Hierarchical Q-Learning (HierQ)

Algorithm 1 Hierarchical Q-Learning (HierQ)

Input:

- Key agent parameters: number of levels in hierarchy $k > 1$, maximum subgoal horizon H, learning rate α, discount rate γ

Output:

- k trained Q-tables $Q_0(s, g, a), ..., Q_{k-1}(s, g, a)$

for M episodes **do** ▷ Train for M episodes
  $s_{k-1} \leftarrow S_{init}$, $g_{k-1} \leftarrow G_{k-1}$ ▷ Sample initial state and task goal
  ▷ Initialize previous state arrays for levels $i, 0 < i < k$
  $Prev\_States_i \leftarrow Array[H^i]$ ▷ Length of level i array is H^i

  while g_{k-1} not achieved **do** ▷ Begin Training
    $a_{k-1} \leftarrow \pi_{k-1_b}(s_{k-1}, g_{k-1})$ ▷ Sample action using ϵ-greedy policy π_{k-1_b}
    $s_{k-1} \leftarrow train - level(k-2,\ s_{k-1},\ a_{k-1})$ ▷ Train next level
  end while
end for

function TRAIN-LEVEL($i :: level, s :: state, g :: goal$)
  $s_i \leftarrow s$, $g_i \leftarrow g$ ▷ Set current state and goal for level i
  for H attempts or until g_n, $i \leq n < k$ achieved **do**
    $a_i \leftarrow \pi_{i_b}(s_i, g_i)$ ▷ Sample action using ϵ-greedy policy π_{i_b}
    if $i > 0$ **then**
      $s_i^{'} \leftarrow train - level(i-1, s_i, a_i)$ ▷ Train level $i-1$ using subgoal a_i
    else
      Execute primitive action a_0 and observe next state $s_0^{'}$
      ▷ Update $Q_0(s, g, a)$ table for all possible subgoal states
      for each state $s_{goal} \in S$ **do**
        $Q_0(s_0, s_{goal}, a_0) \leftarrow (1-\alpha) \cdot Q_0(s_0, s_{goal}, a_0) + \alpha \cdot [R_0 + \gamma max_a Q_0(s_0', s_{goal}, a_0)]$
      end for
      ▷ Add state s_0 to all previous state arrays
      $Prev\_States_i \leftarrow s_0, 0 < i < k$
      ▷ Update $Q_i(s, g, a), 0 < i < k$, tables
      for each level $i, 0 < i < k$ **do**
        for each state $s \in Prev\_States_i$ **do**
          for each goal $s_{goal} \in S$ **do**
            $Q_i(s, s_{goal}, s_0') \leftarrow (1 - \alpha) \cdot Q_i(s, s_{goal}, s_0') + \alpha \cdot [R_i + \gamma max_a Q_i(s_0', s_{goal}, a)$
          end for
        end for
      end for
    end if
    $s_i \leftarrow s_i^{'}$
  end for
  return $s_i^{'}$ ▷ Output current state
end function