**COMARISON BETWEEN A\* & IDA\***

**A\* ALGORITHM**

The A\* algorithm combines features of [uniform-cost search](http://intelligence.worldofcomputing.net/ai-search/uniform-cost-search.html) and [pure heuristic search](http://intelligence.worldofcomputing.net/ai-search/pure-heuristic-search.html) to efficiently compute optimal solutions. A\* algorithm is a best-first search algorithm in which the cost associated with a node is f(n) = g(n) + h(n), where g(n) is the cost of the path from the initial state to node n and h(n) is the heuristic estimate or the cost or a path from node n to a goal. Thus, f(n) estimates the lowest total cost of any solution path going through node n. At each point a node with lowest f value is chosen for expansion. Ties among nodes of equal f value should be broken in favor of nodes with lower h values. The algorithm terminates when a goal is chosen for expansion.

A\* algorithm guides an optimal path to a goal if the heuristic function h(n) is admissible, meaning it never overestimates actual cost. For example, since airline distance never overestimates actual highway distance, and manhatten distance never overestimates actual moves in the gliding tile.

For Puzzle, A\* algorithm, using these evaluation functions, can find optimal solutions to these problems. In addition, A\* makes the most efficient use of the given heuristic function in the following sense: among all shortest-path algorithms using the given heuristic function h(n). A\* algorithm expands the fewest number of nodes.

The main **drawback** of A\* algorithm and indeed of any best-first search is its[memory](http://www.worldofcomputing.net/memory/computer-memory.html) requirement. Since at least the entire open list must be saved, A\* algorithm is severely space-limited in practice, and is no more practical than best-first search algorithm on current machines. For example, while it can be run successfully on the eight puzzle, it exhausts available [memory](http://www.worldofcomputing.net/memory/computer-memory.html) in a matter of minutes on the fifteen puzzle.

**ITERATIVE DEEPENING A\* (IDA\*) SEARCH**

Just as iterative deepening solved the space problem of [breadth-first search](http://intelligence.worldofcomputing.net/ai-search/breadth-first-search.html), iterative deepening A\* (IDA\*) eliminates the memory constraints of [A\* search algorithm](http://intelligence.worldofcomputing.net/ai-search/a-star-algorithm.html) without sacrificing solution optimality. Each iteration of the algorithm is a [depth-first search](http://intelligence.worldofcomputing.net/ai-search/depth-first-search.html) that keeps track of the cost, f(n) = g(n) + h(n), of each node generated. As soon as a node is generated whose cost exceeds a threshold for that iteration, its path is cut off, and the search backtracks before continuing. The cost threshold is initialized to the heuristic estimate of the initial state, and in each successive iteration is increased to the total cost of the lowest-cost node that was pruned during the previous iteration. The algorithm terminates when a goal state is reached whose total cost does not exceed the current threshold.

Since Iterative Deepening A\* performs a series of [depth-first searches](http://intelligence.worldofcomputing.net/ai-search/depth-first-search.html), its memory requirement is linear with respect to the maximum search depth. In addition, if the heuristic function is admissible, IDA\* finds an optimal solution. Finally, by an argument similar to that presented for DFID, IDA\* expands the same number of nodes, asymptotically, as A\* on a tree, provided that the number of nodes, asymptotically, as [A\*](http://intelligence.worldofcomputing.net/ai-search/a-star-algorithm.html) on a tree, provided that the number of nodes grows exponentially with solution cost. These costs, together with the optimality of A\*, imply that

IDA\* is asymptotically optimal in time and space over all [heuristic search](http://intelligence.worldofcomputing.net/ai-search/heuristic-search.html)algorithms that find optimal solutions on a tree. Additional benefits of IDA\* are that it is much easier to implement, and often runs faster than A\*, since it does not incur the overhead of managing the open and closed lists.