Here, we give complete pseudo-code for solving HMM Problem 3, including scaling. This pseudo-code also provides virtually everything needed to solve HMM Problems 1 and 2.

1. Given

   Observation sequence $\mathcal{O} = (\mathcal{O}_0, \mathcal{O}_1, \ldots, \mathcal{O}_{T-1})$.

2. Initialize

   (a) Select $N$ and determine $M$ from $\mathcal{O}$. Recall that the model is denoted $\lambda = (A, B, \pi)$, where $A = \{a_{ij}\}$ is $N \times N$, $B = \{b_j(k)\}$ is $N \times M$, and $\pi = \{\pi_i\}$ is $1 \times N$.

   (b) Initialize the three matrices $A$, $B$, and $\pi$. You can use knowledge of the problem when generating initial values, but if no such information is available (as is often the case), let $\pi_i \approx 1/N$ and let $a_{ij} \approx 1/N$ and $b_j(k) \approx 1/M$. Always be sure that your initial values satisfy the row stochastic conditions (i.e., the elements of each row sum to 1, and each element is between 0 and 1). Also, make sure that the elements of each row are *not* exactly uniform.

   (c) Initialize each of the following.

   `minIters` = minimum number of re-estimation iterations
   $\varepsilon$ = threshold representing negligible improvement in model
   `iters` = 0
   `oldLogProb` = $-\infty$

3. Forward algorithm or $\alpha$-pass

```
// compute α_0(i)
c_0 = 0
for i = 0 to N − 1
    α_0(i) = π_i b_i(O_0)
    c_0 = c_0 + α_0(i)
next i
// scale the α_0(i)
c_0 = 1/c_0
for i = 0 to N − 1
    α_0(i) = c_0 α_0(i)
next i
// compute α_t(i)
for t = 1 to T − 1
    c_t = 0
    for i = 0 to N − 1
        α_t(i) = 0
        for j = 0 to N − 1
            α_t(i) = α_t(i) + α_{t−1}(j) a_{ji}
        next j
        α_t(i) = α_t(i) b_i(O_t)
        c_t = c_t + α_t(i)
    next i
    // scale α_t(i)
    c_t = 1/c_t
    for i = 0 to N − 1
        α_t(i) = c_t α_t(i)
    next i
next t
```

4. Backward algorithm or $\beta$-pass

> // Let $\beta_{T-1}(i) = 1$ scaled by $c_{T-1}$
> for $i = 0$ to $N-1$
>  $\beta_{T-1}(i) = c_{T-1}$
> next $i$
> // $\beta$-pass
> for $t = T-2$ to $0$ by $-1$
>  for $i = 0$ to $N-1$
>   $\beta_t(i) = 0$
>   for $j = 0$ to $N-1$
>    $\beta_t(i) = \beta_t(i) + a_{ij} b_j(\mathcal{O}_{t+1}) \beta_{t+1}(j)$
>   next $j$
>   // scale $\beta_t(i)$ with same scale factor as $\alpha_t(i)$
>   $\beta_t(i) = c_t \beta_t(i)$
>  next $i$
> next t

5. Compute the gammas and di-gammas

> for $t = 0$ to $T-2$
>  $\texttt{denom} = 0$
>  for $i = 0$ to $N-1$
>   for $j = 0$ to $N-1$
>    $\texttt{denom} = \texttt{denom} + \alpha_t(i) a_{ij} b_j(\mathcal{O}_{t+1}) \beta_{t+1}(j)$
>   next $j$
>  next $i$
>  for $i = 0$ to $N-1$
>   $\gamma_t(i) = 0$
>   for $j = 0$ to $N-1$
>    $\gamma_t(i,j) = \big(\alpha_t(i) a_{ij} b_j(\mathcal{O}_{t+1}) \beta_{t+1}(j)\big)/\texttt{denom}$
>    $\gamma_t(i) = \gamma_t(i) + \gamma_t(i,j)$
>   next $j$
>  next $i$
> next $t$
> // Special case for $\gamma_{T-1}(i)$
> $\texttt{denom} = 0$
> for $i = 0$ to $N-1$
>  $\texttt{denom} = \texttt{denom} + \alpha_{T-1}(i)$
> next $i$
> for $i = 0$ to $N-1$
>  $\gamma_{T-1}(i) = \alpha_{T-1}(i)/\texttt{denom}$
> next $i$

6. Re-estimate the model $\lambda = (A, B, \pi)$

```
// re-estimate π
for i = 0 to N − 1
    π_i = γ_0(i)
next i
// re-estimate A
for i = 0 to N − 1
    for j = 0 to N − 1
        numer = 0
        denom = 0
        for t = 0 to T − 2
            numer = numer + γ_t(i, j)
            denom = denom + γ_t(i)
        next t
        a_ij = numer/denom
    next j
next i
// re-estimate B
for i = 0 to N − 1
    for j = 0 to M − 1
        numer = 0
        denom = 0
        for t = 0 to T − 1
            if(O_t == j) then
                numer = numer + γ_t(i)
            end if
            denom = denom + γ_t(i)
        next t
        b_i(j) = numer/denom
    next j
next i
```

7. Compute $\log\big(P(\mathcal{O}\,|\,\lambda)\big)$

```
logProb = 0
for i = 0 to T − 1
    logProb = logProb + log(c_i)
next i
logProb = −logProb
```

8. To iterate or not to iterate, that is the question.

```
iters = iters + 1
δ = |logProb − oldLogProb|
if(iters < minIters or δ > ε) then
    oldLogProb = logProb
    goto 3.
else
    return λ = (A, B, π)
end if
```

Алгоритм Вітербі

**Algorithm 2.5** Dynamic programming without underflow

```
Given:
    Model λ = (A, B, π)
    Observations 𝒪 = (O_0, 𝒪_1, ..., 𝒪_{T-1})
for i = 0, 1, ..., N - 1 do
    δ̂_0(i) = log(π_i b_i(𝒪_0))
end for
for t = 1, 2, ..., T - 1 do
    for i = 0, 1, ..., N - 1 do
        δ̂_t(i) = max_{j∈{0,1,...,N-1}} (δ̂_{t-1}(j) + log(a_{ji}) + log(b_i(𝒪_t)))
    end for
end for
```

Where:

$\lambda = (A, B, \pi)$, $\mathcal{O} = (O_0, \mathcal{O}_1, \ldots, \mathcal{O}_{T-1})$

$\widehat{\delta}_0(i) = \log\big(\pi_i b_i(\mathcal{O}_0)\big)$

$$\widehat{\delta}_t(i) = \max_{j \in \{0,1,\ldots,N-1\}} \Big(\widehat{\delta}_{t-1}(j) + \log(a_{ji}) + \log\big(b_i(\mathcal{O}_t)\big)\Big)$$