Input: Roadmap G, edge length $D(u, v)$, visible faces $S(u, v)$.
Input: Root state v_0, and faces seen S_0.

procedure SEARCH(G, S_0, v_0)
  d_0, q_0, $D_0 \leftarrow 0$ ▷ Depth, score, and distance.
  $U_0 \leftarrow$ NEIGHBORS(G, v_0) ▷ Unvisited neighbors.
  for $i \leftarrow 1$ to N_{iter} **do**
    $p_i \leftarrow \arg\min_{v:U_v \neq \emptyset}(d_v)$ ▷ Most shallow leaf.[a]
    $v_i \leftarrow$ PICK(U_{p_i})
    $d_i \leftarrow d_{p_i} + 1$
    $D_i \leftarrow D_{p_i} + D(v_{p_i}, v_i)$
    $S_i \leftarrow S_{p_i} \cup S(v_{p_i}, v_i)$
    $q_i \leftarrow q_{p_i} + s(S_i, D_i)$
    $U_i \leftarrow$ NEIGHBORS(G, v_i)
    $U_{p_i} \leftarrow U_{p_i} \setminus \{v_i\}$
  return PATH($\arg\max_v(q_v)$)

[a]Taking the most shallow leaf, i.e. leaf of smallest depth, results in a BFS-like search order. An alternative is to minimize the distance D_u instead, however, that would effectively prevent searching beyond jump edges as the distance would be significantly higher.