A*

Optimal: yes

A* is **optimally efficient**: given the information in h, no other optimal search method can expand fewer nodes.

Complete: Unless there are infinitely many nodes with $f(n) < f^\star$. Assume locally finite: (1) finite branching, (2) every operator costs at least $\delta > 0$.

Complexity (time and space): Still exponential because of breadth-first nature. Unless $|h(n) - h^\star| \leq O(log(h^\star(n)))$, with $h^\star$ true cost of getting to goal.

IDA*

Memory is a problem for the A* algorithms.

IDA* is like iterative deepening, but uses an f-cost limit rather than a depth limit.

At each iteration, the cutoff value is the smallest f-cost of any node that exceeded the cutoff on the previous iteration.

Each iteration uses conventional depth-first search.

Recursive best-first search (RBFS)

Similar to a DFS, but keeps track of the f-value of the best alternative path available from any ancestor of the current node.

If current node exceeds this limit, recursion unwinds back to the alternative path, replacing the f-value of each node along the path with the best f-value of its children.

(RBFS remembers the f-value of the best leaf in the forgotten subtree.)

SMA*

Simplified Memory-bounded A* Search

Proceeds just like A*, expanding the best leaf until memory is full.

Drops the **worst** leaf node — the one the highest f-cost; and stores this value in its parent node.

(Won't know which way to go from this node, but we will have some idea of how worthwhile it is to explore the node.)