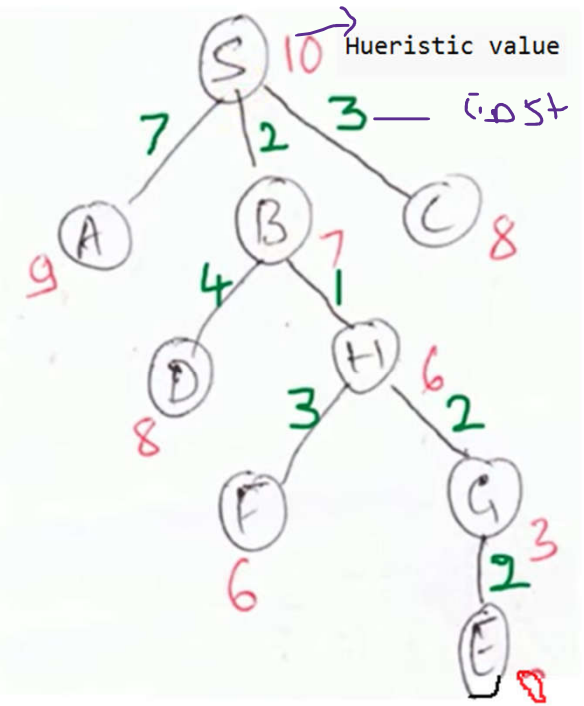
A* Search Algorithm

Open					Closed	
Node	g(n)	h(n)	f(n)		Node	Parent
S	0	10	10			

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

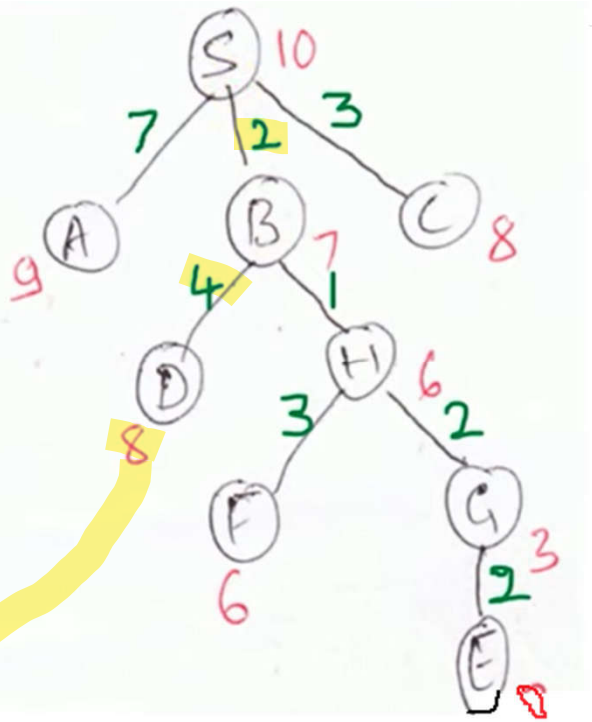
$f(B) = 2 + 7 = 9$

$f(H) = 3 + 6 = 9$



A* Search Algorithm

Open					Closed	
Node	g(n)	h(n)	f(n)		Node	Parent
					S	
C	3	8	11		B	S
D	6	8	14		H	B
A	7	9	16			



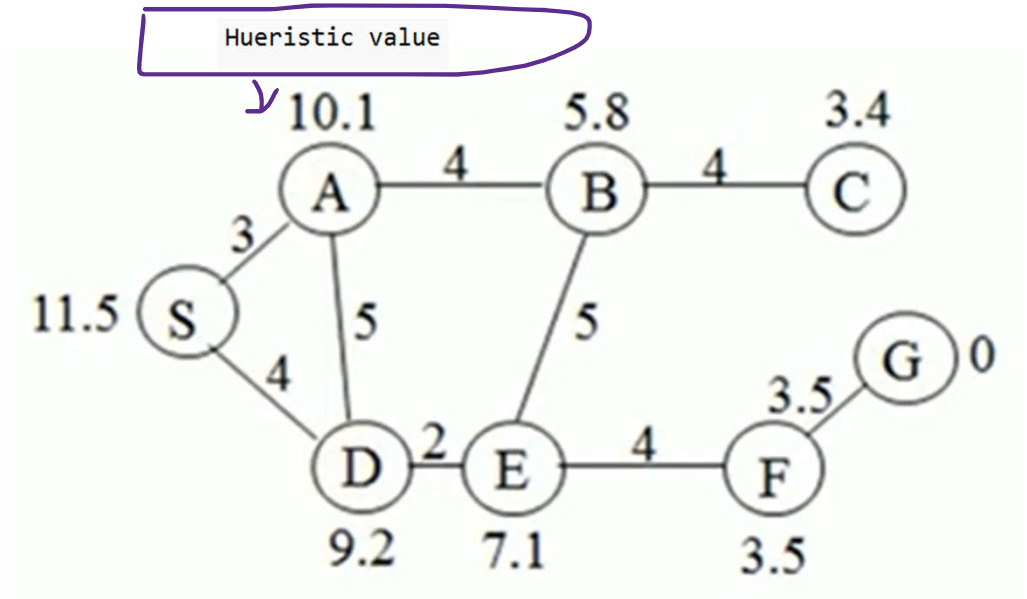
A* Search Algorithm Solved Example 1

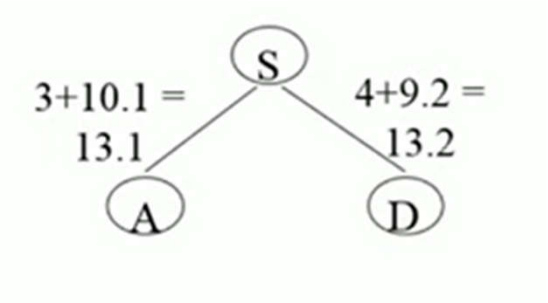
- Algorithm A* (Hart et al., 1968):

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

- $h(n)$ = cost of the cheapest path from node n to a goal state.
- $g(n)$ = cost of the cheapest path from the initial state to node n.

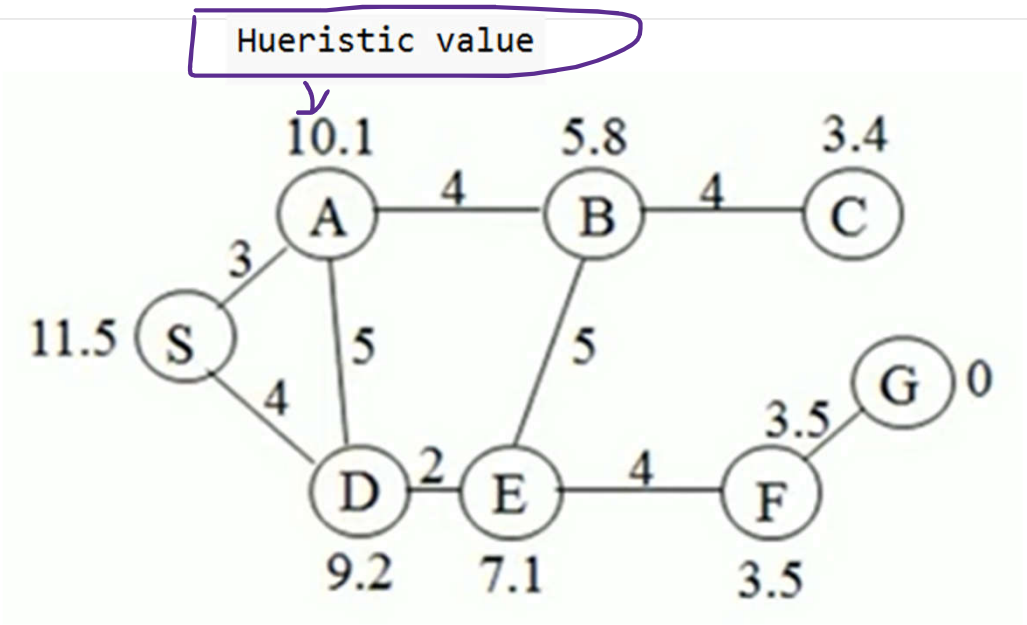
A* Search Algorithm Solved Example 1

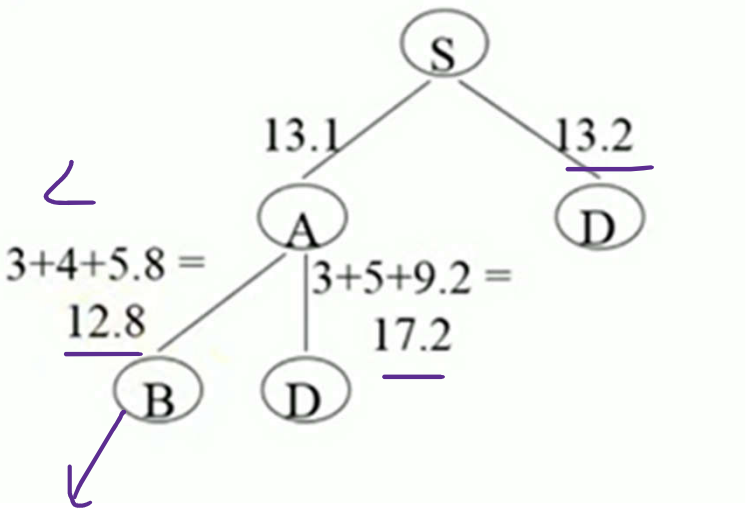




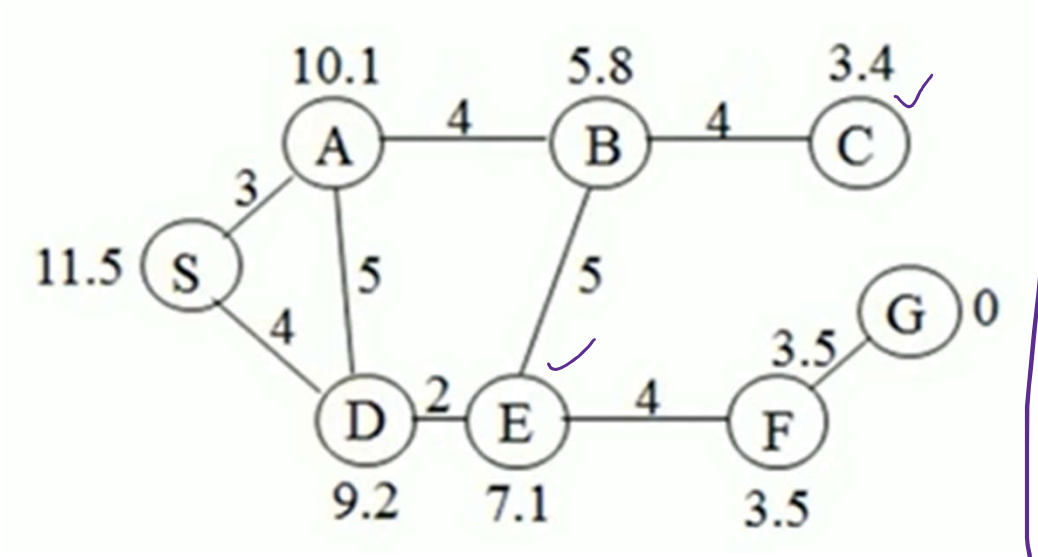
The successors of s are ~~B~~ A and D

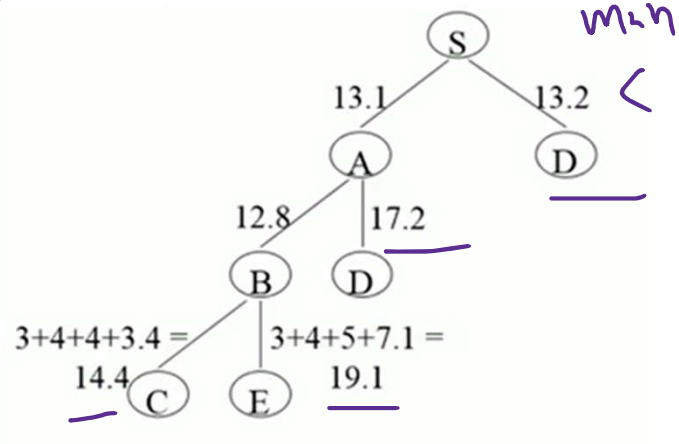
A* Search Algorithm Solved Example 1



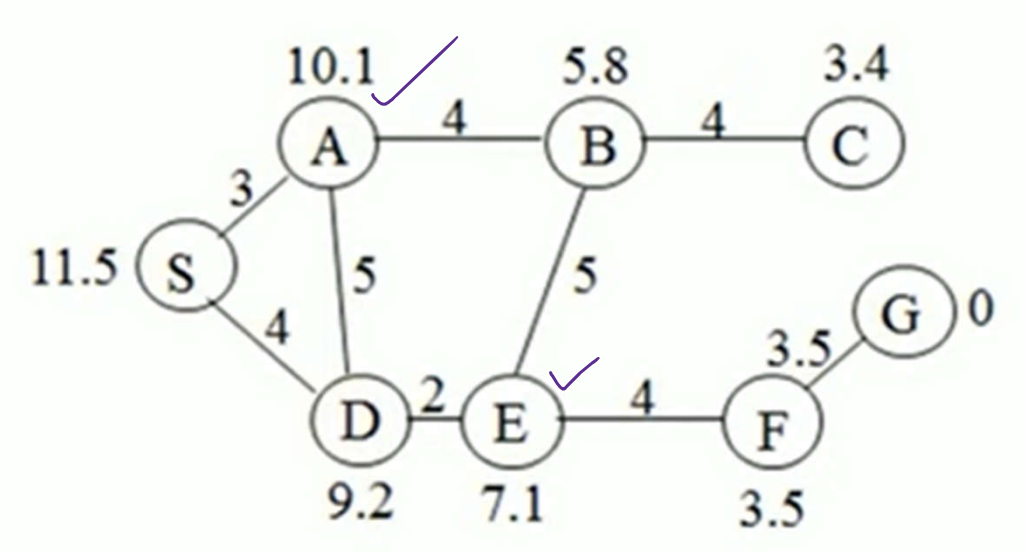


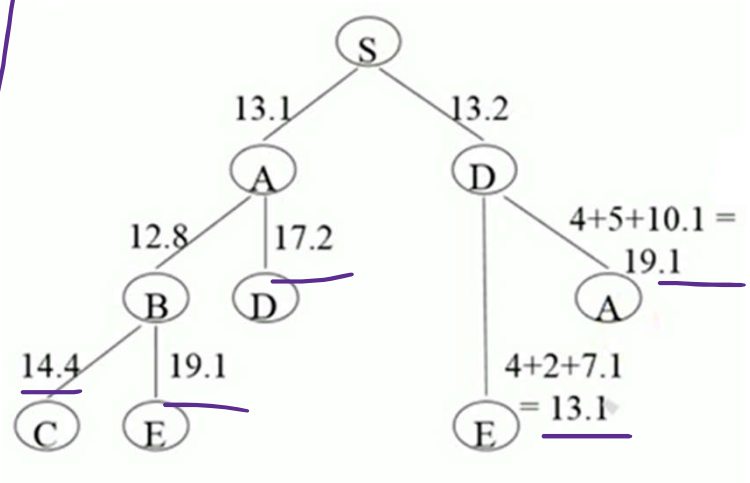
A* Search Algorithm Solved Example 1





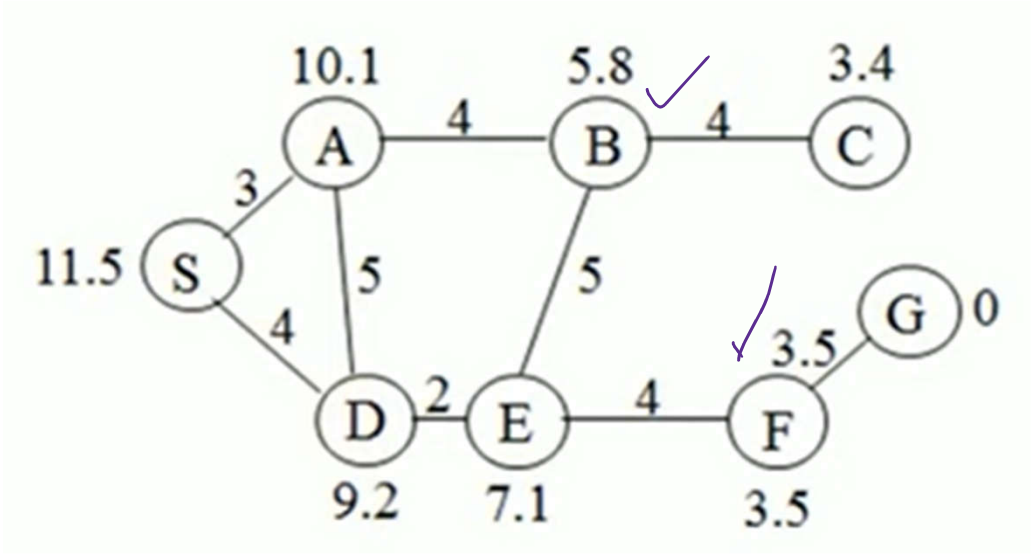
A* Search Algorithm Solved Example 1

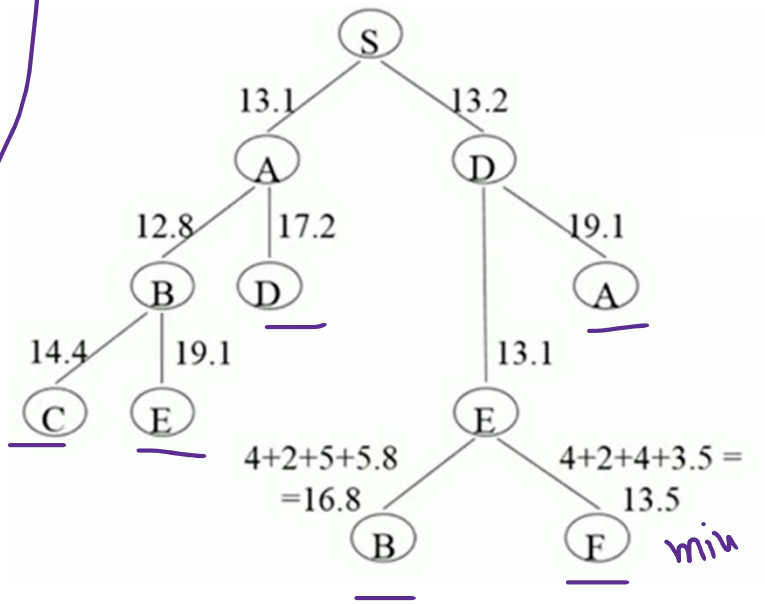




From all leaf nodes minimum is 13.1

A* Search Algorithm Solved Example 1





A* Search Algorithm Solved Example 1