function Best-First-Search(problem, f) returns a solution node or failure

node ← NODE(State=problem.INITIAL)
frontier ← a priority queue ordered by f, with node as an element
reached ← a lookup table, with one entry with key problem.INITIAL and value node

while not IS-EMPTY(frontier) do
    node ← POP(frontier)
    if problem.IS-GOAL(node.STATE) then return node
    for each child in EXPAND(problem, node) do
        s ← child.STATE
        if s is not in reached or child.PATH-COST < reached[s].PATH-COST then
            reached[s] ← child
            add child to frontier
    return failure

function EXPAND(problem, node) yields nodes
s ← node.STATE
for each action in problem.ACTIONS(s) do
    s' ← problem.RESULT(s, action)
    cost ← node.PATH-COST + problem.ACTION-COST(s, action, s')
    yield NODE(State=s', Parent=node, Action=action, Path-Cost=cost)

Figure 3.7 The best-first search algorithm, and the function for expanding a node. The data structures used here are described in Section ???. See Appendix B for yield.