IDA* Search

Step 1: Initialization
Set the root node as the current node, and find the f-score.

Sep 2: Set threshold
Set the cost limit as a **threshold** for a **node** i.e the **maximum f-score** allowed for that node for further explorations.

Step 3: Node Expansion
Expand the current node to its children and find f-scores.

Step 4: Pruning
If for any **node** the **f-score > threshold**, prune that node because it's considered too expensive for that node. and store it in the **visited node list.**

Step 5: Return Path
If the **Goal node** is found then return the **path** from the start node Goal node.

Step 6: Update the Threshold
If the Goal node is not found then **repeat from step 2** by changing the threshold with the minimum pruned value from the **visited node list**. And Continue it until you reach the goal node.

IDA* Search

Advantages

1. G**uaranteed** to find the optimal solution if one exists.
2. Avoids the exponential **time complexity** of traditional Depth First Search. by using an “iterative deepening” approach, where the search depth is gradually increased.
3. Uses a **limited amount of memory** as compared to the A* algorithm because it uses Depth First Search.
4. Admissible heuristic, it **never overestimates** the cost of reaching the goal.
5. Efficient in handling large numbers of states and large branch factors.

IDA* Search

Disadvantages

1. Explore the visited node again and again. it doesn’t keep track of the visited nodes.
2. IDA* may be **slower** to get a solution than other search algorithms like A* or Breadth-First Search because it explores and repeats the explore node again and again.
3. It takes more time and power than the A* algorithm.

	IDDFS	IDA*
1	Systematic	Not Systematic
2	Optimal	Optimal but never expands the node where f-score > Threshold
3	Never expands the same node twice	Expands the same node many times if f-score < Threshold
4	Not good for infinite Search traversal	For infinite Search traversal, it is better than IDDFS.