**A\* Algorithm**

This algorithm is a informed search in which we have a start point and a goal state through which we have to reach our goal with the most optimal path using the heuristic value and then explore all the path to find the best optimal path. It is implemented as below:

* + Represents a node in the search tree.
  + Stores state, parent, g (cost from start), h (heuristic), and f (total estimated cost).
  + Implements \_\_eq\_\_ to compare nodes based on state.
  + Implements the *A search algorithm*\* to find the optimal path.
  + Maintains an open\_list (nodes to explore) and closed\_list (explored nodes).
  + Selects the node with the lowest f = g + h value.
  + Expands neighbors, updates g, h, f, and adds valid nodes to open\_list.
  + Returns the optimal path and total cost when the goal is found.
  + Checks if a neighbor should be added to open\_list.
  + Prevents adding nodes with a worse path cost (g).
  + Placeholder heuristic function (always returns 0, making it behave like Dijkstra's algorithm).
  + Defines the graph as an adjacency list with node connections and costs.
  + Runs *A search*\* from A to G.
  + Prints the optimal path and total cost.