Algorithm 1 Partially Observable Monte-Carlo Planning

```
procedure SEARCH(h)
    repeat
        if h = empty then
            s ~ I
        else
            s ~ B(h)
        end if
        SIMULATE(s, h, 0)
    until TIMEOUT()
    return argmax_b V(hb)
end procedure

procedure ROLLOUT(s, h, depth)
    if γ^depth < ε then
        return 0
    end if
    a ~ π_rollout(h, ·)
    (s', o, r) ~ G(s, a)
    return r + γ.ROLLOUT(s', hao, depth+1)
end procedure

procedure SIMULATE(s, h, depth)
    if γ^depth < ε then
        return 0
    end if
    if h ∉ T then
        for all a ∈ A do
            T(ha) ← (N_init(ha), V_init(ha), ∅)
        end for
        return ROLLOUT(s, h, depth)
    end if
    a ← argmax_b V(hb) + c sqrt(log N(h) / N(hb))
    (s', o, r) ~ G(s, a)
    R ← r + γ.SIMULATE(s', hao, depth + 1)
    B(h) ← B(h) ∪ {s}
    N(h) ← N(h) + 1
    N(ha) ← N(ha) + 1
    V(ha) ← V(ha) + (R − V(ha)) / N(ha)
    return R
end procedure
```