# HMM advanced

Only proceed here if you have answered the open questions in the Quiz!

## Sequence generator

Implement a generator that can generate sequences from given transmission and emission matrices. We provide a rudimentary skeleton script *sequence_generator.py* that you can use. You also need to modify the argument parser and main function. Add an argument for the number of the sequences. Write your generated sequences to a single fasta file. You can use utility functions from *hmm_utility.py* (e.g. *load_tsv*). You may also use the *numpy* function *choice*. Look up its **documentation (https://docs.scipy.org/doc/numpy-1.15.0/reference/generated/numpy.random.choice.html)** online.
Your script has to be uploaded to Codegrade and will be graded manually, but it needs to be functioning in order to be considered for grading. Auto-testing will be performed in the following way:

```
python3 sequence_generator.py A.tsv E.tsv
```

That means it only takes two positional arguments, a transition matrix and an emission matrix.

## Viterbi training

Implement the Viterbi training algorithm as described on page 66 in *Durbin et al.* We provide a skeleton script called *viterbi_training.py*. It takes care of the input and output, while you need to implement the actual algorithm and convergence. You should also use the Viterbi algorithm implementation from *hmm.py.* In contrast to the Baum-Welch algorithm you do not need to maximise for the log likelihood, but you can track whether your transmission and emission parameters change after an iteration. Due to the nature of the Viterbi algorithm there is a discrete solution. Think about why that is.
Upload your script to Codegrade for continuous feedback.

*Note: make sure to upload your hmm.py alongside viterbi_training.py if you import the viterbi algorithm from hmm.py.*