Algorithm 4 pt2ptDistance3(Source indoor position p_s, destination indoor position p_t)

1: $v_s \leftarrow$ getHostPartition(p_s)
2: $v_t \leftarrow$ getHostPartition(p_t)
3: $doors_s \leftarrow P2D_{\sqsubset}(v_s)$
4: $doors_t \leftarrow P2D_{\sqsupset}(v_t)$
5: **for** each door $d_s \in doors_s$ **do**
6: $np \leftarrow$ the partition in $D2P_{\sqsubset}(d_s) \setminus \{v_s\}$
7: **if** $P2D_{\sqsubset}(np) = \{d_s\}$ **and** $np \neq v_t$ **then**
8: remove d_s from $doors_s$
9: **for** each door $d_t \in doors_t$ **do**
10: $dists[d_s][d_t] \leftarrow \infty$
11: $dist_m \leftarrow \infty$
12: **for** each door $d_s \in doors_s$ **do**
13: $doors \leftarrow \emptyset$
14: **for** each door $d_t \in doors_t$ **do**
15: **if** $dists[d_s][d_t] = \infty$ **and** $dist_V(p_s, d_s) + dist_V(p_t, d_t) < dist_m$ **then**
16: add d_t to $doors$
17: initialize a min-heap H
18: **for** each door $d_i \in \Sigma_{door}$ **do**
19: **if** $d_i \neq d_s$ **then**
20: $dist[d_i] \leftarrow \infty$
21: **else**
22: $dist[d_i] \leftarrow 0$
23: enheap($H, \langle d_i, dist[d_i] \rangle$)
24: $prev[d_i] \leftarrow$ null