If X is current robot position, success

Path through X is no longer optimal due to new information. Go through Y

See if neighbors of X can go through new lower cost X due to updated information. Also, update out-of-date costs to each neighbor of X.

Propagate changes to NEW states and descendents of X.

If change in X can lower costs in non-descendent states, queue for processing

If path cost of X can be lowered through neighbor, queue Y for processing

Function: PROCESS-STATE()

L1  \( X = \text{MIN-STATE}() \)
L2  if \( X = \text{NULL} \) then return -1
L3  \( k_{old} = \text{GET-KMIN}() \); DELETE(X)
L4  if \( k_{old} < h(X) \) then
L5    for each neighbor \( Y \) of \( X \):
L6      if \( h(Y) \leq k_{old} \) and \( h(X) > h(Y) + c(Y, X) \) then
L7        \( b(X) = Y; h(X) = h(Y) + c(Y, X) \)
L8    if \( k_{old} = h(X) \) then
L9      for each neighbor \( Y \) of \( X \):
L10     if \( t(Y) = \text{NEW} \) or
L11        \( b(Y) = X \) and \( h(Y) \neq h(X) + c(X, Y) \) or
L12        \( b(Y) \neq X \) and \( h(Y) > h(X) + c(X, Y) \) then
L13        \( b(Y) = X; \text{INSERT}(Y, h(X) + c(X, Y)) \)
L14  else
L15    for each neighbor \( Y \) of \( X \):
L16      if \( t(Y) = \text{NEW} \) or
L17        \( b(Y) = X \) and \( h(Y) \neq h(X) + c(X, Y) \) then
L18        \( b(Y) = X; \text{INSERT}(Y, h(X) + c(X, Y)) \)
L19      else
L20        if \( b(Y) \neq X \) and \( h(Y) > h(X) + c(X, Y) \) then
L21          \text{INSERT}(X, h(X))
L22      else
L23        if \( b(Y) \neq X \) and \( h(X) > h(Y) + c(Y, X) \) and
L24          \( t(Y) = \text{CLOSED} \) and \( h(Y) > k_{old} \) then
L25          \text{INSERT}(Y, h(Y))
L26  return \text{GET-KMIN}()
Path through X no longer optimal (RAISE state)

Path through X is still optimal

A new cheaper path may have been found in sensory update

---

### Function: PROCESS-STATE()

<table>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>L1</td>
<td>( X = \text{MIN-STATE}(\ ) )</td>
<td>Select ( X ) in OPEN with minimum ( k )</td>
</tr>
<tr>
<td>L2</td>
<td>if ( X = \text{NULL} ) then return (-1)</td>
<td>If OPEN empty, then failure</td>
</tr>
<tr>
<td>L3</td>
<td>( k_{old} = \text{GET-KMIN}() ); DELETE(( X ))</td>
<td>Get min value of ( k ) on OPEN, set ( t(X) = \text{CLOSED} )</td>
</tr>
<tr>
<td>L4</td>
<td>if ( k_{old} &lt; h(X) ) then</td>
<td></td>
</tr>
<tr>
<td>L5</td>
<td>for each neighbor ( Y ) of ( X ):</td>
<td>If cheaper to go thru nhbr ( Y ), then relink path from ( X )</td>
</tr>
<tr>
<td>L6</td>
<td>if ( h(Y) \leq k_{old} ) and ( h(X) &gt; h(Y) + c(Y, X) ) then</td>
<td></td>
</tr>
<tr>
<td>L7</td>
<td>( b(X) = Y; h(X) = h(Y) + c(Y, X) )</td>
<td></td>
</tr>
<tr>
<td>L8</td>
<td>if ( k_{old} = h(X) ) then</td>
<td>Nhbr ( Y ) is unvisited</td>
</tr>
<tr>
<td>L9</td>
<td>for each neighbor ( Y ) of ( X ):</td>
<td>Cost is out of date</td>
</tr>
<tr>
<td>L10</td>
<td>if ( t(Y) = \text{NEW or} )</td>
<td>Path thru ( Y ) is higher cost than thru ( X )</td>
</tr>
<tr>
<td>L11</td>
<td>( (b(Y) = X \text{ and } h(Y) \neq h(X) + c(X, Y)) \text{ or} )</td>
<td>Relink ( Y )’s path thru ( X ), put on OPEN</td>
</tr>
<tr>
<td>L12</td>
<td>( (b(Y) \neq X \text{ and } h(Y) &gt; h(X) + c(X, Y)) \text{ then} )</td>
<td></td>
</tr>
<tr>
<td>L13</td>
<td>( b(Y) = X; \text{INSERT}(Y, h(X) + c(X, Y)) )</td>
<td></td>
</tr>
<tr>
<td>L14</td>
<td>else</td>
<td></td>
</tr>
<tr>
<td>L15</td>
<td>for each neighbor ( Y ) of ( X ):</td>
<td>Nhbr ( Y ) is unvisited</td>
</tr>
<tr>
<td>L16</td>
<td>if ( t(Y) = \text{NEW or} )</td>
<td>Cost is out of date</td>
</tr>
<tr>
<td>L17</td>
<td>( (b(Y) = X \text{ and } h(Y) \neq h(X) + c(X, Y)) \text{ then} )</td>
<td>Relink ( Y )’s path thru ( X ), put on OPEN</td>
</tr>
<tr>
<td>L18</td>
<td>( b(Y) = X; \text{INSERT}(Y, h(X) + c(X, Y)) )</td>
<td>This step prevents loops in plan</td>
</tr>
<tr>
<td>L19</td>
<td>else</td>
<td></td>
</tr>
<tr>
<td>L20</td>
<td>if ( b(Y) \neq X \text{ and } h(Y) &gt; h(X) + c(X, Y) ) then</td>
<td>Path thru ( Y ) may be lower cost alternative</td>
</tr>
<tr>
<td>L21</td>
<td>( \text{INSERT}(X, h(X)) )</td>
<td></td>
</tr>
<tr>
<td>L22</td>
<td>else</td>
<td></td>
</tr>
<tr>
<td>L23</td>
<td>if ( b(Y) \neq X \text{ and } h(Y) &gt; h(X) + c(Y, X) \text{ and} )</td>
<td></td>
</tr>
<tr>
<td>L24</td>
<td>( t(Y) = \text{CLOSED} \text{ and } h(Y) &gt; k_{old} \text{ then} )</td>
<td>Put ( Y ) on OPEN for later processing</td>
</tr>
<tr>
<td>L25</td>
<td>( \text{INSERT}(Y, h(Y)) )</td>
<td></td>
</tr>
<tr>
<td>L26</td>
<td>return ( \text{GET-KMIN}() )</td>
<td></td>
</tr>
</tbody>
</table>
h(G)=0;
do{
    k_{min}=\text{PROCESS-STATE}();
}while(k_{min} != -1 &\& \text{ start state not removed from open list});
if(k_{min} == -1)
    \{ \text{ goal unreachable; exit;} \}
else{
    do{
        do{
            \text{ trace optimal path();}
        }\text{while ( goal is not reached &\& map == environment);}

        if( \text{ goal_is_reached})
            \{ \text{ exit;} \}
        else {
            Y= \text{ State of discrepancy reached trying to move from some State X;}
            \text{ MODIFY-COST}(Y,X,\text{ newc}(Y,X));
            do
                \{ 
                    k_{min}=\text{PROCESS-STATE}();
                }\text{while}(k_{min} < h(X) &\& k_{min} != -1);
            if(k_{min} == -1)
                exit();
            \}
        }\text{while(1);}
    }
}

\textbf{Function: MODIFY-COST (X, Y, cval)}
\begin{align*}
L1 & \quad c(X, Y) = cval \\
L2 & \quad \text{if } t(X) = \text{CLOSED then INSERT}(X, h(X)) \\
L3 & \quad \text{return } GET\_KMIN()
\end{align*}