

## A\* Algorithm pseudocode

The goal node is denoted by `node_goal` and the source node is denoted by `node_start`

We maintain two lists: **OPEN** and **CLOSE**:

**OPEN** consists on nodes that have been visited but not expanded (meaning that successors have not been explored yet). This is the list of pending tasks.

**CLOSE** consists on nodes that have been visited *and* expanded (successors have been explored already and included in the open list, if this was the case).

```
1 Put node_start in the OPEN list with  $f(\text{node\_start}) = h(\text{node\_start})$  (initialization)
2 while the OPEN list is not empty {
3   Take from the open list the node node_current with the lowest
4    $f(\text{node\_current}) = g(\text{node\_current}) + h(\text{node\_current})$ 
5   if node_current is node_goal we have found the solution; break
6   Generate each state node_successor that come after node_current
7   for each node_successor of node_current {
8     Set successor_current_cost =  $g(\text{node\_current}) + w(\text{node\_current}, \text{node\_successor})$ 
9     if node_successor is in the OPEN list {
10      if  $g(\text{node\_successor}) \leq \text{successor\_current\_cost}$  continue (to line 20)
11    } else if node_successor is in the CLOSED list {
12      if  $g(\text{node\_successor}) \leq \text{successor\_current\_cost}$  continue (to line 20)
13      Move node_successor from the CLOSED list to the OPEN list
14    } else {
15      Add node_successor to the OPEN list
16      Set  $h(\text{node\_successor})$  to be the heuristic distance to node_goal
17    }
18    Set  $g(\text{node\_successor}) = \text{successor\_current\_cost}$ 
19    Set the parent of node_successor to node_current
20  }
21  Add node_current to the CLOSED list
22 }
23 if(node_current != node_goal) exit with error (the OPEN list is empty)
```

