30/30

Will Argus
A12802324

ECE 208 Homework 4

Problem 1:

Time complexity: The beginning of the function is list creation so it is not considered. Next are 4 for loops, of complexity *O(n)* for the first three and *O(k)* for the last one, where n and k are length of sequence and number of matching states in the Hidden Markov Model, respectively. The nested for loops that come next are of complexity *O(nk)*, as they iterate through all characters in the sequence for every matching state in the HMM. After this comes a for loop of complexity *O(k)*, and finally, after a large amount of if statements, there is a for loop for finding the alignment. This for loop has order *O(n)* if the length of the sequence is greater than the number of matching states in the HMM and *O(k)* if the number if matching states is greater than the length of the sequence. Overall this term doesn't matter though, because the complexity is dominated by the two nested for loops:

O(n)+ O(n)+ O(n)+ O(k)+ O(nk)+ O(k)+ max(O(n), O(k))).

Meaning that the time complexity is: *O(nk),* (where n and k are length of sequence and number of matching states in the Hidden Markov Model, respectively.)

Problem 2:

Time complexity: As the forward algorithm is very similar to the Viterbi algorithm in terms of execution, with the exception of using the log_sum_exp() function in place of the maximum function used in the Viterbi algorithm. As such, the time complexity is the same: ***O(nk),*** (where n and k are length of sequence and number of matching states in the Hidden Markov Model, respectively.)

Recursive formula of the Forward algorithm:

Where log_sum_exp is defined in HMM.py

$$V_j^M(i) = \log e_j^M(x_i) + \text{log\_sum\_exp} \begin{cases} V_{j-1}^M(i-1) + \log a_{M_{j-1}M_j} \\ V_{j-1}^I(i-1) + \log a_{I_{j-1}M_j} \\ V_{j-1}^D(i) + \log a_{D_{j-1}M_j} \end{cases}$$

$$V_j^I(i) = \log e_j^I(x_i) + \text{log\_sum\_exp} \begin{cases} V_j^M(i-1) + \log a_{M_jI_j} \\ V_j^I(i-1) + \log a_{I_jI_j} \\ V_j^D(i) + \log a_{D_jI_j} \end{cases}$$

$$V_j^D[i] = \text{log\_sum\_exp} \begin{cases} V_{j-1}^M(i-1) + \log a_{M_{j-1}D_j} \\ V_{j-1}^I(i-1) + \log a_{I_{j-1}D_j} \\ V_{j-1}^D(i) + \log a_{D_{j-1}D_j} \end{cases}$$

Problem 3:

Fscore was chosen to be used over Vscore because it used log sum so it took into account all paths, not just the most likely path. Since when trying to match a query to an HMM model, a better choice is the score that takes all paths into account and gives the overall likelihood. Another way to say it is that a given query could have 1 path through a model which generated a good Vscore, but no other paths. Thus that HMM model may not be the best for that query, but the Vscore would never show that. The Fscore, on the other hand, takes into account all paths. So another HMM model with several paths for the query, even if none of the paths had an individual higher score than the Vscore, would be a better choice for the HMM model for that query since overall it is more likely. This higher probability would only be reflected in the Fscore, hence why it was chosen over the Vscore. It is worth noting, that had the Vscore been use, the query and HMM model assignments would have still come out the same.

Appendix

For reference: the query number and its corresponding HMM model, with Vscore and Fscore shown in that order.

```
-m Data/PF00937.hmm -q Data/queries.fas
3 -991.41601 -985.77977
```

```
-m Data/PF01635.hmm -q Data/queries.fas
9 -490.42536 -487.42458
```

```
-m Data/PF02723.hmm -q Data/queries.fas
8 -156.82297 -154.74157
```

```
-m Data/PF08779.hmm -q Data/queries.fas
4 -212.5441 -211.76325


-m Data/PF09399.hmm -q Data/queries.fa
7 -157.88638 -157.87882


-m Data/PF11289.hmm -q Data/queries.fas
2 -457.35524 -456.0625


-m Data/PF11395.hmm -q Data/queries.fas
5 -54.01351 -54.01263


-m Data/PF12093.hmm -q Data/queries.fas
6 -251.98692 -251.54712



-m Data/PF12133.hmm -q Data/queries.fas
1 -97.86634 -97.8384


-m Data/PF17635.hmm -q Data/queries.fas
10 -122.72555 -122.66593



-m Data/PF19219.hmm -q Data/queries.fas
```