|  |
| --- |
| *Algorithm: A Search Algorithm*\* |
| Input: Start node *S*, Goal node *G*, Graph or grid representation, Heuristic function *h*(*n*) |
| Output: Optimal path from *S* to *G* |
|  |
| 1. Initialize: |
| * + Open list ← {*S*} |
| * + Closed list ← ∅ |
| * + *g*(*S*) ← 0 |
| * + *f* (*S*) ← *h*(*S*) |
| 1. While Open list is not empty do: |
| 1. Select node *n* from Open list with the smallest *f* (*n*). |
| 1. If *n* = *G* then: |
| 1. Return the reconstructed path from *S* to *G*. |
| 1. Remove *n* from Open list and add to Closed list. |
| 1. For each neighbor *m* of *n*: |
| 1. If *m* ∈ Closed list then continue. |
| 1. Tentative cost *gtentative* ← *g*(*n*) + *cost*(*n*, *m*). |
| 1. If *m* ∈/ Open list then: |
| 1. Add *m* to Open list. |
| 1. Else if *gtentative* ≥ *g*(*m*) then continue. |
| 1. Update: |
| 1. *g*(*m*) ← *gtentative*. |
| 1. *f* (*m*) ← *g*(*m*) + *h*(*m*). |
| 1. Parent of *m* ← *n*. |
| 1. End While |
| 1. If Open list is empty then: |
| 1. Return "Path not found". |
| 1. End |

**Explanation**:

 Open list: Stores nodes to be evaluated.

 Closed list: Stores nodes already evaluated.

 Heuristic *h*(*n*): Estimated cost to goal.

 Function *f* (*n*): Combines *g*(*n*) (actual cost) and *h*(*n*) (estimated cost).