```
    while H is not empty do
        ⟨d_i, dist[d_i]⟩ ← deheap(H)
        if d_i ∈ doors then
            doors ← doors \ {d_i}
            if dist_m > dist_V(p_s, d_s) + dist[d_i] + dist_V(p_t, d_i)
            then
                dist_m ← dist_V(p_s, d_s) + dist[d_i] + dist_V(p_t, d_i)
            (v, d_j) ← prev[d_i]
            while d_j ≠ d_s do
                if d_j ∈ doors_s and d_j > d_s then
                    dists[d_j][d_i] ← dist[d_i] − dist[d_j]
                    if dist_m > dist_V(p_s, d_j) + dists[d_j][d_i] +
                    dist_V(p_t, d_i) then
                        dist_m ← dist_V(p_s, d_j) + dists[d_j][d_i] +
                        dist_V(p_t, d_i)
                (v, d_j) ← prev[d_j]
            if doors = ∅ then
                break
        else if d_i ∈ doors_s and d_i < d_s then
            for each door d_j ∈ doors do
                dists[d_s][d_j] ← dist[d_i] + dists[d_i][d_j]
                if dist_m > dist_V(p_s, d_s) + dists[d_s][d_j] +
                dist_V(p_t, d_j) then
                    dist_m ← dist_V(p_s, d_s) + dists[d_s][d_j] +
                    dist_V(p_t, d_j)
            break
        mark door d_i as visited
        parts ← D2P_⊏(d_i)
        for each partition v ∈ parts do
            for each unvisited door d_j ∈ P2D(v) do
                if d_j ∈ P2D_⊏(v) then
                    if dist[d_i] + G_dist.f_d2d(v, d_i, d_j) < dist[d_j] then
                        dist[d_j] ← dist[d_i] + G_dist.f_d2d(v, d_i, d_j)
                        prev[d_j] ← (v, d_i)
    return dist_m
```