

# Hidden Markov Models

Slides by Carl Kingsford

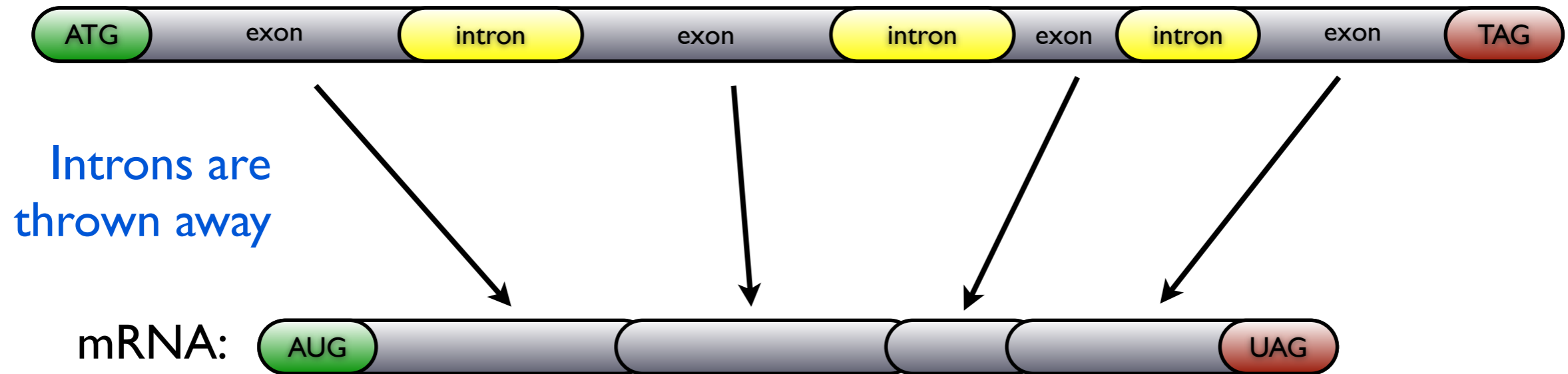
Based on Chapter 11 of Jones & Pevzner, *An Introduction to Bioinformatics Algorithms*

# Eukaryotic Genes & Exon Splicing

Prokaryotic (bacterial) genes look like this:



Eukaryotic genes usually look like this:



This spliced RNA is what is translated into a protein.

# Checking a Casino



Fair coin:  
 $\text{Pr}(\text{Heads}) = 0.5$



Biased coin:  
 $\text{Pr}(\text{Heads}) = 0.75$



Suppose either a fair or biased coin was used to generate a sequence of heads & tails. But we don't know which type of coin was actual used.

Heads/Tails: ↑ ↑ ↓ ↓ ↓ ↓ ↑ ↑ ↑ ↑ ↓ ↑ ↓ ↑ ↓ ↑

# Checking a Casino



Fair coin:  
 $\text{Pr}(\text{Heads}) = 0.5$



Biased coin:  
 $\text{Pr}(\text{Heads}) = 0.75$



Suppose either a fair or biased coin was used to generate a sequence of heads & tails. But we don't know which type of coin was actual used.

Heads/Tails: ↑ ↑ ↓ ↓ ↓ ↓ ↑ ↑ ↑ ↑ ↓ ↑ ↓ ↑ ↓ ↑

# Checking a Casino



Fair coin:  
 $\text{Pr}(\text{Heads}) = 0.5$



Biased coin:  
 $\text{Pr}(\text{Heads}) = 0.75$




Suppose either a fair or biased coin was used to generate a sequence of heads & tails. But we don't know which type of coin was actual used.

Heads/Tails: ↑ ↑ ↓ ↓ ↓ ↓ ↑ ↑ ↑ ↑ ↓ ↑ ↓ ↑ ↓ ↑

How could we guess which coin was more likely?

# Compute the Probability of the Observed Sequence

Fair coin:  $\Pr(\text{Heads}) = 0.5$   
Biased coin:  $\Pr(\text{Heads}) = 0.75$

$x =$        

$\Pr(x \mid \text{Fair}) =$  **0.5** **0.5** **0.5** **0.5** **0.5** **0.5** **0.5**

$\Pr(x \mid \text{Biased}) =$  **0.75** **0.75** **0.25** **0.25** **0.25** **0.25** **0.75**

# Compute the Probability of the Observed Sequence

Fair coin:  $\Pr(\text{Heads}) = 0.5$   
Biased coin:  $\Pr(\text{Heads}) = 0.75$

$x = \uparrow \uparrow \downarrow \downarrow \downarrow \downarrow \uparrow$

$$\Pr(x \mid \text{Fair}) = 0.5 \times 0.5 \times 0.5 \times 0.5 \times 0.5 \times 0.5 \times 0.5 = 0.5^7 = 0.0078125$$

$$\Pr(x \mid \text{Biased}) = 0.75 \times 0.75 \times 0.25 \times 0.25 \times 0.25 \times 0.25 \times 0.75 = 0.001647949$$

# Compute the Probability of the Observed Sequence

Fair coin:  $\Pr(\text{Heads}) = 0.5$   
Biased coin:  $\Pr(\text{Heads}) = 0.75$

$x = \uparrow \uparrow \downarrow \downarrow \downarrow \downarrow \uparrow$

$$\Pr(x \mid \text{Fair}) = 0.5 \times 0.5 \times 0.5 \times 0.5 \times 0.5 \times 0.5 \times 0.5 = 0.5^7 = 0.0078125$$

$$\Pr(x \mid \text{Biased}) = 0.75 \times 0.75 \times 0.25 \times 0.25 \times 0.25 \times 0.25 \times 0.75 = 0.001647949$$

The *log-odds* score:

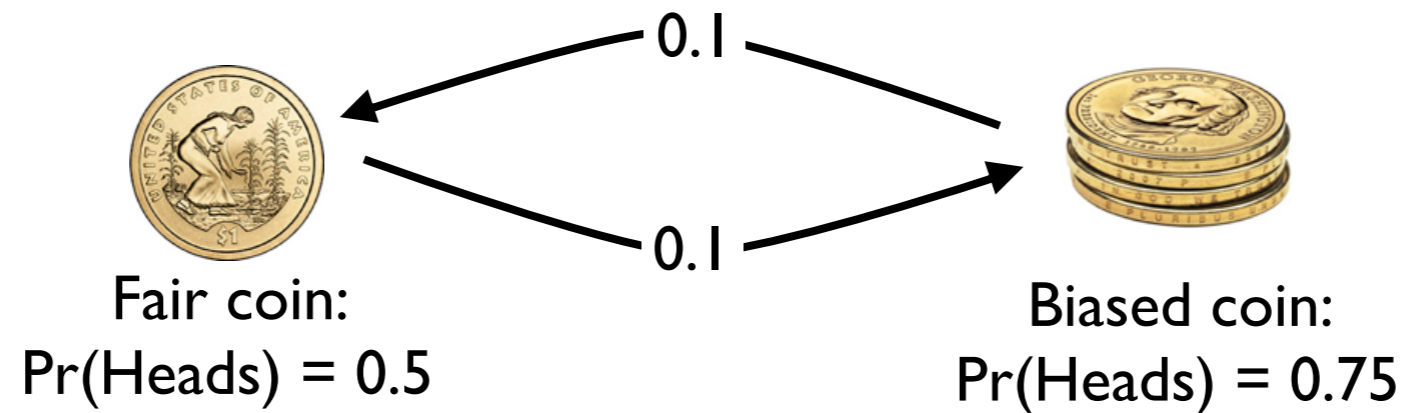
$$\log_2 \frac{\Pr(x \mid \text{Fair})}{\Pr(x \mid \text{Biased})} = \log_2 \frac{0.0078}{0.0016} = 2.245$$

> 0. Hence “Fair” is a better guess.



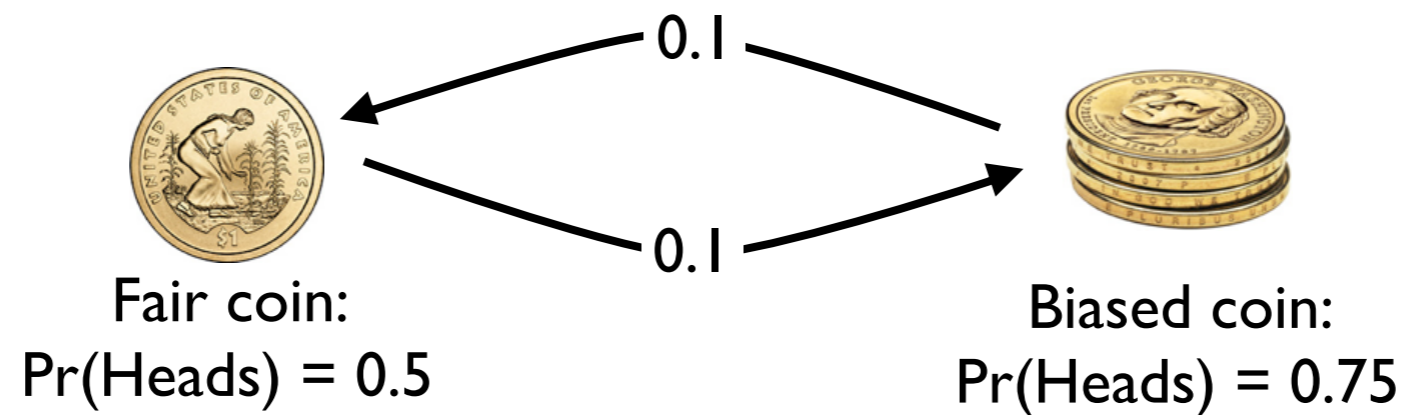
# What if the casino switches coins?

Fair coin:  $\Pr(\text{Heads}) = 0.5$   
Biased coin:  $\Pr(\text{Heads}) = 0.75$   
Probability of switching coins = 0.1



# What if the casino switches coins?

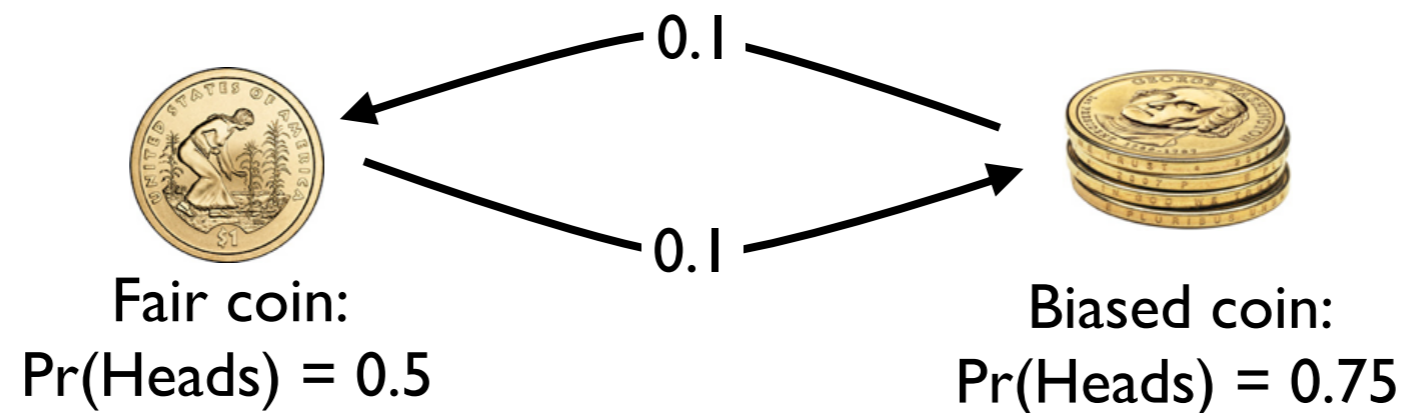
Fair coin:  $\Pr(\text{Heads}) = 0.5$   
Biased coin:  $\Pr(\text{Heads}) = 0.75$   
Probability of switching coins = 0.1



How can we compute the probability of the entire sequence?

# What if the casino switches coins?

Fair coin:  $\Pr(\text{Heads}) = 0.5$   
Biased coin:  $\Pr(\text{Heads}) = 0.75$   
Probability of switching coins = 0.1



How can we compute the probability of the entire sequence?

How could we guess which coin was more likely at each position?

What does this have to do with  
biology?

atg gat ggg agc aga tca gat cag atc agg gac gat aga cga tag tga

# What does this have to do with biology?

Before:

How likely is it that this sequence was generated by a fair coin?

Which parts were generated by a biased coin?

atg gat ggg agc aga tca gat cag atc agg gac gat aga cga tag tga

# What does this have to do with biology?

Before:

How likely is it that this sequence was generated by a fair coin?

Which parts were generated by a biased coin?

Now:

How likely is it that this is a gene?

Which parts are the start, middle and end?

atg gat ggg agc aga tca gat cag atc agg gac gat aga cga tag tga

# What does this have to do with biology?

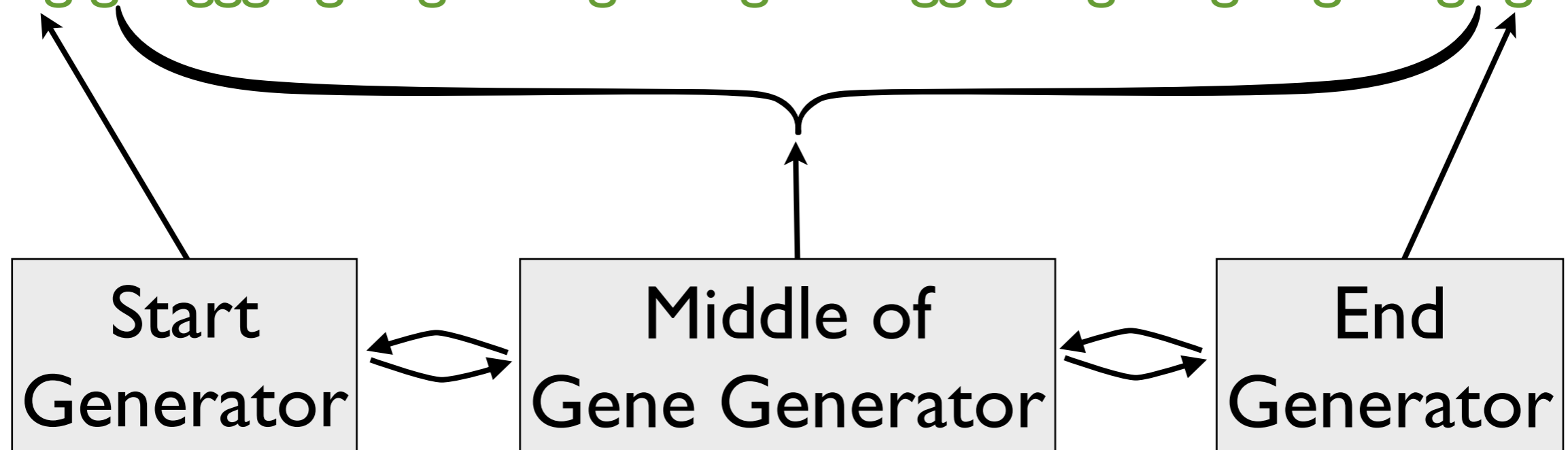
Before:

How likely is it that this sequence was generated by a fair coin?  
Which parts were generated by a biased coin?

Now:

How likely is it that this is a gene?  
Which parts are the start, middle and end?

atg gat ggg agc aga tca gat cag atc agg gac gat aga cga tag tga

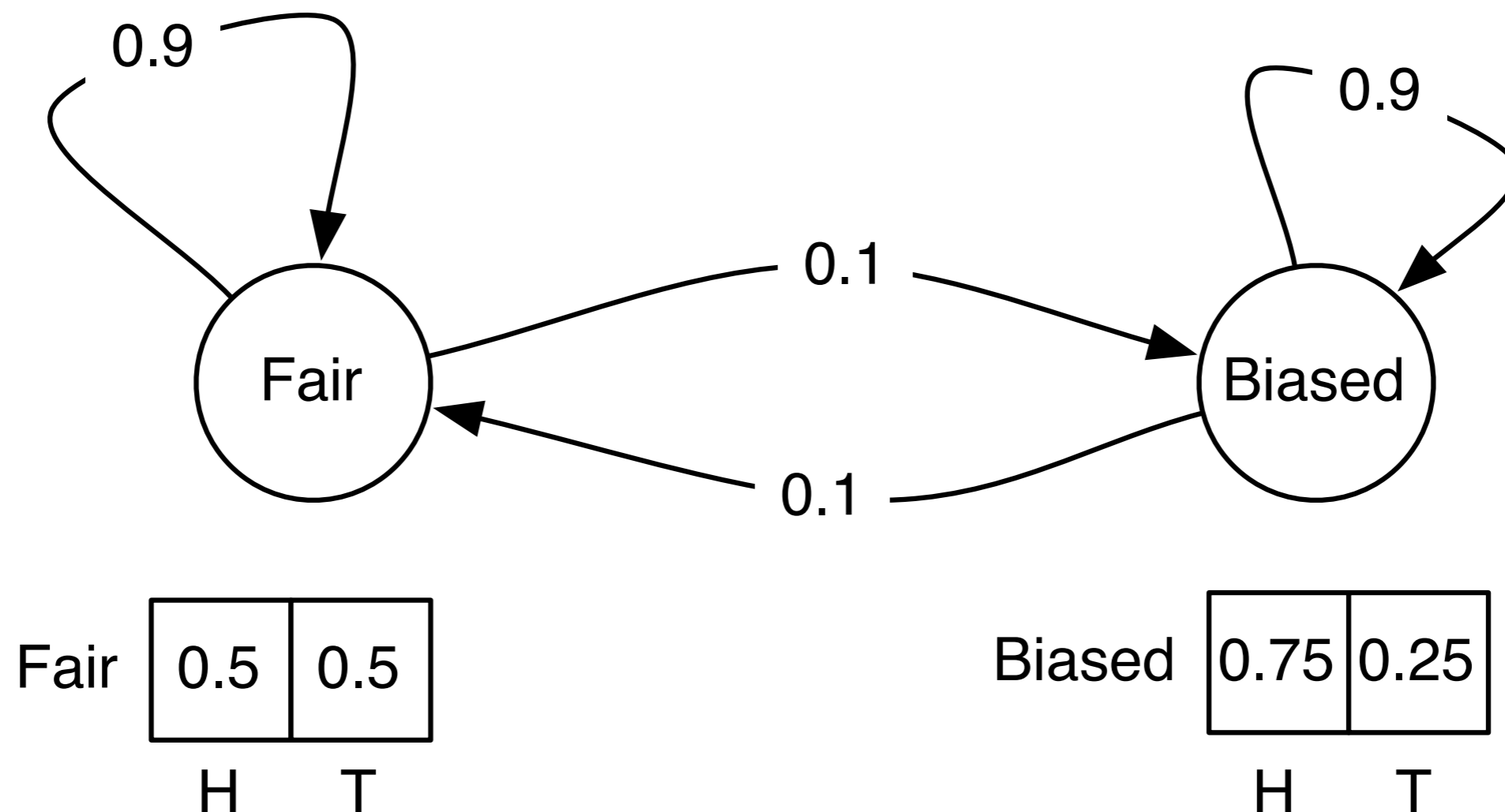


# Hidden Markov Model (HMM)

Fair coin:  $\text{Pr}(\text{Heads}) = 0.5$

Biased coin:  $\text{Pr}(\text{Heads}) = 0.75$

Probability of switching coins = 0.1





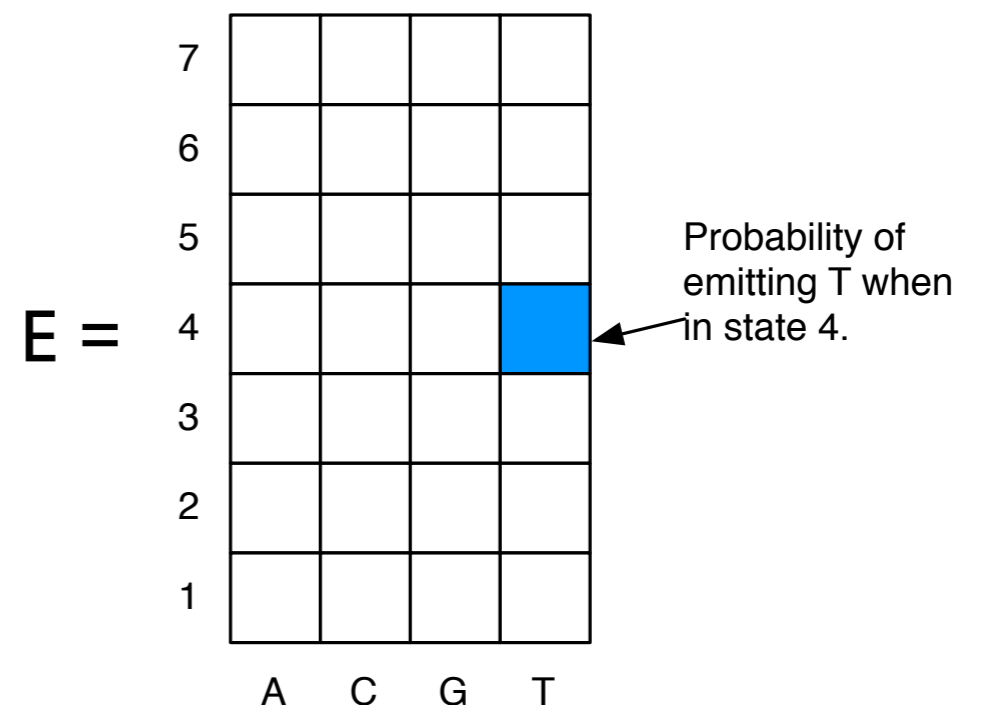
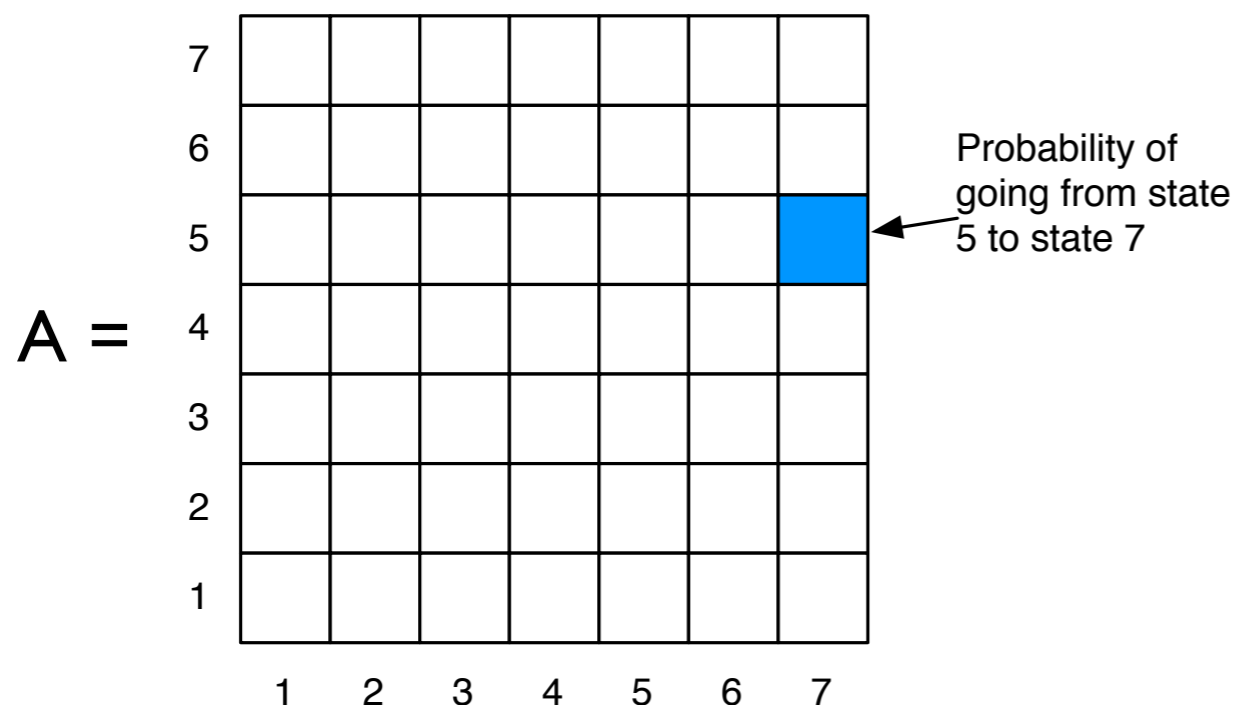
# Formal Definition of a HMM

$\Sigma$  = alphabet of symbols.

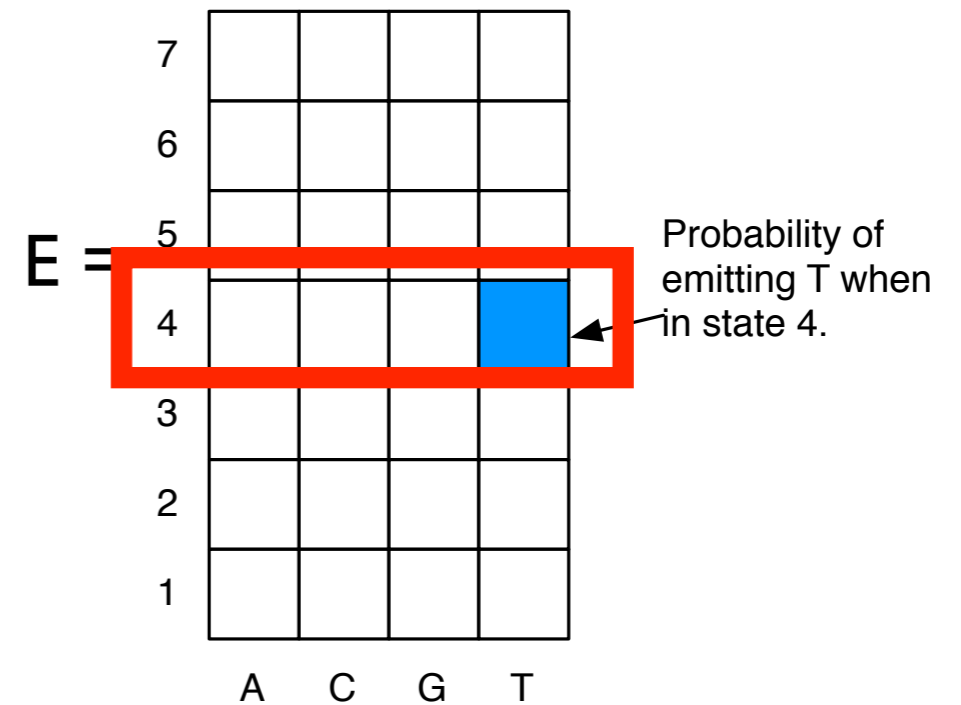
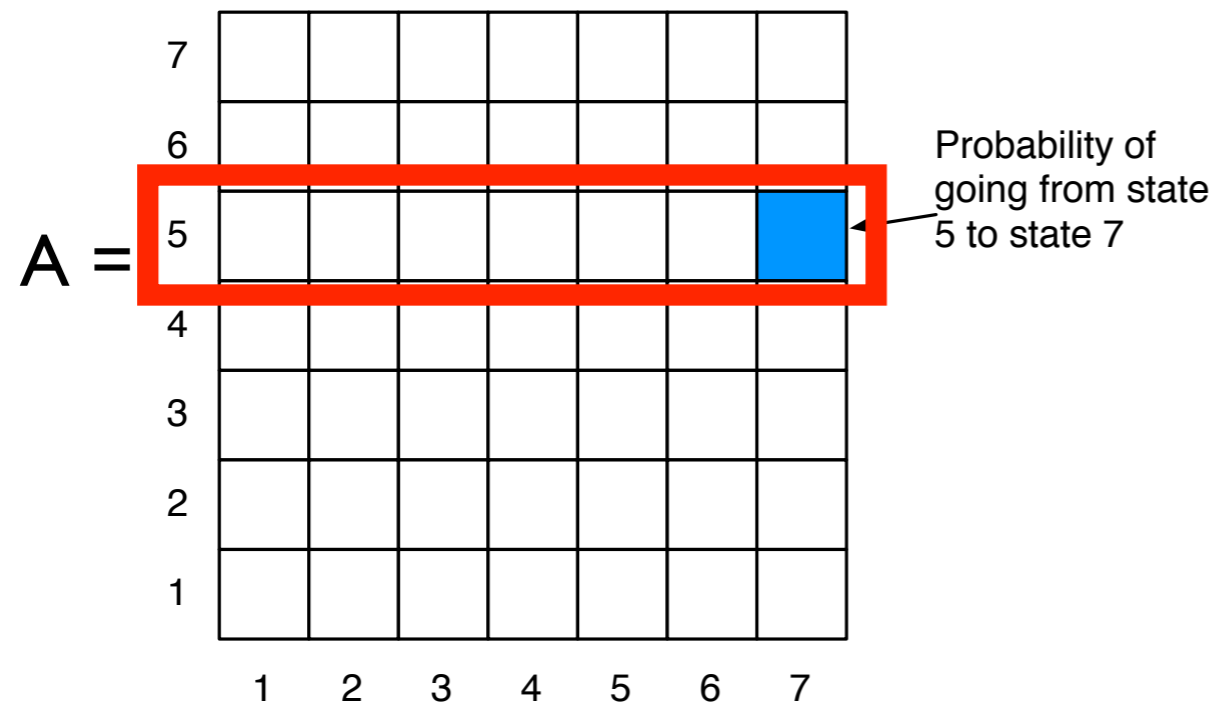
$Q$  = set of states.

$A$  = an  $|Q| \times |Q|$  matrix where entry  $(k,l)$  is the probability of moving from state  $k$  to state  $l$ .

$E$  = a  $|Q| \times |\Sigma|$  matrix, where entry  $(k,b)$  is the probability of emitting  $b$  when in state  $k$ .

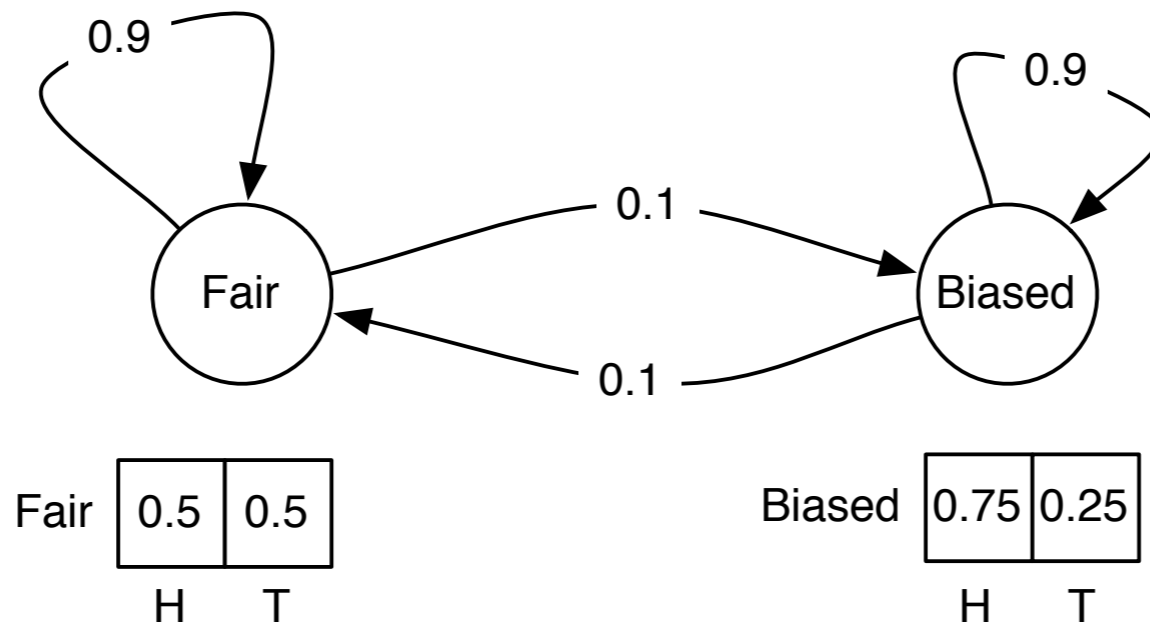


# Constraints on A and E



Sum of the # in each row must be 1.

# Computing Probabilities Given Path



$x =$      ↓   ↑   ↓   ↑   ↑   ↑   ↓   ↑   ↑   ↓

$\pi =$      **F**   **F**   **F**   **B**   **B**   **B**   **B**   **F**   **F**   **F**

$\Pr(x_i | \pi_i) =$      0.5   0.5   0.5   0.75   0.75   0.75   0.25   0.5   0.5   0.5

$\Pr(\pi_i \rightarrow \pi_{i+1}) =$  0.1   0.9   0.9   0.1   0.9   0.9   0.9   0.1   0.1   0.1

# The Decoding Problem

Given  $x$  and  $\pi$ , we can compute:

- $\Pr(x \mid \pi)$ : product of  $\Pr(x_i \mid \pi_i)$
- $\Pr(\pi)$ : product of  $\Pr(\pi_i \rightarrow \pi_{i+1})$
- $\Pr(x, \pi)$ : product of all the  $\Pr(x_i \mid \pi_i)$  and  $\Pr(\pi_i \rightarrow \pi_{i+1})$

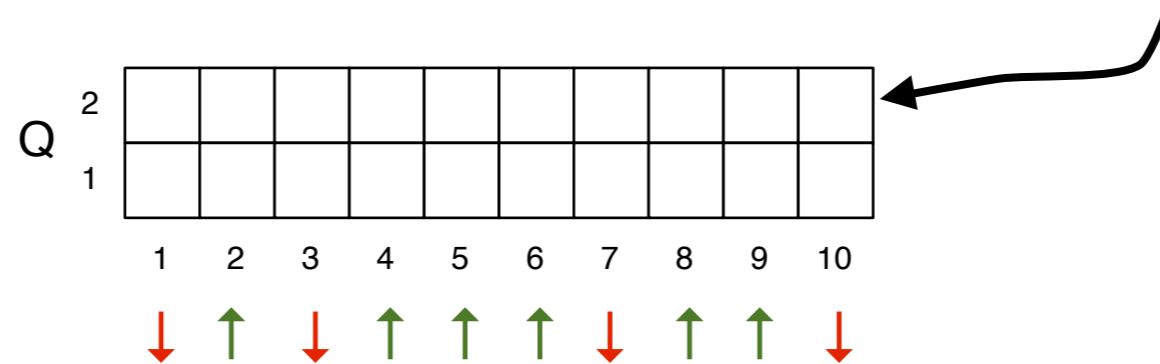
$$\Pr(x, \pi) = \Pr(\pi_0 \rightarrow \pi_1) \prod_{i=1}^n \Pr(x_i \mid \pi_i) \Pr(\pi_i \rightarrow \pi_{i+1})$$

But they are “hidden” Markov models because  $\pi$  is unknown.

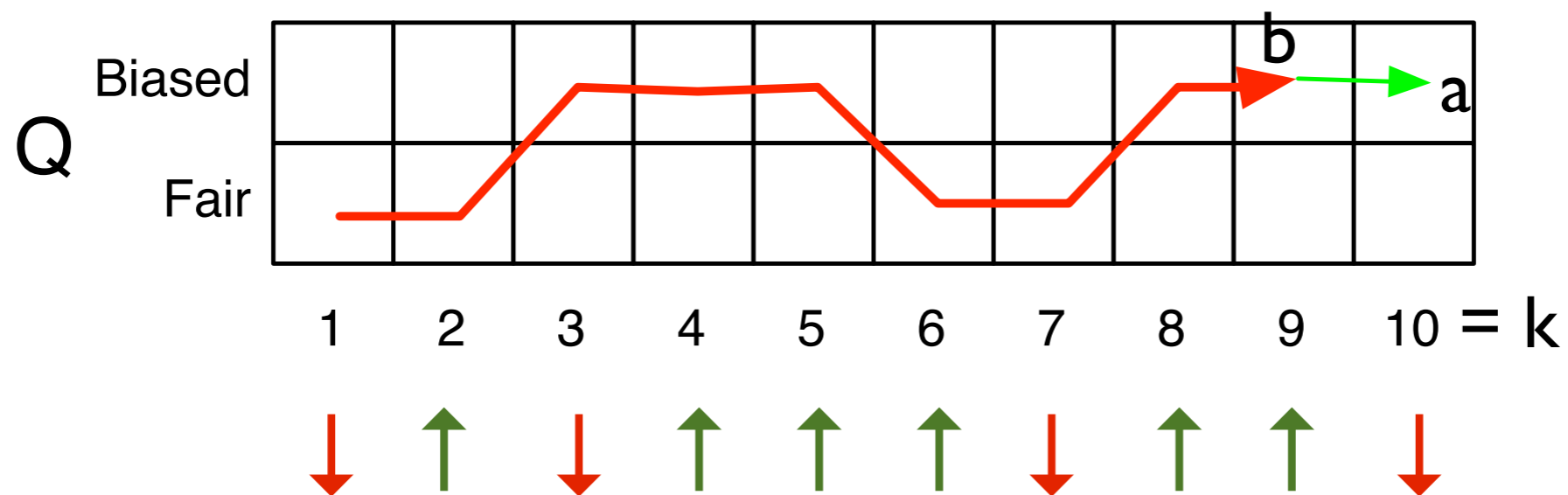
**Decoding Problem:** Given a sequence  $x_1x_2x_3\dots x_n$  generated by an HMM  $(\Sigma, Q, A, E)$ , find a path  $\pi$  that maximizes  $\Pr(x, \pi)$ .

# The Viterbi Algorithm to Find Best Path

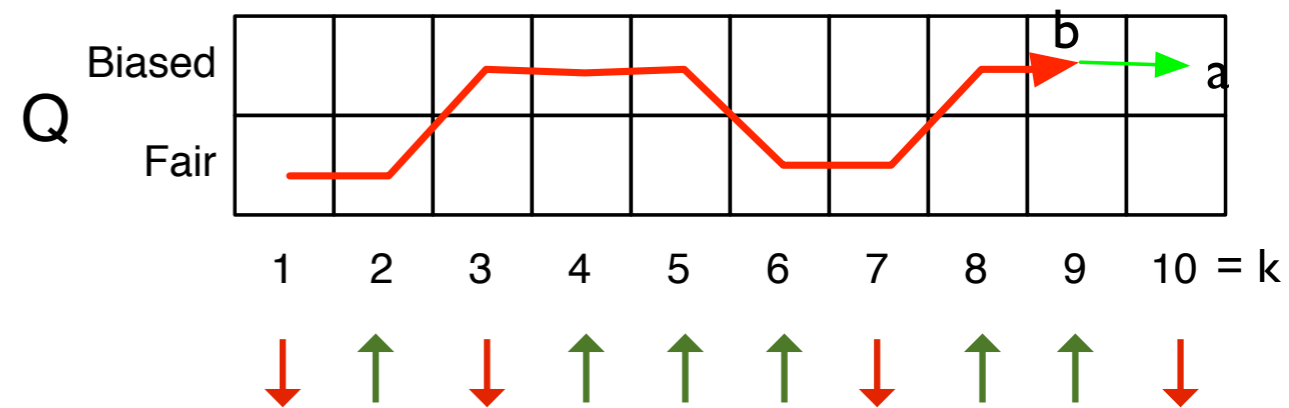
$A[a, k]$  := the probability of the **best** path for  $x_1 \dots x_k$  that ends at state  $a$ .



$A[a, k]$  = the path for  $x_1 \dots x_{k-1}$  that goes to some state  $b$  times cost of a transition from  $b$  to  $i$ , and then to output  $x_k$  from state  $a$ .



# Viterbi DP Recurrence



$$A[a, k] = \max_{b \in Q} \{ \underbrace{A[b, k - 1]}_{\text{Best path for } x_1..x_k \text{ ending in state } b} \times \underbrace{\text{Pr}(b \rightarrow a)}_{\text{Probability of transitioning from state } b \text{ to state } a} \times \underbrace{\text{Pr}(x_k \mid \pi_k = a)}_{\text{Probability of outputting } x_k \text{ given that the } k\text{th state is } a} \}$$

Over all possible previous states.

Best path for  $x_1..x_k$  ending in state  $b$

Probability of transitioning from state  $b$  to state  $a$

Probability of outputting  $x_k$  given that the  $k$ th state is  $a$ .

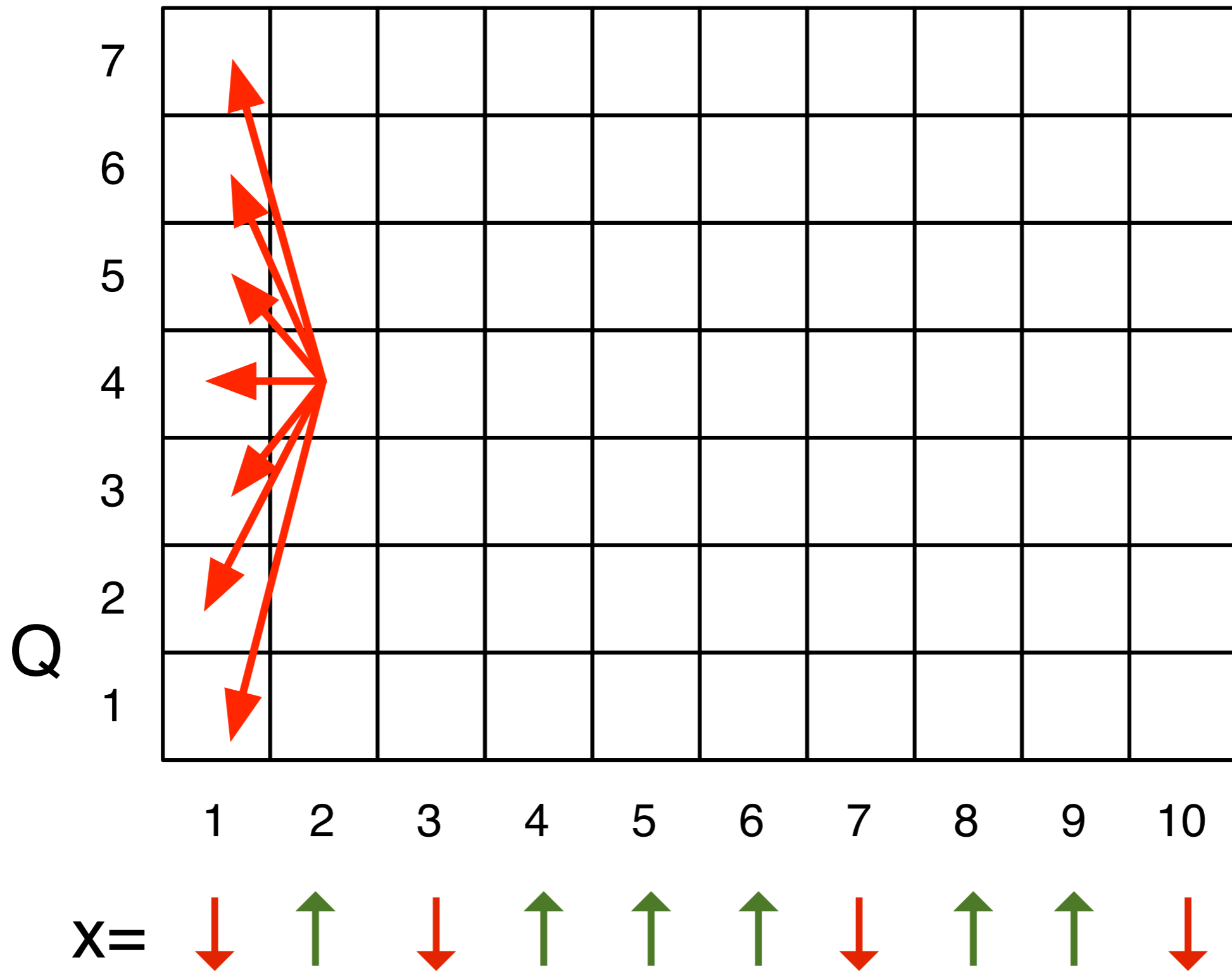
Base case:

$$A[a, 1] = \underbrace{\text{Pr}(\pi_1 = a)}_{\text{Probability that the first state is } a} \times \underbrace{\text{Pr}(x_1 \mid \pi_1 = a)}_{\text{Probability of emitting } x_1 \text{ given the first state is } a}$$

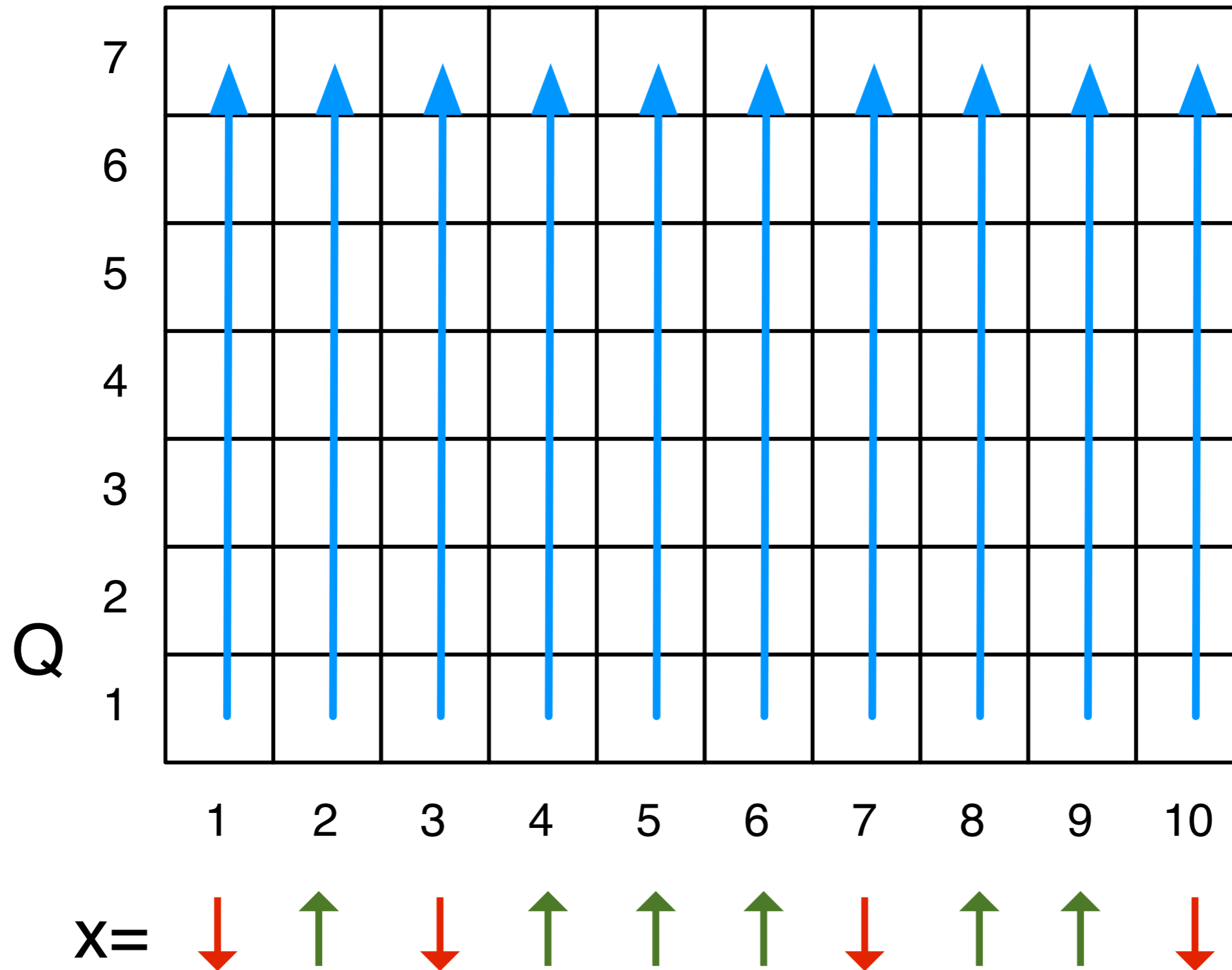
Probability that the first state is  $a$

Probability of emitting  $x_1$  given the first state is  $a$ .

# Which Cells Do We Depend On?

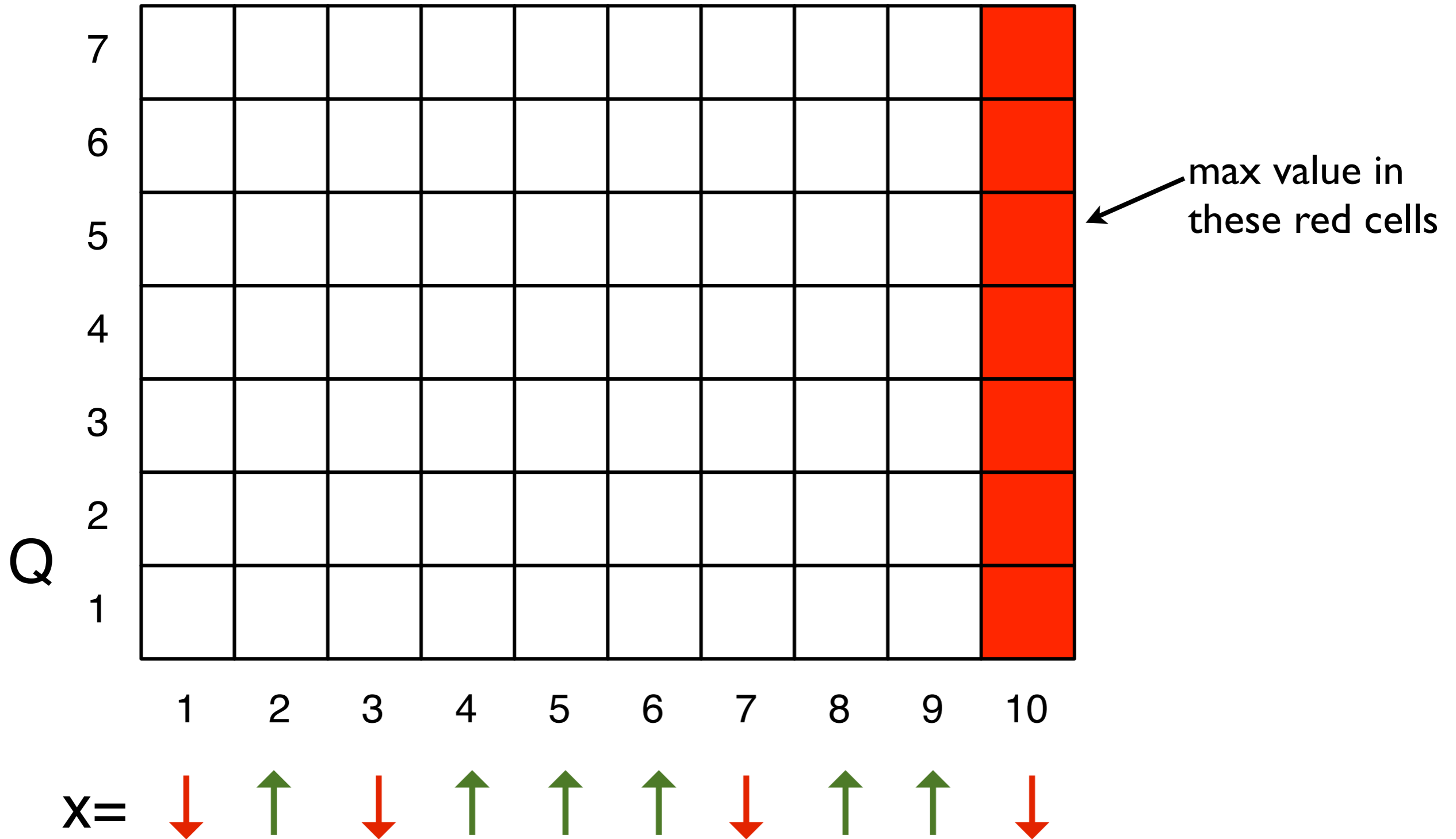


# Order to Fill in the Matrix:

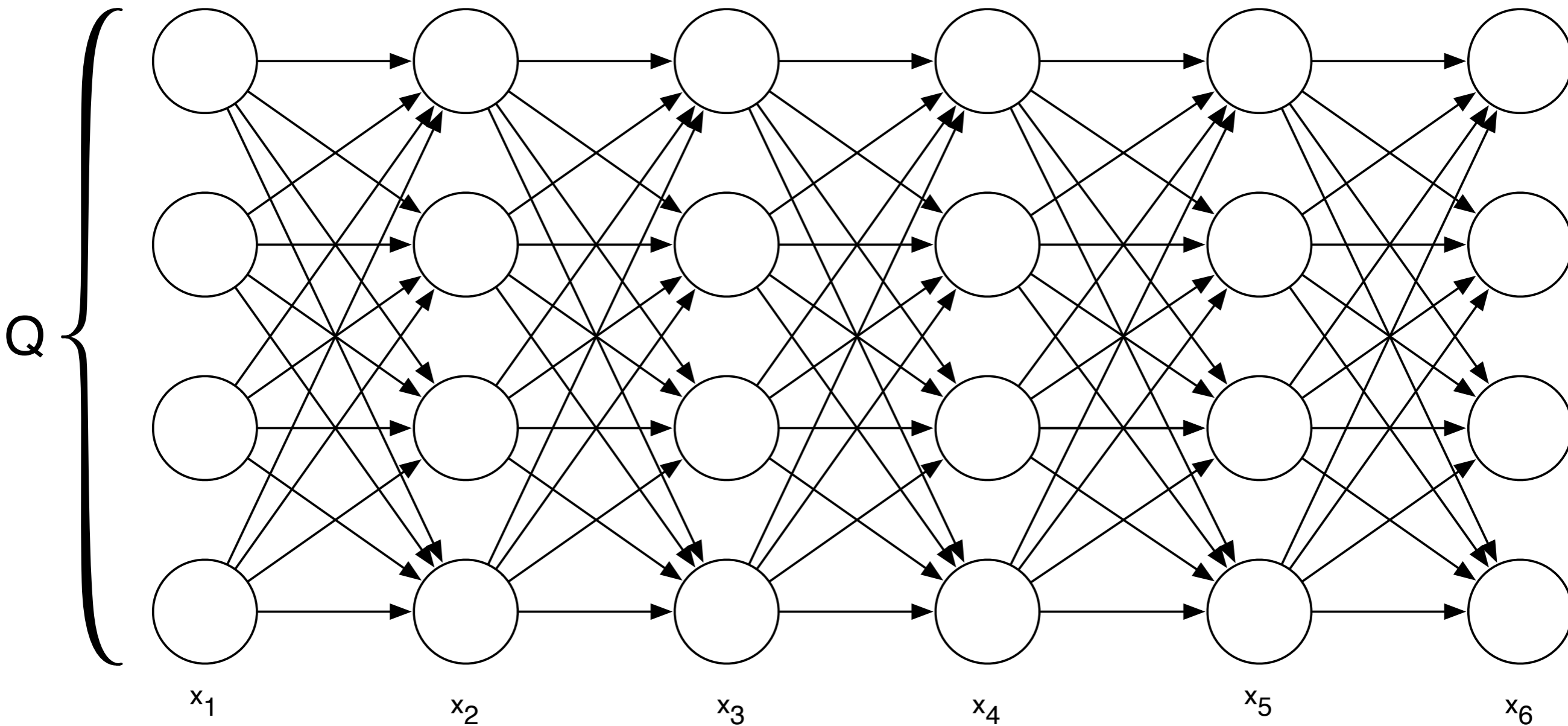




# Where's the answer?



# Graph View of Viterbi



# Running Time

- # of subproblems =  $O(n|Q|)$ , where  $n$  is the length of the sequence.
- Time to solve a subproblem =  $O(|Q|)$
- Total running time:  $O(n|Q|^2)$

# Using Logs

Typically, we take the log of the probabilities to avoid multiplying a lot of terms:

$$\begin{aligned}\log(A[a, k]) &= \max_{b \in Q} \{\log(A[b, k-1]) \times \Pr(b \rightarrow a) \times \Pr(x_k \mid \pi_k = a)\} \\ &= \max_{b \in Q} \{\log(A[b, k-1]) + \log(\Pr(b \rightarrow a)) + \log(\Pr(x_k \mid \pi_k = a))\}\end{aligned}$$

Remember:  $\log(ab) = \log(a) + \log(b)$

Why do we want to avoid multiplying lots of terms?

# Using Logs

Typically, we take the log of the probabilities to avoid multiplying a lot of terms:

$$\begin{aligned}\log(A[a, k]) &= \max_{b \in Q} \{\log(A[b, k - 1] \times \Pr(b \rightarrow a) \times \Pr(x_k \mid \pi_k = a))\} \\ &= \max_{b \in Q} \{\log(A[b, k - 1]) + \log(\Pr(b \rightarrow a)) + \log(\Pr(x_k \mid \pi_k = a))\}\end{aligned}$$

Remember:  $\log(ab) = \log(a) + \log(b)$

Why do we want to avoid multiplying lots of terms?

Multiplying leads to very small numbers:

$$0.1 \times 0.1 \times 0.1 \times 0.1 \times 0.1 = 0.00001$$

This can lead to underflow.

Taking logs and adding keeps numbers bigger.

# Estimating HMM Parameters

$$\begin{aligned} (\mathbf{x}^{(1)}, \boldsymbol{\pi}^{(1)}) &= \begin{matrix} x_1^{(1)} & x_2^{(1)} & x_3^{(1)} & x_4^{(1)} & x_5^{(1)} & \dots & x_n^{(1)} \\ \pi_1^{(1)} & \pi_2^{(1)} & \pi_3^{(1)} & \pi_4^{(1)} & \pi_5^{(1)} & \dots & \pi_n^{(1)} \end{matrix} \\ (\mathbf{x}^{(2)}, \boldsymbol{\pi}^{(2)}) &= \begin{matrix} x_1^{(2)} & x_2^{(2)} & x_3^{(2)} & x_4^{(2)} & x_5^{(2)} & \dots & x_n^{(2)} \\ \pi_1^{(2)} & \pi_2^{(2)} & \pi_3^{(2)} & \pi_4^{(2)} & \pi_5^{(2)} & \dots & \pi_n^{(2)} \end{matrix} \end{aligned}$$

Training examples  
where outputs and  
paths are known.

# of times transition  
 $a \rightarrow b$  is observed.

$$\Pr(a \rightarrow b) = \frac{A_{ab}}{\sum_{q \in Q} A_{aq}}$$

# of times  $x$  was  
observed to be  
output from state  $a$ .

$$\Pr(x | a) = \frac{E_{xa}}{\sum_{x \in \Sigma} E_{xa}}$$

# Pseudocounts

# of times transition  
 $a \rightarrow b$  is observed.

$$\Pr(a \rightarrow b) = \frac{A_{ab}}{\sum_{q \in Q} A_{aq}}$$

# of times  $x$  was  
observed to be  
output from state  $a$ .

$$\Pr(x | a) = \frac{E_{xa}}{\sum_{x \in \Sigma} E_{xa}}$$

What if a transition or emission is never observed in the training data?  
 $\Rightarrow$  0 probability

Meaning that if we observe an example with that transition or emission in the real world, we will give it 0 probability.

But it's unlikely that our training set will be large enough to observe every possible transition.

Hence: we take  $A_{ab} = (\text{\#times } a \rightarrow b \text{ was observed}) + 1$  ← “pseudocount”  
Similarly for  $E_{xa}$ .

# Viterbi Training

- **Problem:** typically, in the real world we only have examples of the output  $x$ , and we don't know the paths  $\pi$ .

## Viterbi Training Algorithm:

1. Choose a random set of parameters.
2. Repeat:
  1. Find the best paths.
  2. Use those paths to estimate new parameters.

This is an local search algorithm.

It's also an example of a "Gibbs sampling" style algorithm.

The Baum-Welch algorithm is similar, but doesn't commit to a single best path for each example.



# Some probabilities in which we are interested

What is the probability of observing a string  $x$  under the assumed HMM?

$$\Pr(x) = \sum_{\pi} \Pr(x, \pi)$$

What is the probability of observing  $x$  using a path where the  $i^{\text{th}}$  state is  $a$ ?

$$\Pr(x, \pi_i = a) = \sum_{\pi: \pi_i = a} \Pr(x, \pi)$$

What is the probability that the  $i^{\text{th}}$  state is  $a$ ?

$$\Pr(\pi_i = a | x) = \frac{\Pr(x, \pi_i = a)}{\Pr(x)}$$

# The Forward Algorithm

How do we compute this:

$$\Pr(x, \pi_k = a) = \Pr(x_1, \dots, x_i, \pi_i = a) \Pr(x_{i+1}, \dots, x_n \mid \pi_i = a)$$

Recall the recurrence to compute **best** path for  $x_1 \dots x_k$  that ends at state  $a$ :

$$A[a, k] = \max_{b \in Q} \{ A[b, k - 1] \times \Pr(b \rightarrow a) \times \Pr(x_k \mid \pi_k = a) \}$$

We can compute the probability of emitting  $x_1, \dots, x_k$  using **some** path that ends in  $a$ :

$$F[a, k] = \sum_{b \in Q} F[b, k - 1] \times \Pr(b \rightarrow a) \times \Pr(x_k \mid \pi_k = a)$$

# The Forward Algorithm

How do we compute this:

$$\Pr(x, \pi_k = a) = \Pr(x_1, \dots, x_i, \pi_i = a) \Pr(x_{i+1}, \dots, x_n \mid \pi_i = a)$$

Recall the recurrence to compute **best** path for  $x_1 \dots x_k$  that ends at state  $a$ :

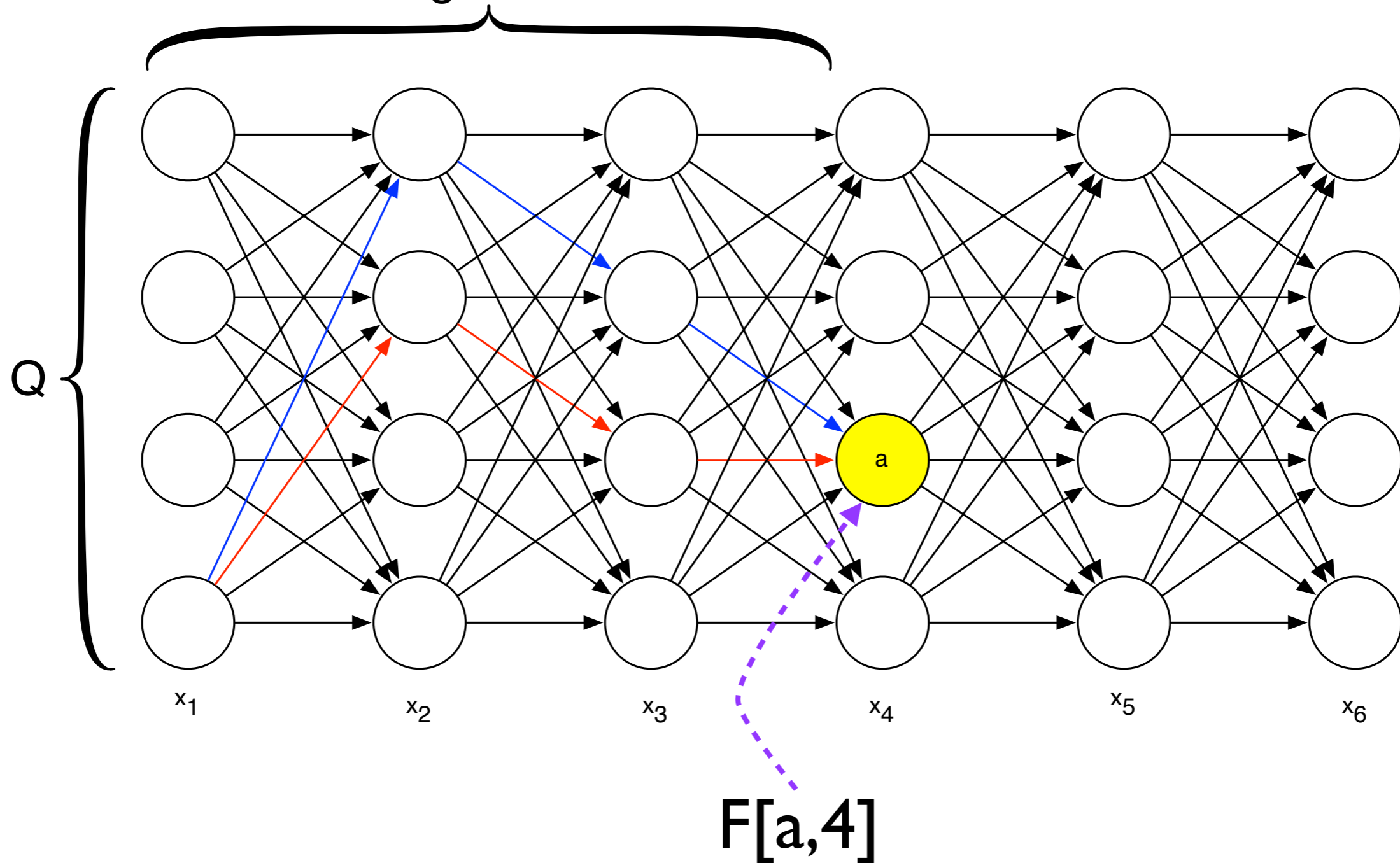
$$A[a, k] = \max_{b \in Q} \{ A[b, k - 1] \times \Pr(b \rightarrow a) \times \Pr(x_k \mid \pi_k = a) \}$$

We can compute the probability of emitting  $x_1, \dots, x_k$  using **some** path that ends in  $a$ :

$$F[a, k] = \sum_{b \in Q} F[b, k - 1] \times \Pr(b \rightarrow a) \times \Pr(x_k \mid \pi_k = a)$$

# The Forward Algorithm

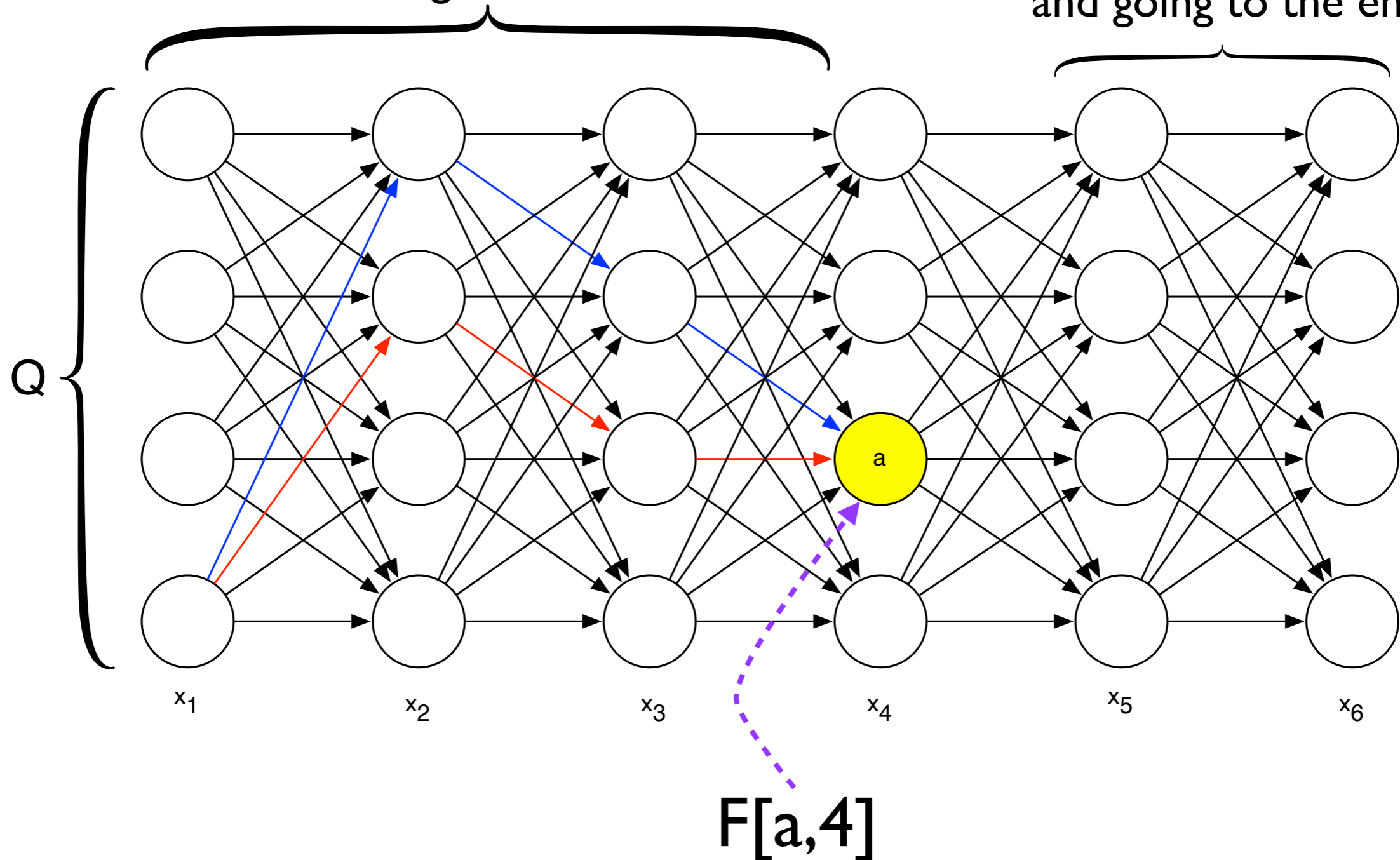
Computes the total probability  
of all the paths of length  $k$   
ending in state  $a$ .



# The Forward Algorithm

Computes the total probability of all the paths of length  $k$  ending in state  $a$ .

Still need to compute the probability of paths leaving  $a$  and going to the end.



# The Backward Algorithm

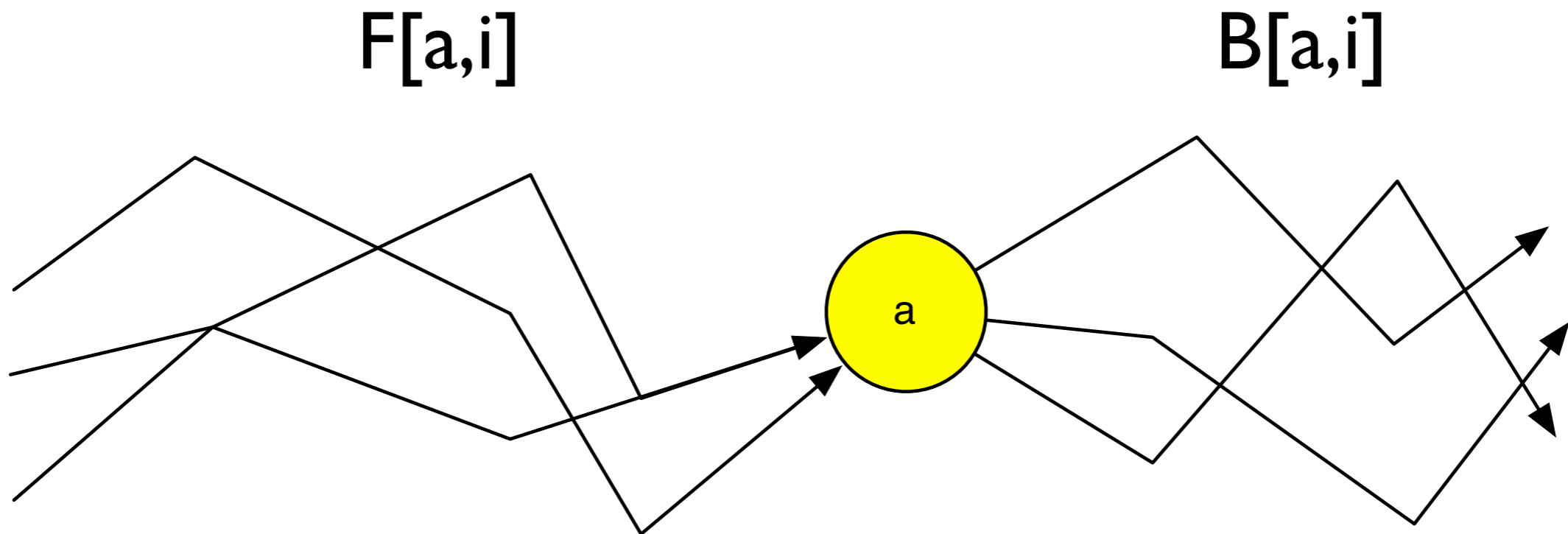
The same idea as the forward algorithm, we just start from the *end* of the input string and work towards the beginning:

$B[a,k]$  = “the probability of generating string  $x_{k+1}, \dots, x_n$  starting from state  $b$ ”

$$B[a, k] = \sum_{b \in Q} \underbrace{B[b, k + 1]}_{\substack{\text{Prob for} \\ x_{k+1} \dots x_n \\ \text{starting in} \\ \text{state } b}} \times \underbrace{\text{Pr}(a \rightarrow b)}_{\substack{\text{Probability} \\ \text{going from} \\ \text{state } a \text{ to } b}} \times \underbrace{\text{Pr}(x_{k+1} \mid \pi_{k+1} = b)}_{\substack{\text{Probability of emitting} \\ x_{k+1} \text{ given that the next} \\ \text{state is } b.}}$$

# The Forward-Backward Algorithm

$$\Pr(\pi_i = a \mid x) = \frac{\Pr(x, \pi_i = k)}{\Pr(x)} = \frac{F[a, i] \cdot B[a, i]}{\Pr(x)}$$



# Recap

- Hidden Markov Model (HMM) model the generation of strings.
- They are governed by a string alphabet ( $\Sigma$ ), a set of states ( $Q$ ), a set of transition probabilities  $A$ , and a set of emission probabilities for each state ( $E$ ).
- Given a string and an HMM, we can compute:
  - The most probable path the HMM took to generate the string (Viterbi).
  - The probability that the HMM was in a particular state at a given step (forward-backward algorithm).
- Algorithms are based on dynamic programming.
- *Finding* good parameters is a much harder problem.  
The Baum-Welch algorithm is an oft-used heuristic algorithm.