Algorithm 1 Uniform Cost Search (Graph)

```
1: function UNIFORM-COST-SEARCH(problem) returns a solution or failure
2:     node ← a node with STATE = problem.INITIAL-STATE, PATH-COST = 0
3:     frontier ← a priority queue ordered by PATH-COST, with node as the only element
4:     explored ← an empty set
5:
6:     loop do
7:         if frontier is ∅ then return failure
8:         node ← POP(frontier)
9:         if problem.GOAL-TEST(node.STATE) then return SOLUTION(node)
10:
11:        add node.STATE to explored
12:        for each action in problem.ACTIONS(node.STATE) do
13:            child ← CHILD-NODE(problem, node, action)
14:            if child.STATE ∉ explored or frontier then
15:                frontier.APPEND(child)
16:            else if child.STATE ∈ frontier with higher PATH-COST then
17:                frontier.REPLACE(node-with-higher-cost, child)
```