Legend (Glossary of Symbols)

$g(s)$	The current cost estimate from state s to the goal.
$rhs(s)$	The one-step lookahead cost estimate of $g(s)$.
$h(s_1, s_2)$	Heuristic estimate from s_1 to s_2 (e.g., Euclidean or Manhattan distance).
km	A cumulative modifier for the heuristic to account for movement; used to detect changes.
s_{start}	The agent's current position (start of the current planning segment).
s_{goal}	The target (goal) position.
U	A priority queue storing states to process, ordered by keys.
$c(u, v)$	The cost of moving from node u to node v.
$Succ(u)$	Set of successor states of u (i.e., nodes reachable from u).
$Pred(u)$	Set of predecessor states of u (i.e., nodes that can reach u).
s	A generic state (node) in the graph or grid.

Algorithm 1 D* Lite: CalculateKey

```
1: procedure CalculateKey($s$)
2:     return $[\min(g(s), rhs(s)) + h(s_{start}, s) + km, \min(g(s), rhs(s))]$
3: end procedure
```

Algorithm 2 D* Lite: Initialize

```
1: procedure Initialize
2:     $U \leftarrow \emptyset$                                  ▷ Priority queue
3:     $km \leftarrow 0$
4:     for all $s \in S$ do
5:         $rhs(s) \leftarrow \infty$
6:         $g(s) \leftarrow \infty$
7:     end for
8:     $rhs(s_{goal}) \leftarrow 0$
9:     Insert $s_{goal}$ into $U$ with key CalculateKey($s_{goal}$)
10: end procedure
```

Algorithm 3 D* Lite: UpdateVertex

```
1: procedure UpdateVertex($u$)
2:     if $u \neq s_{goal}$ then
3:         $rhs(u) \leftarrow \min_{s' \in Succ(u)} (c(u, s') + g(s'))$
4:     end if
5:     if $u \in U$ then
6:         Remove $u$ from $U$
7:     end if
8:     if $g(u) \neq rhs(u)$ then
9:         Insert $u$ into $U$ with key CalculateKey($u$)
10:     end if
11: end procedure
```

Algorithm 4 D* Lite: ComputeShortestPath

```
procedure ComputeShortestPath
    while $U$.TopKey() $<$ CalculateKey($s_{start}$) or $rhs(s_{start}) \neq g(s_{start})$ do
        $u \leftarrow U$.Top()
        $k_{old} \leftarrow U$.TopKey()
        Remove $u$ from $U$
        $k_{new} \leftarrow$ CalculateKey($u$)
        if $k_{old} < k_{new}$ then
            Insert $u$ into $U$ with key $k_{new}$
        else if $g(u) > rhs(u)$ then
            $g(u) \leftarrow rhs(u)$
            for all $p \in Pred(u)$ do
                UpdateVertex($p$)
            end for
        else
            $g_{old} \leftarrow g(u)$
            $g(u) \leftarrow \infty$
            for all $p \in Pred(u) \cup \{u\}$ do
                UpdateVertex($p$)
            end for
        end if
    end while
end procedure
```

Algorithm 5 D* Lite: MainLoop

```
procedure MainLoop
    $s_{last} \leftarrow s_{start}$
    Initialize()
    ComputeShortestPath()
    while $s_{start} \neq s_{goal}$ do
        if $g(s_{start}) = \infty$ then
            return No path exists
        end if
        $s_{start} \leftarrow \arg\min_{s' \in Succ(s_{start})} (c(s_{start}, s') + g(s'))$
        Move to $s_{start}$
        Scan for changed edge costs
        if any edge costs changed then
            $km \leftarrow km + h(s_{last}, s_{start})$
            $s_{last} \leftarrow s_{start}$
            for all changed edges $(u, v)$ do
                Update cost $c(u, v)$
                UpdateVertex($u$)
            end for
            ComputeShortestPath()
        end if
    end while
end procedure
```