### Decoding Problem

*Find an optimal hidden path in an HMM given a string of its emitted symbols.*

**Input:** A string *x* emitted by an HMM (*Σ*, *States*, *Transition*, *Emission*).

**Output:** A path *π* that maximizes the probability Pr(*x*, *π*) over all possible paths through this HMM. Please be as close to the book as possible.

**Input Format.** The first line of the input contains the outcome string *x*. The second line of the input is “--------” (a delimiter). The third line of the input is the list of symbols in the alphabet *Σ* (space-separated). The fourth line of the input is “--------” (a delimiter). The fifth line of the input is the list of states *States* (space-separated). The sixth line of the input is “--------” (a delimiter). The next |*States*|+1 lines are the transition matrix *Transition*, depicted as a tab-delimited |*States*| by |*States*| matrix as shown in the sample dataset. The next line is “--------” (a delimiter). The remaining lines are the emission matrix *Emission*, depicted as a tab-delimited |*States*| by |*Σ*| matrix as shown in the sample dataset. You may assume that transitions from the initial state occur with equal probability.

**Output Format.** A path *π* that maximizes the probability Pr(*x*, *π*) over all possible paths through this HMM. Each probability should be written to at least 3 decimal places. (**Note:** more than one solution may exist, in which case you may output any one.)

**Constraints.** |*x*| = |*π*| = 100; 2 ≤ |*States*| ≤ 4; |*Σ*| = 3

**SAMPLE DATASET:**

Input:

xyxzz

--------

x y z

--------

A B

--------

A B

A 0.641 0.359

B 0.729 0.271

--------

x y z

A 0.117 0.691 0.192

B 0.097 0.42 0.483

Output:

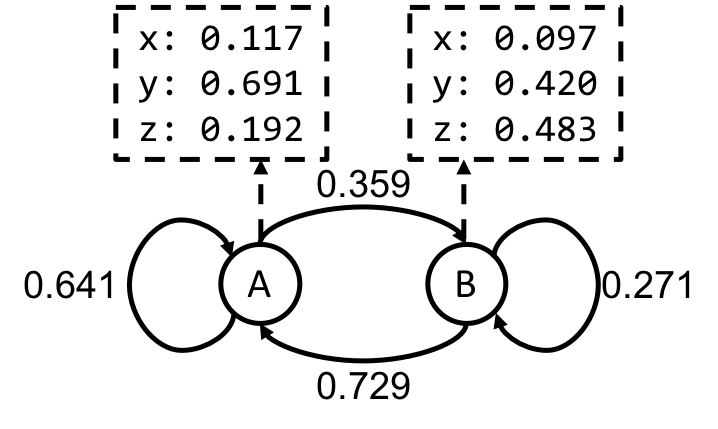
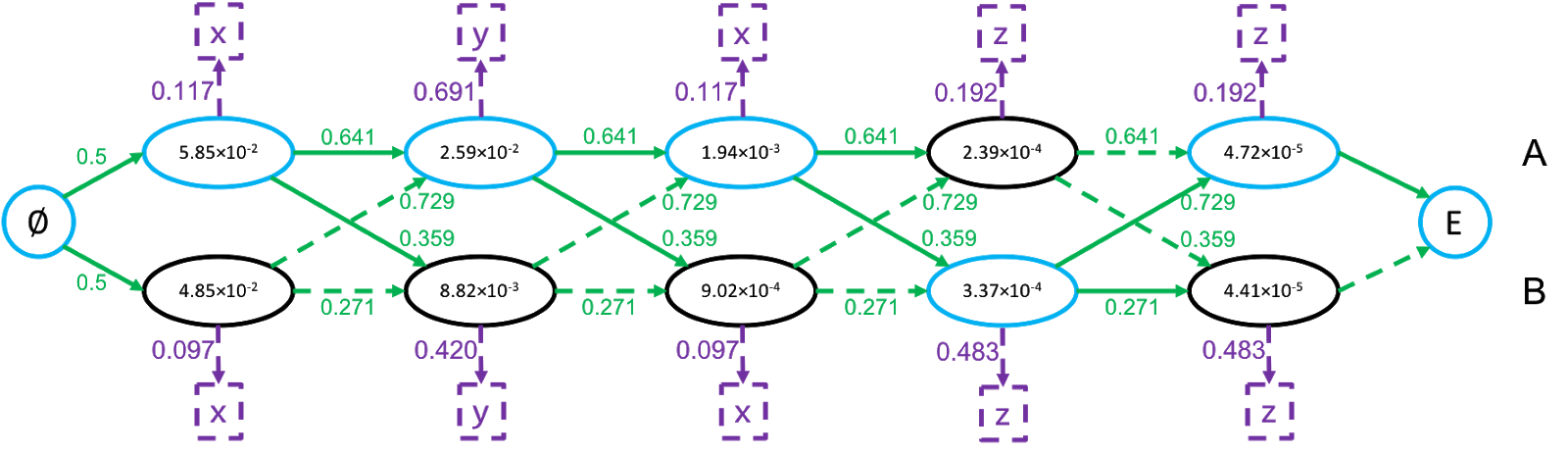
AAABA

The first figure below is a visualization of this HMM, in which transitions are shown as solid edges (labeled by transition probabilities) and emissions are shown as dashed lines (labeled by emission probabilities). AAABA is a hidden path *π* that maximizes the (unconditional) probability Pr(*x*, *π*) for the string *x* = xyxzz.

The second figure below represents the Dynamic Programming graph solving the problem using the Viterbi algorithm, in which row *i* and column *j* denotes the probability of the highest-probability path in which the *j*-th letter of *x* was emitted while in state *i*. Green edges denote probabilities *transition*(*πi*-1,*πi*), and purple edges denote probabilities *emission*(*πi*,*xi*). The actual weight on a given edge from node (*l*,*i*-1) to node (*k*,*i*) has *weighti*(*l*,*k*) = *transition*(*πi*-1,*πi*) \* *emission*(*πi*,*xi*). The solid edge pointing to node (*k*,*i*) denotes the last edge in the hidden path *π* = *π*1...*k* maximizing Pr(*x*,*π*). Thus, the solution is denoted by the path from the start node to the end node that only contains solid edges. The nodes along this optimal path have been highlighted in blue.

For example, the value in node (A,2) is:

max{(5.85×10-2×0.641×0.691), (4.85×10-2×0.729×0.691)} = 2.59×10-2

****

**TEST DATASET 1:**

Input:

xyxy

--------

x y

--------

A B

--------

A B

A 0.5 0.5

B 0.5 0.5

--------

x y

A 0.1 0.9

B 0.9 0.1

Output:

BABA

This dataset makes sure that your code is correctly using emission probabilities when determining the optimal path for an emitted string. All transition probabilities are equal, so the optimal path is dependent on the emission probabilities at each state. If your code does not output the correct answer for this dataset it is likely that your code is incorrectly using the input emission probability matrix. Make sure that your recurrence is for state s and character k. Note that the emission probability used is the probability of the current state emitting the current character, not the previous state emitting the current or previous character. Mixing up which emission probability to use is a common mistake.

**TEST DATASET 2:**

Input:

xyxy

--------

x y

--------

A B

--------

A B

A 0.9 0.1

B 0.1 0.9

--------

x y

A 0.5 0.5

B 0.5 0.5

Output:

AAAA OR BBBB

This dataset makes sure that your code is correctly using transition probabilities in the calculation of the optimal path. More specifically, this dataset checks to make sure that your solution can handle self-transitions out of states. If your code doesn’t output AAAA or BBBB then your code likely mishandles self-transition probabilities. Be sure that your implementation of the Viterbi algorithm considers the current state as a possible previous state.

**TEST DATASET 3:**

Input:

x

--------

x y

--------

A B

--------

A B

A 0.4 0.6

B 0.2 0.8

--------

x y

A 0.55 0.45

B 0.5 0.5

Output:

A

This dataset makes sure that your code can handle inputs in which the string *x* is only one character long. The output should be the state that has the maximum probability of emitting the one character in string *x*. In this dataset the state A has the highest probability of emitting character x, so the output is just state A. If your output doesn’t match the correct output make sure that your implementation doesn’t include off-by-one indexing errors that could cause characters in string *x* to be ignored.

**TEST DATASET 4:**

Input:

zxyxy

--------

x y z

--------

A

--------

A

A 1

--------

x y z

A 0.5 0.5 0

Output:

AAAAA

This dataset makes sure that your code can handle inputs in which there is only one possible state. The correct answer has a probability of 0 of emitting string *x* but is still correct since there’s no other possible path. If your output doesn’t match the correct output make sure your code does not exclude paths with a probability of 0 in determining final answers.

**TEST DATASET 5:**

Input:

xx

--------

x y

--------

A B C

--------

A B C

A 0.7 0.1 0.2

B 0.5 0.3 0.2

C 1 0 0

--------

x y

A 0 1

B 0.5 0.5

C 1 0

Output:

BC

This dataset makes sure that your code is not using a greedy approach to finding the optimal path for the given string *x*. Greedy approaches that pick the state that maximizes the output probability will first choose state C, since it is guaranteed to emit an x character. The only possible transition from state C is to state A, which cannot emit an x character, so the final path CA from a greedy algorithm will have a 0 probability of emitting xx. If your code outputs CA instead of BC it is likely that you are implementing a greedy algorithm.