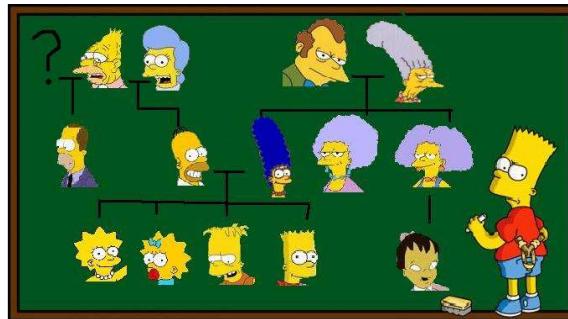


HMM Review

Some basic concepts: Probabilities over large collections, Markov models, and some of their family tree..



SPFLODD, 22 Jan 2018

Bhiksha Raj

Distributions over discrete sets

- Computing probabilities over very large sets of distinctly identifiable elements
 - Sets which can be mapped to sets of integers
 - Or even to all of \mathbb{Z}
 - i.e. *countable* sets

Discrete sets: Examples

- The set of all integers with exactly N digits
 - How large is this set?
- The set of all integers with *up to* N digits
 - How large is this set?
- Other examples?

Discrete sets: Examples

- The set of all integers with exactly N digits
- The set of all integers with *up to* N digits
- Other examples?
 - All possible sentences you can speak in 15 seconds.
 - All possible character sequences
 - All musical melodies in C major
 - All melodies in C major that require less than 5 minutes to sing

Discrete sets: Examples

- Segmentation:
 - *I returned and saw under the sun that the race is not to the swift nor the battle to the strong neither yet bread to the wise nor yet riches to men of understanding nor yet favour to men of skill but time and chance happen to them all*
 - How many possible sentences?
 - Assuming any sequence of characters is a valid word?
 - Assuming words must come from a fixed dictionary?

Discrete sets: Examples

- Segmentation:
 - *I returned and saw under the sun that the race is not to the swift nor the battle to the strong neither yet bread to the wise nor yet riches to men of understanding nor yet favor to men of skill but time and chance happen to them all*
- With correction:
 - *I retruned and saw unnder the sun that the race is not to the svift nor the batle to the strong neither yet bread to the weis enor yet riches to men of andurstend innor yet feyvor to men of skill buttyme and chance happen to them all*

A characteristic of the examples you saw..

- *Sequences*
- Ordering is important
 - *I returned and saw ≠ nsalreaturnwed*

Assigning probabilities to discrete sets

- All entries in the set are equally likely?

Assigning probabilities to discrete sets

- All entries in the set are equally likely?
- The probability of any entry with N elements in the sequence is proportional to $1/N$

Assigning probabilities to discrete sets

- All entries in the set are equally likely?
- The probability of any entry with N elements in the sequence is proportional to $1/N$
- The *total* probability of all entries of length N is proportional to $1/N$

Assigning probabilities to discrete sets

- All entries in the set are equally likely?
- The probability of any entry with N elements in the sequence is proportional to $1/N$
- The *total* probability of all entries of length N is proportional to $1/N$
- The probability of any entry with N elements in the sequence is proportional to $1/N^{1+\varepsilon}$
 - $\varepsilon > 0$

Assigning Probabilities over Discrete Sets

- Segmentation:
 - *I returned and saw under the sun that the race is not to the swift nor the battle to the strong...*
 - “All generatable sentences are equally likely”?

Assigning Probabilities over Discrete Sets

- Segmentation:
 - *I returned and saw under the sun that the race is not to the swift nor the battle to the strong...*
 - “All generatable sentences are equally likely”?
- With correction:
 - *I retruned and saw unnder the sun that the race is not to the svift nor the batle to the strong...*
 - “All generatable sentences are equally likely”?

Assigning Probabilities over Discrete Sets

- Segmentation:
 - *I returned and saw under the sun that the race is not to the swift nor the battle to the strong...*
 - “All generatable sentences are equally likely”?
- Why is this a bad idea?

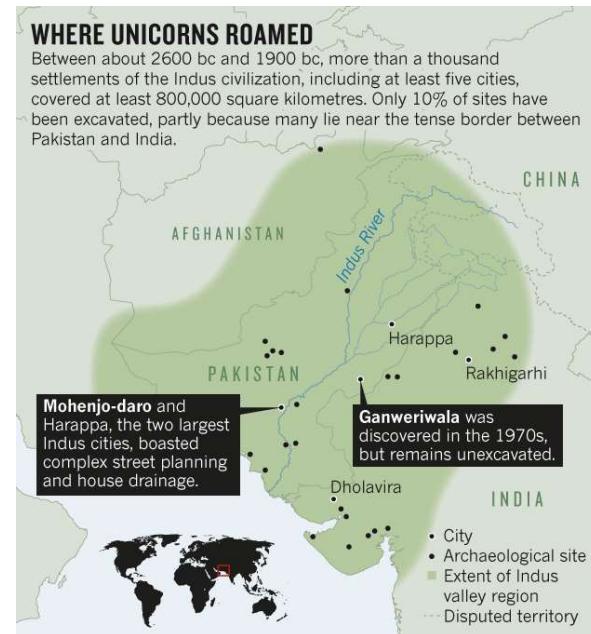
Assigning Probabilities over Discrete Sets

- Segmentation:
 - *I returned and saw under the sun that the race is not to the swift nor the battle to the strong...*
 - “All generatable sentences are equally likely”?
- Why is this a bad idea?
 - Because this assumes *no structure*
 - Structure:
 - Some sequences are more likely than others
 - Some sequences never happen
- Knowing that there is structure is not the same as knowing the structure

The Indus Valley Civilization

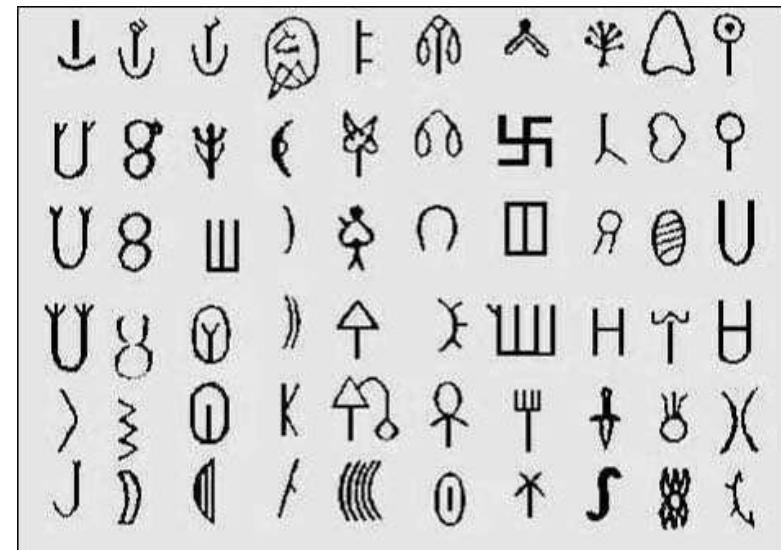


The mysterious Indus unicorn on a roughly 4,000-year-old sealstone, found at the Mohenjo-daro site. (from Nature, April 16)



- One of the oldest civilizations, known to have been extremely advanced
 - *Only known advanced civilization that left no evidence of warfare, or even armies!!!*
- But nobody has yet deciphered their script!

The Indus Valley Script



- The Indus script is made up of partially pictographic signs and human and animal motifs including a puzzling 'unicorn'. These are inscribed on miniature steatite (soapstone) seal stones, terracotta tablets and occasionally on metal. The designs are “little masterpieces of controlled realism, with a monumental strength in one sense out of all proportion to their size and in another entirely related to it”, wrote the best-known excavator of the Indus civilization, Mortimer Wheeler, in 19681.
 - From Nature, April 2016

A Controversy

The Collapse of the Indus-Script Thesis: The Myth of a Literate Harappan Civilization

Steve Farmer, Richard Sproat, and Michael Witzel¹

Abstract

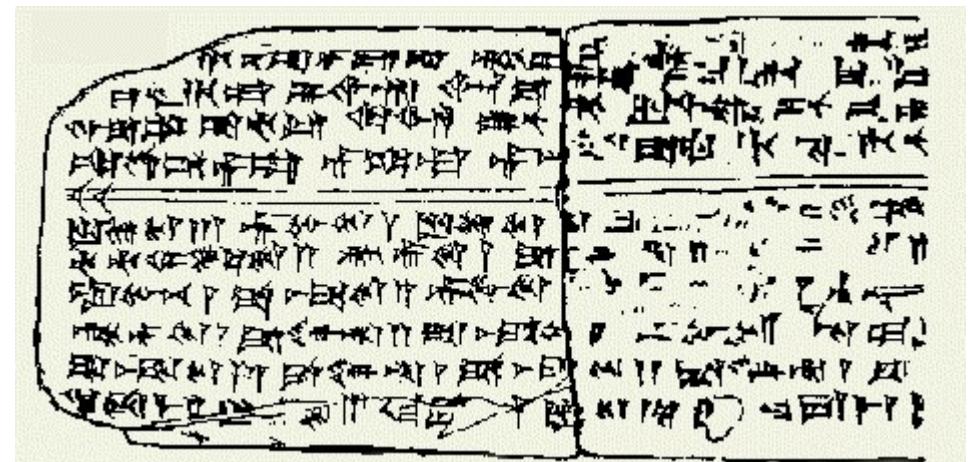
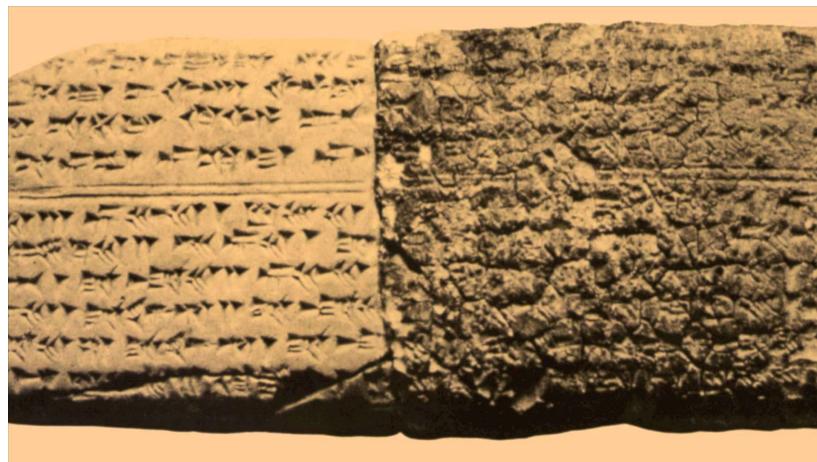
Archaeologists have long claimed the Indus Valley as one of the four literate centers of the early ancient world, complete with long texts written on perishable materials. We demonstrate the impossibility of the lost-manuscript thesis and show that Indus symbols were not even evolving in linguistic directions after at least 600 years of use. Suggestions as to how Indus symbols were used are noted in nonlinguistic symbol systems in the Near East that served key religious, political, and social functions without encoding speech or serving as formal memory aids. Evidence is reviewed that the Harappans' lack of a true script may have been tied to the role played by their symbols in controlling large multilingual populations; parallels are drawn to the later resistance of Brahmin elites to the literate encoding of Vedic sources and to similar phenomena in esoteric traditions outside South Asia. Discussion is provided on some of the political and academic forces that helped sustain the Indus-script myth for over 130 years and on ways in which our findings transform current views of the Indus Valley and of literacy in the ancient world in general.

- Electronic Journal of Vedic Studies, 2004

The not a language hypothesis

- Indus valley inscriptions are only between 5 and 14 characters long
- The distribution of symbols is no different from the Zipfian distribution of decorative symbols in various illiterate cultures
 - The entropy of the distributions shows a lack of *structure* associated with a real language of any kind
- Still an ongoing controversy
 - I suspect Sproat is right that these don't represent true script.
 - This does not mean the I-V civilization didn't have a script; just that they did not record it on stone...
- **Example of *not knowing if there is structure!***

World's oldest known music: Hurrian Hymn

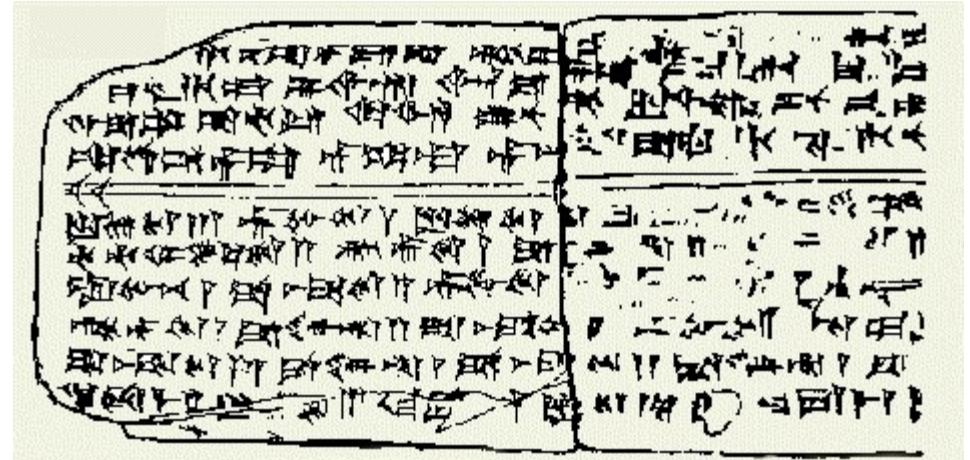
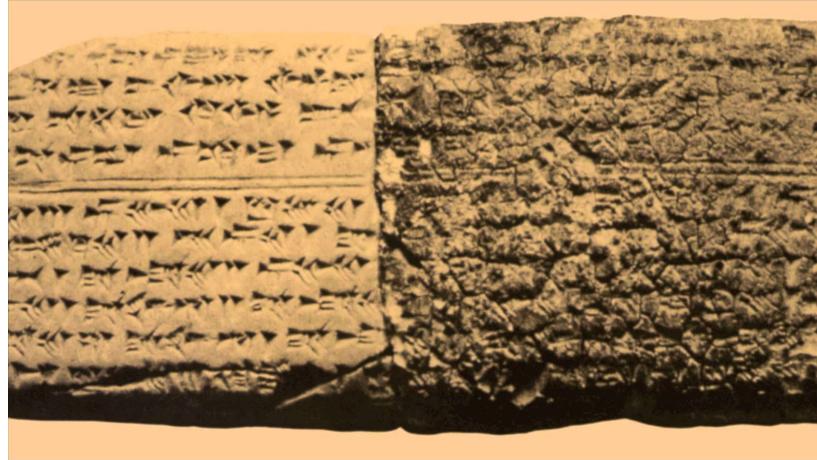


- Music transcriptions discovered in the Amorite-Canaanite city of Ugarit (Ras Shamra in modern day Syria)
 - From Ca 1400BC!
- Oldest notated music known

Hurrian Hymns

- 36 songs found in all
 - Songs to Amorite gods
- Of these, only Hymn no. 6 written to Nikkal, the goddess of orchards, is complete
- *Known to be music*
 - Amorites wrote in Akkadian, which is understood
 - Attendant inscriptions describe the music
 - Some of them include the name of the composer
 - Though not Hymn no. 6
 - Even includes instruction that it must be sung by a singer with a lyre

What is missing???

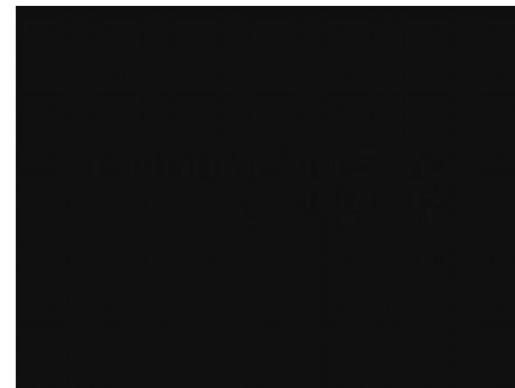


- What do those symbols mean??
- What are the notes?
 - Or the cadence?
- No analogy to any other music available
- *Structure is known to be present, but the structure is not understood!!*

The outcomes..

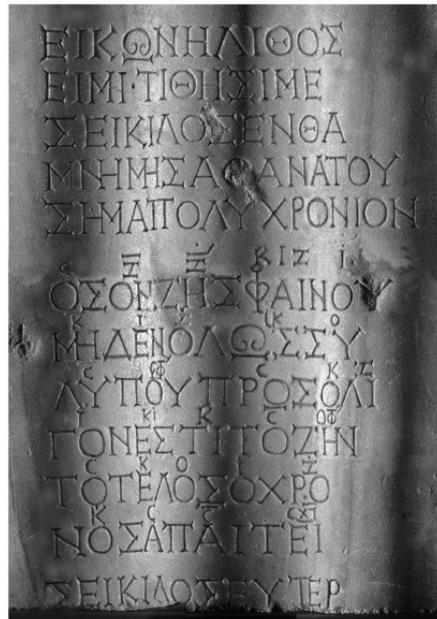


<http://www.urkesh.org/urkeshpublic/music.htm>



- Hm...

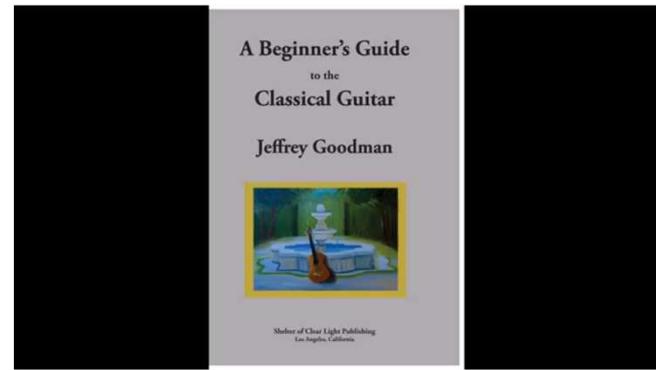
When interpretation is not a problem



Οσον ζῆς φαίνου
μηδὲν ὅλως σὺ λυποῦ
πρὸς ὀλίγον ἔστι τὸ ζῆν
τὸ τέλος ὁ χρόνος ἀπαιτεῖ.

- Seikilos Epitaph
 - Somewhere between 200BC and 100 AD, probably closer to the latter
 - Either from Seikilos to his wife, Euterpe, or simply the epitaph of Seikilos, son of Euterpos
- The musical notation is understood
 - Oldest *complete* piece of music

Seikilos Epitaph



"Ο- σον ζῆς φαί - νον μη- δὲν ὅ - λως σὺ λυ-ποῦ πρὸς ὁ- λί-γον ἐσ - τὶ τὸ ζῆν τὸ τέλος ὁ χρό-νος ἀ - παι-τεῖ.

Hóson zêis phaínou

While you live, shine

mēdén hólōs sy lypoū

have no grief at all

pros olígon estí to zén

life exists only for a short while

to télos ho chrónos apaiteî

and time demands an end.

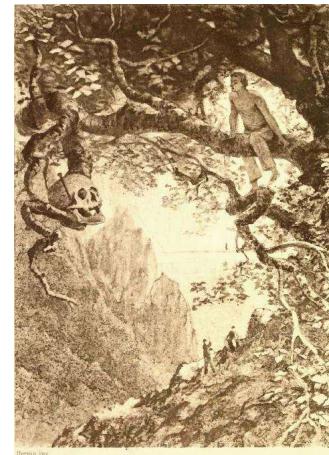
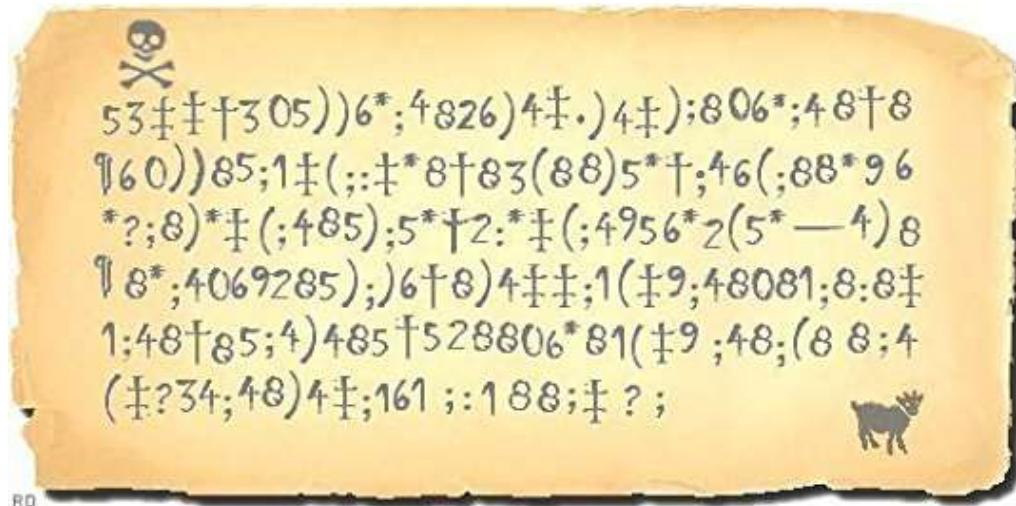
How did we interpret this?



Ο- σον ζῆς φαί - νον μη- δὲν ὁ- λως σὺ λυ-ποῦ πρὸς ὁ- λι-γον ἐσ- - τὶ τὸ ζῆν τὸ τέλος ὁ χρό-νος ἀ- παι-τεῖ.

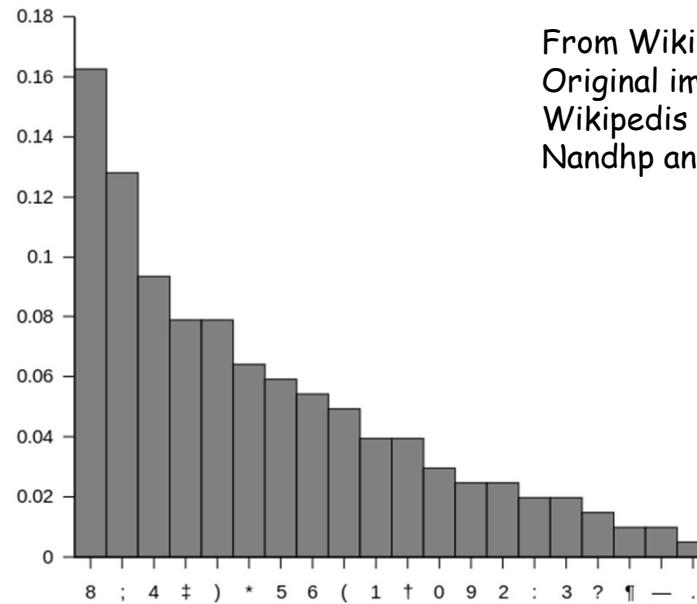
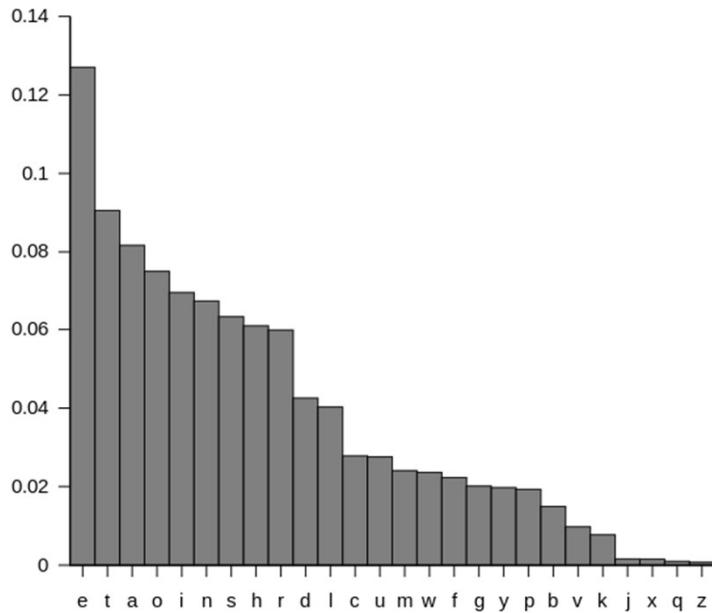
- So how does one go from the information on the left to the information on the right?

The Gold Bug (Edgar Allan Poe)



- Whut?

The Gold Bug (Edgar Allan Poe)

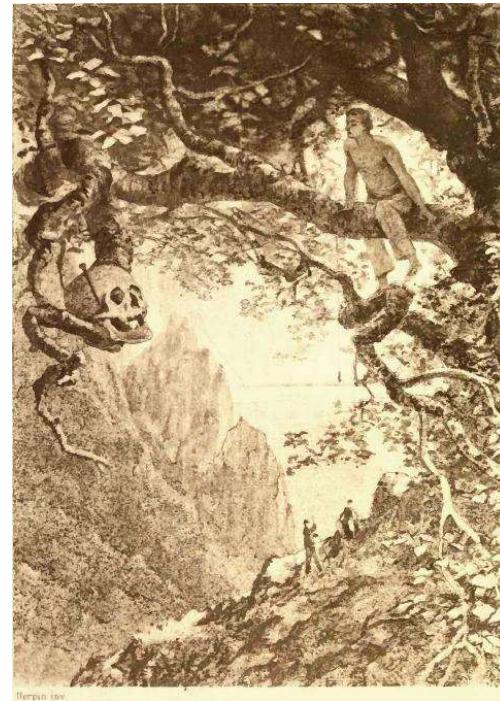


From Wikipedia:
Original images by
Wikpedis users
Nandhp and Atar

- Comparing *letter frequency* histograms
 - If that fails, you can use frequencies of *bigrams* of characters (most frequent character pair: “TH”)
 - <http://norvig.com/mayzner.html>

The Gold Bug (Edgar Allan Poe)

53‡†305))6*:4826)4‡.)4‡);806*:48†8
agooodglassinthebishopshostelinthedede
¶60))85;;]8*:‡*8†83(88)5*†;46(;88*96
vilsseattwentyonedegreesandthirteenmi
?;8)‡(;485);5*†2:*‡(;4956*2(5*—4)8
nutesnortheastandbynorthmainbranchse
¶8*:4069285);)6†8)4‡‡;1(‡9;48081;8:8‡
venthlimbeastsideshootfromthelefteyeo
1;48†85;4)485†528806*81(‡9;48;(88;4
fthedeadsheadabeelinefromthetreeth
(‡?34;48)4‡;161;:188;‡?
roughtheshotfiftyfeetout



- Substitution cipher; based on comparison of trivial surface level statistics
 - Even works for music if its from the same genre

More more complex things



Ca 1400-1430

- The Voynich manuscript
 - In cipher?
 - Still undeciphered
 - What do we do?

Need to *characterize* the language

- Statistically model the *structure* of the language
 - Model the probability distribution over a potentially infinite set, such that every entry gets assigned a realistic probability
 - The probability must sum to 1.0 obviously
- Any ideas?

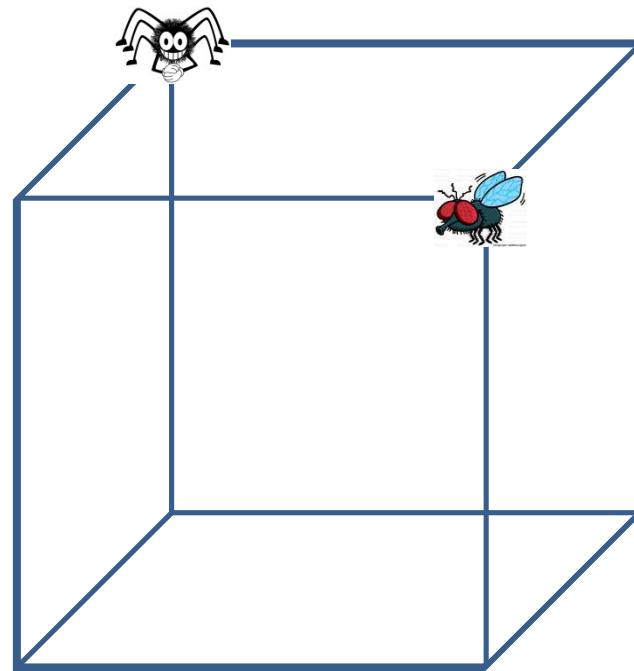
Need to *characterize* the language

- Statistically model the *structure* of the language
- In this course we will largely model language as *Markov*
 - I.e. produced by a *Markov process*
 - Hypothesis: Markov process models nicely capture the structure of the language
 - And can be used to generate, categorize, compare languages etc.
- Lets understand Markov processes a little..

**SWITCHING
GEARS
ABIT...**



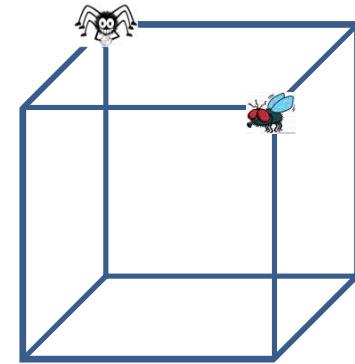
The story of Flider and Spy



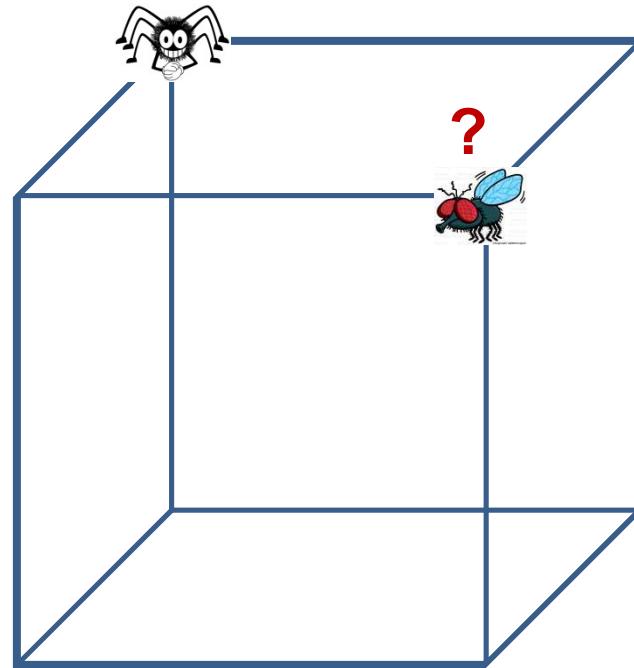
- Flider the spider is at the far corner of the room, and Spy the fly is sleeping happily at the near corner

The story of Flider and Spy

- Flider only walks along edges
- She begins walking along one of the three edges at random
- She takes one minute to cover the distance from one corner to the other along any edge
- When she arrives at the new corner, she randomly chooses one of the three edges and continues walking (she may even turn back)

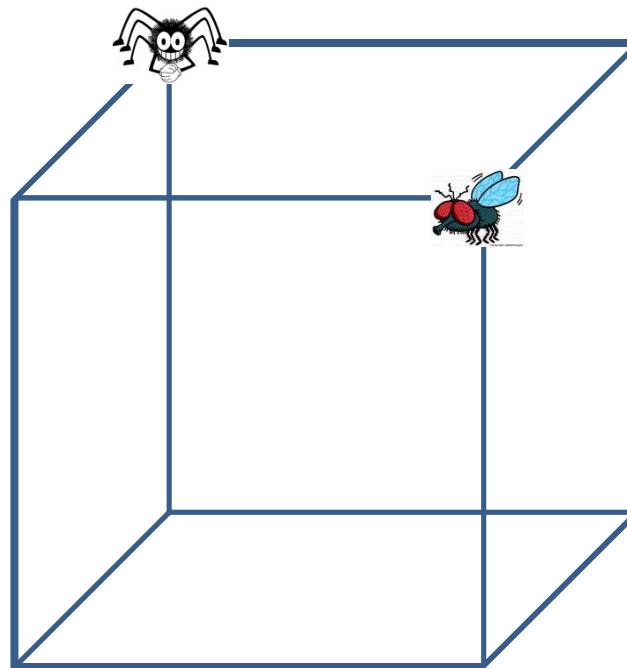


The story of Flider and Spy



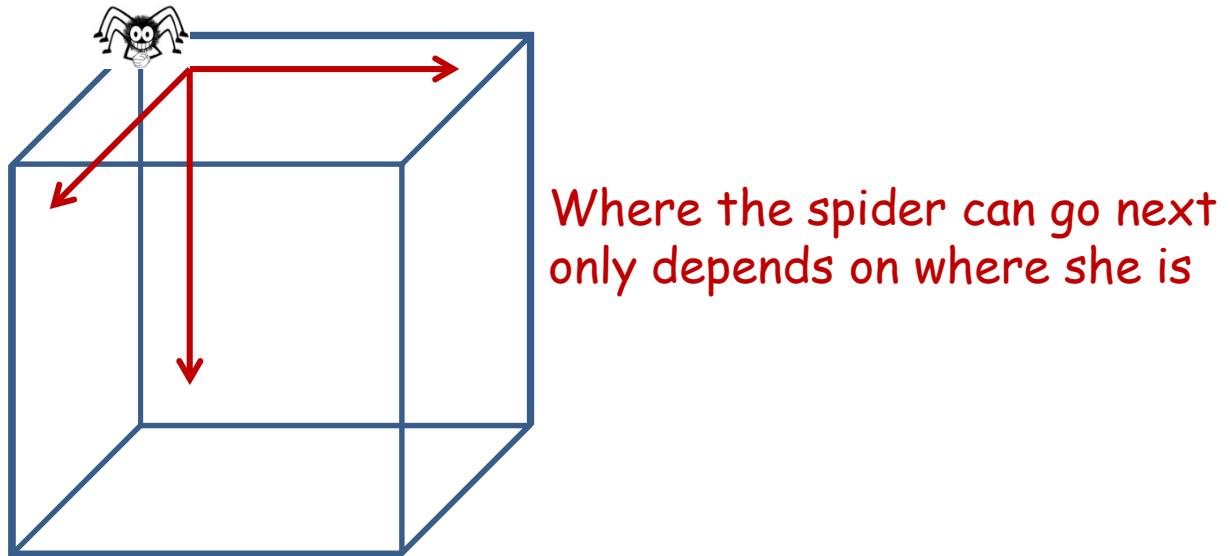
- What is the life expectancy of Spy?

Markov process



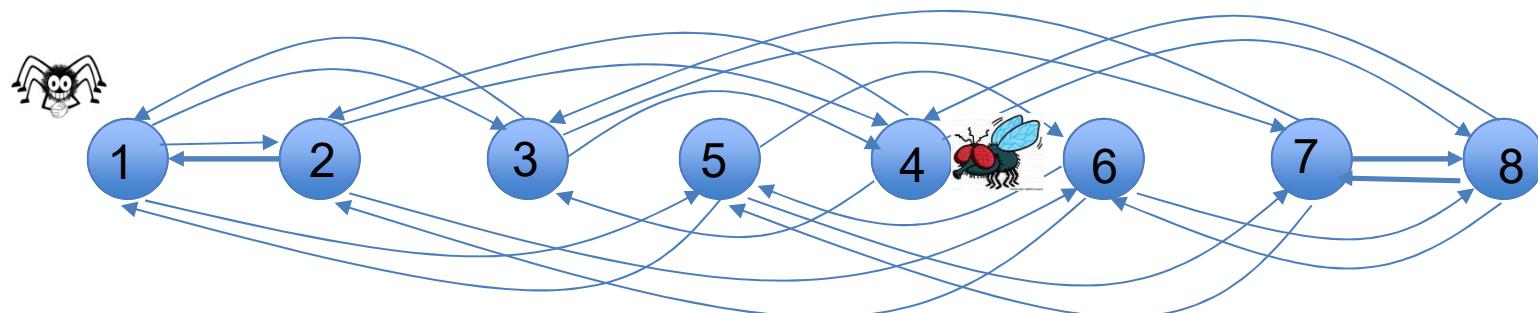
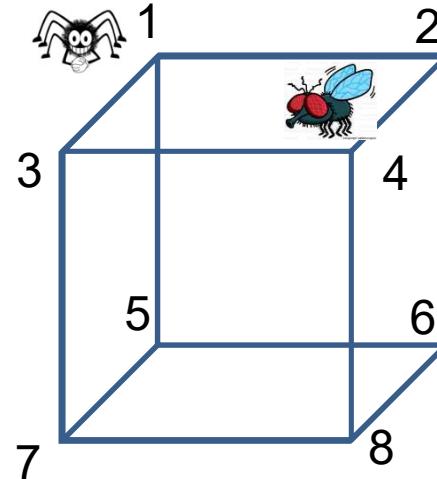
- **Markov Process:** Does not matter how you got here, only matters where you are

The World as we model It



- Definition of Markov property:
 - The state of the system has a Markov property if the future only depends on the present
$$P(S_{t+1}|S_0, S_1, \dots, S_t) = P(S_{t+1}|S_t)$$
- States can be *defined* to have this property

Discrete-State Markov Process



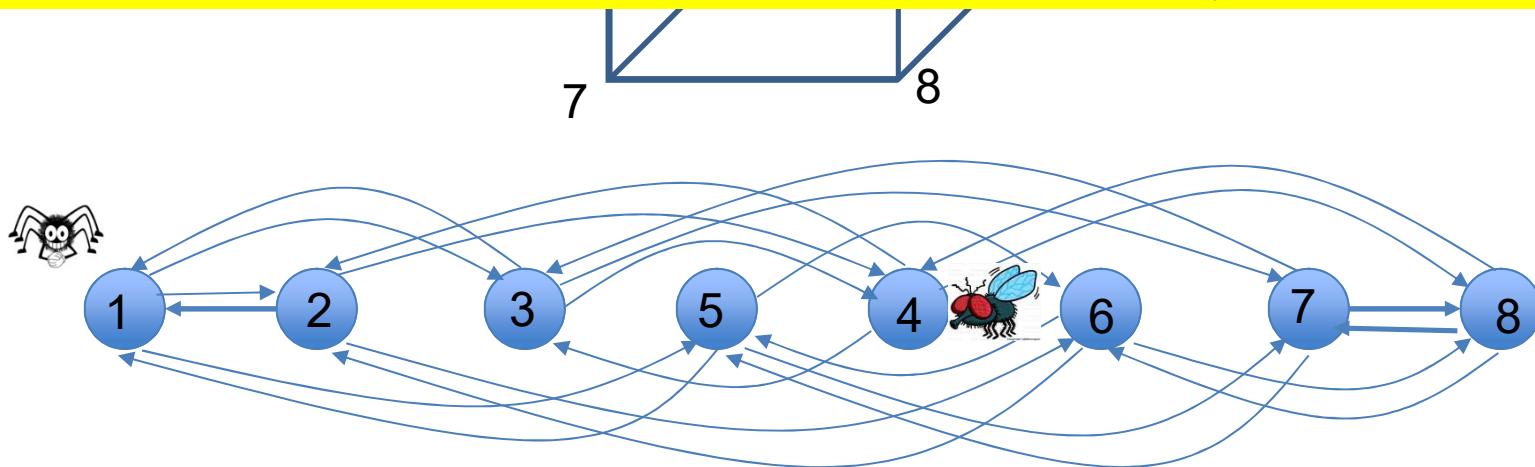
- AKA Markov Chain
- The process (flider) can be in one of a number of states (corners)
- At any state the process can transition into another states, based on a probability distribution that is only dependent on the current state

Discrete-State Markov Process

What characterizes this process?

- a) Number of states
- b) The complete set of transition probabilities $P(S_i|S_j)$ that determine the probability with which the process transitions to S_i , when its in S_j

What is the complete set of transition probabilities $P(S_i|S_j)$ in this example?



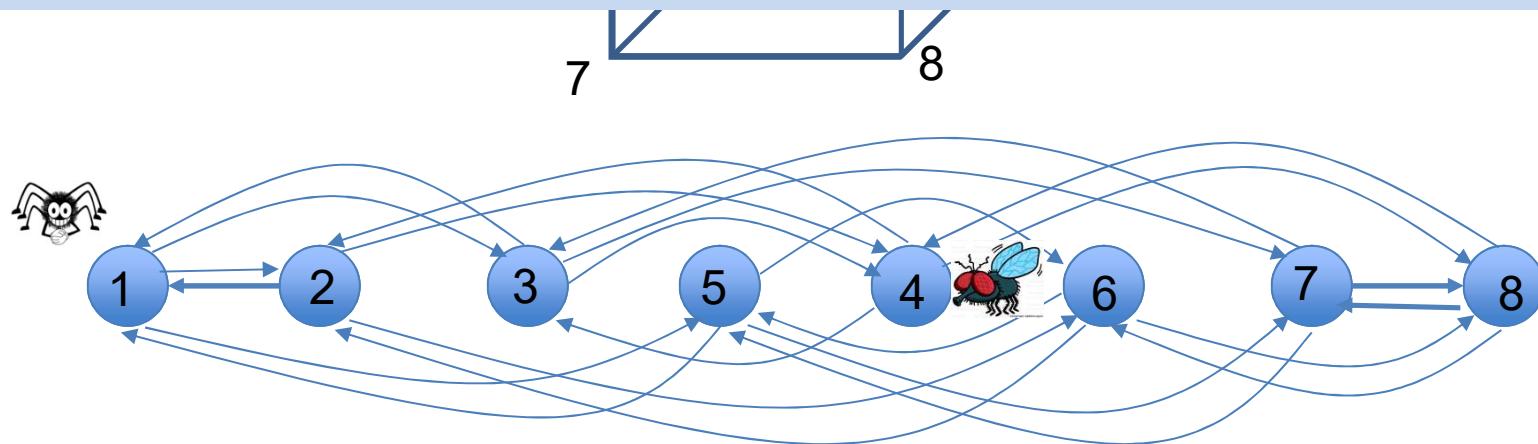
- AKA Markov Chain
- The process (flider) can be in one of a number of states (corners)
- At any state the process can transition into another states, based on a probability distribution that is only dependent on the current state

Discrete-State Markov Process

What characterizes this process?

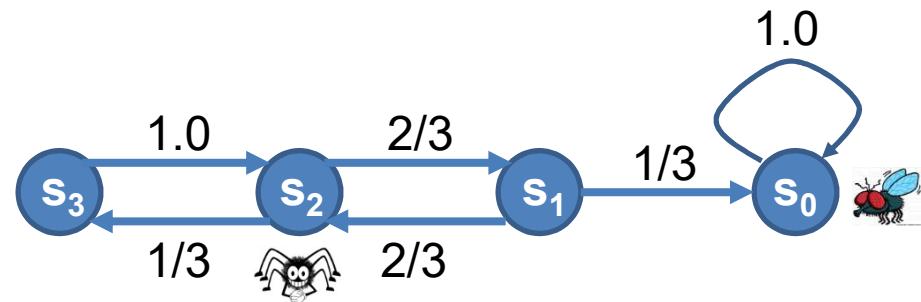
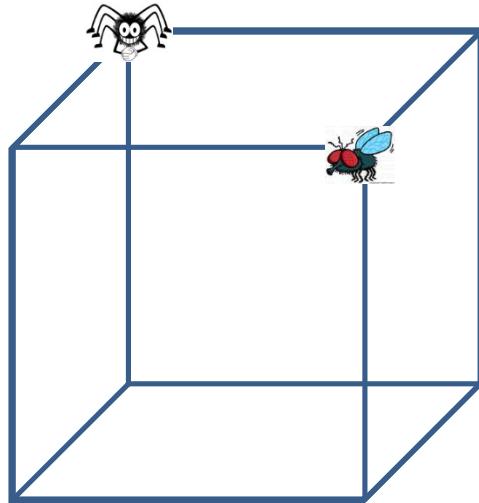
- a) Number of states
- b) The complete set of transition probabilities $P(S_i|S_j)$ that determine the probability with which the process transitions to S_i , when its in S_j

Is there a different way of characterizing this process, which is also Markov?



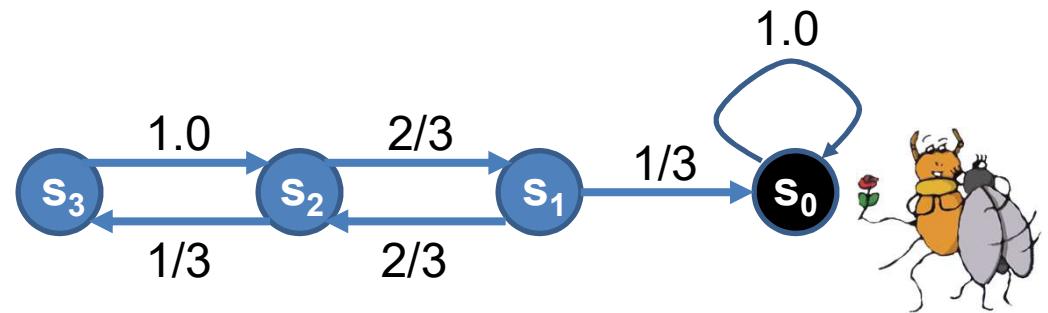
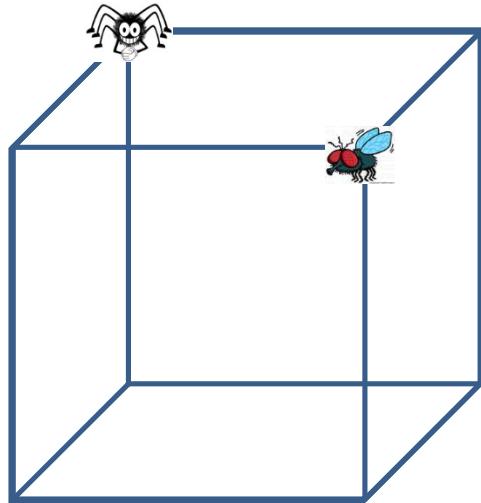
- AKA Markov Chain
- The process (flider) can be in one of a number of states (corners)
- At any state the process can transition into another states, based on a probability distribution that is only dependent on the current state

An alternate Markov representation



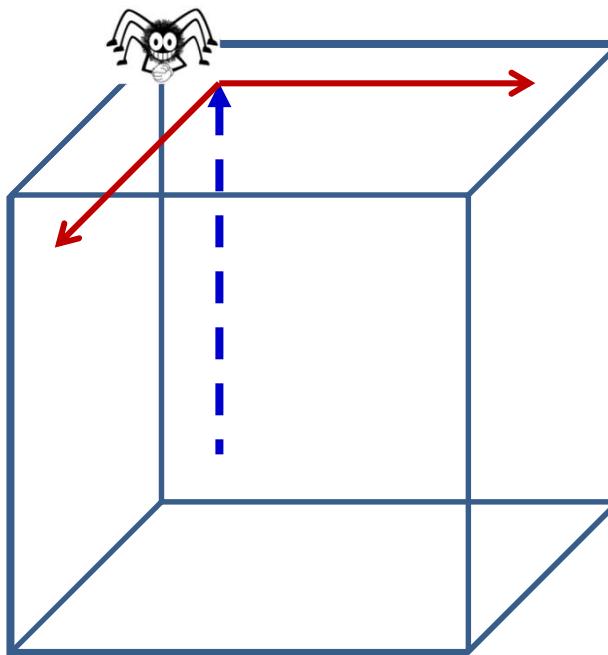
- A Markov process definition may not be unique!
 - The same set of outcomes may be derived from multiple processes!

The absorbing state



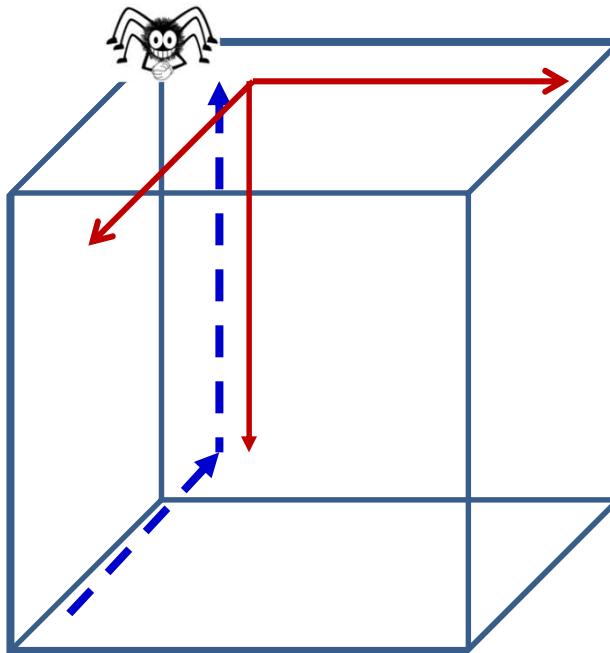
- What happens when Flider catches Spy?
 - There's no more wandering to be done
 - This is an *absorbing state* with *self-transition probability = 1*

Introducing.... Glider!!



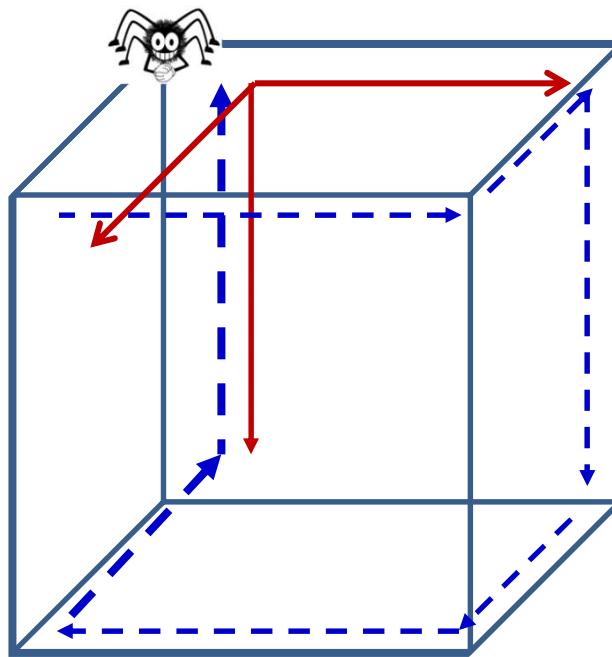
- Glider, Flider's brother, never turns around during his wanderings
 - On arriving at any corner, he chooses one of the two “forward” paths randomly.
 - The future possibilities depend on the edge he arrived from
 - Is he Markovian?

Introducing.... Rider!!



- Rider, Glider's picky twin is even more fastidious
 - She considers his past *two* corners before choosing the next one to go to.
 - Is she Markovian?

Introducing.... Rider!!



- Schneider considers *every place he's been, and the order in which he's been there, since he was born!*
 - Is he Markovian?

Introducing.. The “information” state

- By appropriately defining an *information state*, **any** discrete process can be modelled as Markov
- Simply modelling a process as Markov imparts no new insight into its structure!!

Need to *characterize* the language

- Statistically model the *structure* of the language
- In this course we will largely model language as *Markov*
 - I.e. produced by a *Markov process*
 - Hypothesis: Markov process models nicely capture the structure of the language
 - And can be used to generate, categorize, compare languages etc.
- Lets understand...
Really?????

Need to *characterize* the language

- Statistically model the *structure* of the language
- In this course we will largely model language as *Markov*
 - I.e. produced by a *Markov process*
 - Hypothesis: Markov process models nicely capture the structure of the language
 - And can be used to generate, categorize, compare languages etc.

- Lets understand...

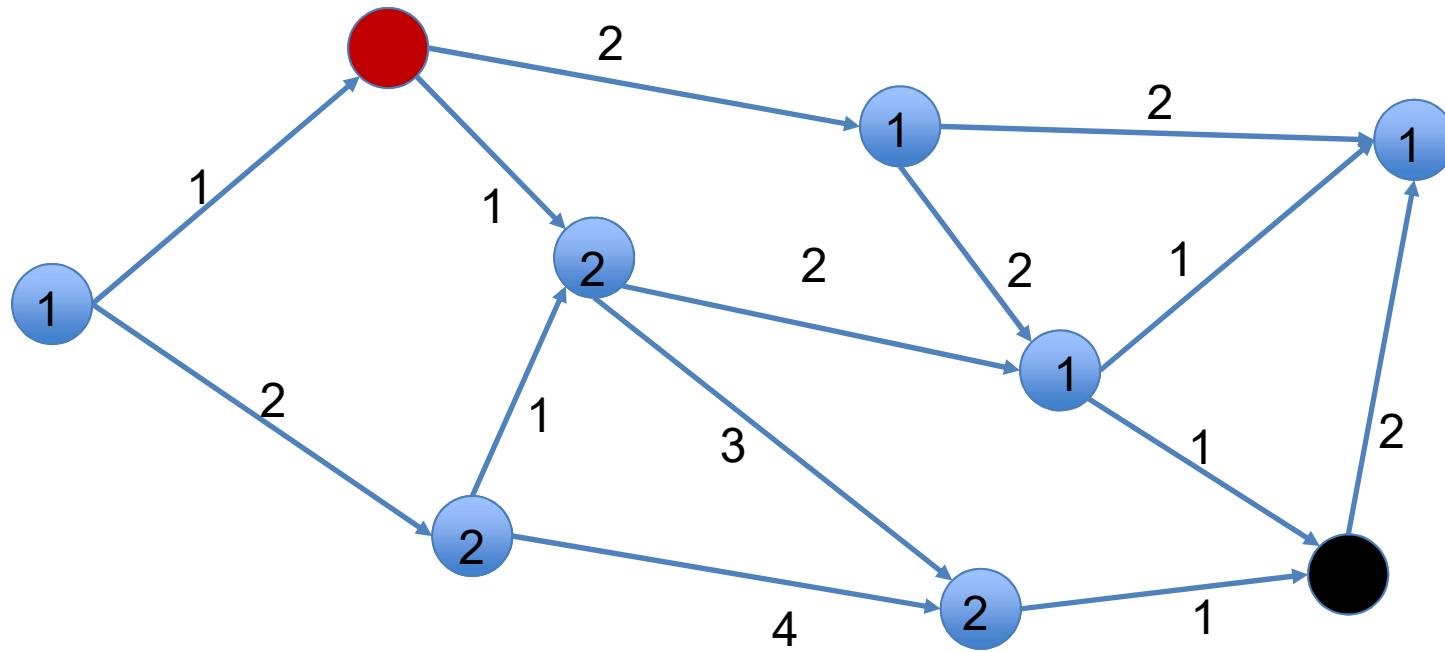
Really????



Only if you define the information state properly

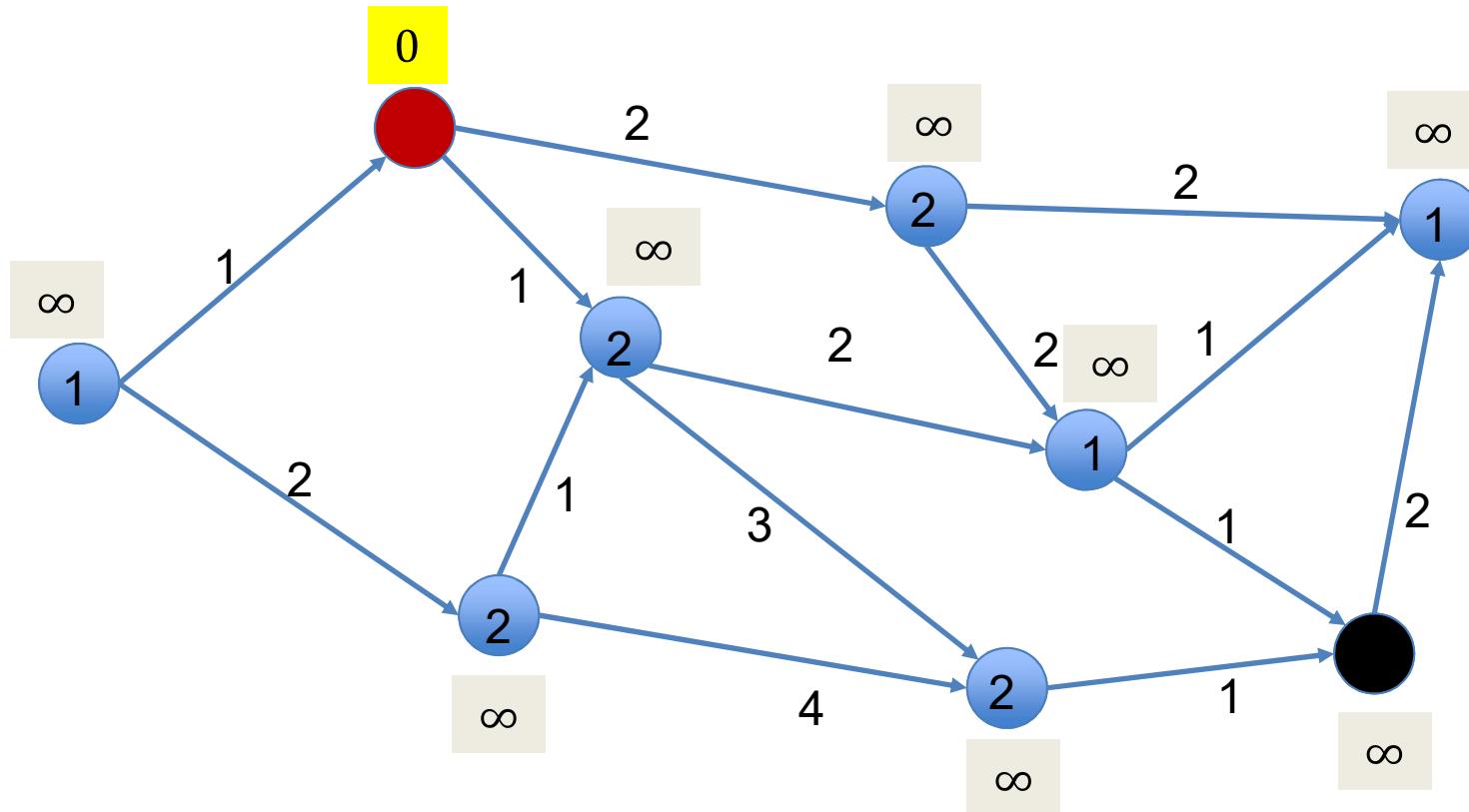
- An on that happy note....

The shortest path algorithm



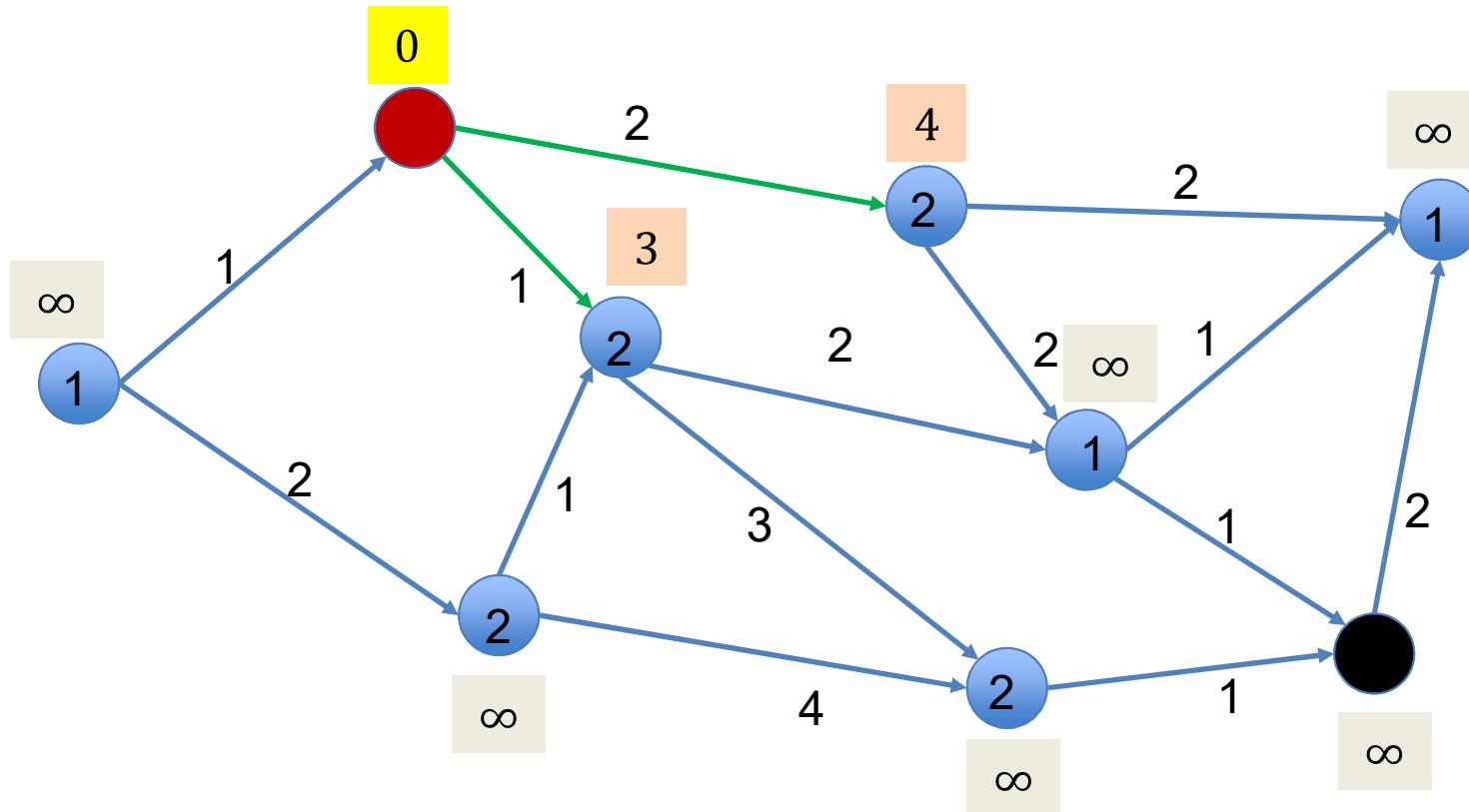
- What is the shortest path from the red to the black node?

Dijkstra's Algorithm



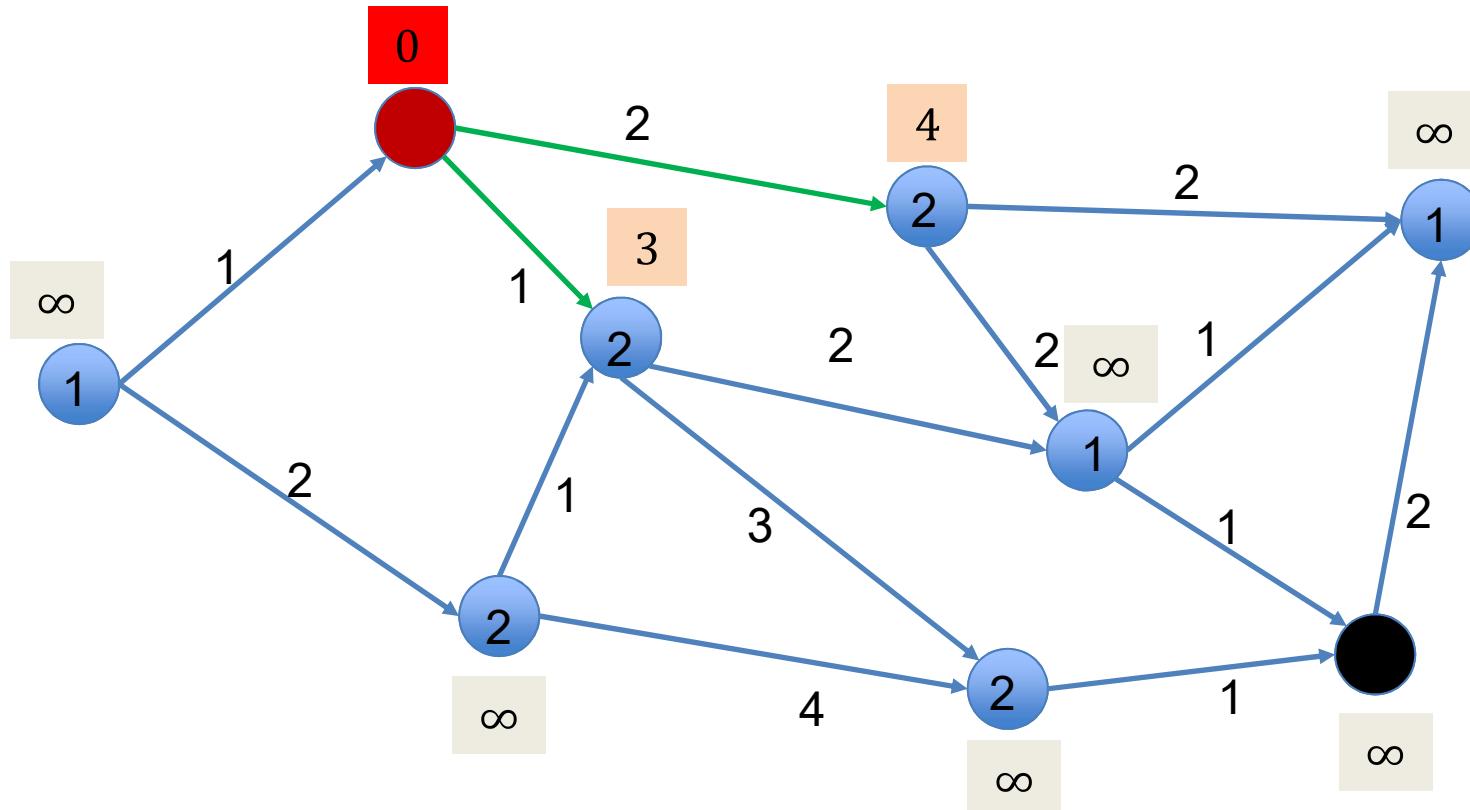
- Set the cost of the initial node to 0 and all other nodes to infinity
- Start at the initial node

Dijkstra's Algorithm



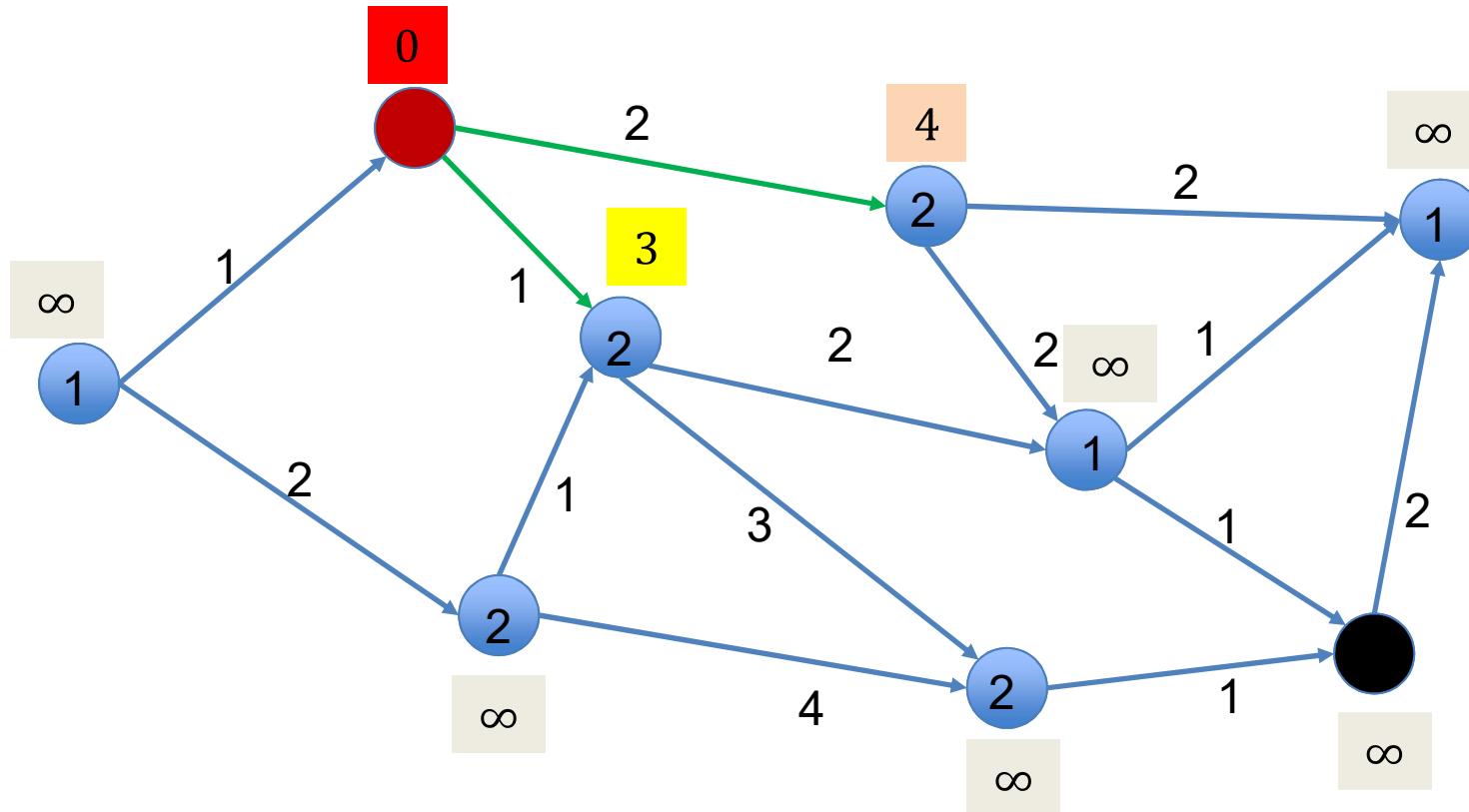
- “Expand” the initial node out along all outgoing paths
- At the destination nodes, record the cost

Dijkstra's Algorithm



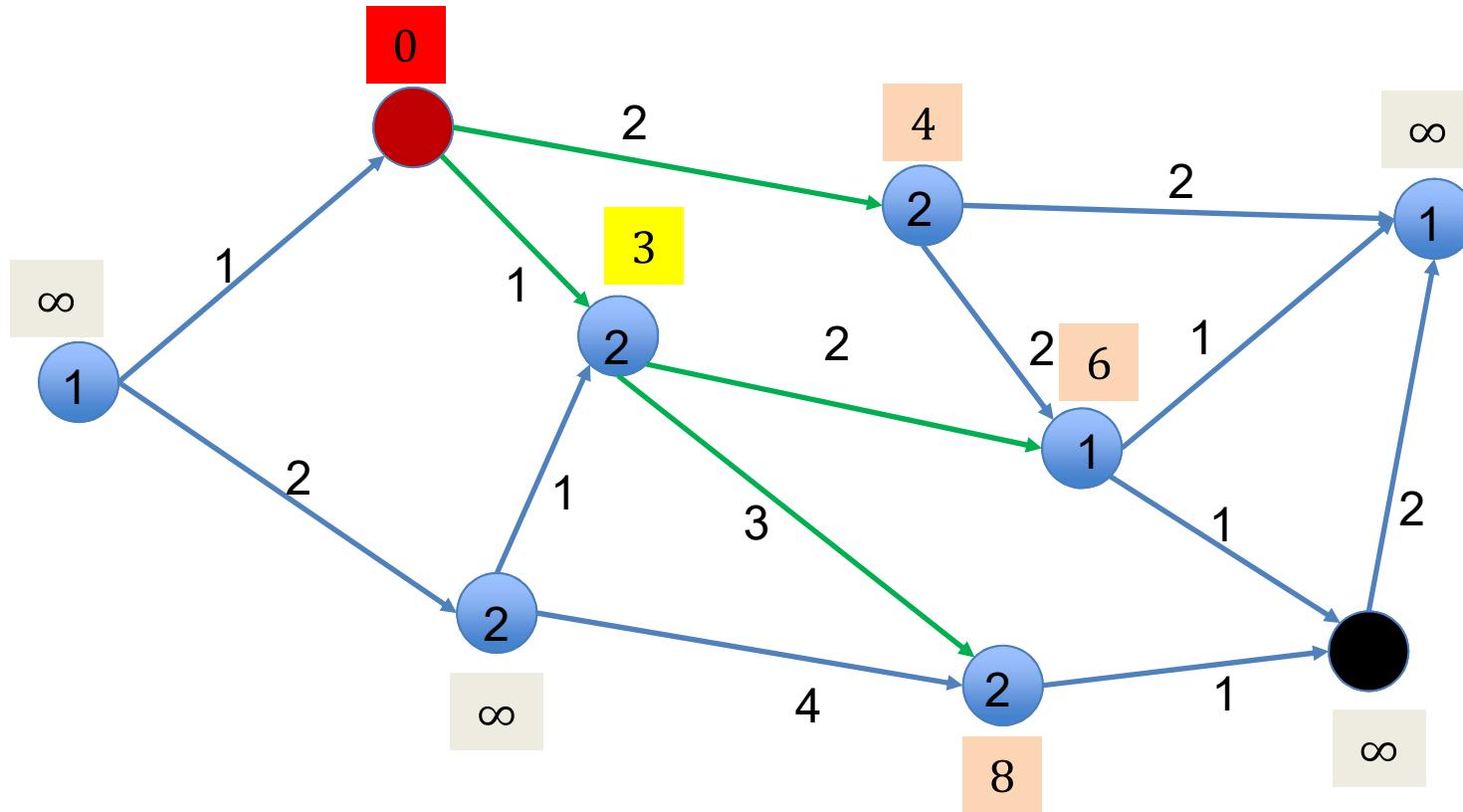
- Mark initial node as “done”

Dijkstra's Algorithm



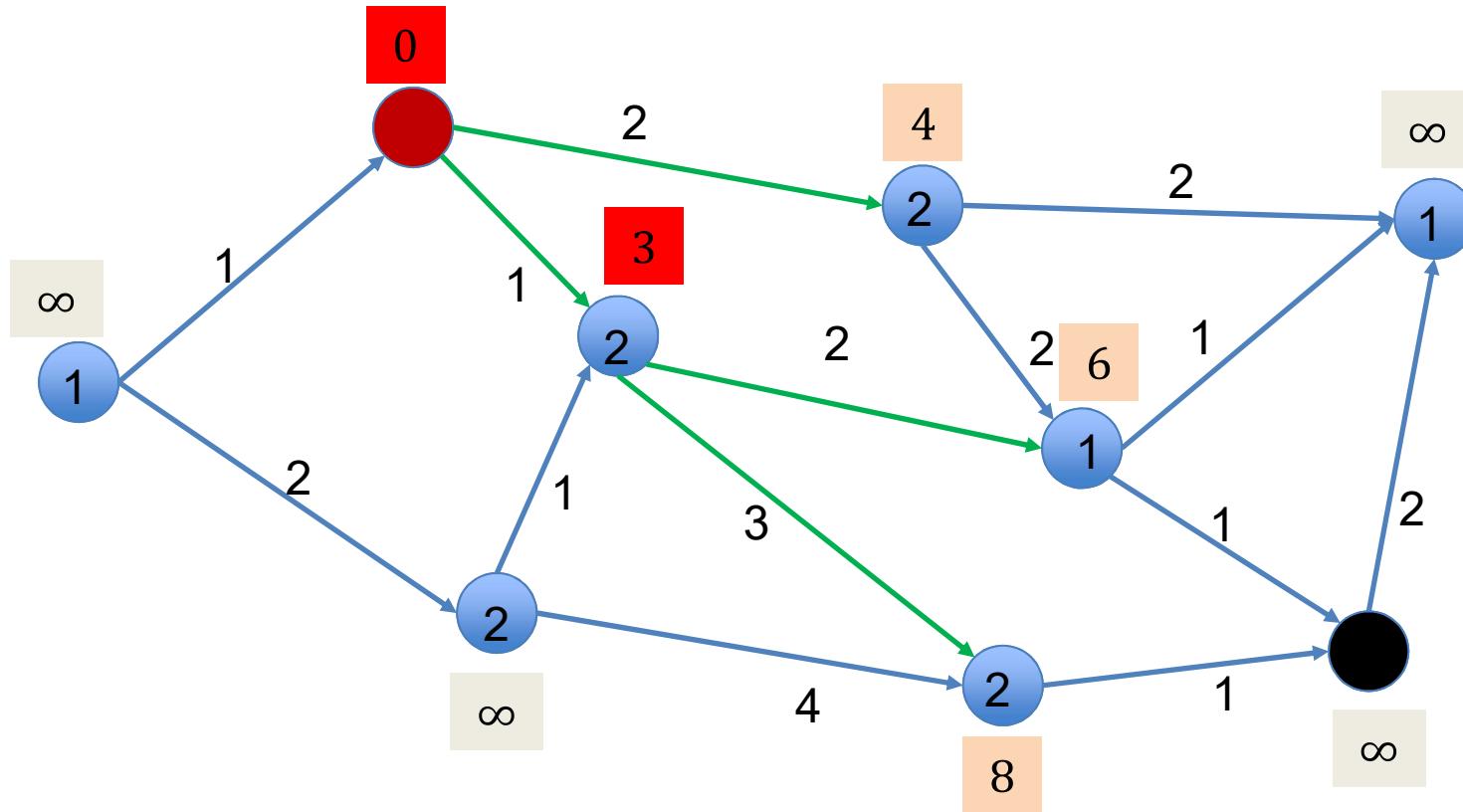
- Select the node with the lowest cost

Dijkstra's Algorithm



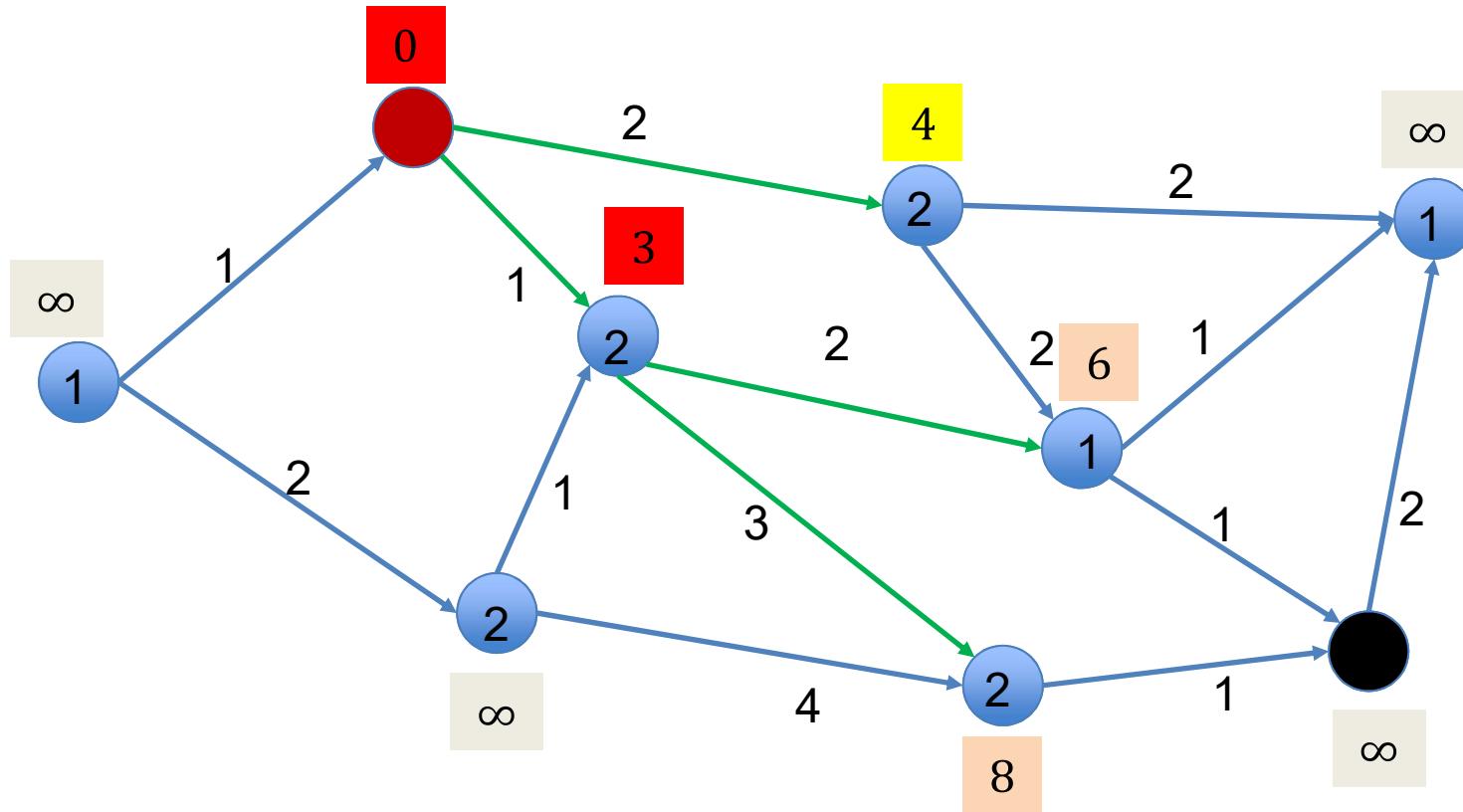
- “Expand” it along all outgoing edges
- Node path costs at destination nodes

Dijkstra's Algorithm



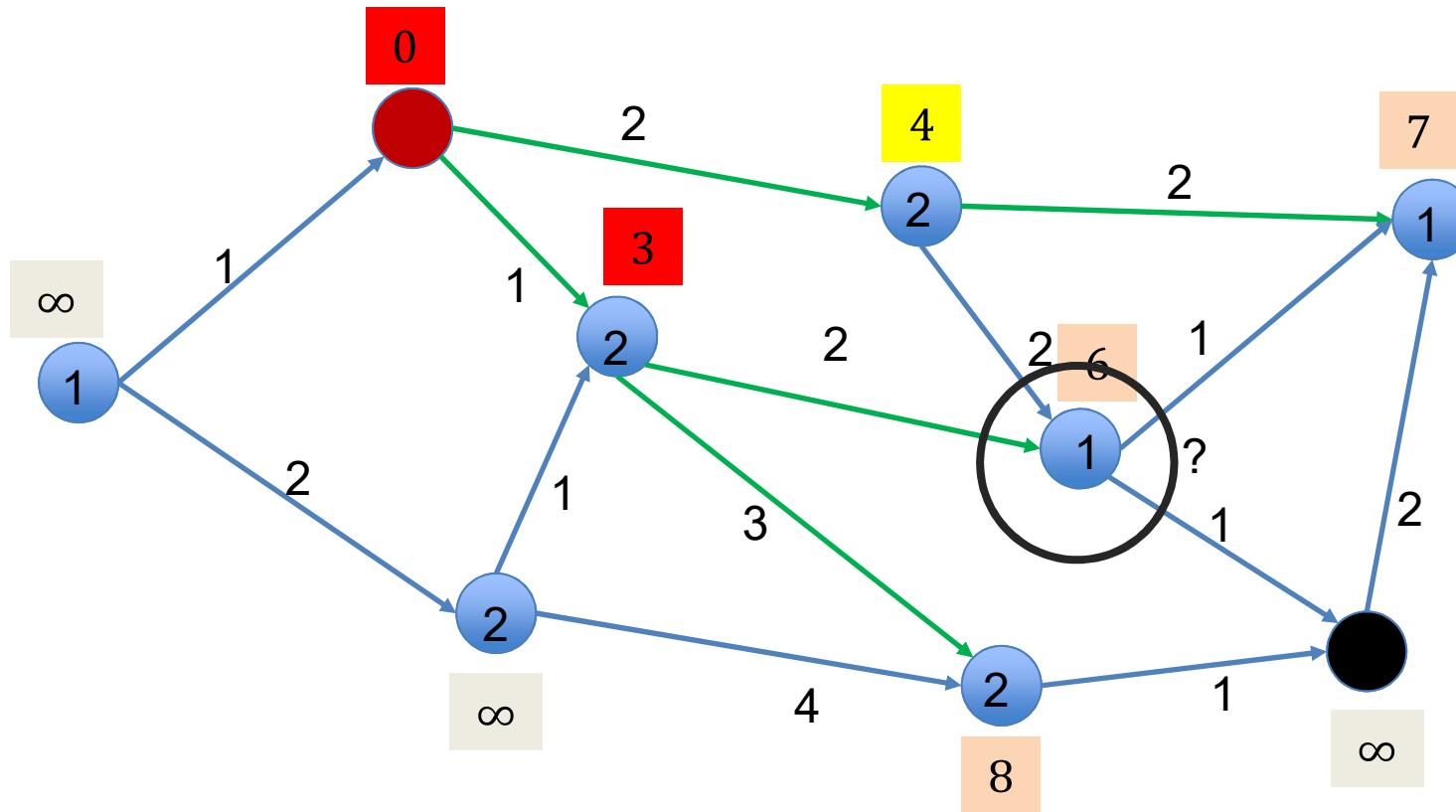
- Mark node as “done”

Dijkstra's Algorithm



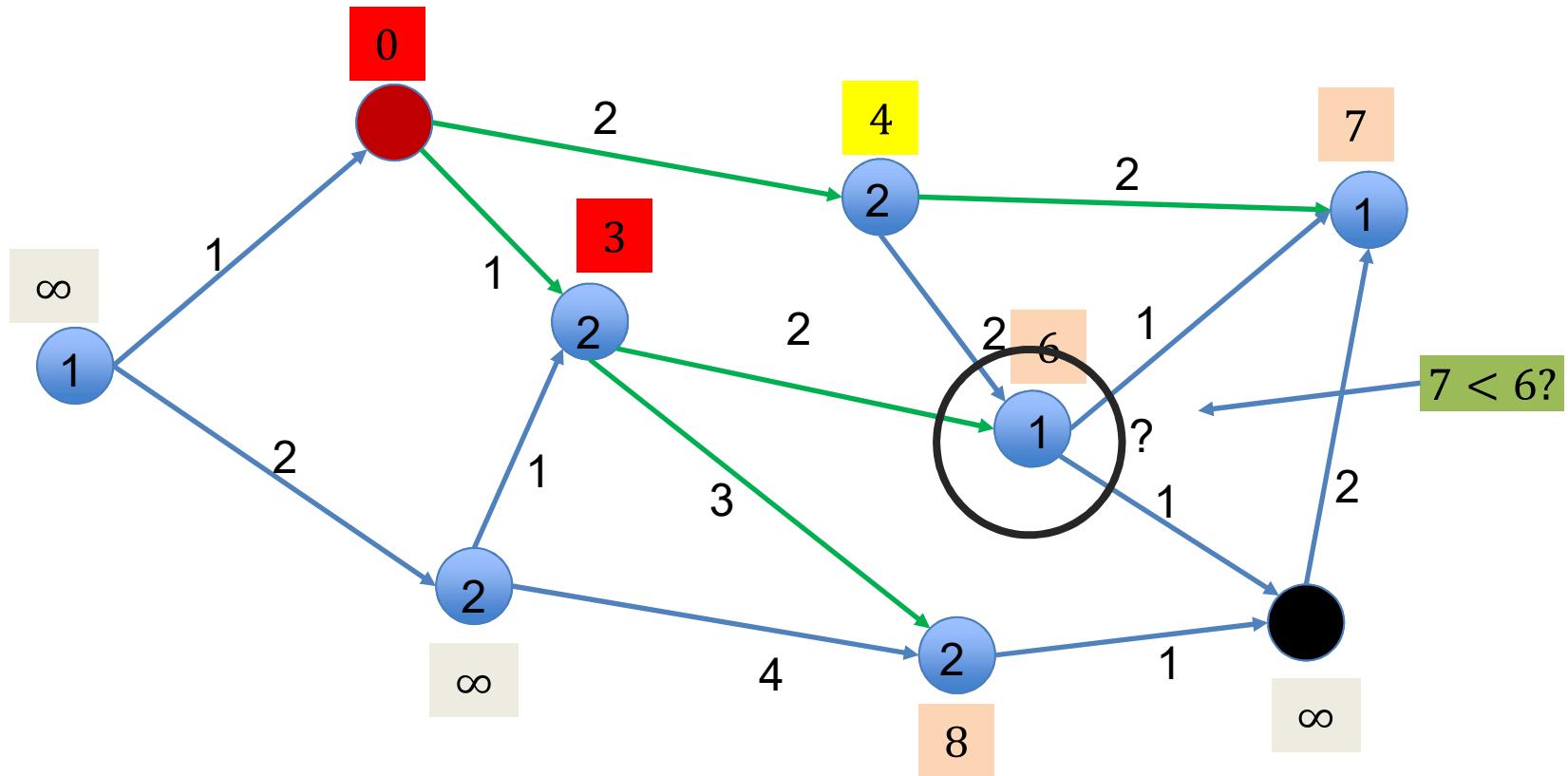
- Select the node with the lowest cost

Dijkstra's Algorithm



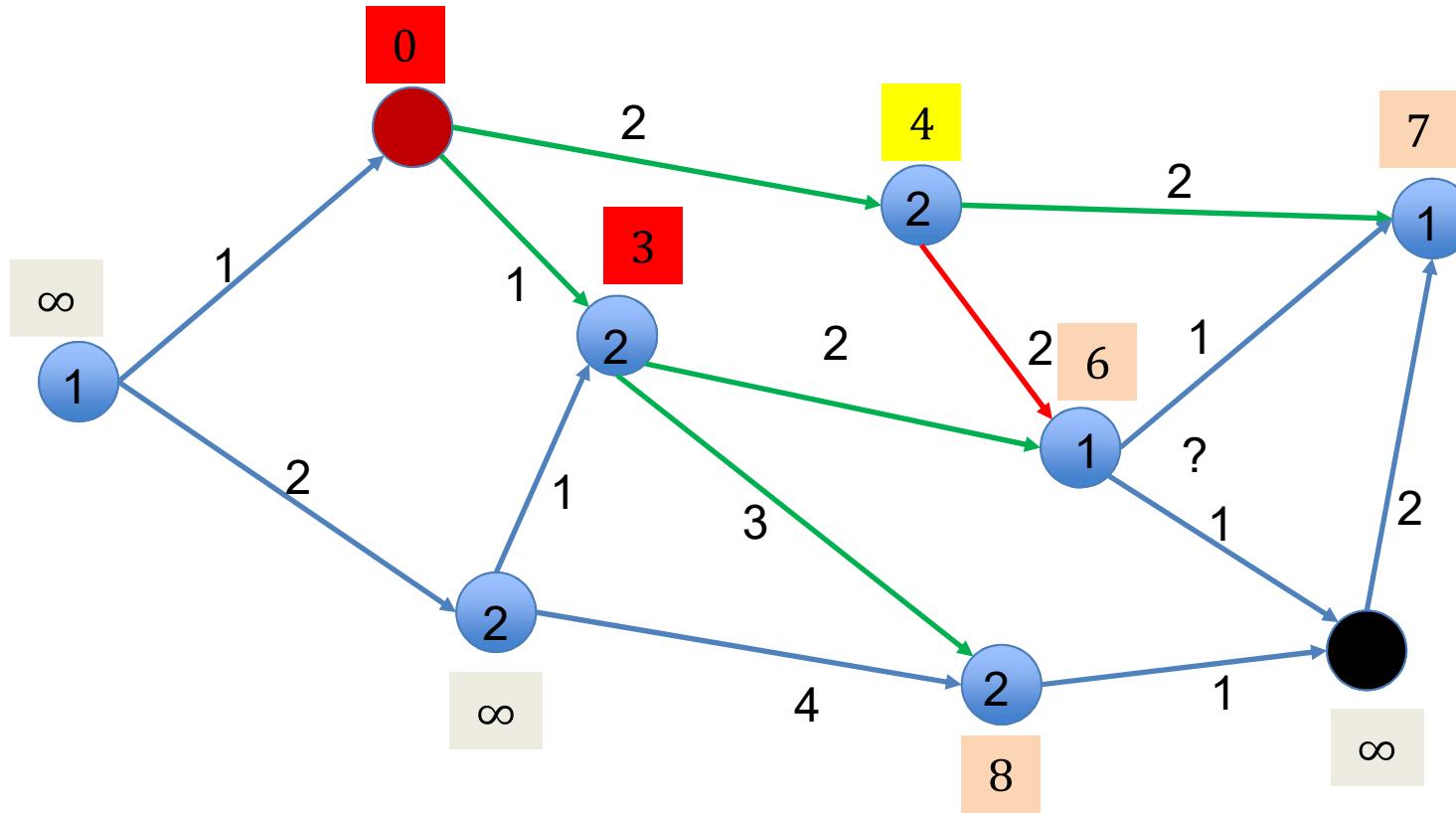
- Expand it out as before
- What do we do when an expansion arrives at a node that *already* has a cost assigned to it?

Dijkstra's Algorithm



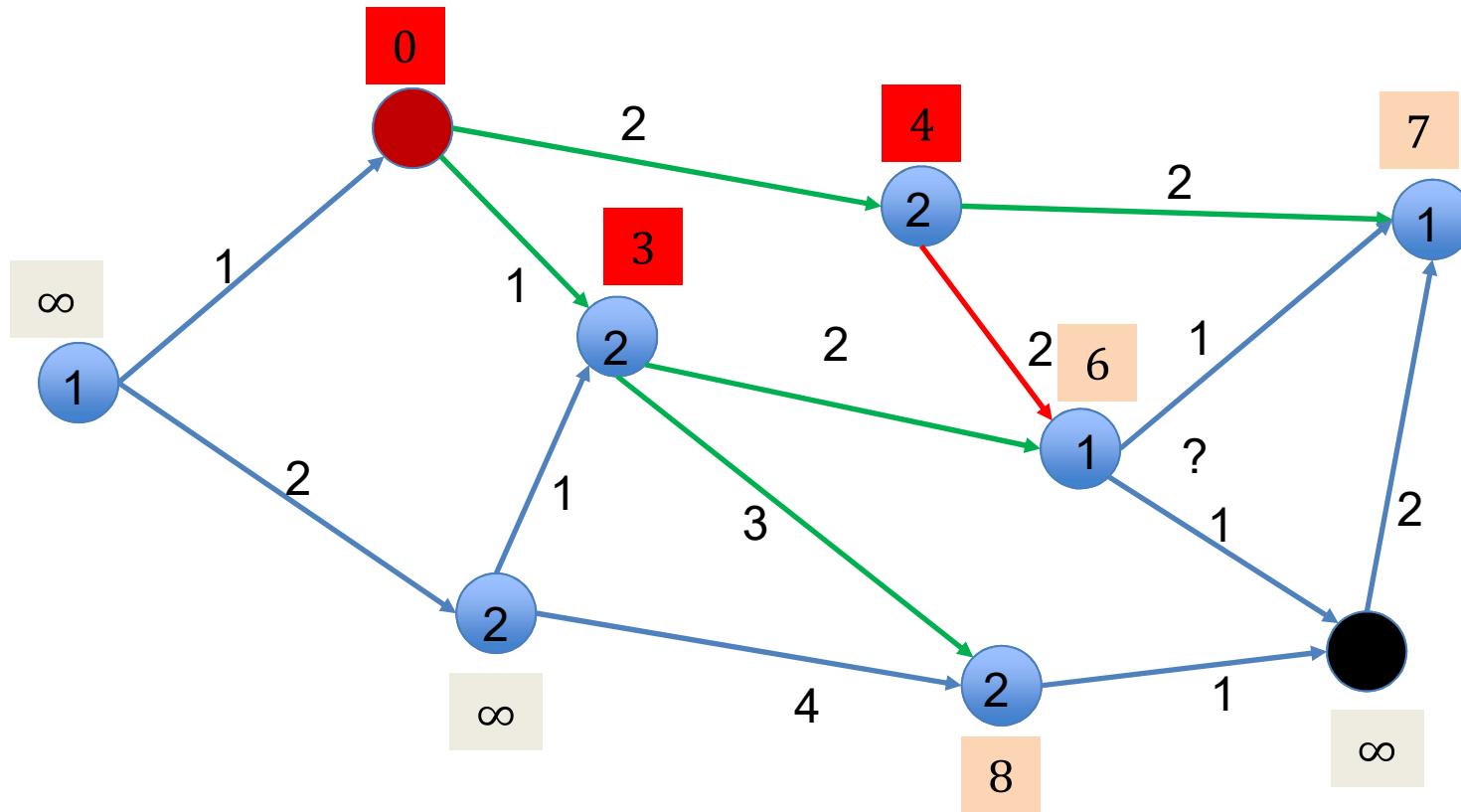
- Retain the *lower* of the current cost at the node and the cost of the expanded incoming edge

Dijkstra's Algorithm

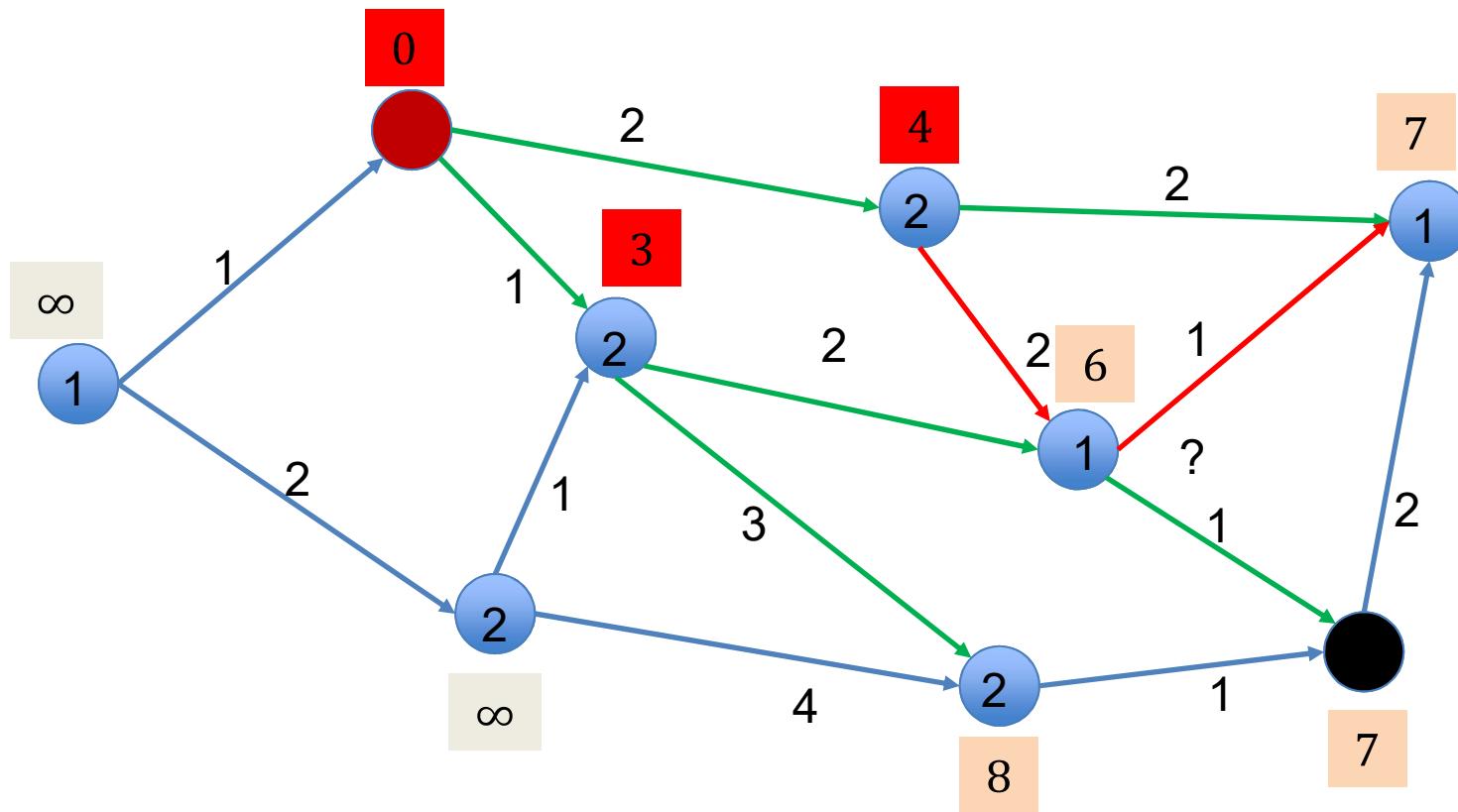


- Retain the *lower* of the current cost at the node and the cost of the expanded incoming edge

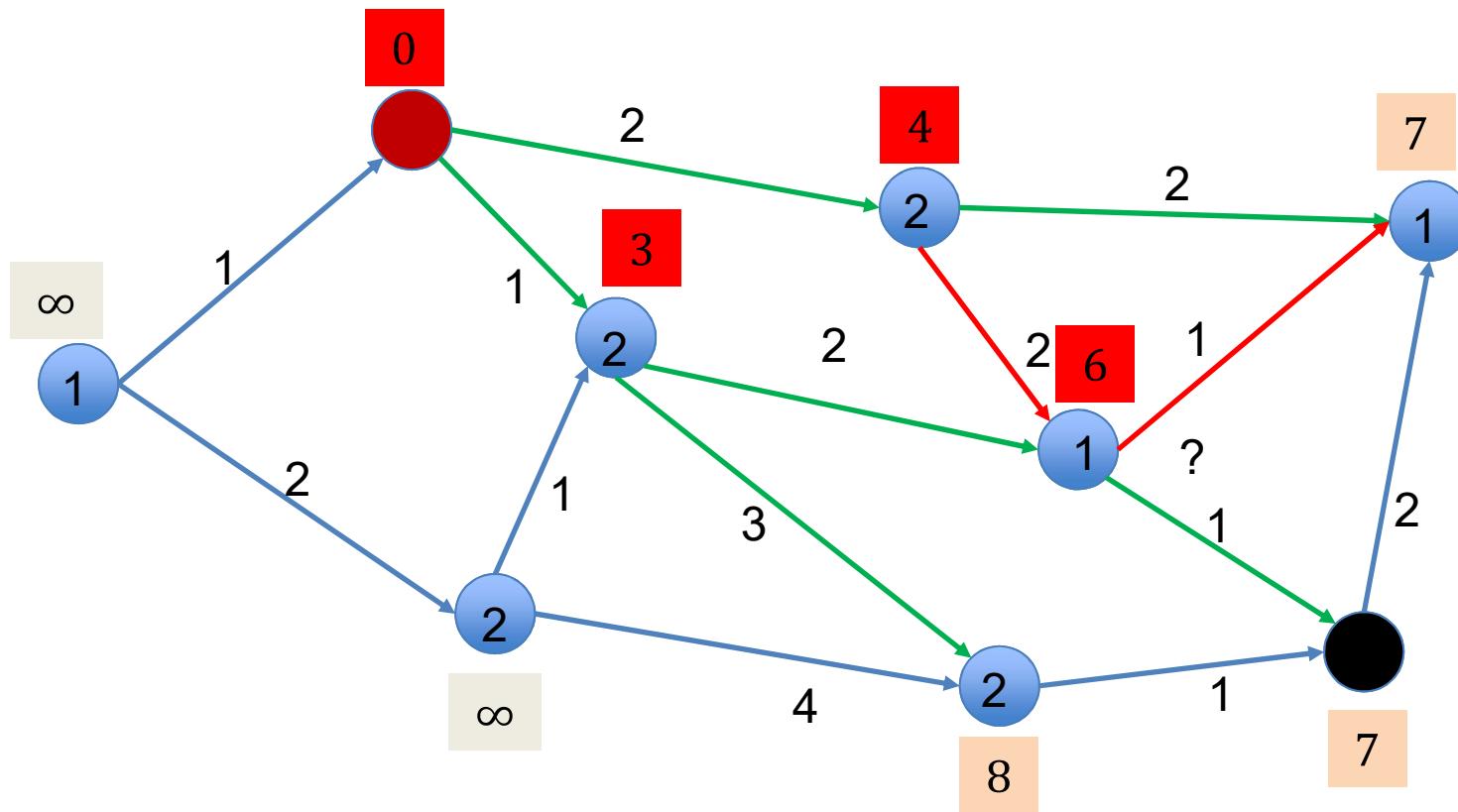
Dijkstra's Algorithm



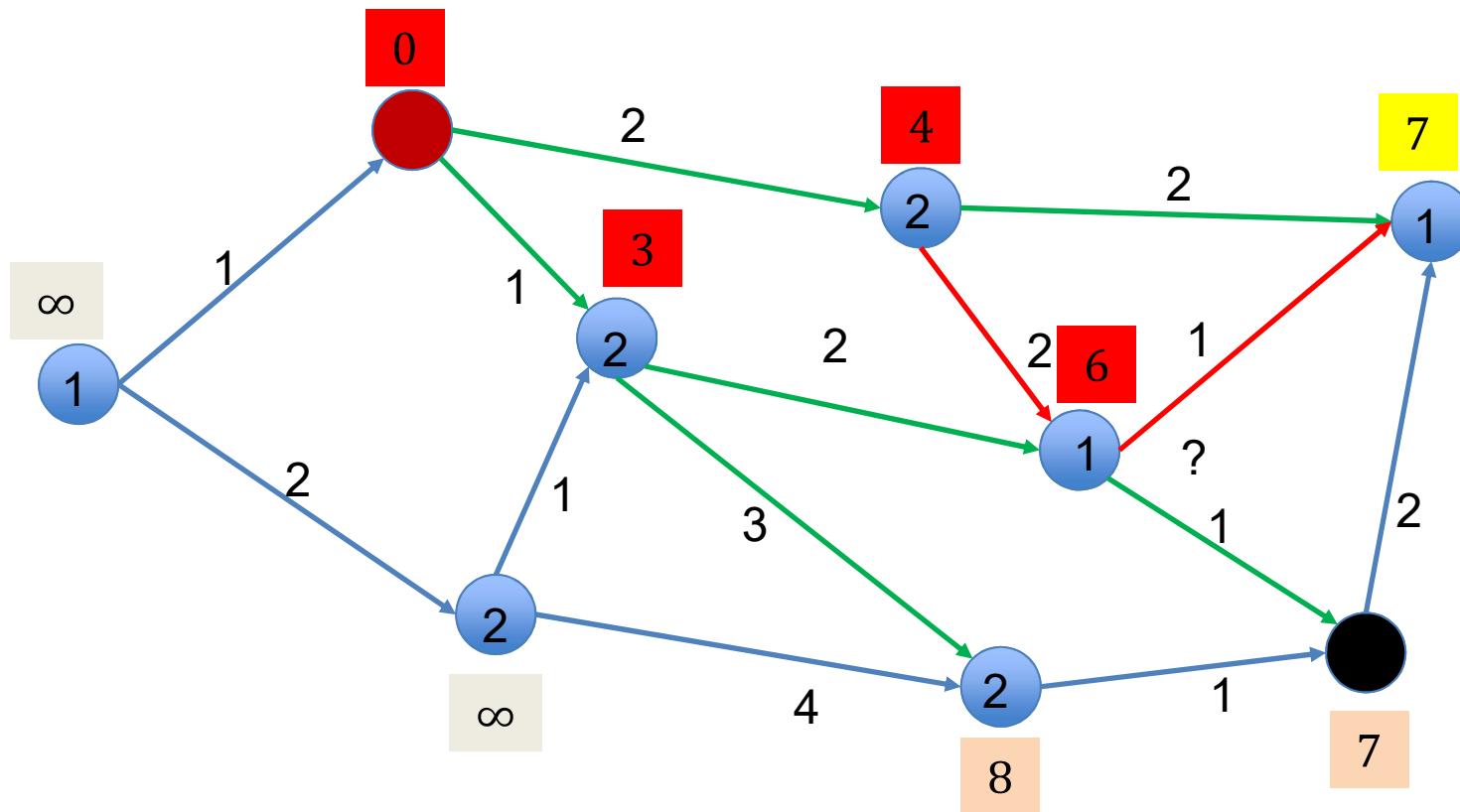
Dijkstra's Algorithm



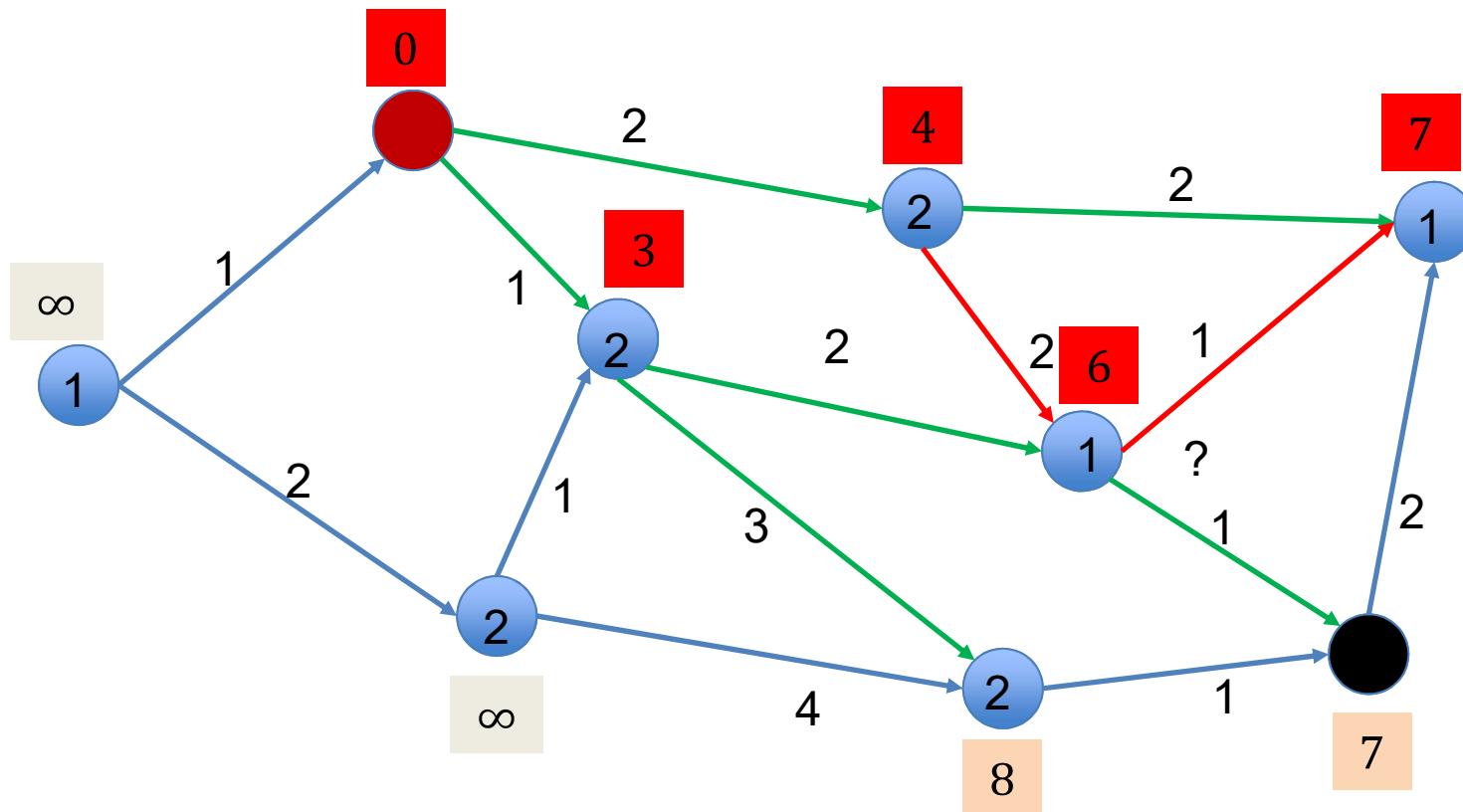
Dijkstra's Algorithm



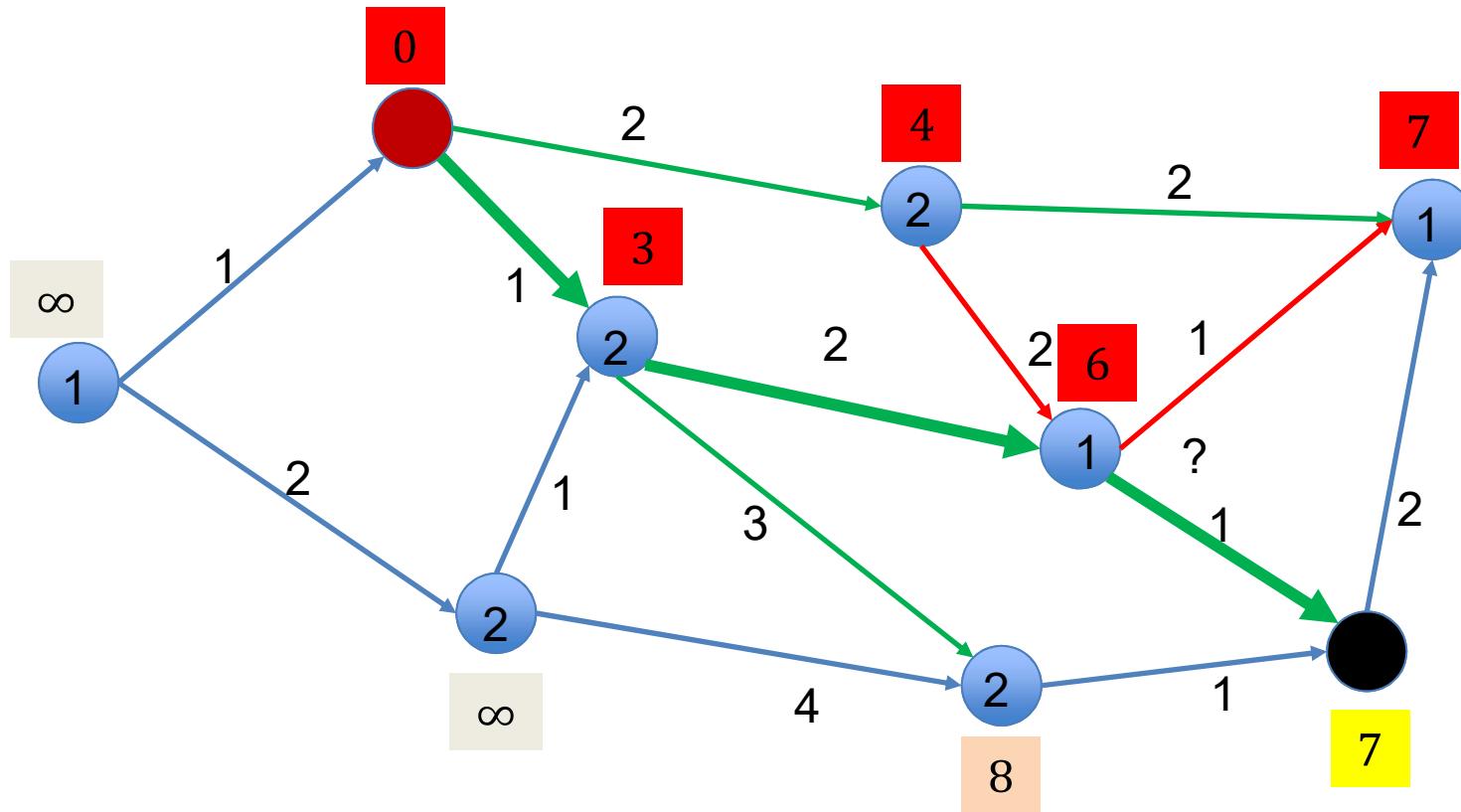
Dijkstra's Algorithm



Dijkstra's Algorithm

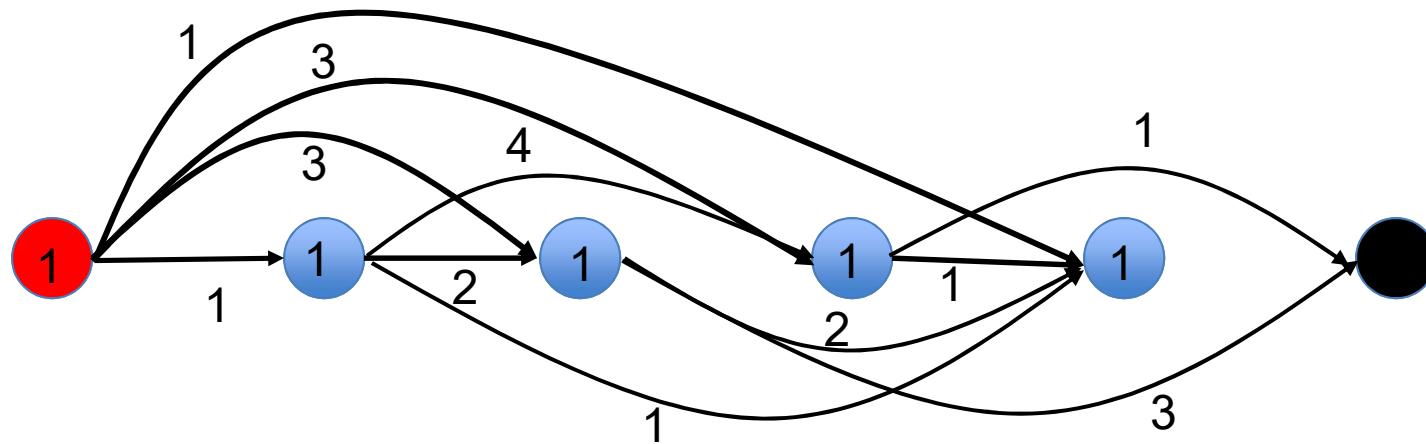


Dijkstra's Algorithm



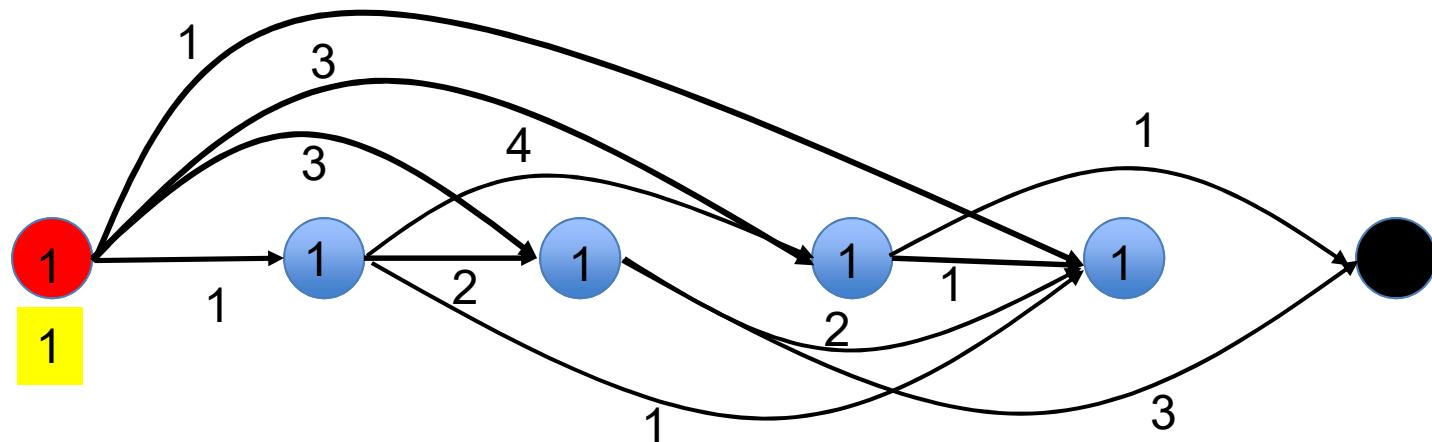
- When the least cost node is the target node, we're done
- We have the cost of the shortest path from the red to the black node

A variation on the problem



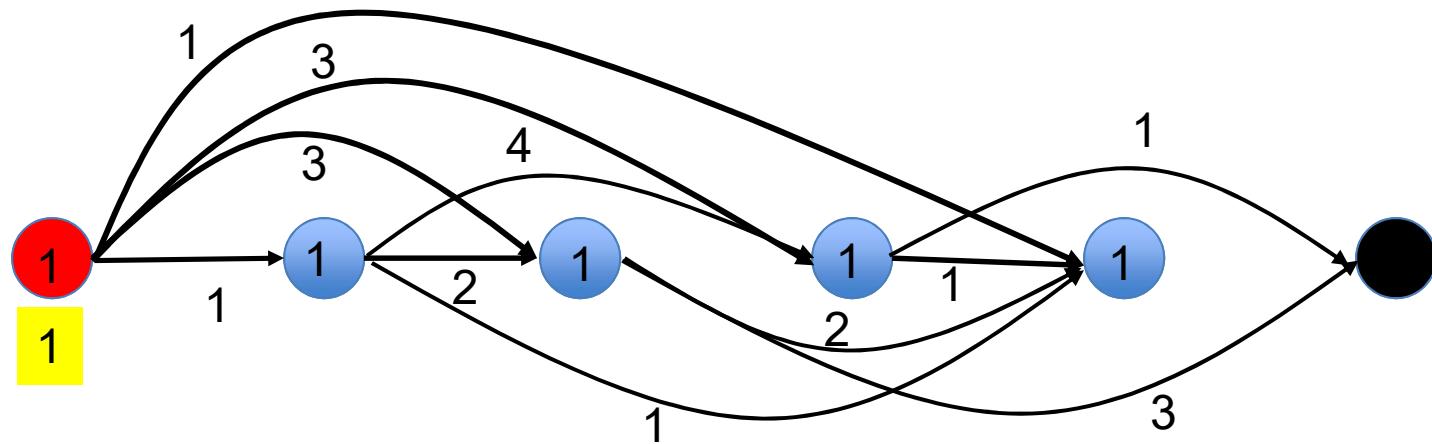
- When the graph is topologically sorted..
 - Assumption: No nodes before the identified source node
 - In practice, we can “kill” all earlier nodes and their outgoing edges without affecting the result

A variation on the problem



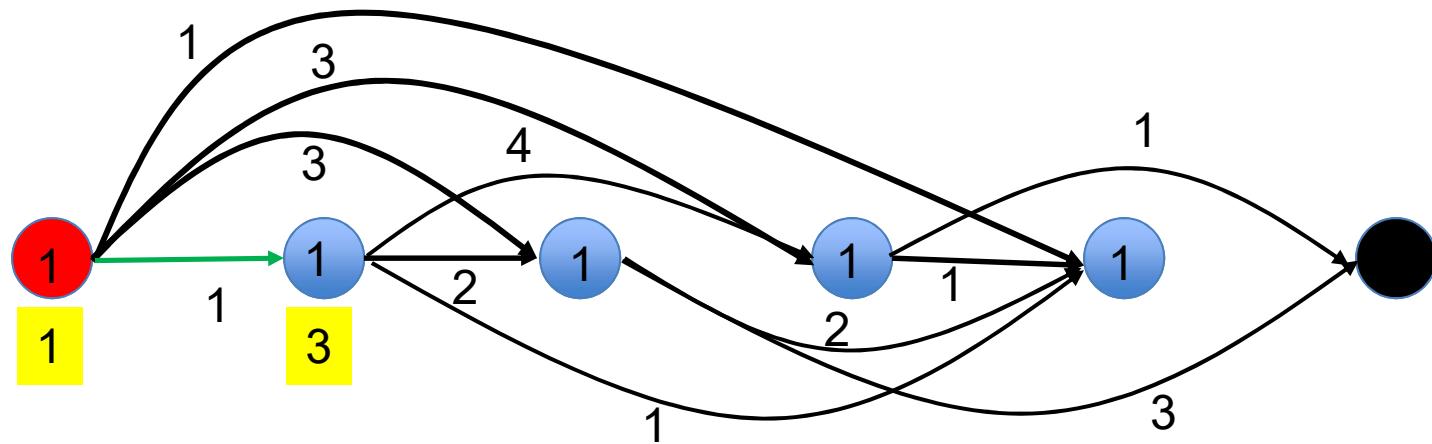
- Start from the source node

A variation on the problem



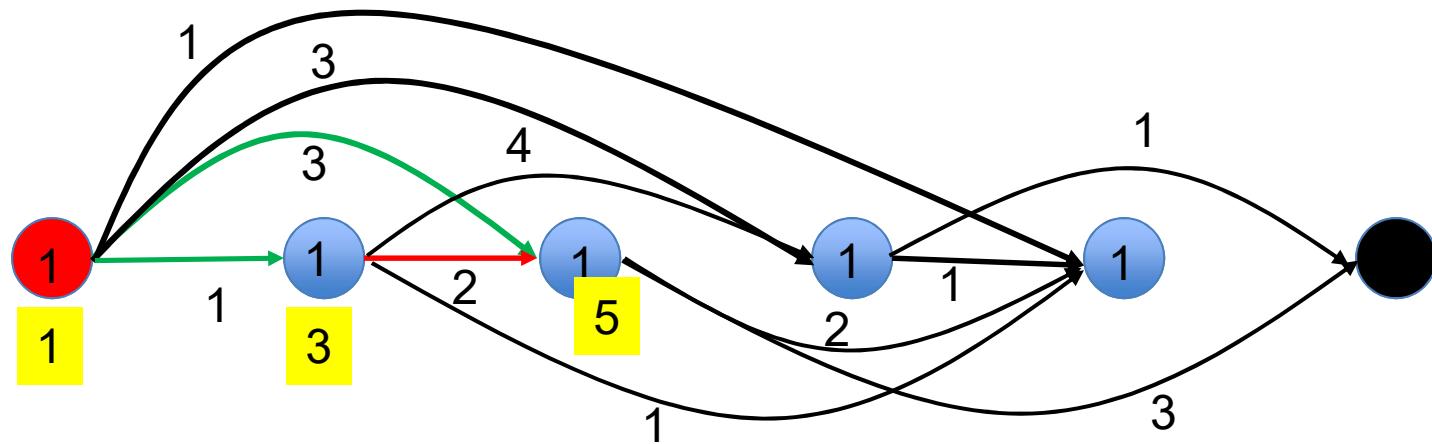
- Start from the source node
- Going left to right, for each subsequent node, retain the *lowest* incoming cost

A variation on the problem



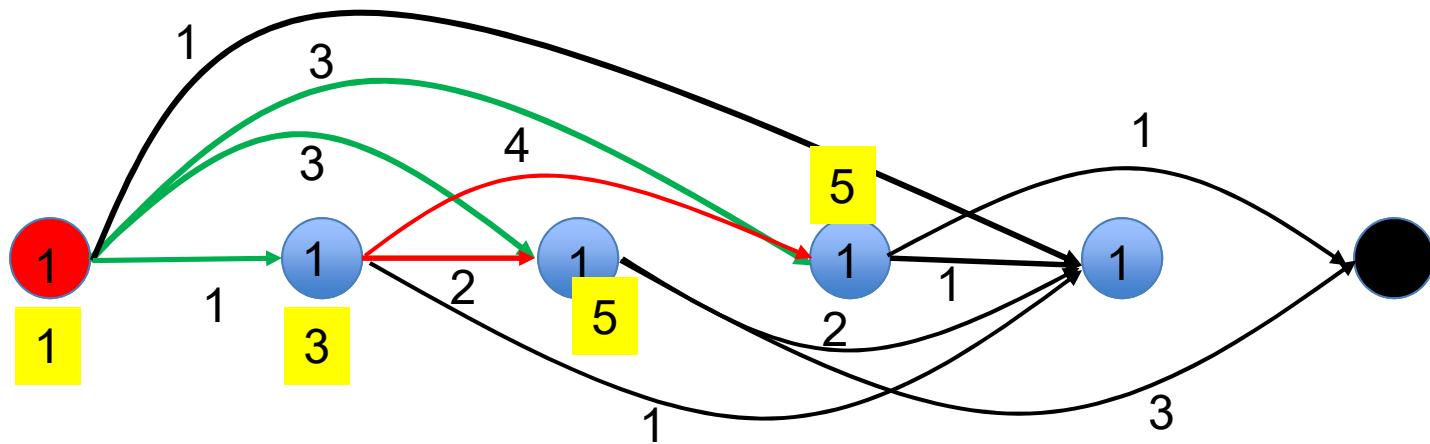
- Start from the source node
- Going left to right, for each subsequent node, retain the *lowest* incoming cost

A variation on the problem



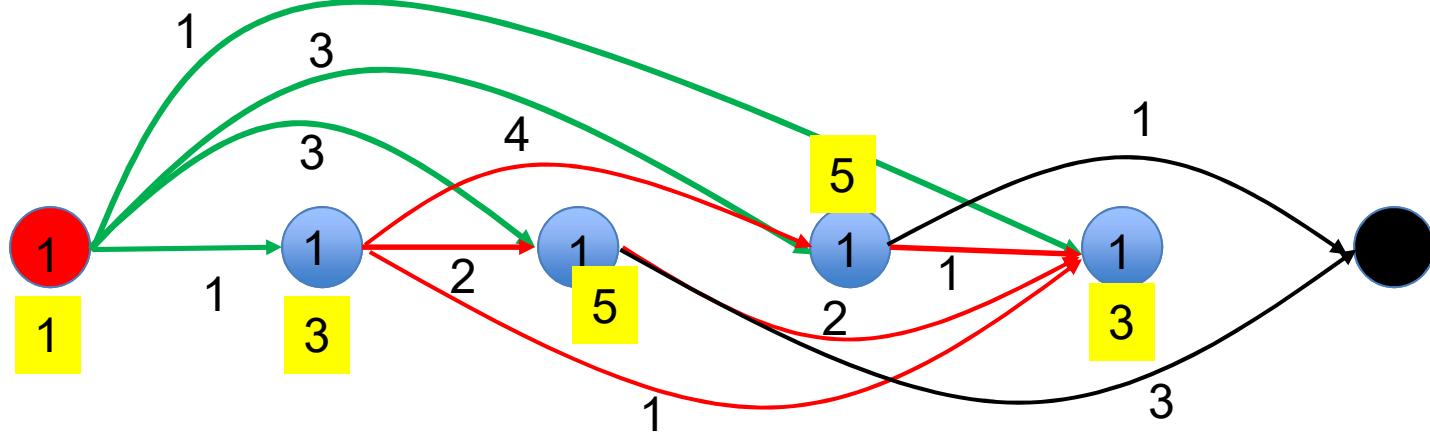
- Start from the source node
- Going left to right, for each subsequent node, retain the *lowest* incoming cost

A variation on the problem



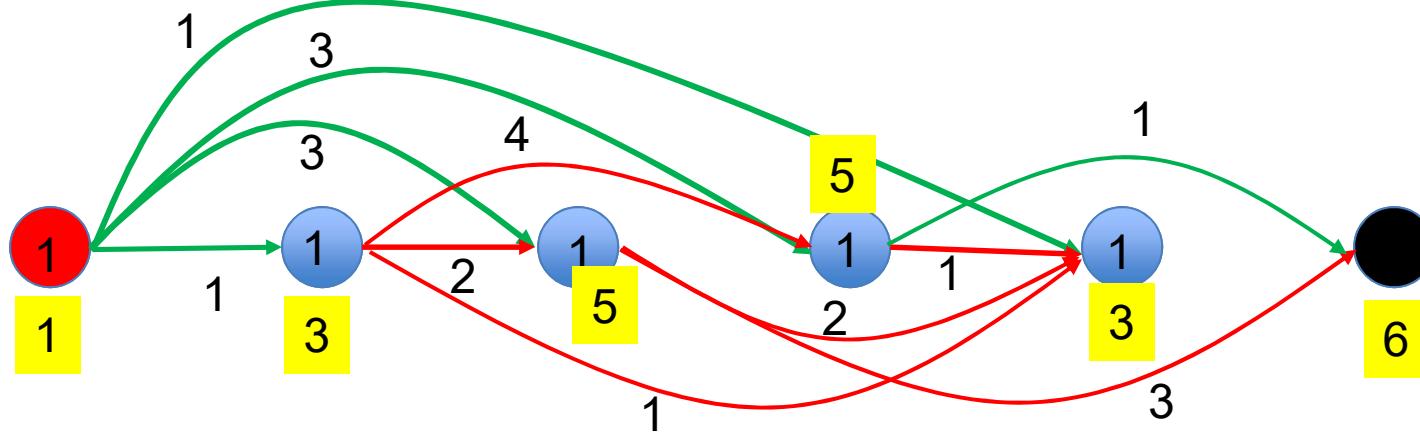
- Start from the source node
- Going left to right, for each subsequent node, retain the *lowest* incoming cost

A variation on the problem



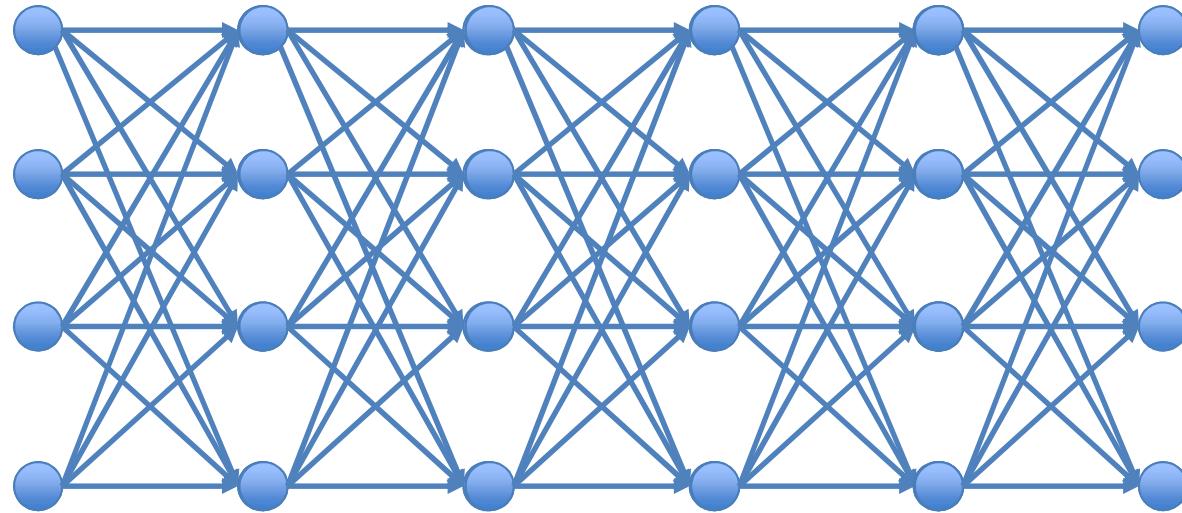
- Start from the source node
- Going left to right, for each subsequent node, retain the *lowest* incoming cost

A variation on the problem



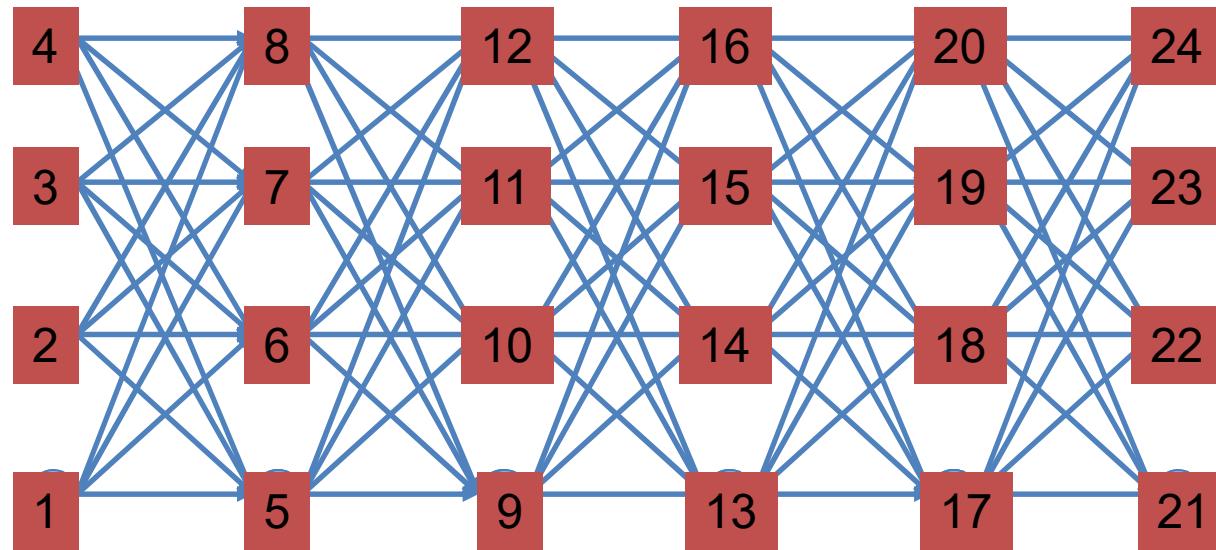
- Start from the source node
- Going left to right, for each subsequent node, retain the *lowest* incoming cost

Is this graph topologically sorted?



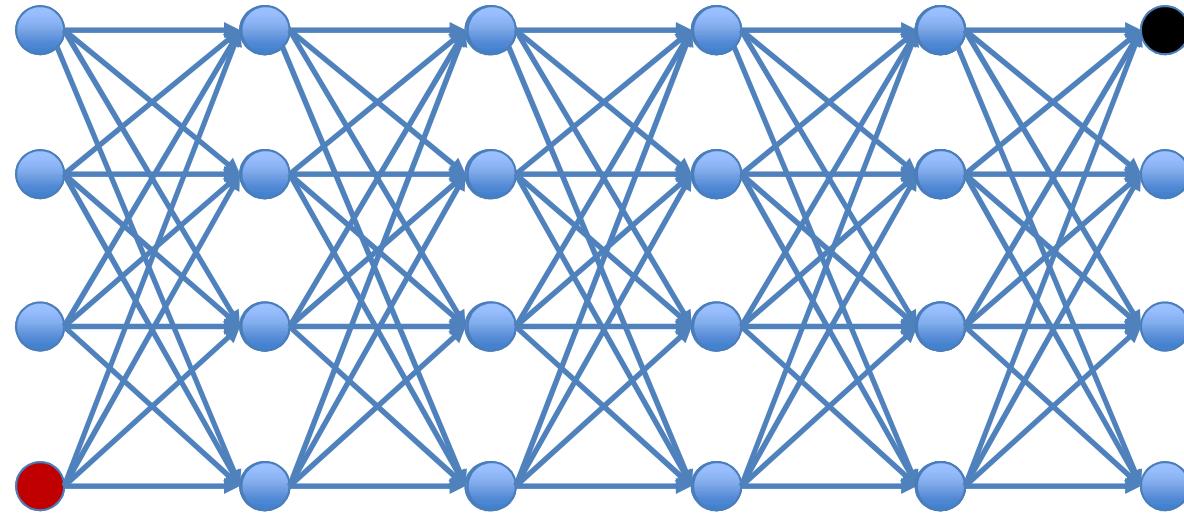
- Nodes on each column only connect to the nodes in the previous column(s)

Is this graph topologically sorted?



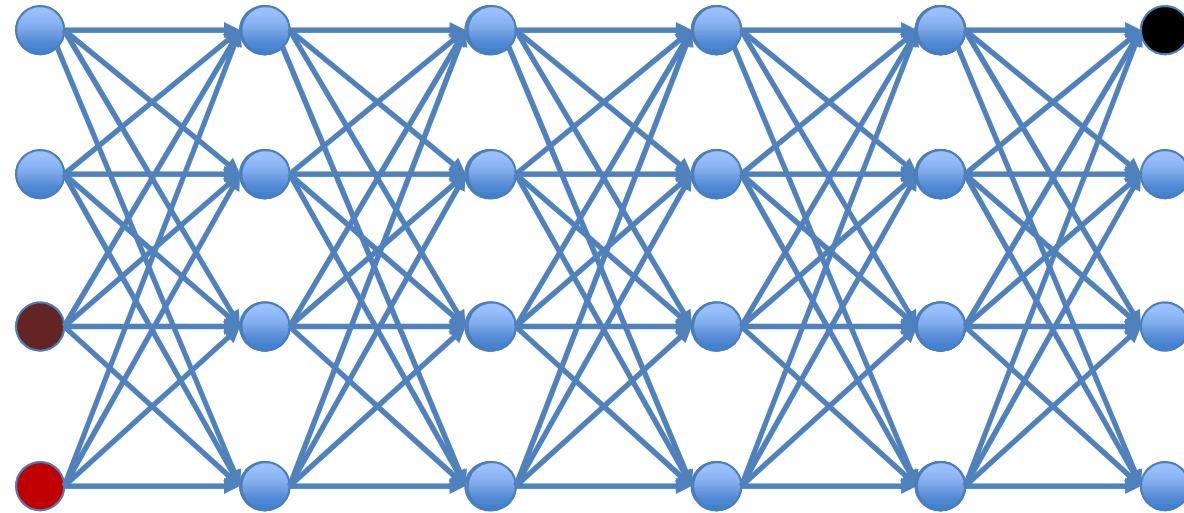
- Nodes on each column only connect to the nodes in the previous column(s)
- The sorted best path algorithm applies!

Shortest path algorithm



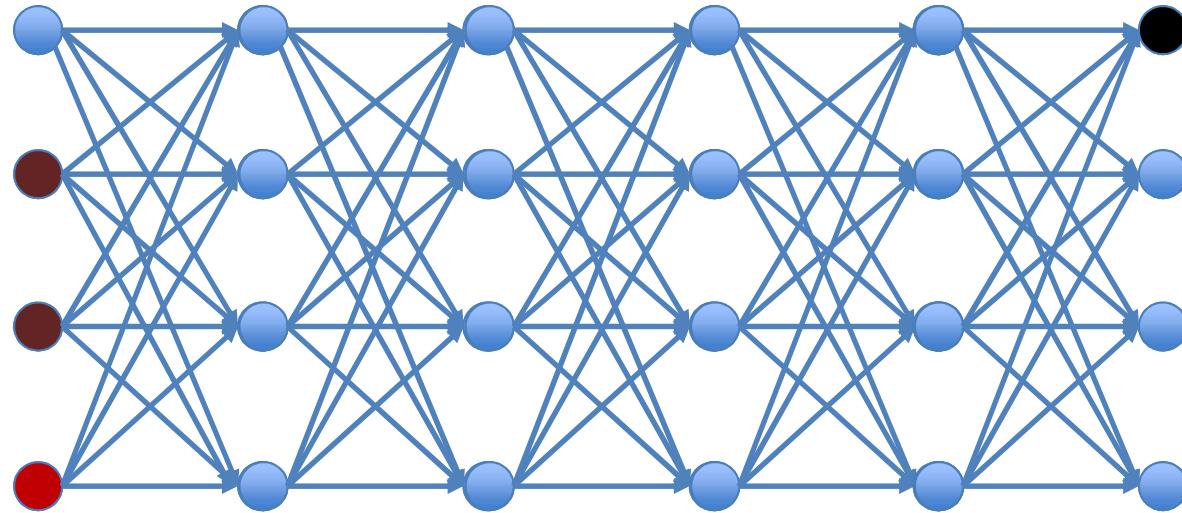
- Order of evaluation to find the shortest path between the given source and destination

Shortest path algorithm



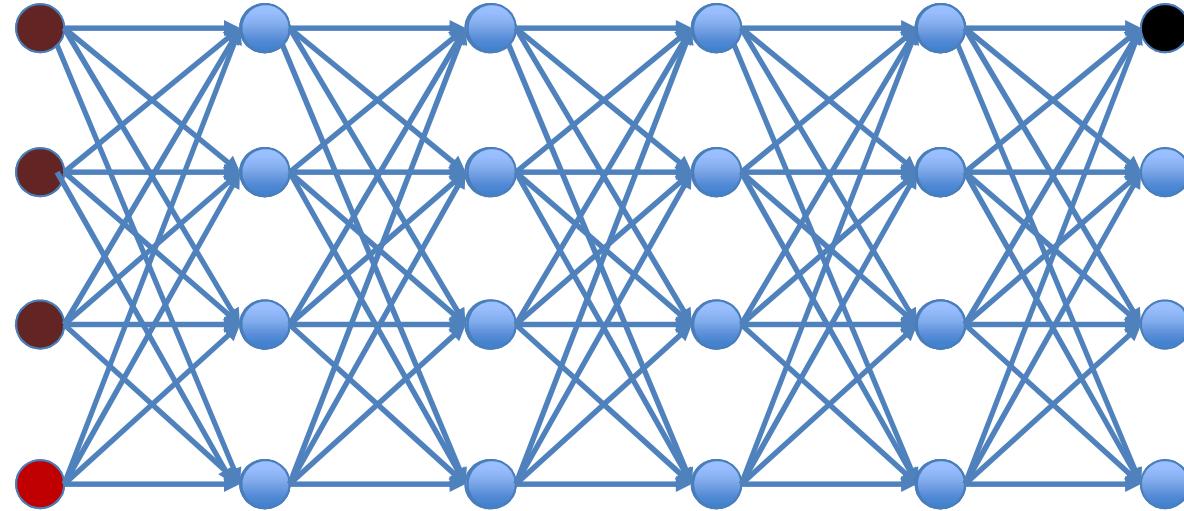
- Order of evaluation to find the shortest path between the given source and destination

Shortest path algorithm



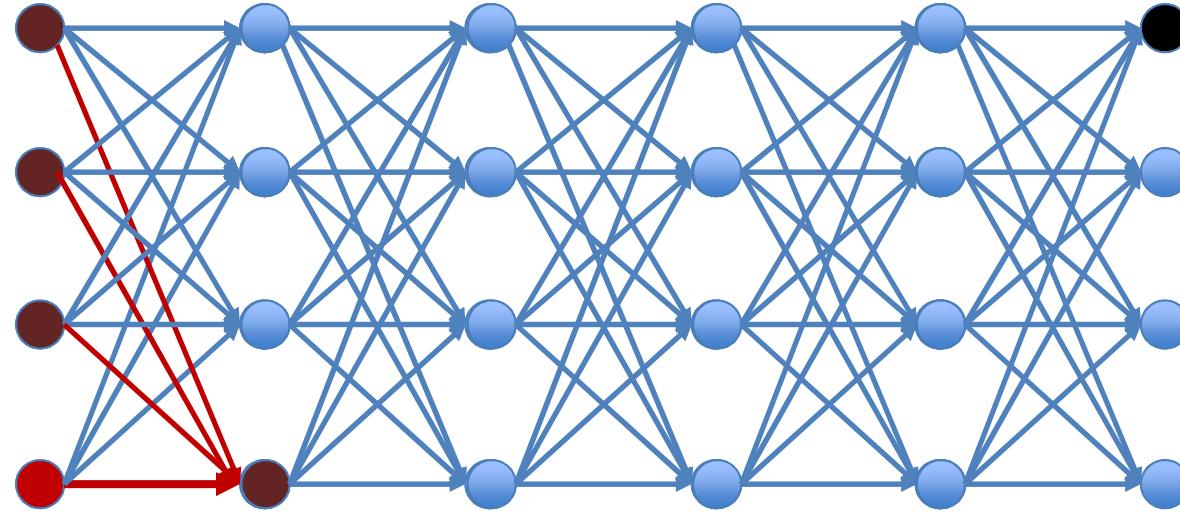
- Order of evaluation to find the shortest path between the given source and destination

Shortest path algorithm



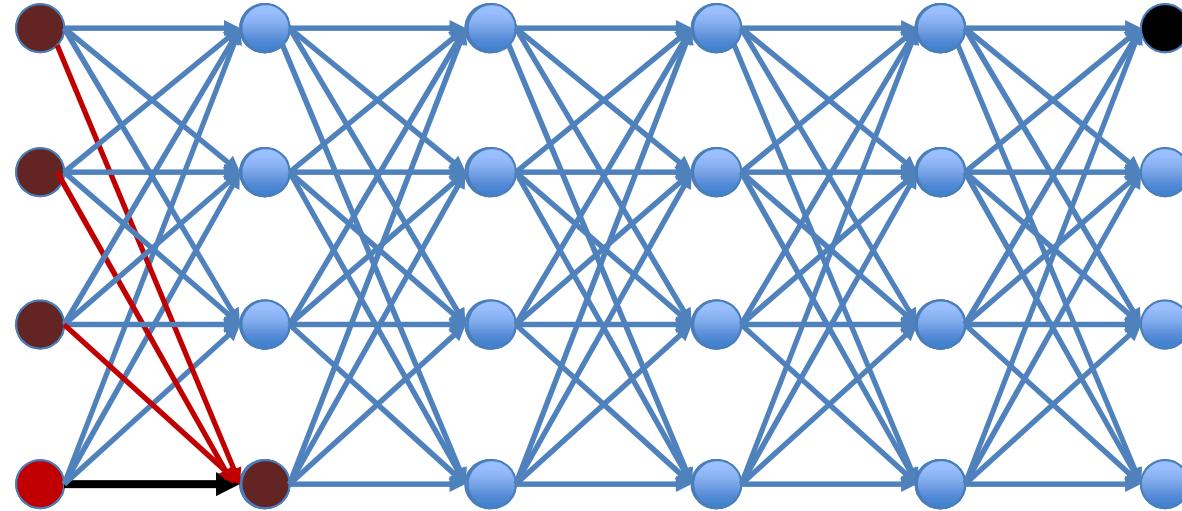
- Order of evaluation to find the shortest path between the given source and destination

Shortest path algorithm



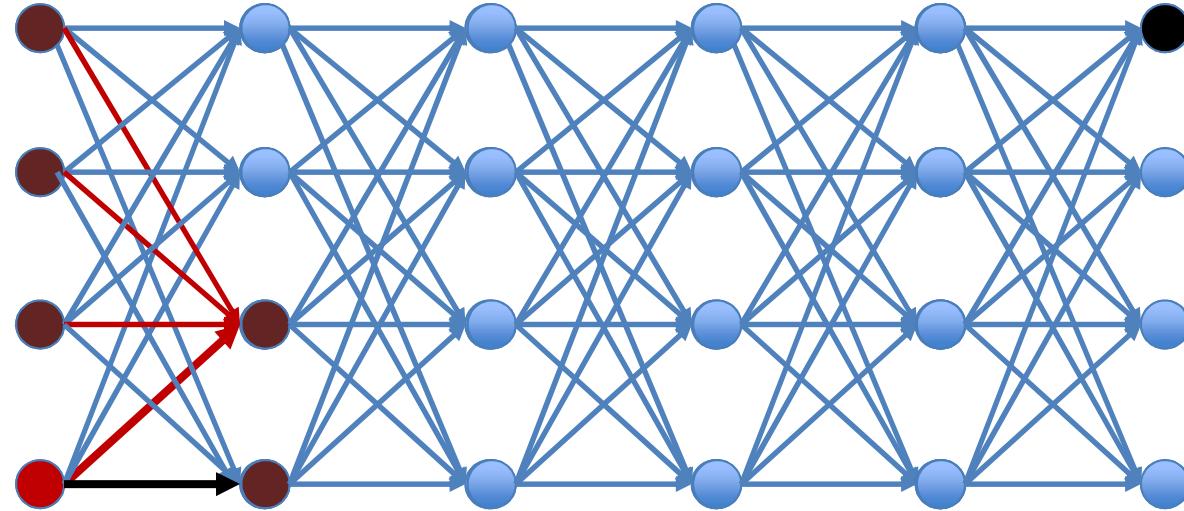
- Order of evaluation to find the shortest path between the given source and destination

Shortest path algorithm



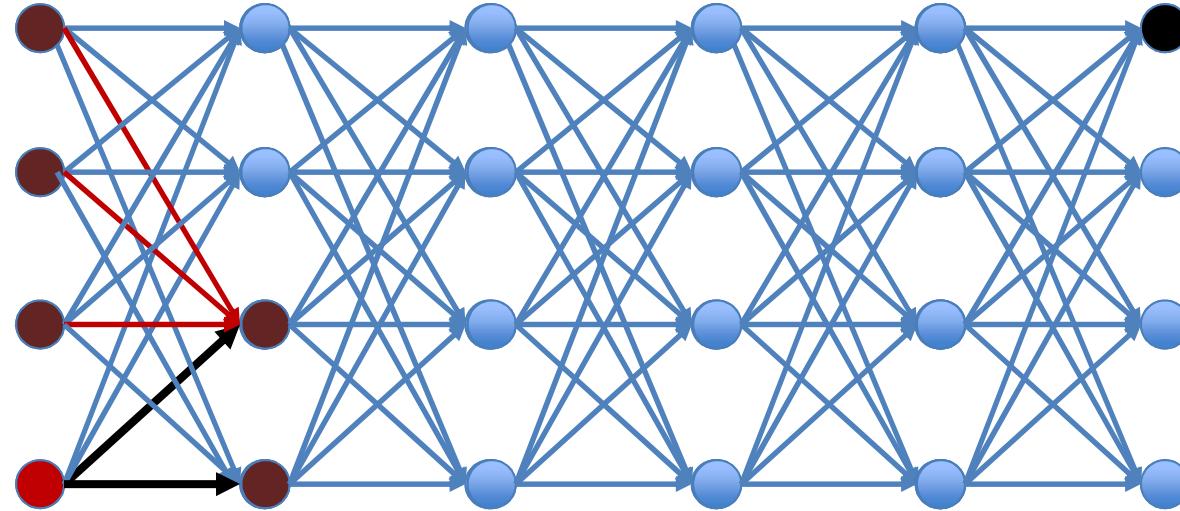
- Order of evaluation to find the shortest path between the given source and destination

Shortest path algorithm



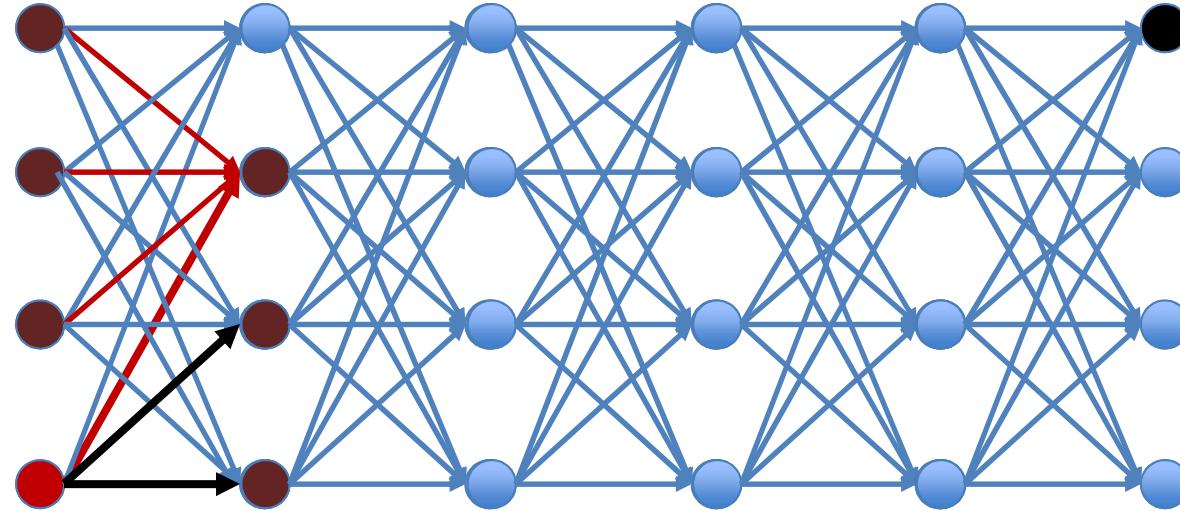
- Order of evaluation to find the shortest path between the given source and destination

Shortest path algorithm



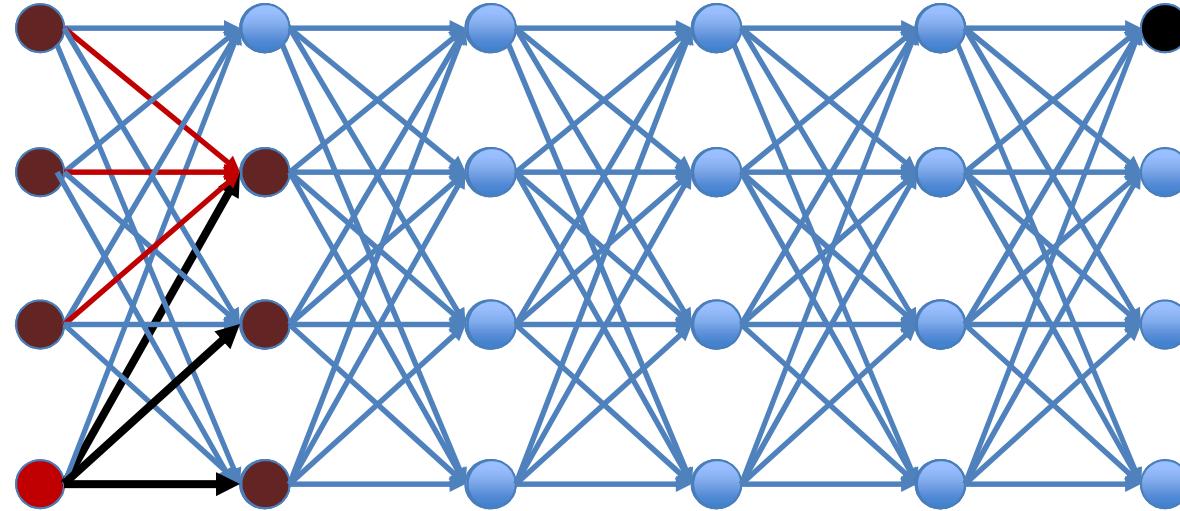
- Order of evaluation to find the shortest path between the given source and destination

Shortest path algorithm



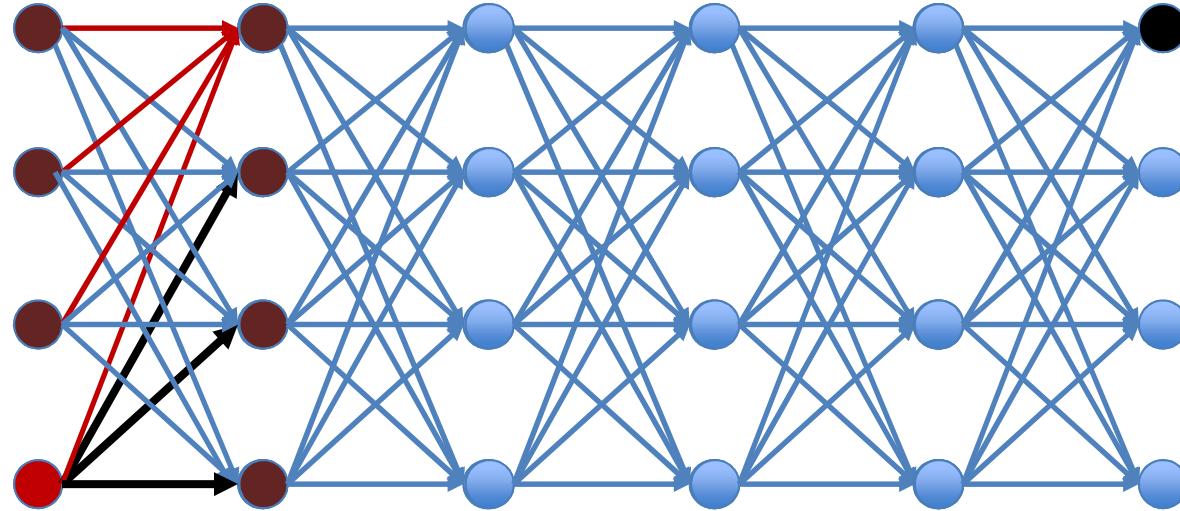
- Order of evaluation to find the shortest path between the given source and destination

Shortest path algorithm



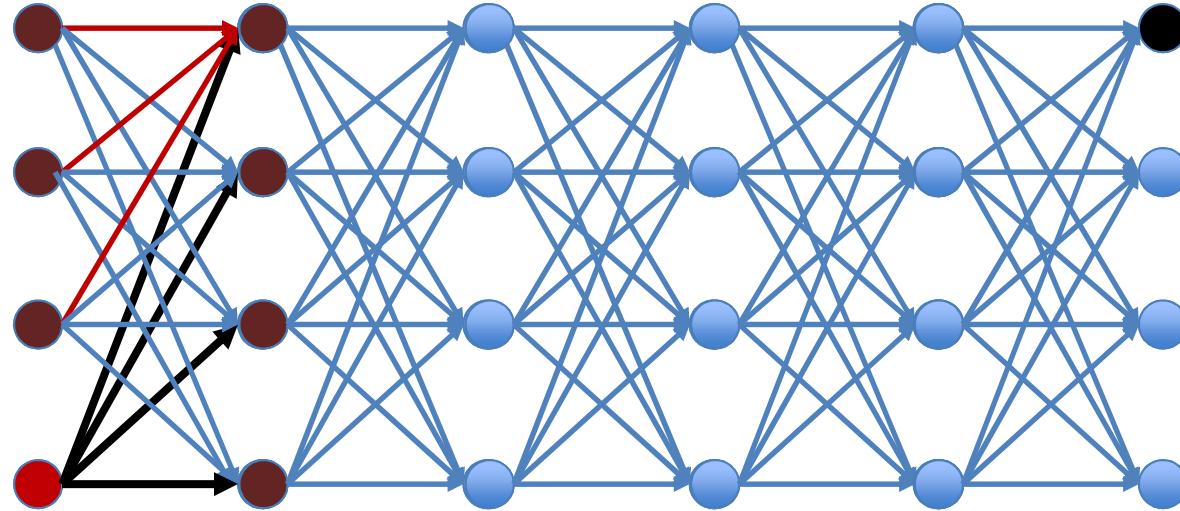
- Order of evaluation to find the shortest path between the given source and destination

Shortest path algorithm



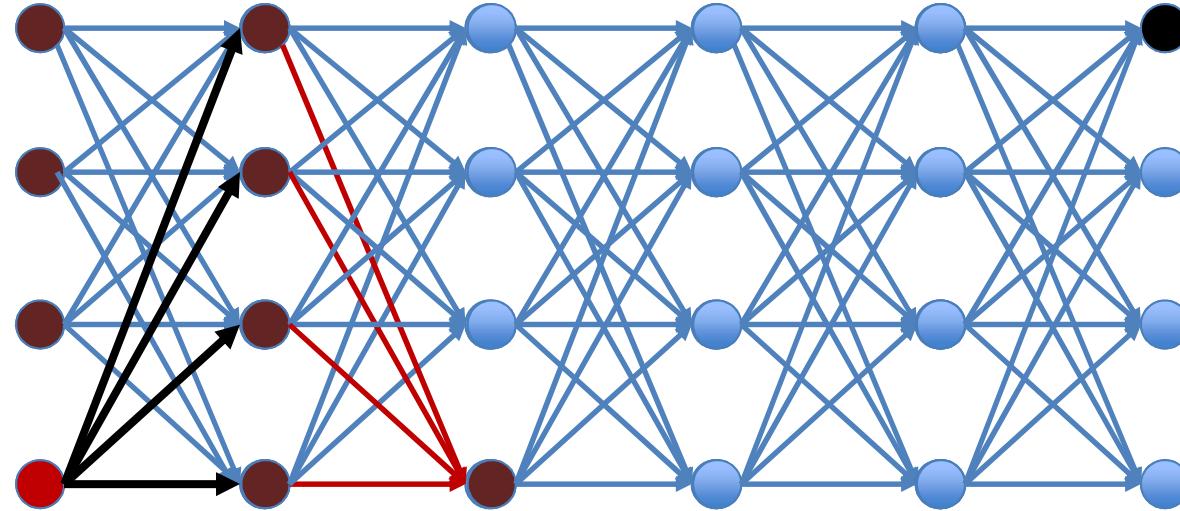
- Order of evaluation to find the shortest path between the given source and destination

Shortest path algorithm



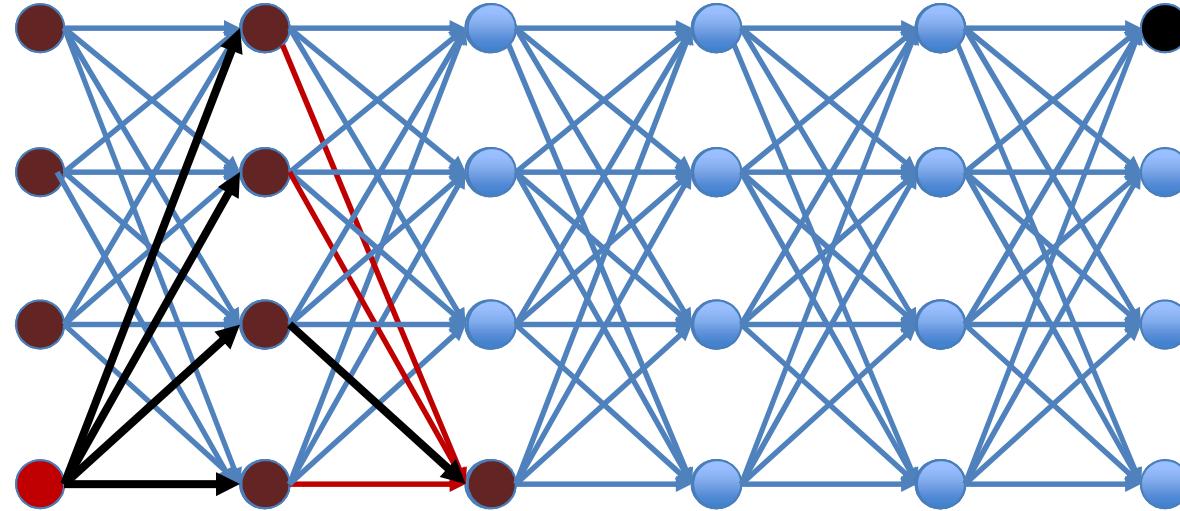
- Order of evaluation to find the shortest path between the given source and destination

Shortest path algorithm



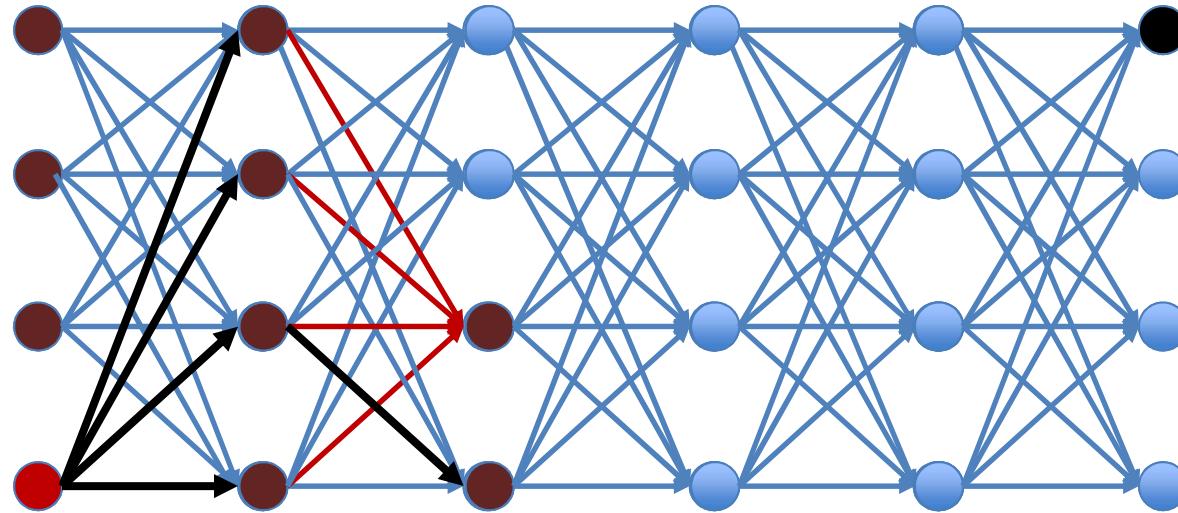
- Order of evaluation to find the shortest path between the given source and destination

Shortest path algorithm



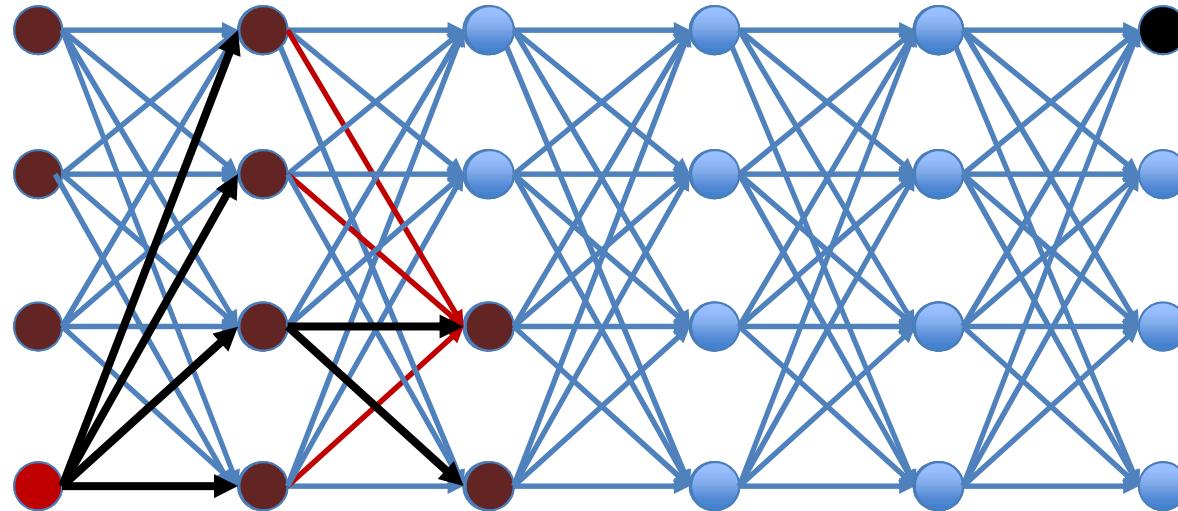
- Order of evaluation to find the shortest path between the given source and destination

Shortest path algorithm



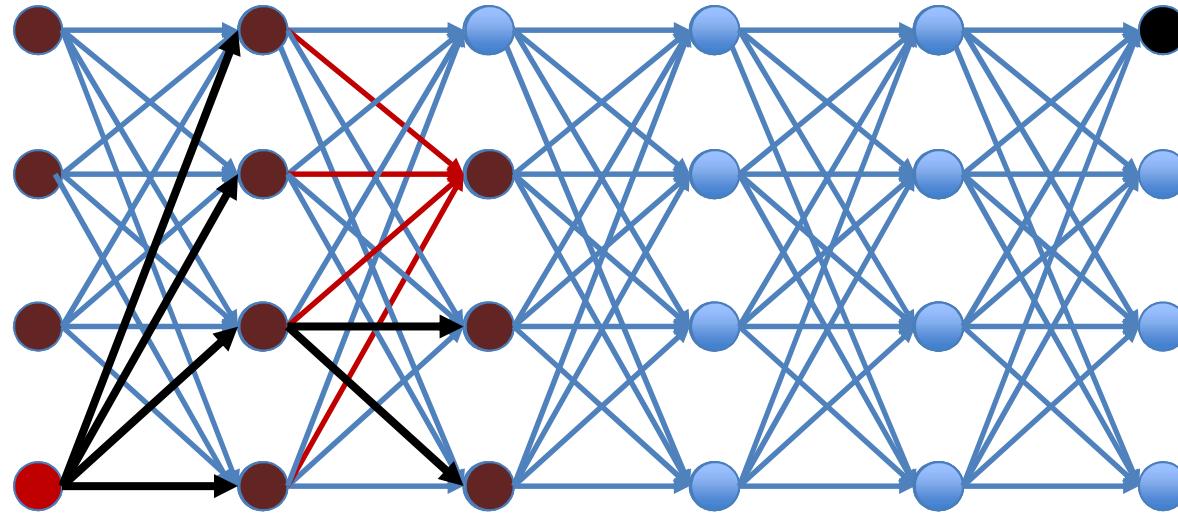
- Order of evaluation to find the shortest path between the given source and destination

Shortest path algorithm



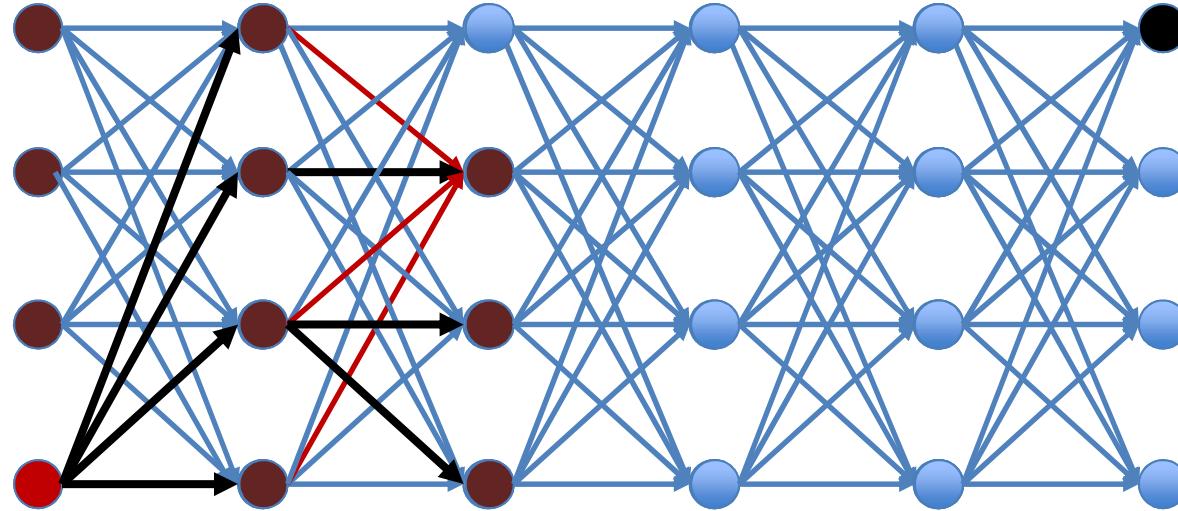
- Order of evaluation to find the shortest path between the given source and destination

Shortest path algorithm



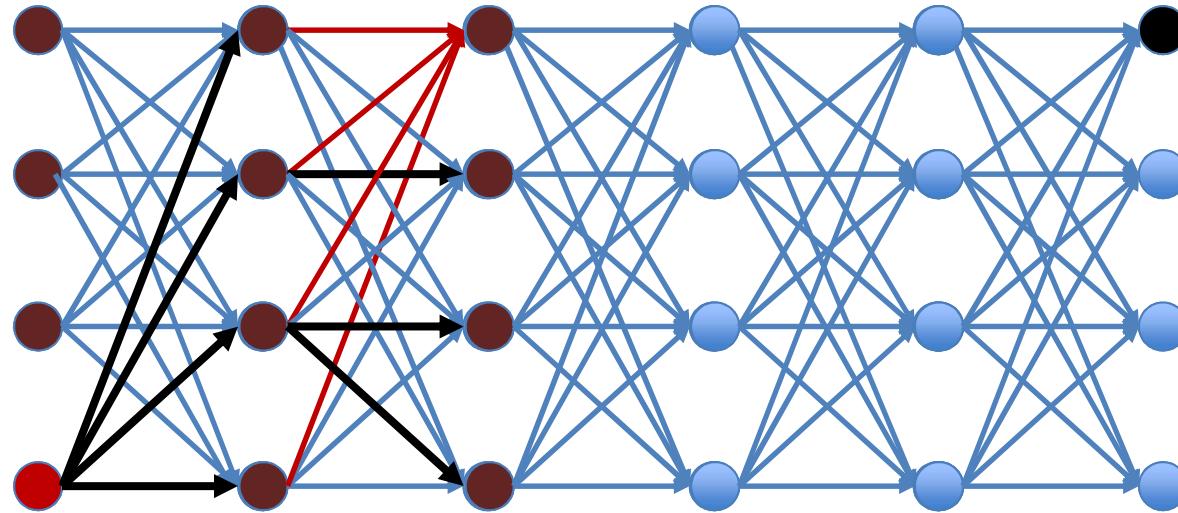
- Order of evaluation to find the shortest path between the given source and destination

Shortest path algorithm



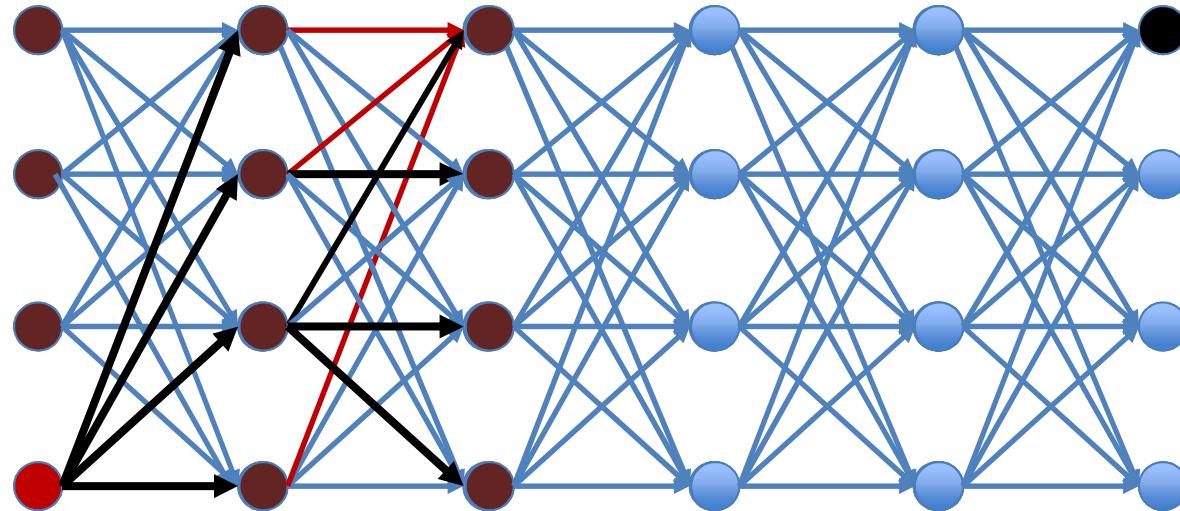
- Order of evaluation to find the shortest path between the given source and destination

Shortest path algorithm



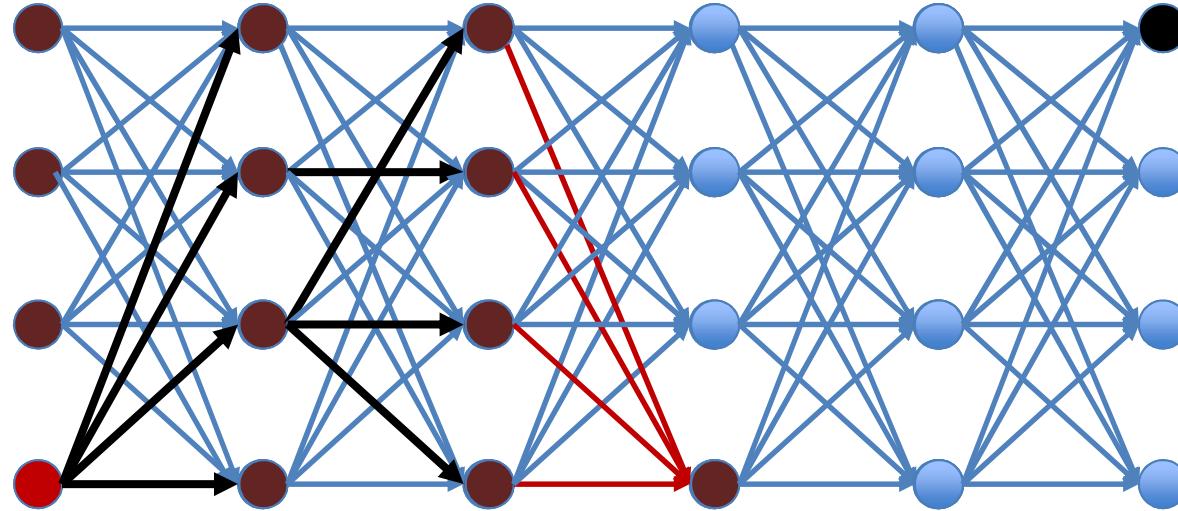
- Order of evaluation to find the shortest path between the given source and destination

Shortest path algorithm



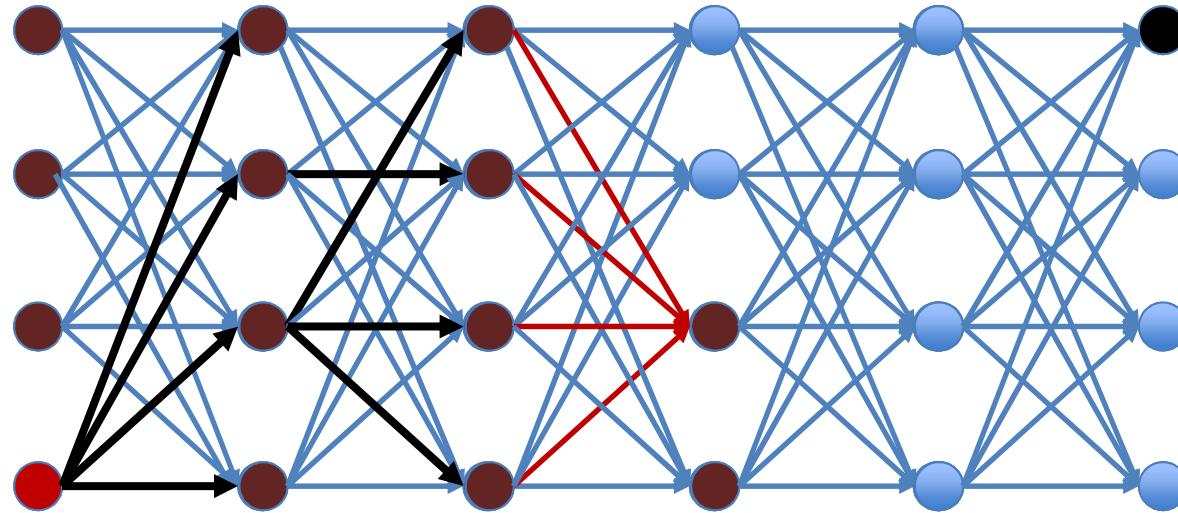
- Order of evaluation to find the shortest path between the given source and destination

Shortest path algorithm



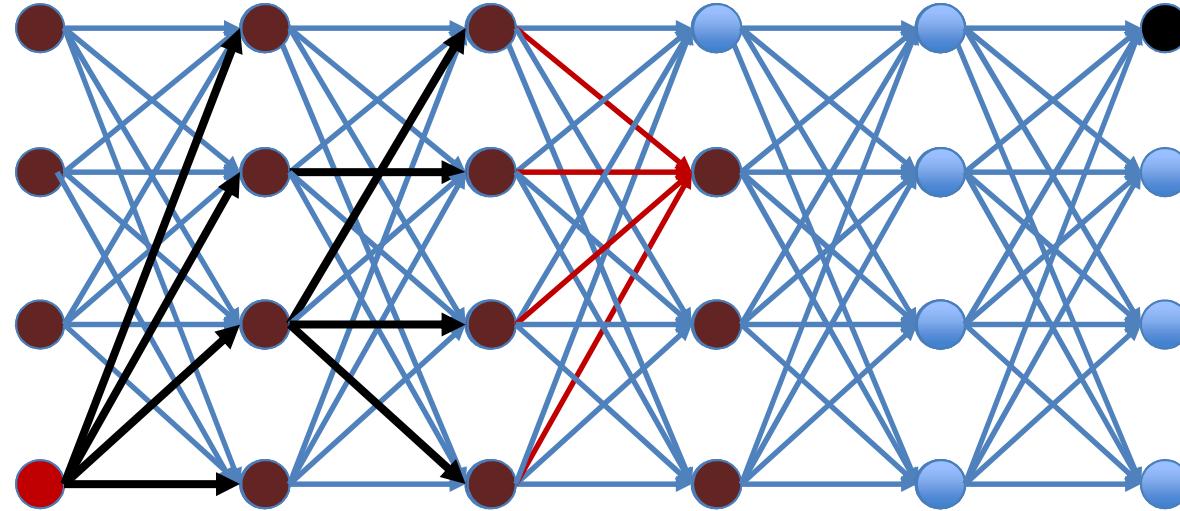
- Order of evaluation to find the shortest path between the given source and destination

Shortest path algorithm



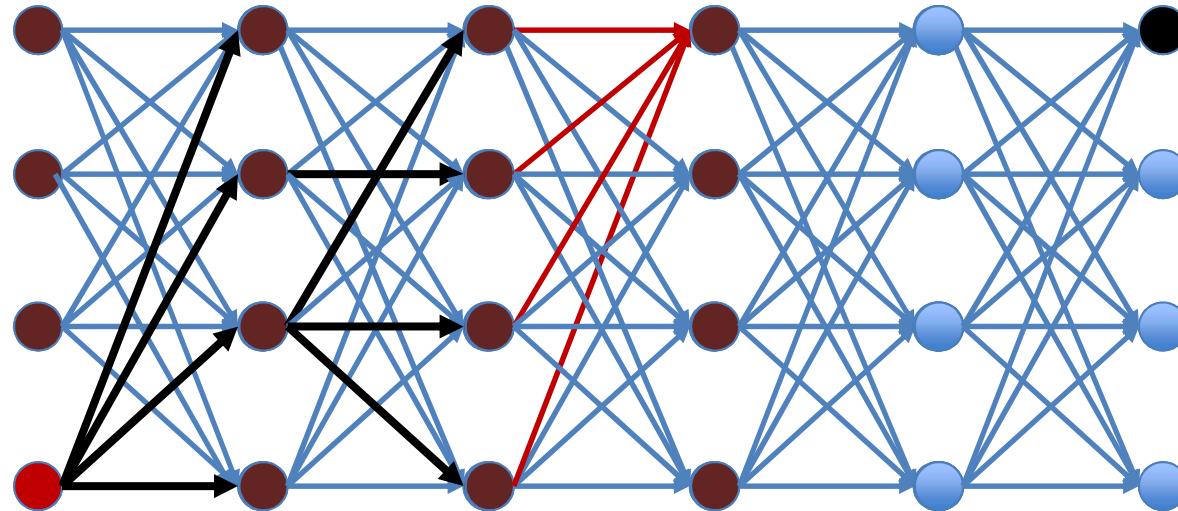
- Order of evaluation to find the shortest path between the given source and destination

Shortest path algorithm



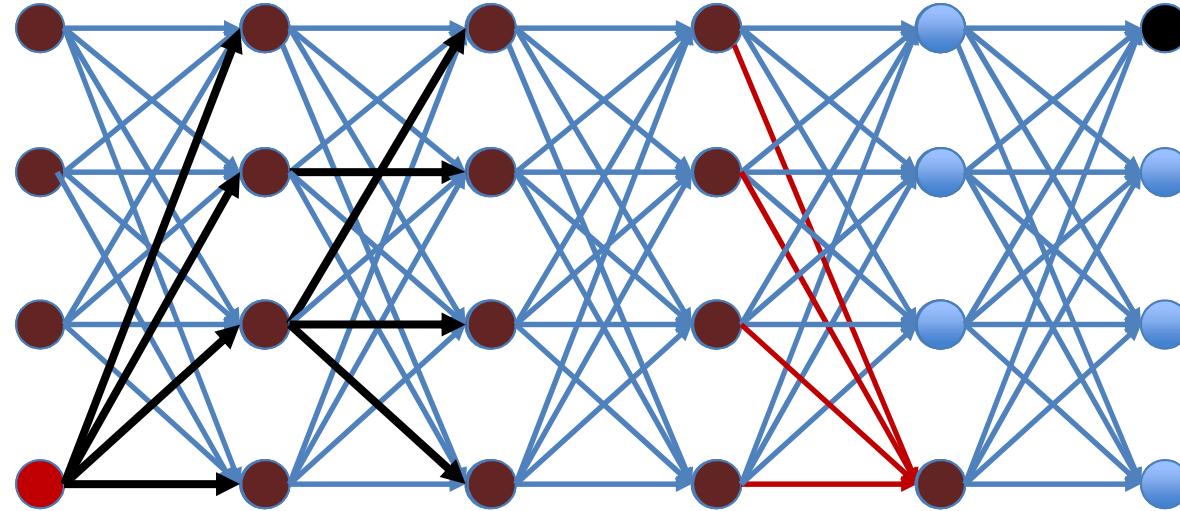
- Order of evaluation to find the shortest path between the given source and destination

Shortest path algorithm



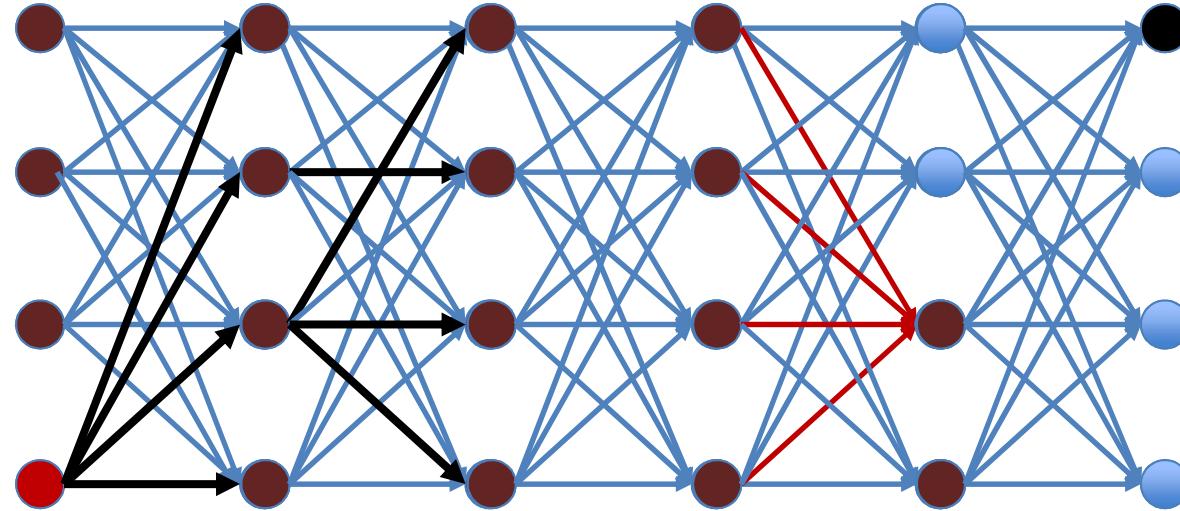
- Order of evaluation to find the shortest path between the given source and destination

Shortest path algorithm



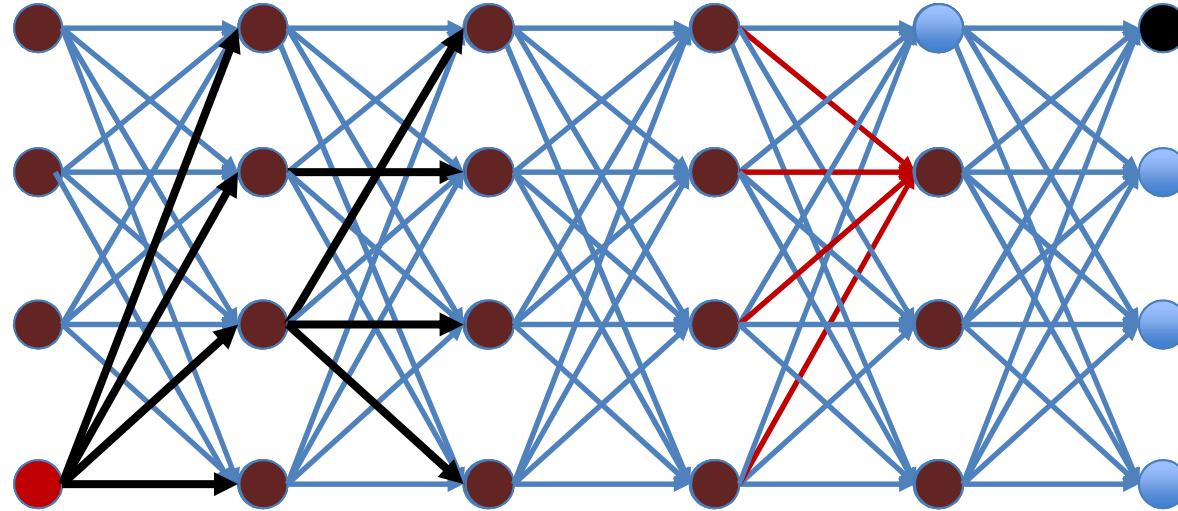
- Order of evaluation to find the shortest path between the given source and destination

Shortest path algorithm



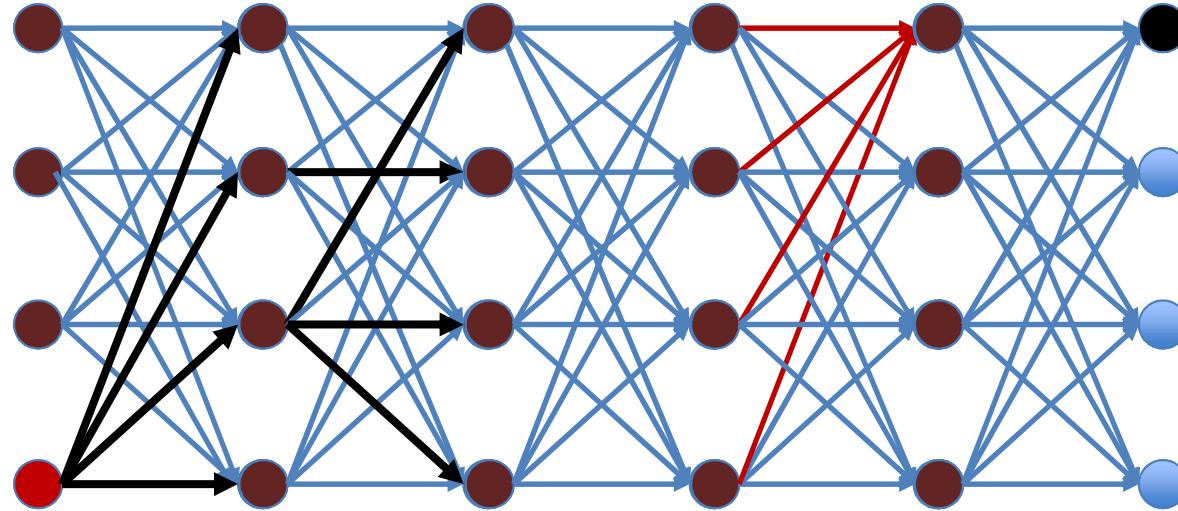
- Order of evaluation to find the shortest path between the given source and destination

Shortest path algorithm



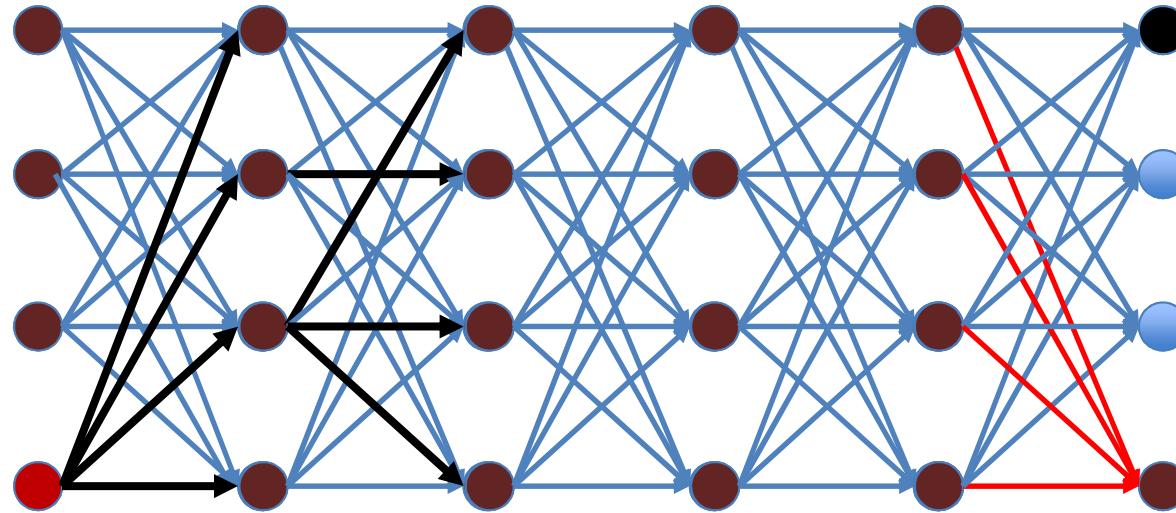
- Order of evaluation to find the shortest path between the given source and destination

Shortest path algorithm



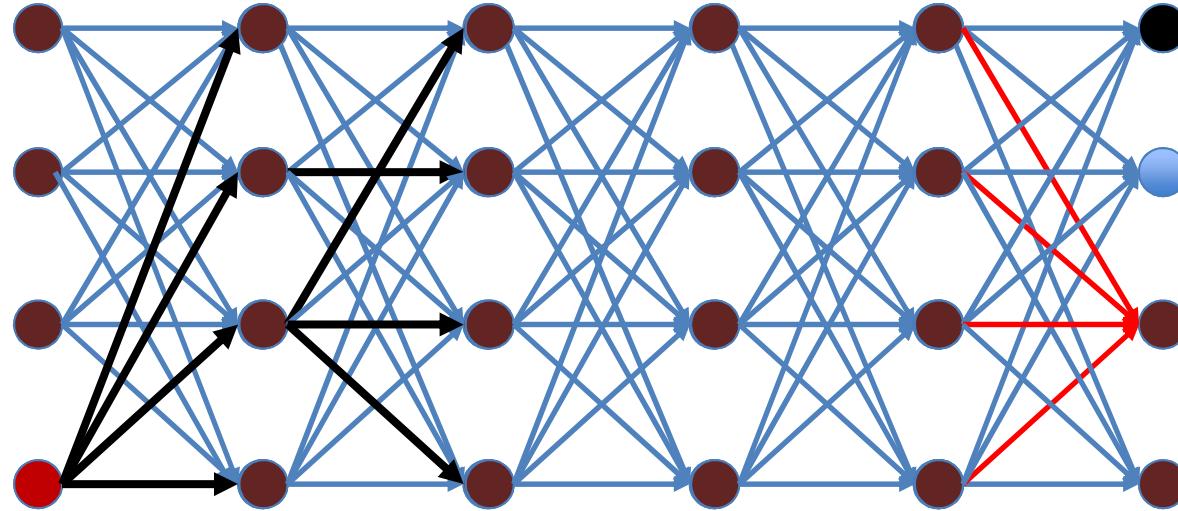
- Order of evaluation to find the shortest path between the given source and destination

Shortest path algorithm



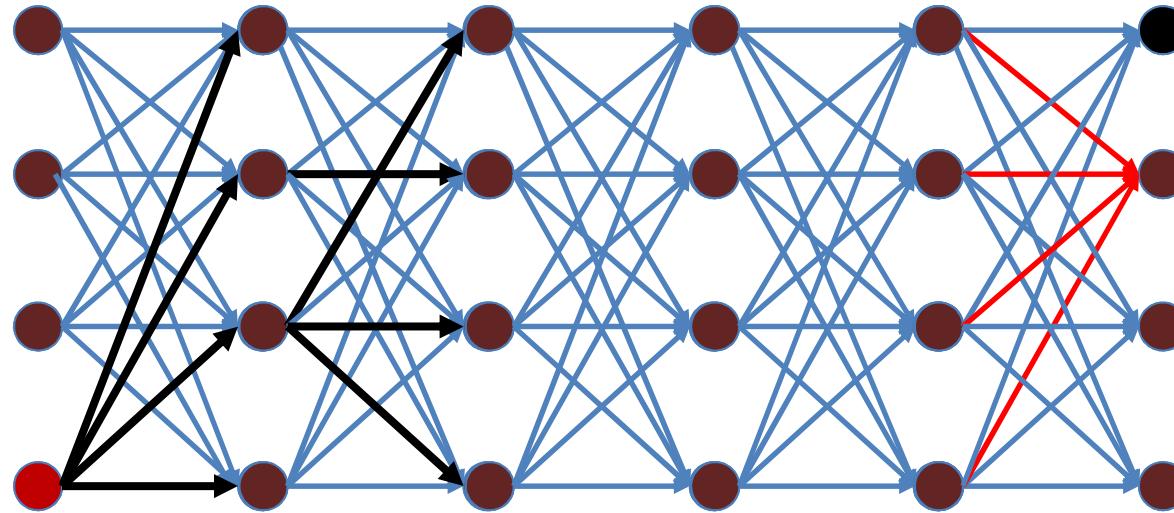
- Order of evaluation to find the shortest path between the given source and destination

Shortest path algorithm



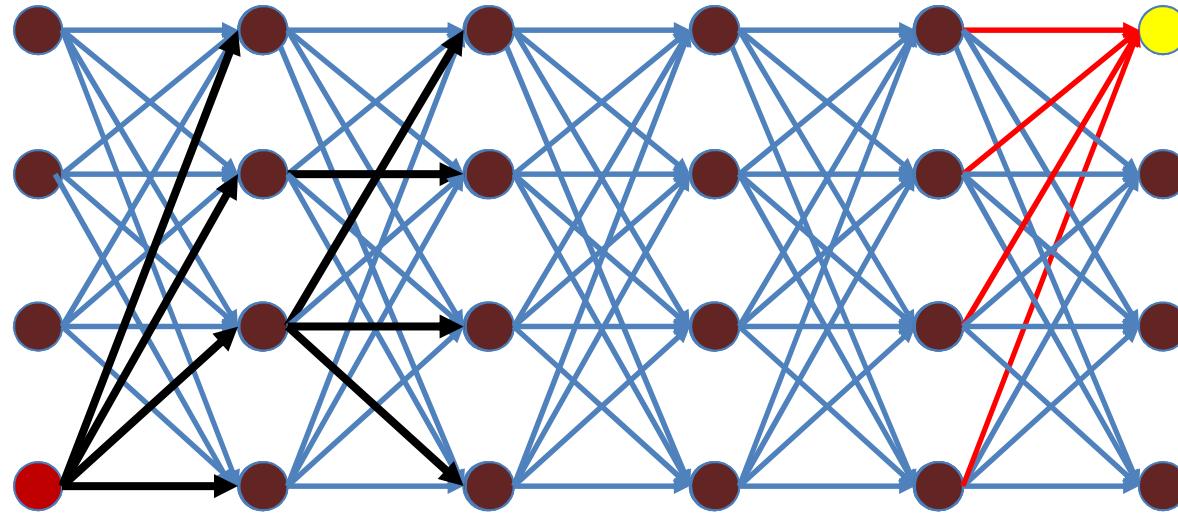
- Order of evaluation to find the shortest path between the given source and destination

Shortest path algorithm



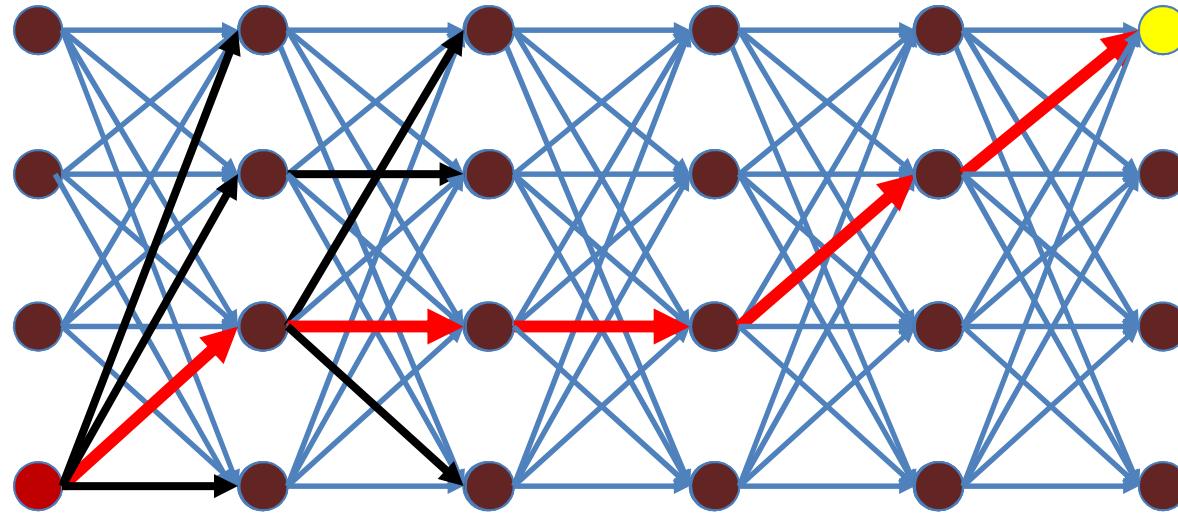
- Order of evaluation to find the shortest path between the given source and destination

Shortest path algorithm



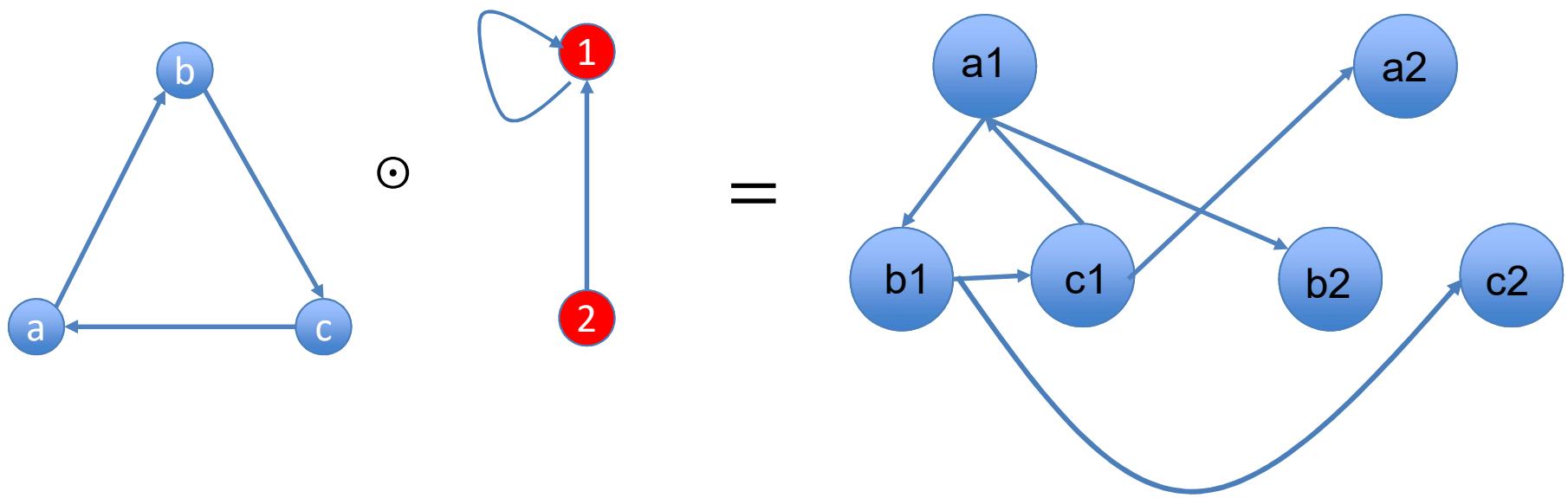
- Order of evaluation to find the shortest path between the given source and destination

Shortest path algorithm



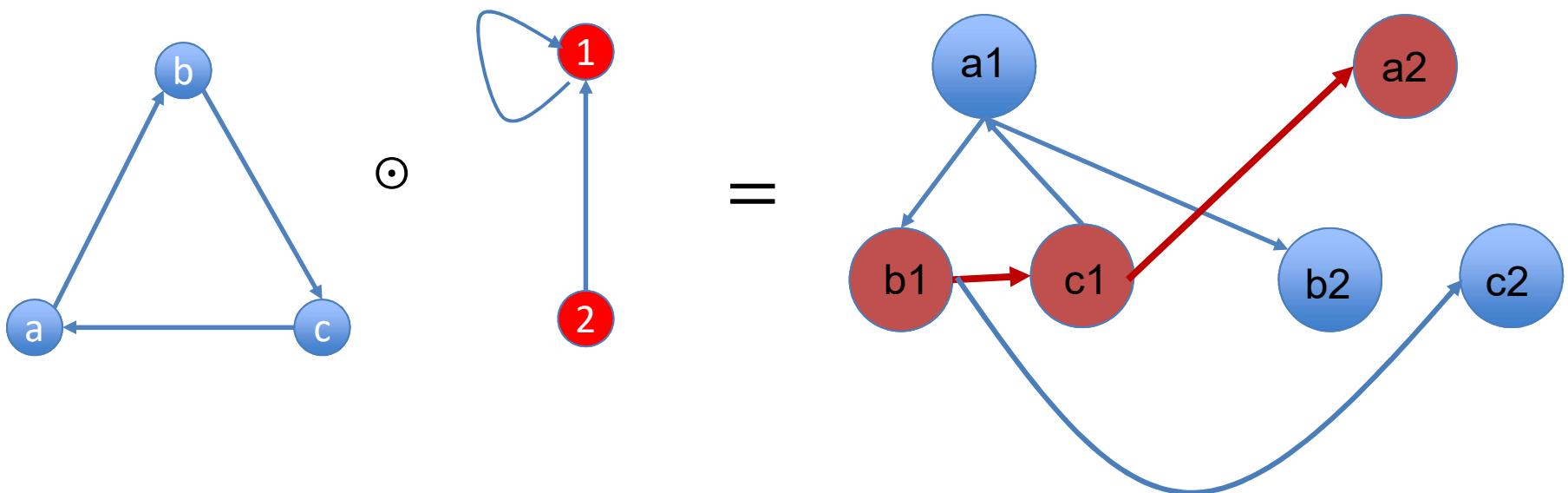
- Order of evaluation to find the shortest path between the given source and destination

Composing directed graphs



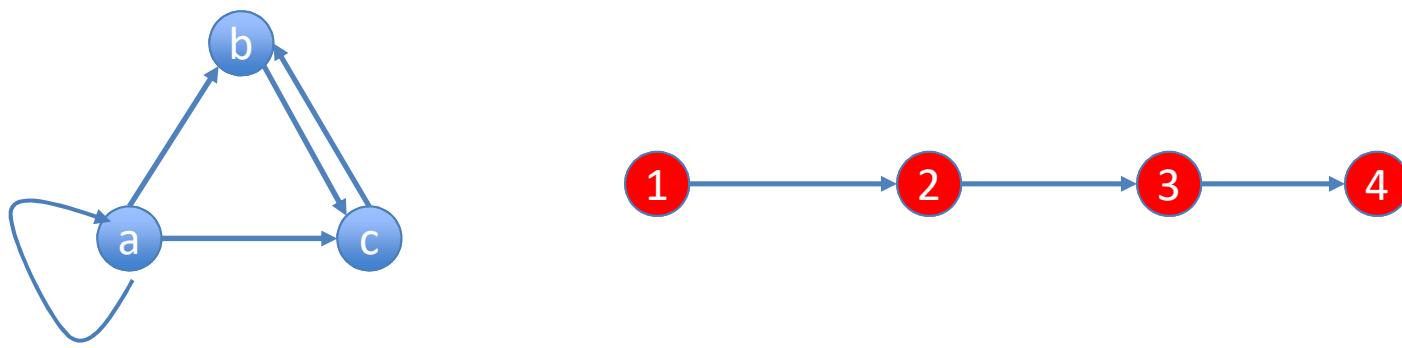
- Two “product” of two graphs
 - Each vertex on the child graph represents an unordered pair of vertices from the parent graphs
 - Two vertices on the child graph are connected by an edge if corresponding vertices in the parent graphs are also connected

Composing directed graphs



- Two “product” of two graphs
 - Each vertex on the child graph represents a pair of vertices from the parents
 - Every path through the child graph conforms to the structure of both parents

Composing graphs



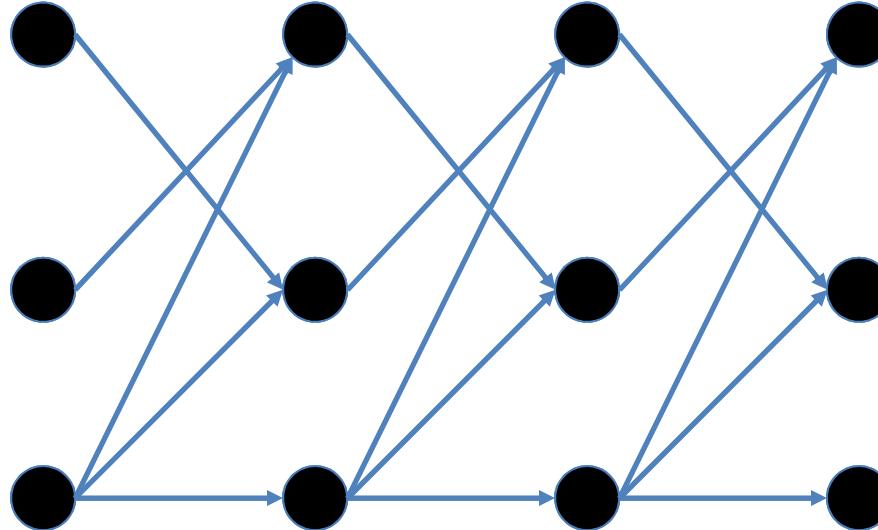
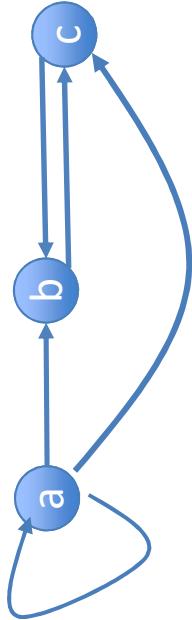
- A graph and a linear graph

Composing graphs



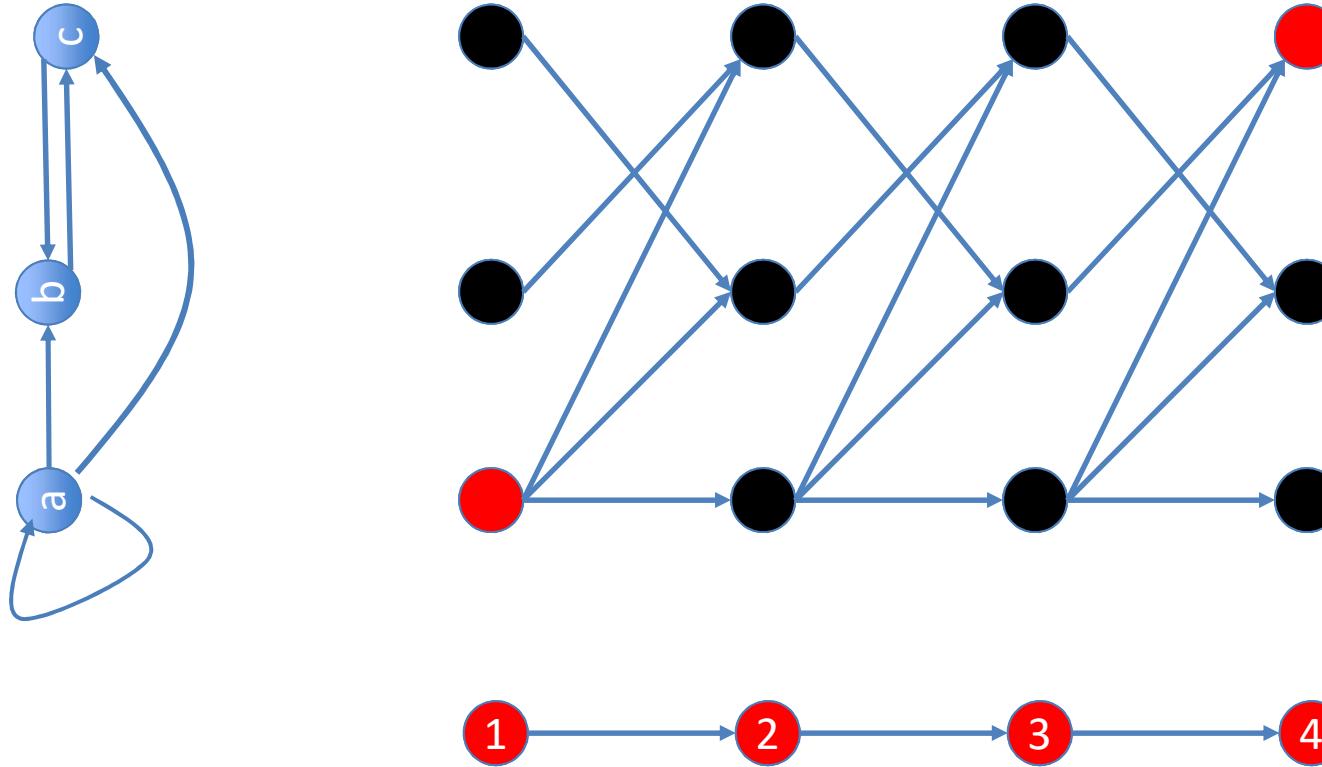
- A graph and a linear graph

The Trellis



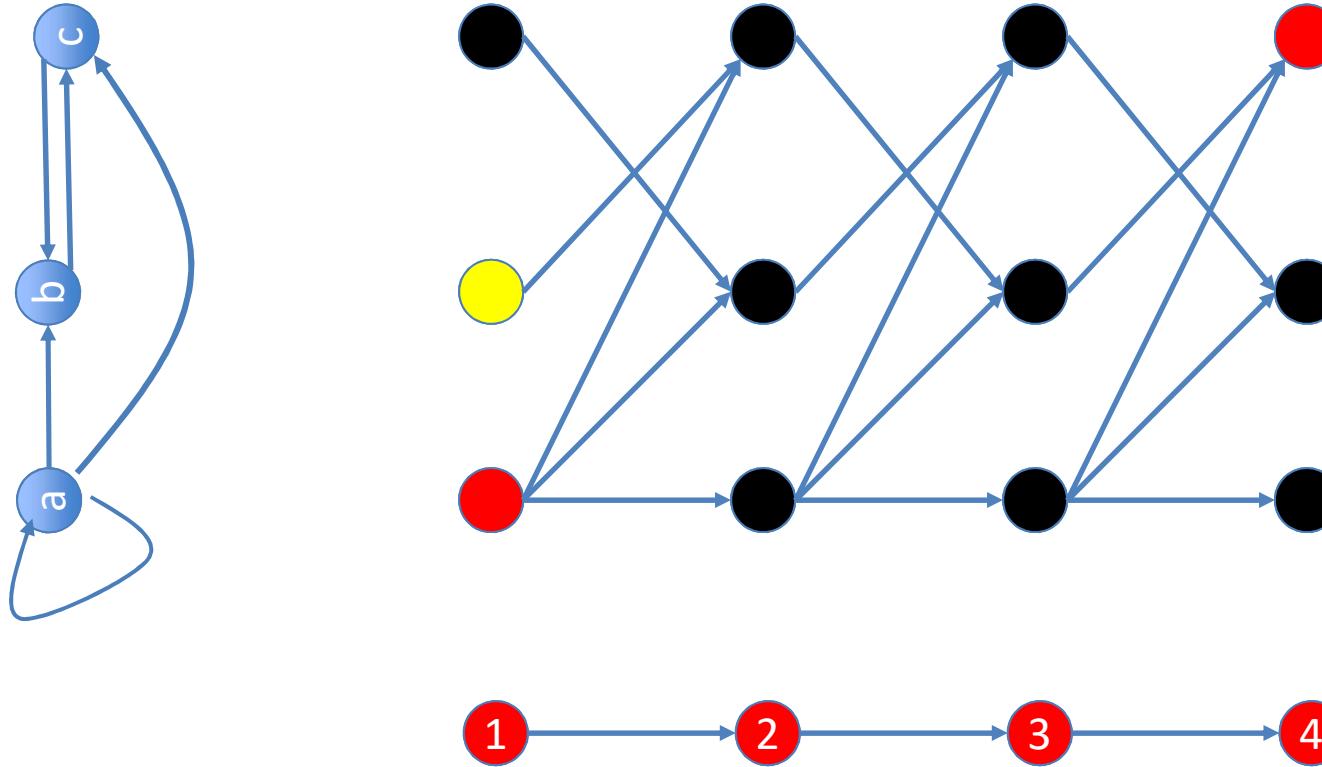
- The composition of a directed graph and a linear graph results in a *trellis*

Best Path Through The Trellis



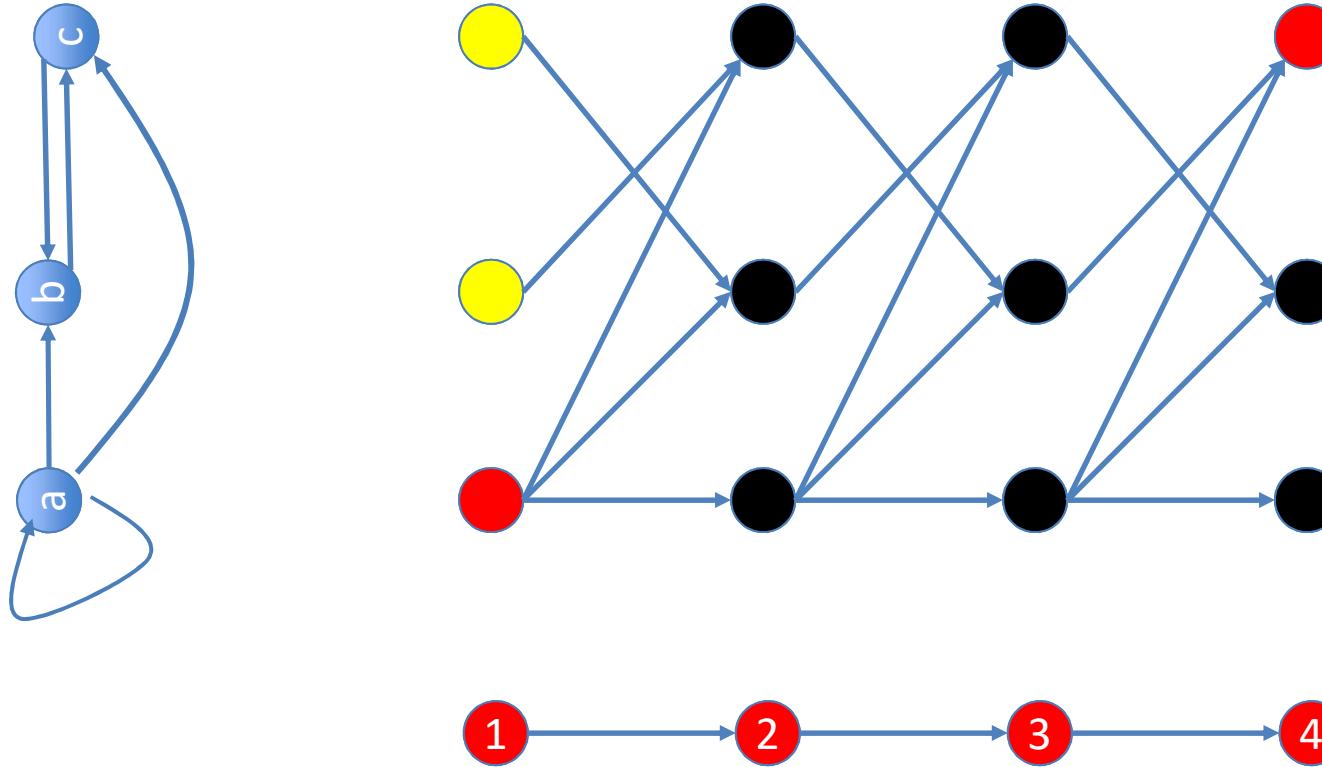
- The composition of a directed graph and a linear graph results in a *trellis*

Best Path Through The Trellis



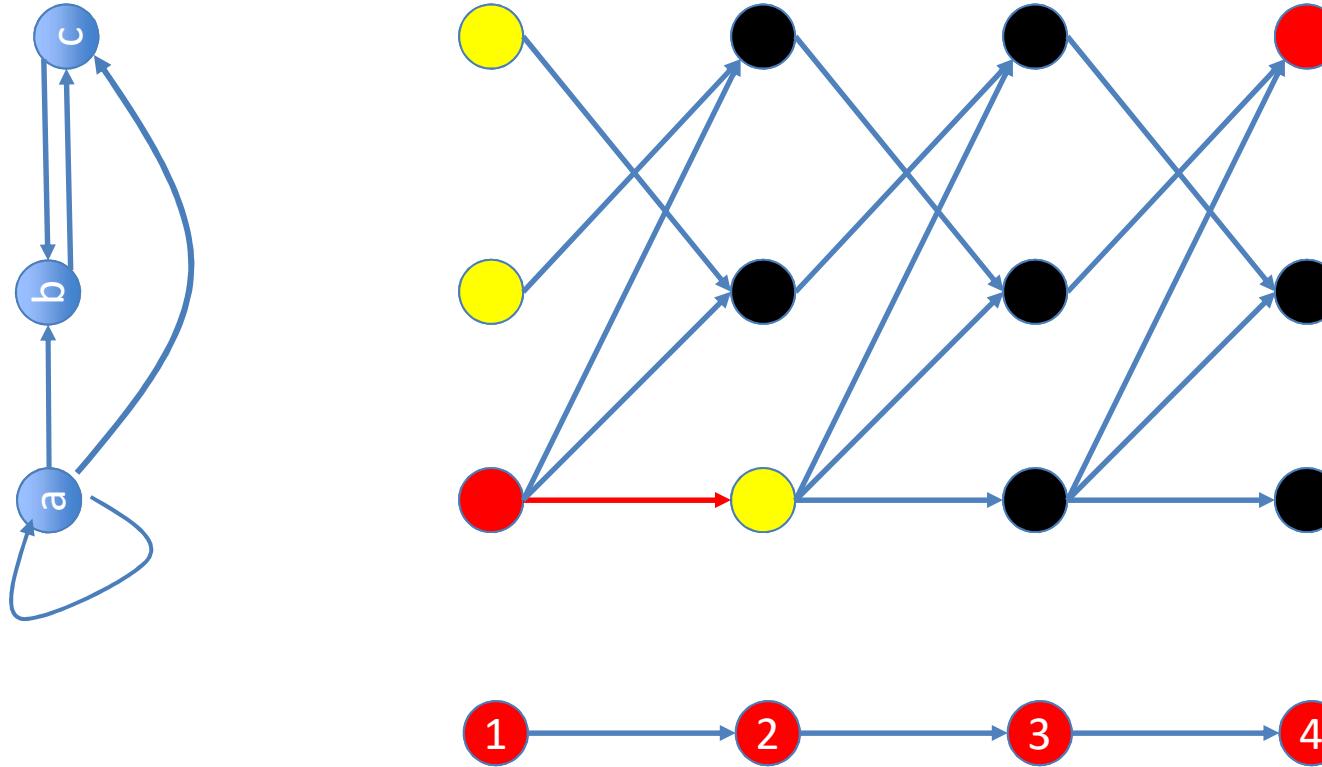
- The composition of a directed graph and a linear graph results in a *trellis*

Best Path Through The Trellis



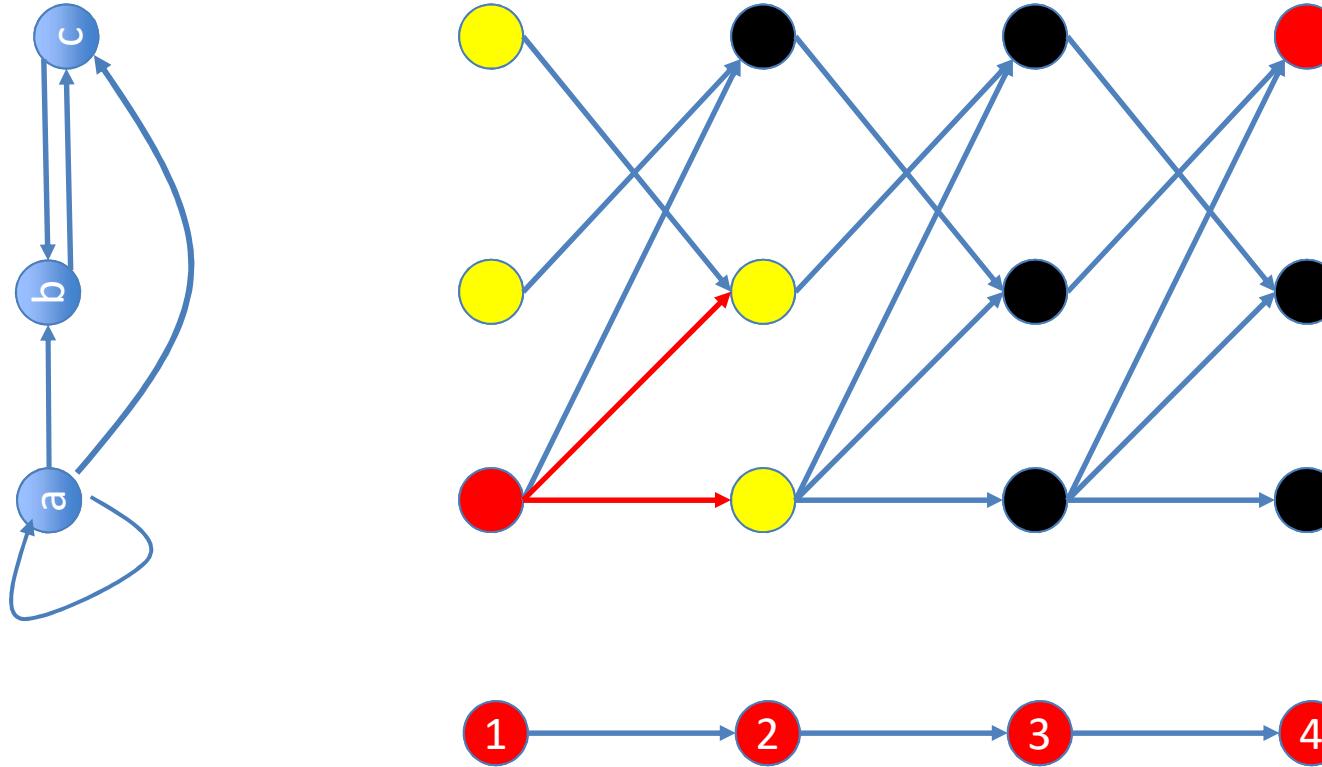
- The composition of a directed graph and a linear graph results in a *trellis*

Best Path Through The Trellis



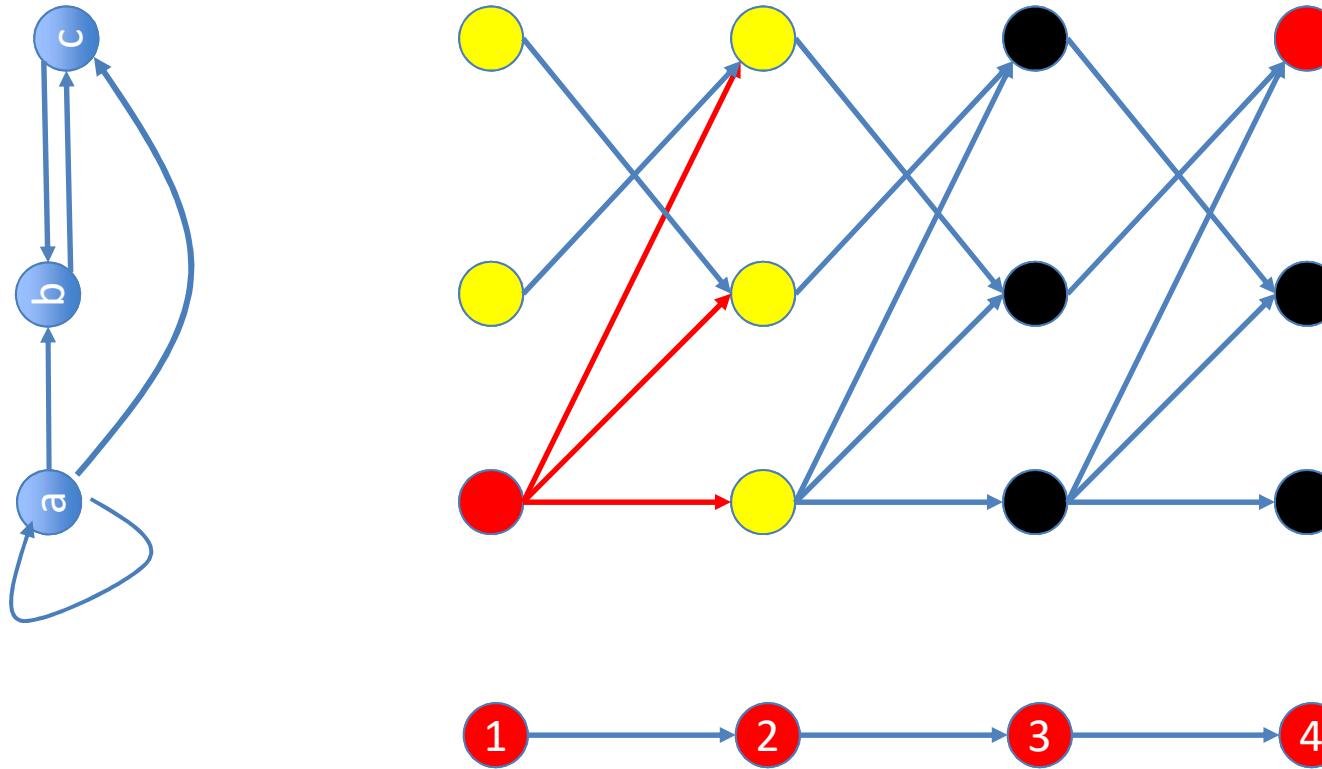
- The composition of a directed graph and a linear graph results in a *trellis*

Best Path Through The Trellis



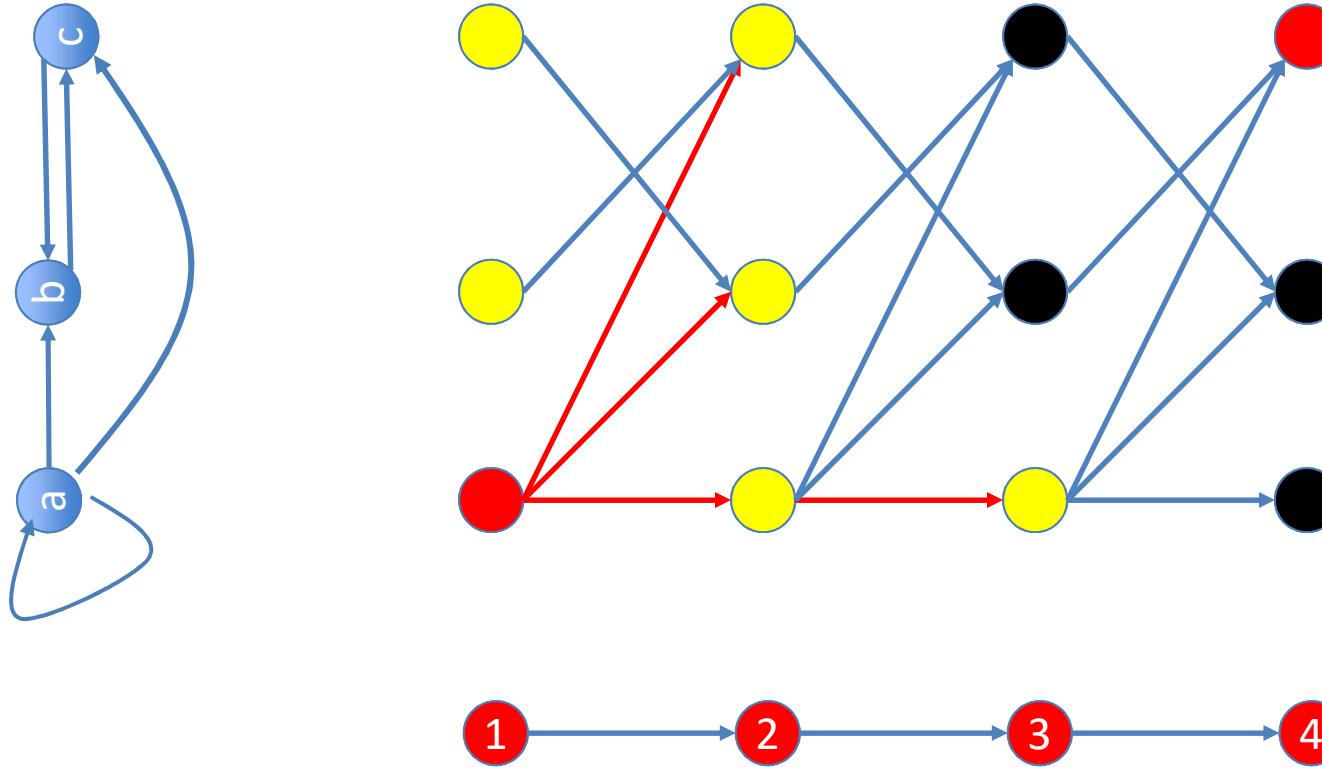
- The composition of a directed graph and a linear graph results in a *trellis*

Best Path Through The Trellis



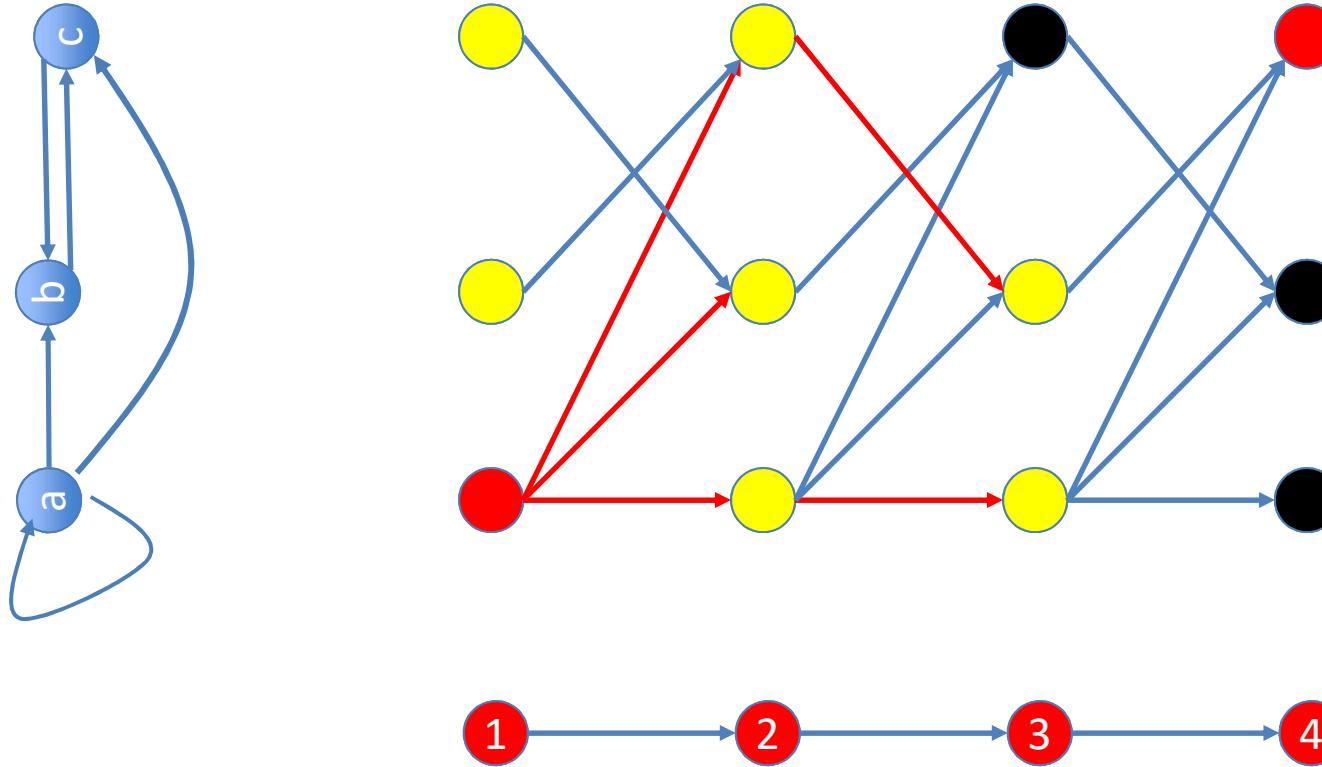
- The composition of a directed graph and a linear graph results in a *trellis*

Best Path Through The Trellis



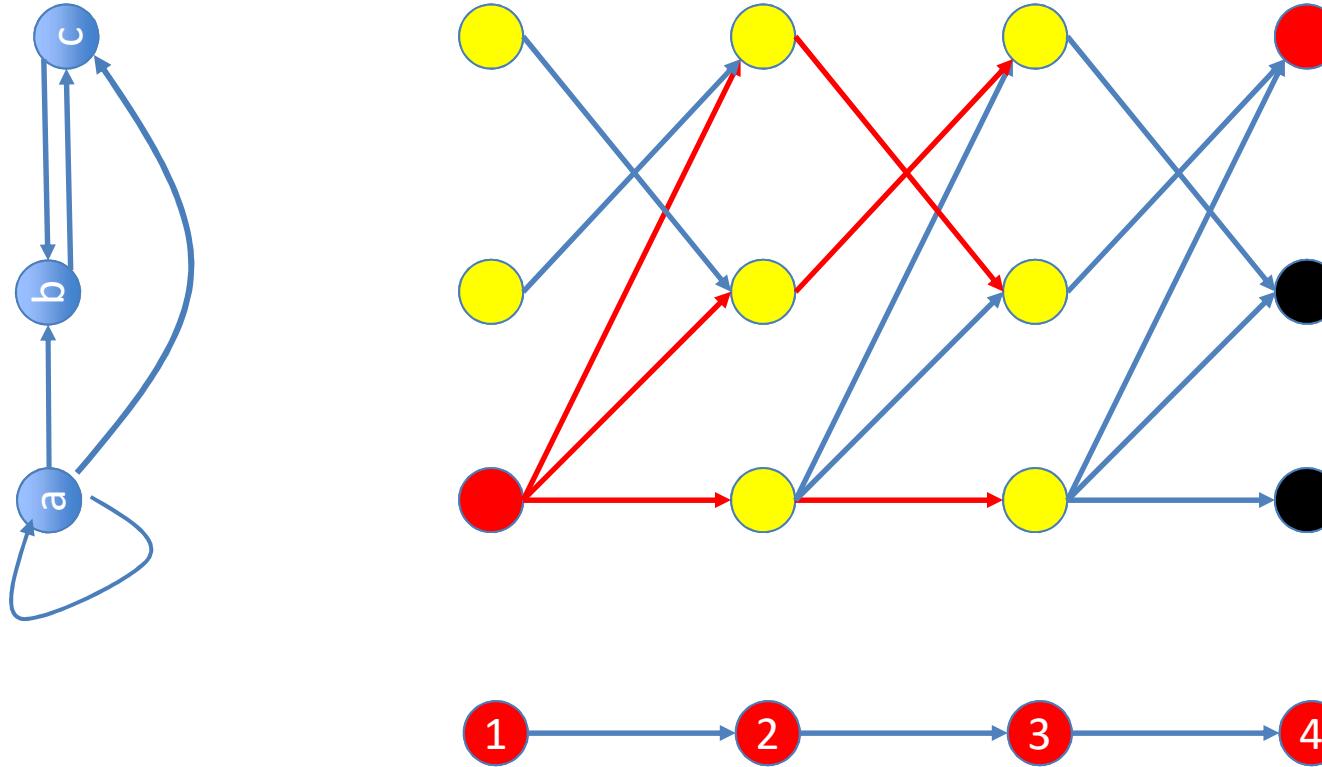
- The composition of a directed graph and a linear graph results in a *trellis*

Best Path Through The Trellis



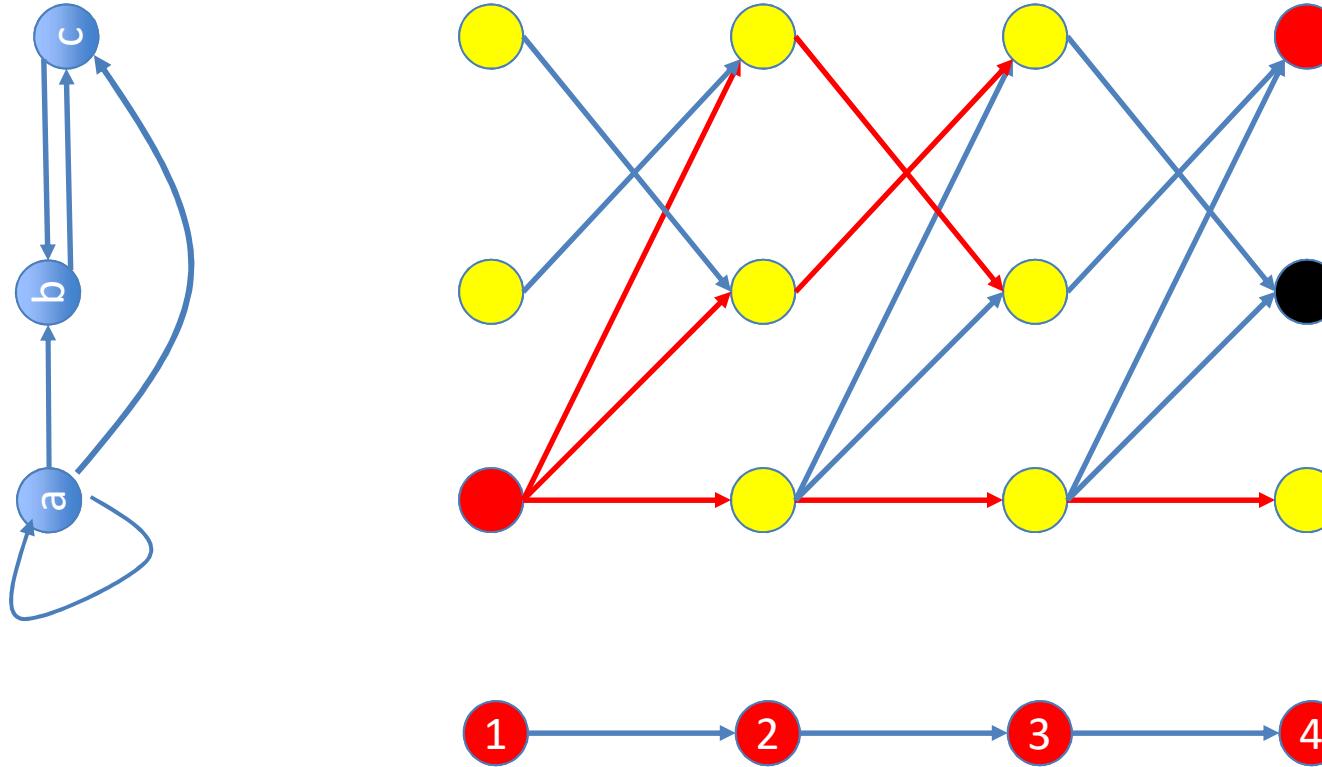
- The composition of a directed graph and a linear graph results in a *trellis*

Best Path Through The Trellis



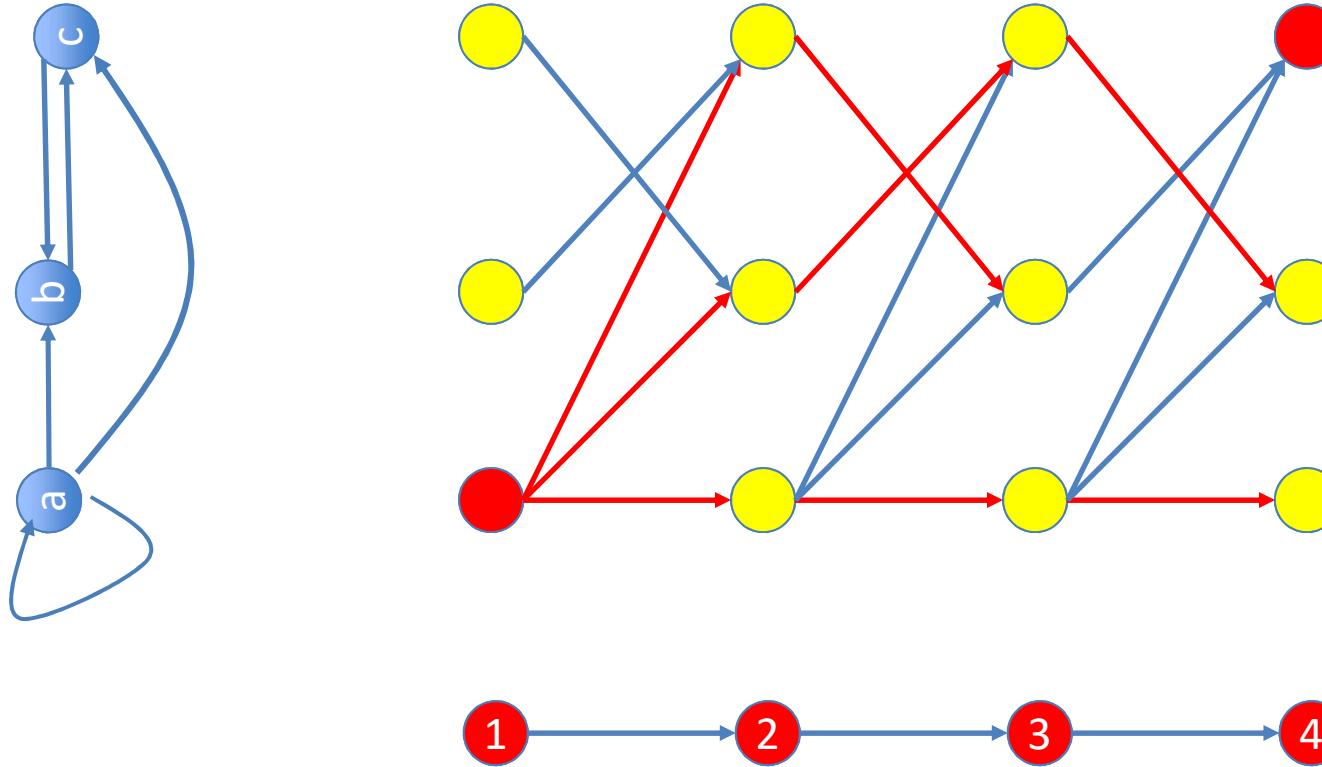
- The composition of a directed graph and a linear graph results in a *trellis*

Best Path Through The Trellis



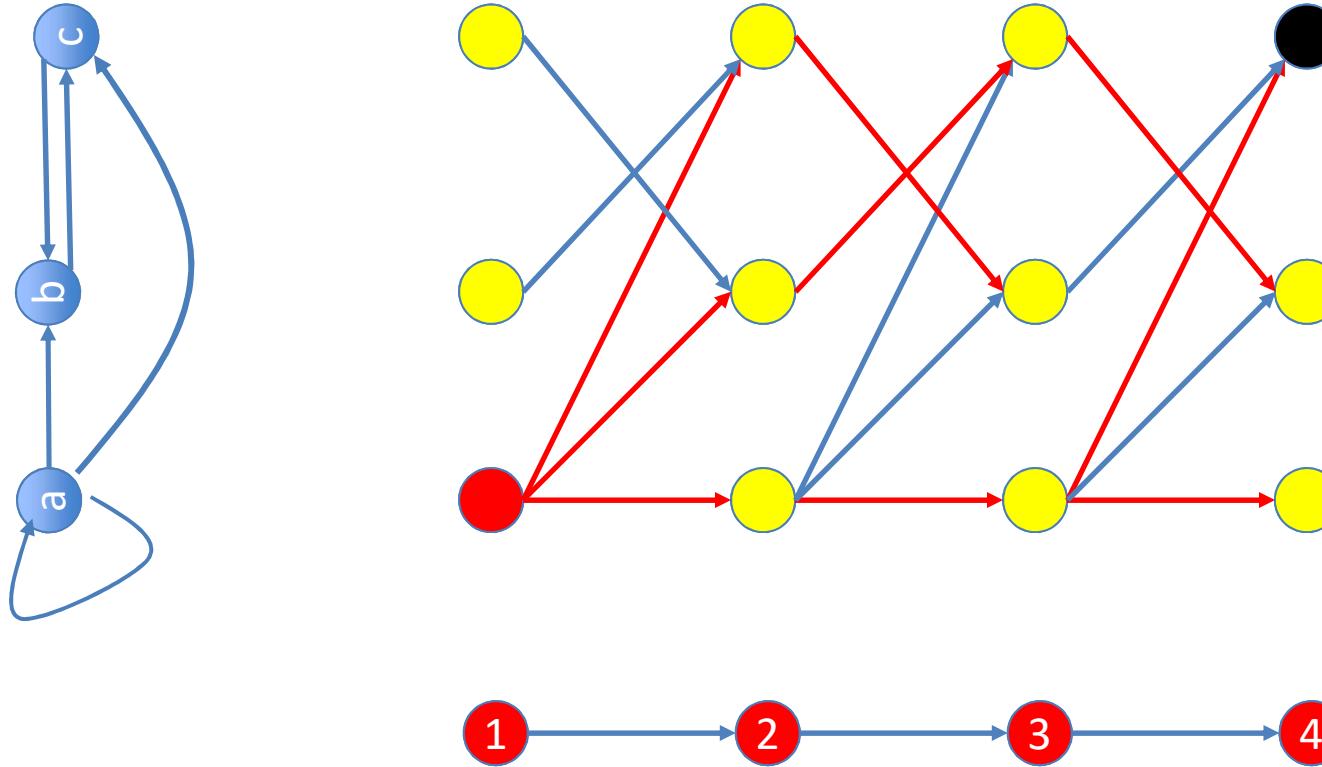
- The composition of a directed graph and a linear graph results in a *trellis*

Best Path Through The Trellis



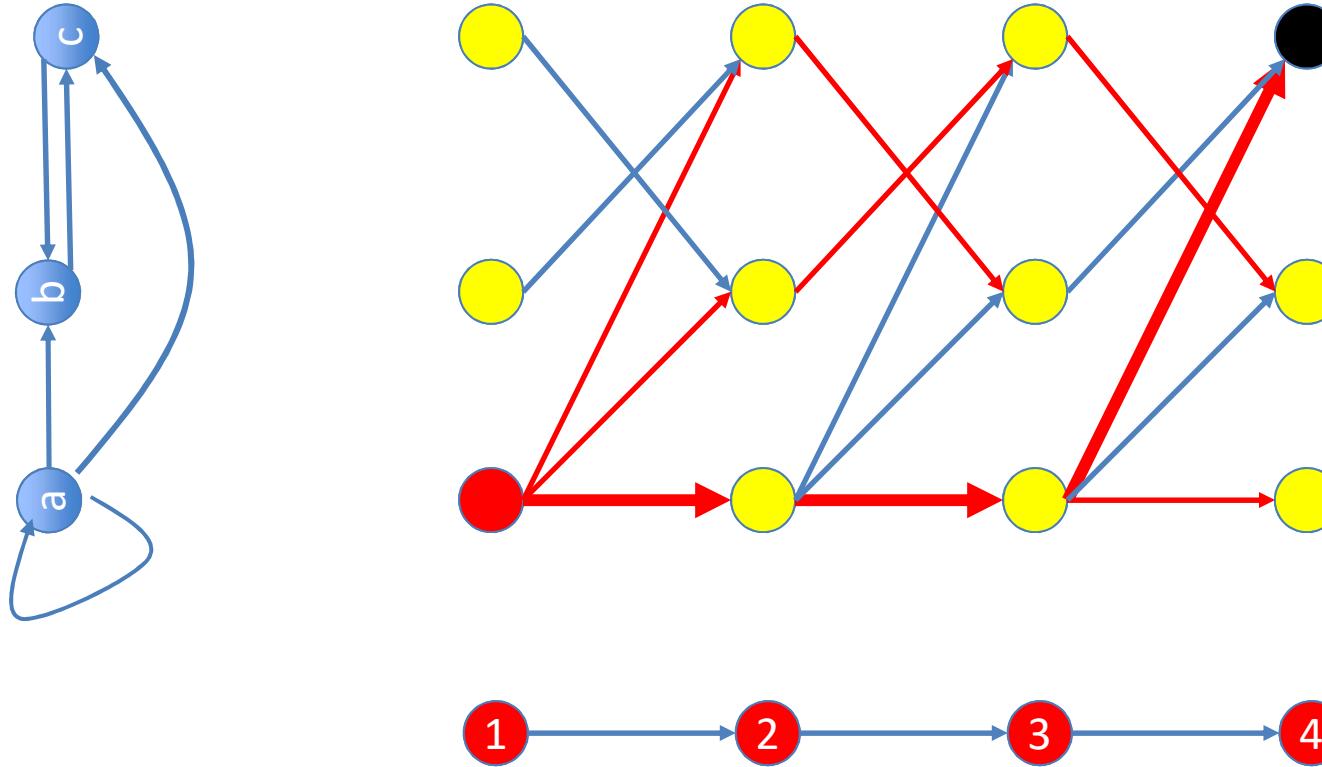
- The composition of a directed graph and a linear graph results in a *trellis*

Best Path Through The Trellis



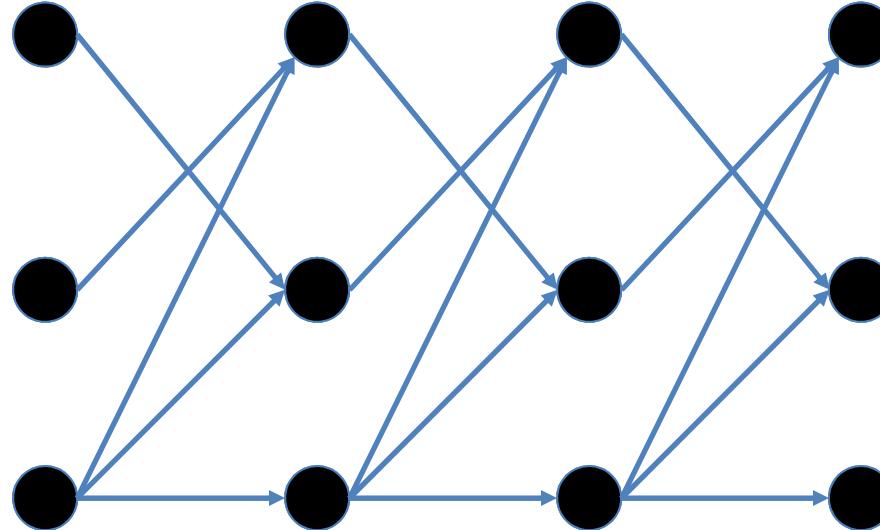
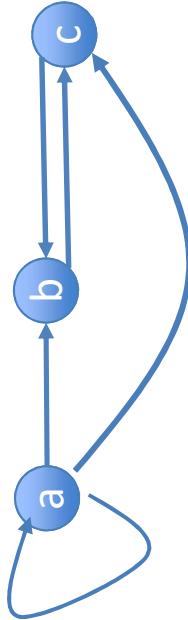
- The composition of a directed graph and a linear graph results in a *trellis*

Best Path Through The Trellis



- You just saw the *Viterbi* algorithm!!
- Again, the best path conforms to sequential constraints imposed by both parents

The Trellis

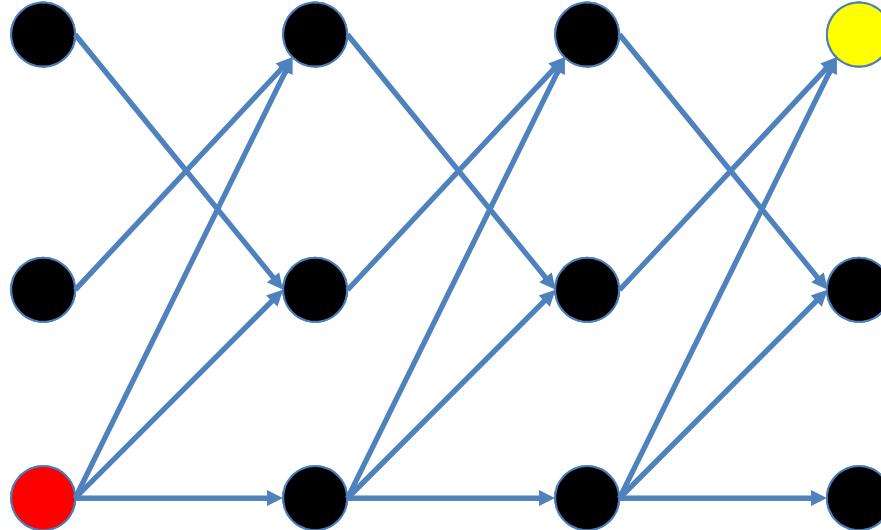
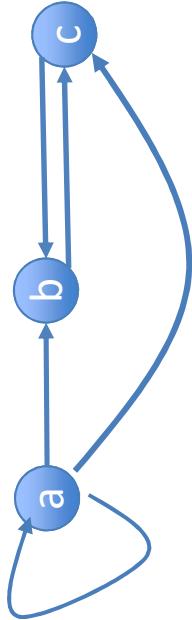


- What are the edge and node costs?
 - $\text{Cost}(\text{node}(a,3)) = f(a,3)$
 - $\text{Cost}(\text{edge}(a3 \rightarrow b=3)) = g(\text{edge}(a \rightarrow b), \text{edge}(3 \rightarrow 4))$

Viterbi algorithm

- Assuming graph 1 (vertical) has N nodes (indexed 1..N) and graph 2 (linear horizontal) has M nodes (indexed 1..M)
 - Will not specify *how* node and edge costs on trellis are derived (for later)
 - Note – referring to “indexed” rather than labeled; we will use numeric indices for both parent graphs for convenience. In the child graph, a node label i,j refers to a node composed from node number i of parent graph 1, and node number j of parent graph j
- Case 1:
 - Parent (vertical) graph 1 constraint: all paths *must* start at node 1 and terminate at node “n”
 - All paths through parent graph 2 must start at node 1 and terminate at node M

CASE 1



- Require best path from red to yellow nodes

Viterbi algorithm

- Initialize:

```
Cost[1:M, 1:N] = infinity
```

```
Bestpredecessor[1:M, 1:N] = null
```

- Algorithm:

```
Cost[1,1] = nodecost(node(1,1))
```

```
for i = 2:M
```

```
    for j = 1:N
```

```
        BP = argmin_k(Cost[i-1,k] + edgecost((i-1,k),(i,j)))
```

```
        Cost[i,j] = Cost[i-1,BP] + edgecost((i-1,BP),(i,j)) + nodecost(i,j)
```

```
        Bestpredecessor[i,j] = BP
```

- Final overall cost:

```
Finalcost = Cost[M,N]
```

- Actual sequence of states (from parent 1):

```
State[M] = N
```

```
for i = M downto 2
```

```
    State[i-1] = Bestpredecessor(i, State[i])
```

Viterbi algorithm

- Initialize:

```
Cost[1:M, 1:N] = infinity
```

```
Bestpredecessor[1:M, 1:N] = null
```

- Algorithm:

```
Cost[1,1] = nodecost(node(1,1))
```

```
for i = 2:M
```

```
    for j = 1:N
```

```
        BP = argmin_k(Cost[i-1,k] + edgecost((i-1,k),(i,j)))
```

```
        Cost[i,j] = Cost[i-1,BP] + edgecost((i-1,BP),(i,j)) + nodecost(i,j)
```

```
        Bestpredecessor[i,j] = BP
```

- Final overall cost:

```
Finalcost = Cost[M,N]
```

- Actual sequence of states (from parent 1):

```
State[M] = N
```

```
for i = M downto 2
```

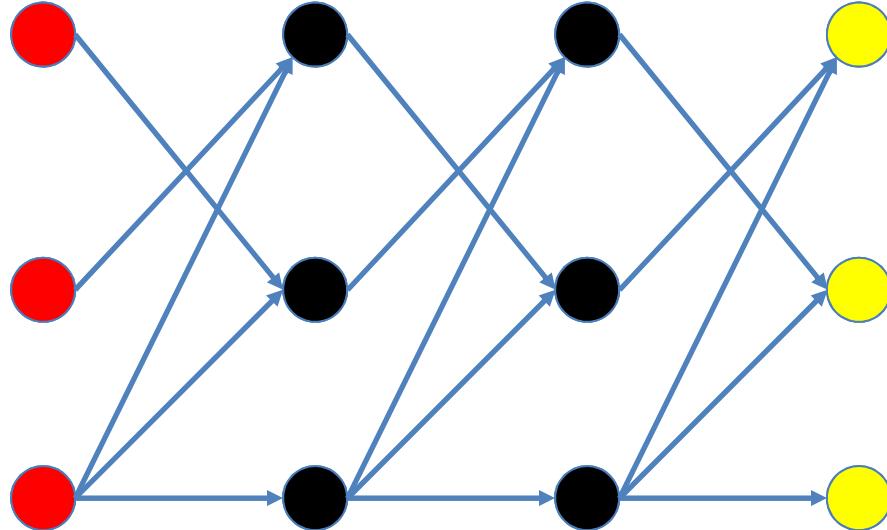
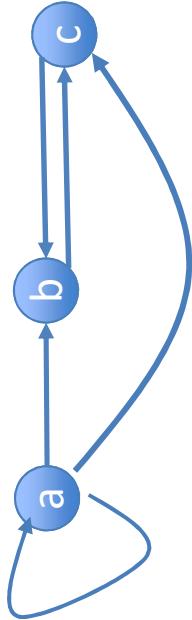
```
    State[i-1] = Bestpredecessor(i, State[i])
```

We leave how you will implement this `argmin_k` to your discretion. Sometimes, one can take advantage of the structure of graph 1 to reduce the cost of this operation significantly.

Viterbi algorithm: argmin_k

- `argmin_k(Cost[i-1,k] + edgecost((i-1,k), (i,j))):`
 `bestcost = inf`
 `BP = null`
 `for k = 1:N`
 `costk = Cost[i-1,k] + edgecost((i-1,k), (i,j))`
 `if (costk < bestcost)`
 `bestcost = costk`
 `BP = k`
 `return BP`

CASE 2



- More generic: Find best path from *any* red node to *any* yellow node
 - i.e. Of all paths from a red node to a yellow node, find the best one

Viterbi algorithm for Case 2

- Initialize:

```
Cost[1:M, 1:N] = infinity  
Bestpredecessor[1:M, 1:N] = null  
for i = 1:N Cost[1,i] = nodecost(node(1,i))
```

- Algorithm:

```
for i = 2:M  
    for j = 1:N  
        BP = argmin_k(Cost[i-1,k] + edgecost((i-1,k),(i,j)))  
        Cost[i,j] = Cost[i-1,BP] + edgecost((i-1,BP),(i,j)) + nodecost(i,j)  
        Bestpredecessor[i,j] = BP
```

- Final overall cost:

```
BestFinalNode = argmin_k(Cost[M,k])  
Bestcost = Cost[M, BestFinalNode]
```

- Actual sequence of states (from parent 1):

```
State[M] = BestFinalNode  
for i = M downto 2  
    State[i-1] = Bestpredecessor(i, State[i])
```

Lecture Outline

1. Markov models
2. Hidden Markov models
3. Viterbi algorithm

MARKOV MODELS

One View of Text

- Sequence of symbols (bytes, letters, characters, morphemes, words, ...)
 - Let Σ denote the set of symbols.
- Lots of possible sequences. (Σ^* is infinitely large.)
- Probability distributions over Σ^* ?

Trivial Distributions over Σ^*

- Give probability 0 to sequences with length greater than B ; uniform over the rest.
- Use data: with N examples, give probability $1/N$ to each observed sequence, 0 to the rest.
- What if we want *every* sequence to get some probability?
 - Need a probabilistic *model family* and algorithms for constructing the model from *data*.

A History-Based Model

$$p(\text{start}, w_1, w_2, \dots, w_n, \text{stop}) = \prod_{i=1}^{n+1} \gamma(w_i \mid w_1, w_2, \dots, w_{i-1})$$

- Generate each word from left to right, conditioned on what came before it.

A History-Based Model

$$p(\text{start}, w_1, w_2, \dots, w_n, \text{stop}) = \prod_{i=1}^{n+1} \gamma(w_i | w_1, w_2, \dots, w_{i-1})$$

- Generate each word from left to right, conditioned on what came before it.

Note these guys..

Die / Dice



one die



two dice

start

one die per history:





one die per history:



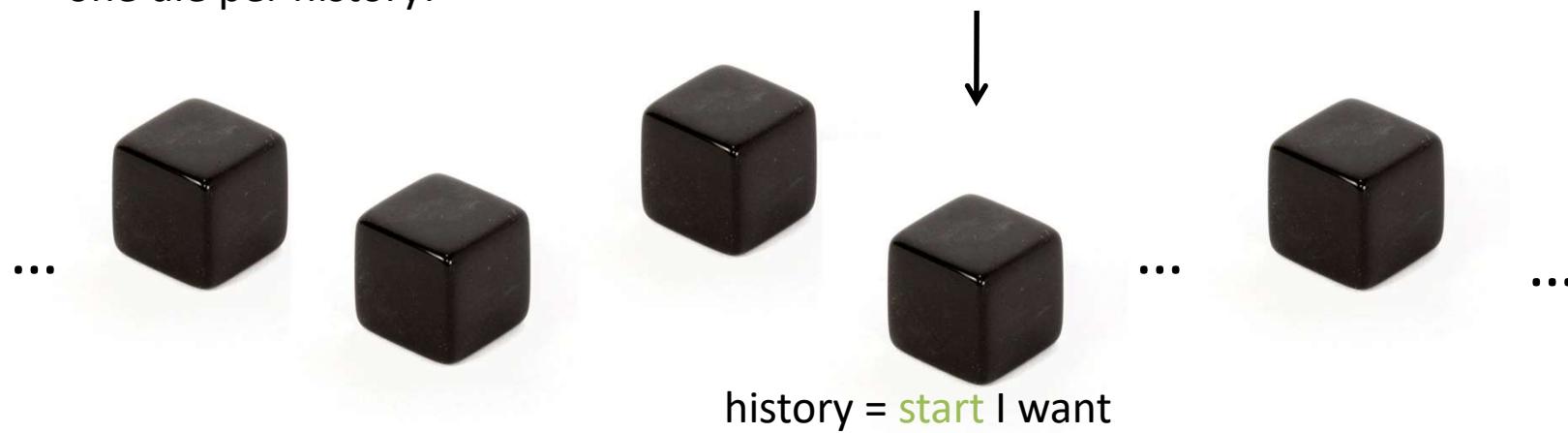
`start | want`

one die per history:





one die per history:



start I want a flight

one die per history:



start I want a flight to



one die per history:



...

...

...

history = start I want a flight

start I want a flight to Lisbon

one die per history:



start I want a flight to Lisbon .

one die per history:



history = **start** I want a flight to Lisbon

start I want a flight to Lisbon . stop



one die per history:



history = start I want a flight to Lisbon .

Is this process Markov?

start I want a flight to Lisbon . stop



one die per history:



A History-Based Model

$$p(\text{start}, w_1, w_2, \dots, w_n, \text{stop}) = \prod_{i=1}^{n+1} \gamma(w_i \mid w_1, w_2, \dots, w_{i-1})$$

- Generate each word from left to right, conditioned on what came before it.
- Very rich representational power!
- How many parameters?
- What is the probability of a sentence not seen in training data?

A Bag of Words Model

$$p(\text{start}, w_1, w_2, \dots, w_n, \text{stop}) = \prod_{i=1}^{n+1} \gamma(w_i)$$

- Every word is independent of every other word.

start

one die:



start |

one die:



start I want

one die:



start I want a

one die:



start I want a flight

one die:



start I want a flight to

one die:



start I want a flight to Lisbon

one die:



start I want a flight to Lisbon .

one die:



start I want a flight to Lisbon . stop

one die:



A Bag of Words Model

$$p(\text{start}, w_1, w_2, \dots, w_n, \text{stop}) = \prod_{i=1}^{n+1} \gamma(w_i)$$

- Every word is independent of every other word.
- Strong assumptions mean this model cannot fit the data very closely.
- How many parameters?
- What is the probability of a sentence not seen in training data?

A Bag of Words Model

$$p(\text{start}, w_1, w_2, \dots, w_n, \text{stop}) = \prod_{i=1}^{n+1} \gamma(w_i)$$

- Every word is independent of every other word.
- Strong assumptions mean this model cannot fit the data very closely.
- How many parameters?
- What is the probability of a sentence not seen in training data?

Is this a Markov Model?

A Bag of Words Model

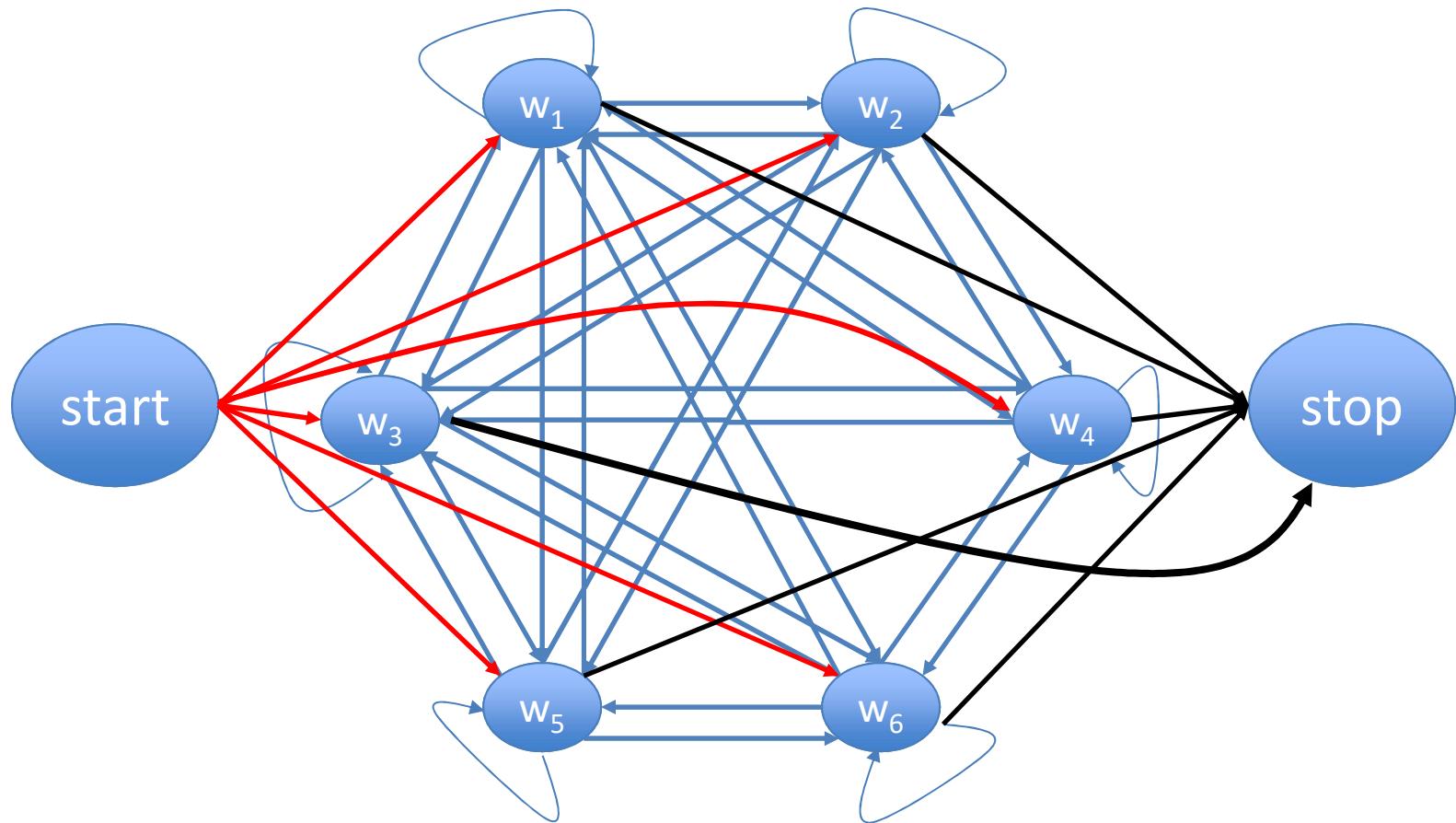
$$p(\text{start}, w_1, w_2, \dots, w_n, \text{stop}) = \prod_{i=1}^{n+1} \gamma(w_i)$$

- Every word is independent of every other word.
- Strong assumptions mean this model cannot fit the data very closely.
- How many parameters?
- What is the probability of a sentence not seen in training data?

Is this a Markov Model?
What is the information state?

Is there an absorbing state?

The Markov Model



- All incoming edges to a word w_i carry the same probability $P(w_i)$

First Order Markov Model

- Happy medium?

$$p(\text{start}, w_1, w_2, \dots, w_n, \text{stop}) = \prod_{i=1}^{n+1} \gamma(w_i \mid w_{i-1})$$

- Condition on the most recent symbol in history.

start

one die per history:



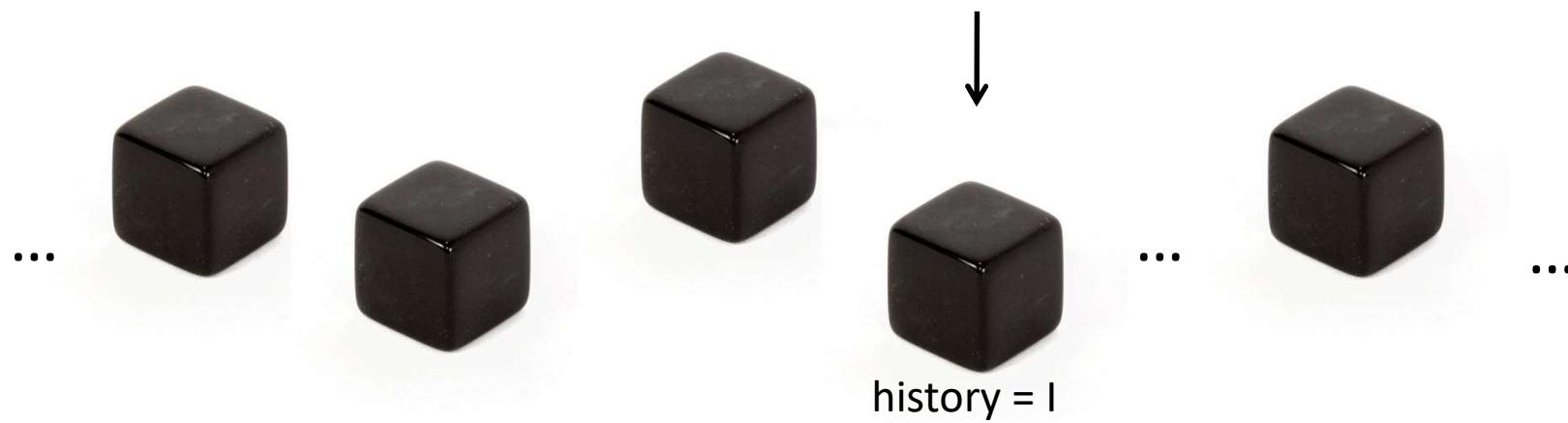


one die per history:



start  want

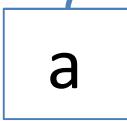
one die per history:



start | want a

one die per history:



start I want  flight

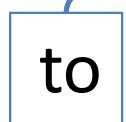
one die per history:



start I want a **flight** to

one die per history:



start I want a flight  Lisbon

one die per history:



start I want a flight to Lisbon .

one die per history:



3



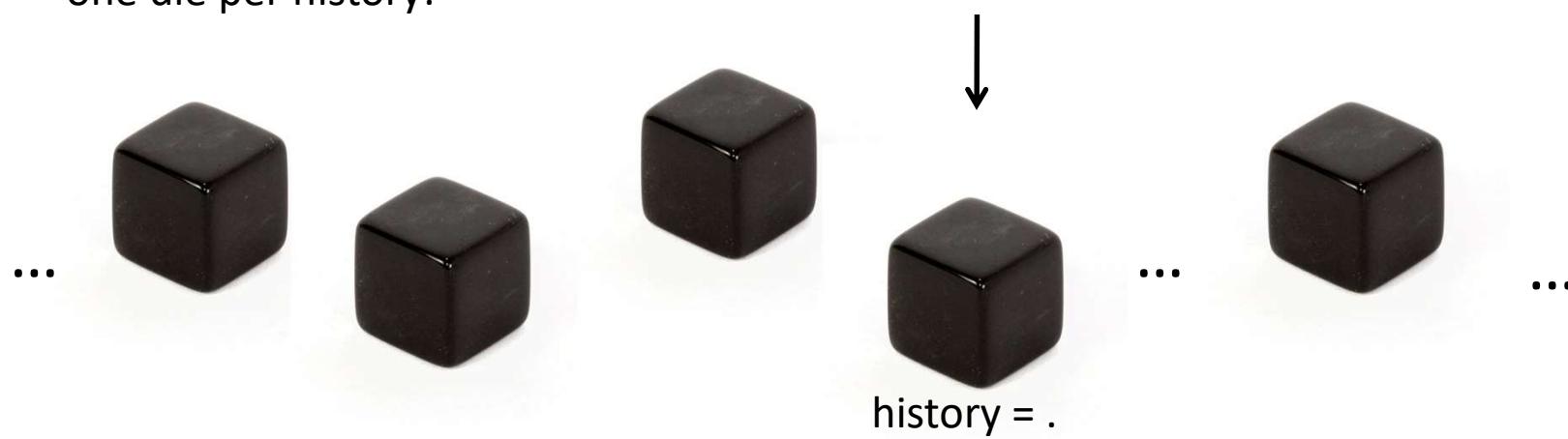
2



history = Lisbon

start I want a flight to Lisbon . stop

one die per history:



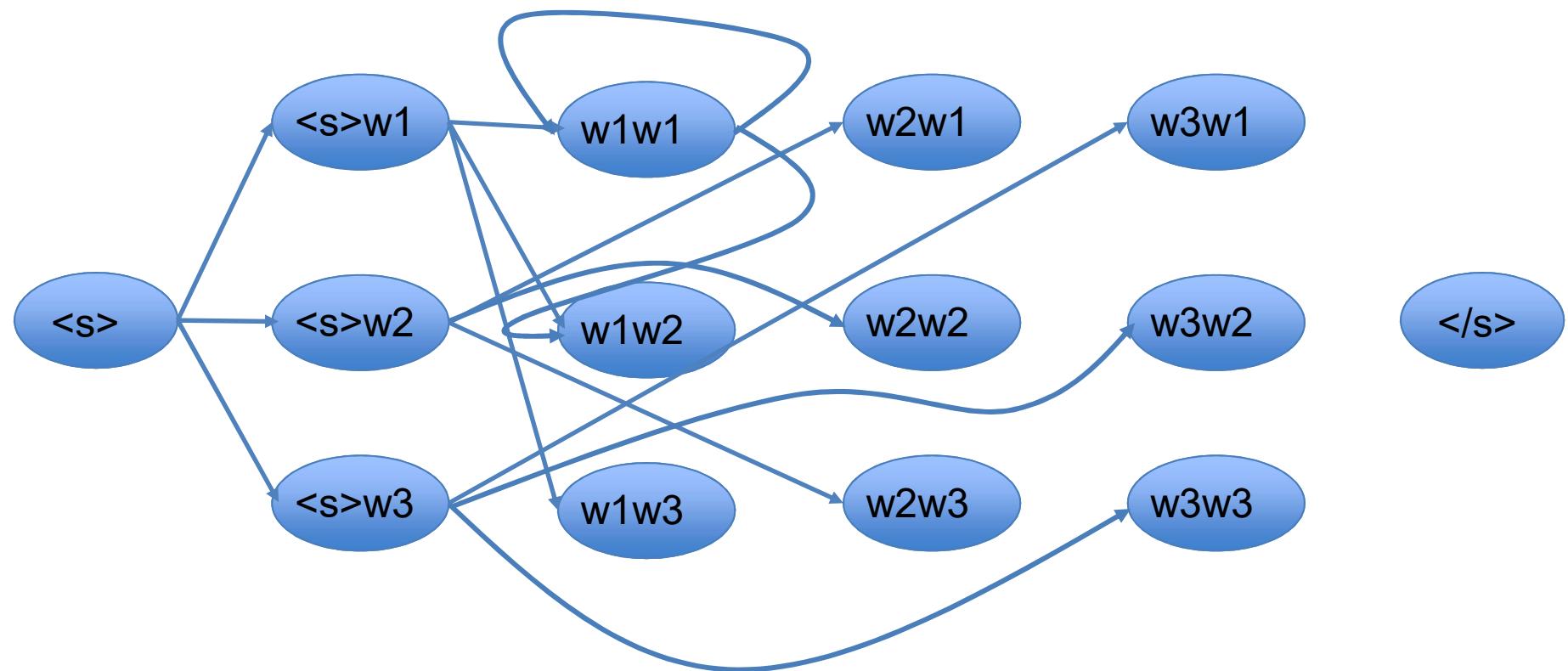
First Order Markov Model

- Happy medium?

$$p(\text{start}, w_1, w_2, \dots, w_n, \text{stop}) = \prod_{i=1}^{n+1} \gamma(w_i \mid w_{i-1})$$

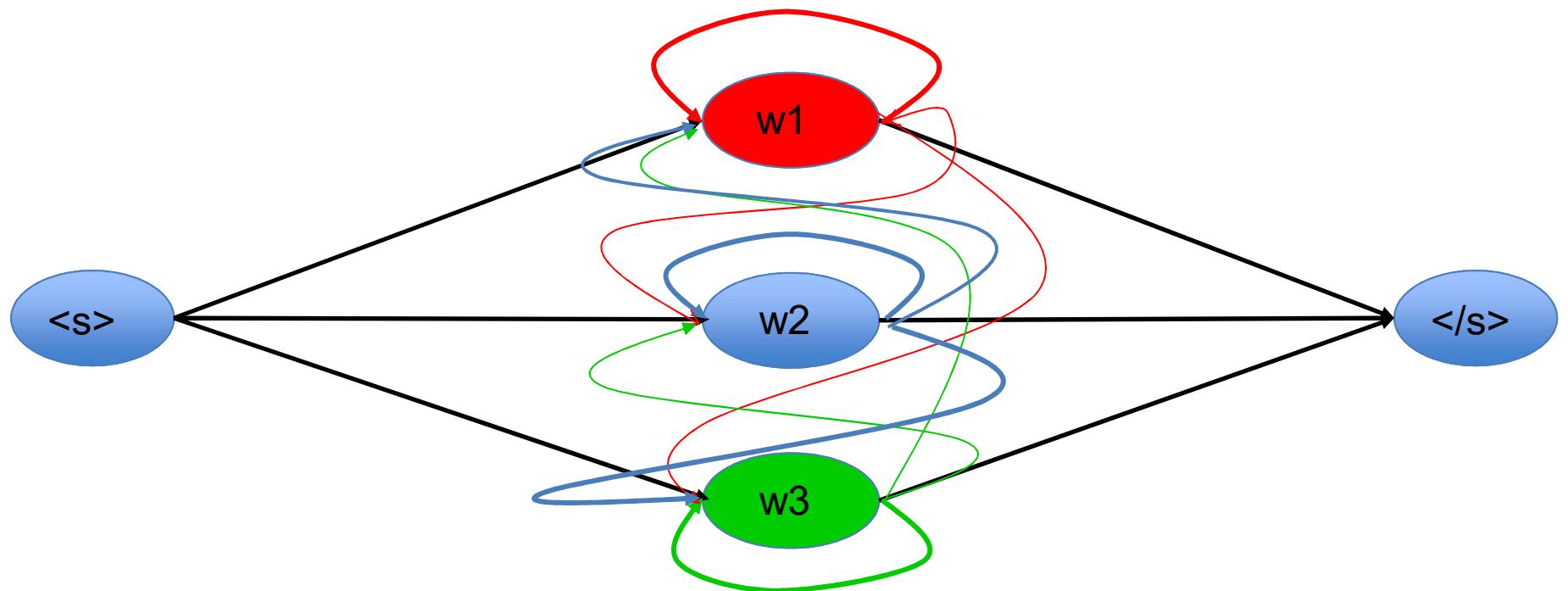
- Condition on the most recent symbol in history.
- Independence assumptions?
- Number of parameters?
- Sentences not seen in training?

Markov Model



- Incomplete figure, but lets simplify it

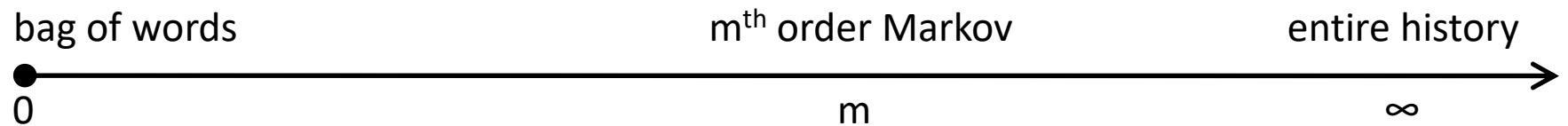
Markov Model



- What are the outgoing probabilities on the edges?

mth Order Markov Models

$$p(\text{start}, w_1, w_2, \dots, w_n, \text{stop}) = \prod_{i=1}^{n+1} \gamma(w_i \mid w_{i-m}, \dots, w_{i-1})$$

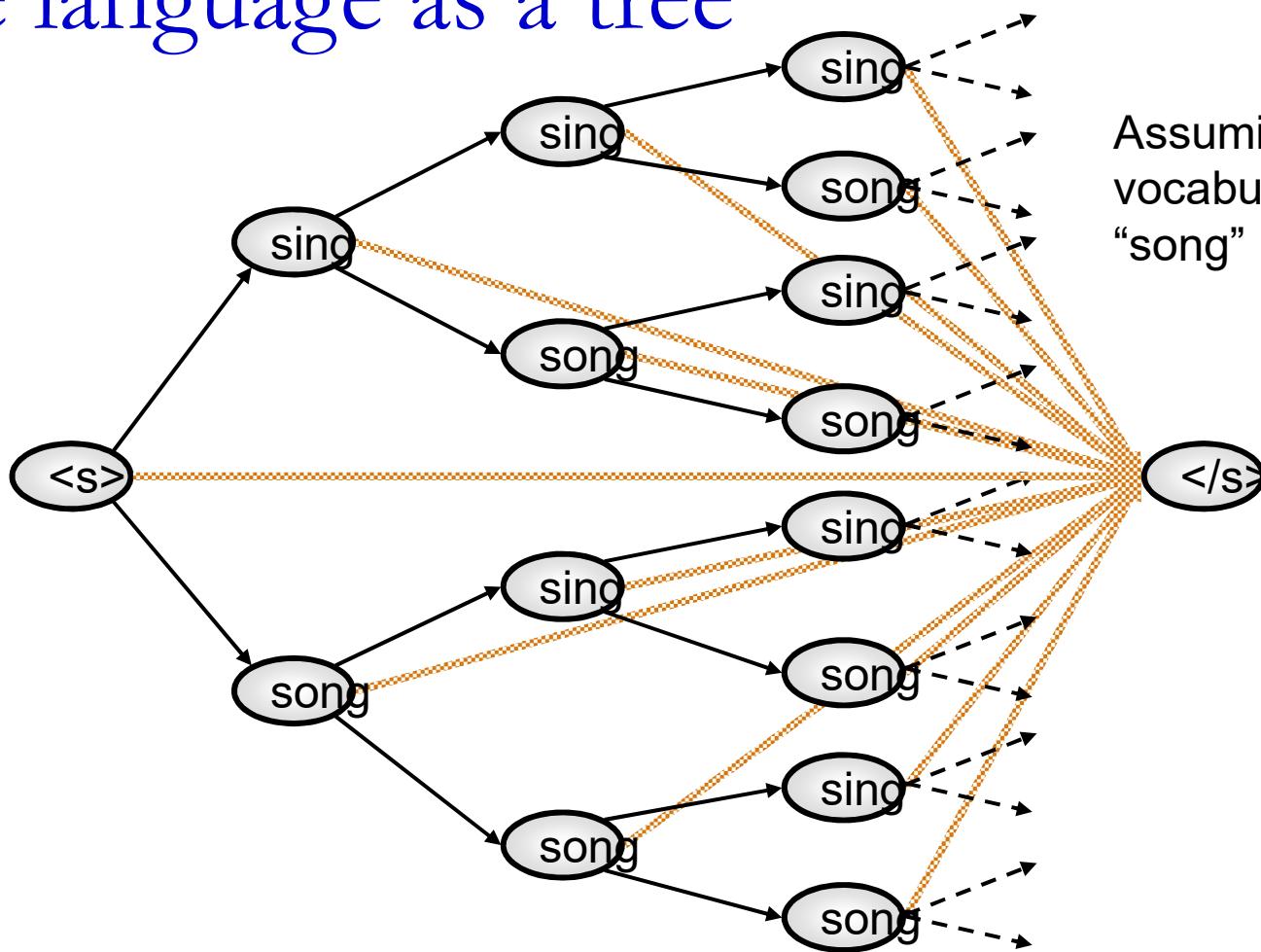


fewer parameters

stronger independence assumptions

richer expressive power

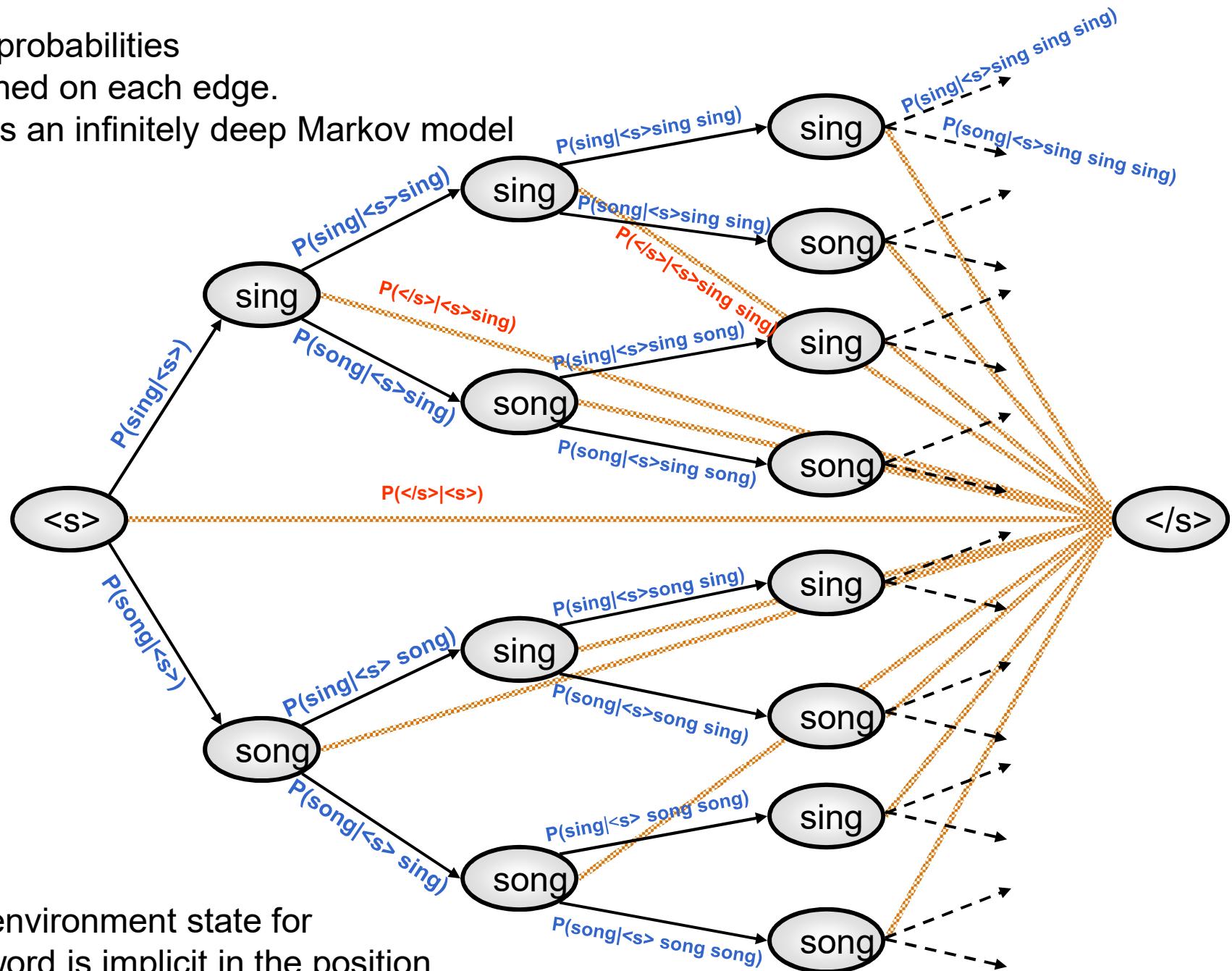
The language as a tree



Assuming a two-word vocabulary: “sing” and “song”

- *The complete tree of sentences for a two-word language*

With probabilities
attached on each edge.
This is an infinitely deep Markov model

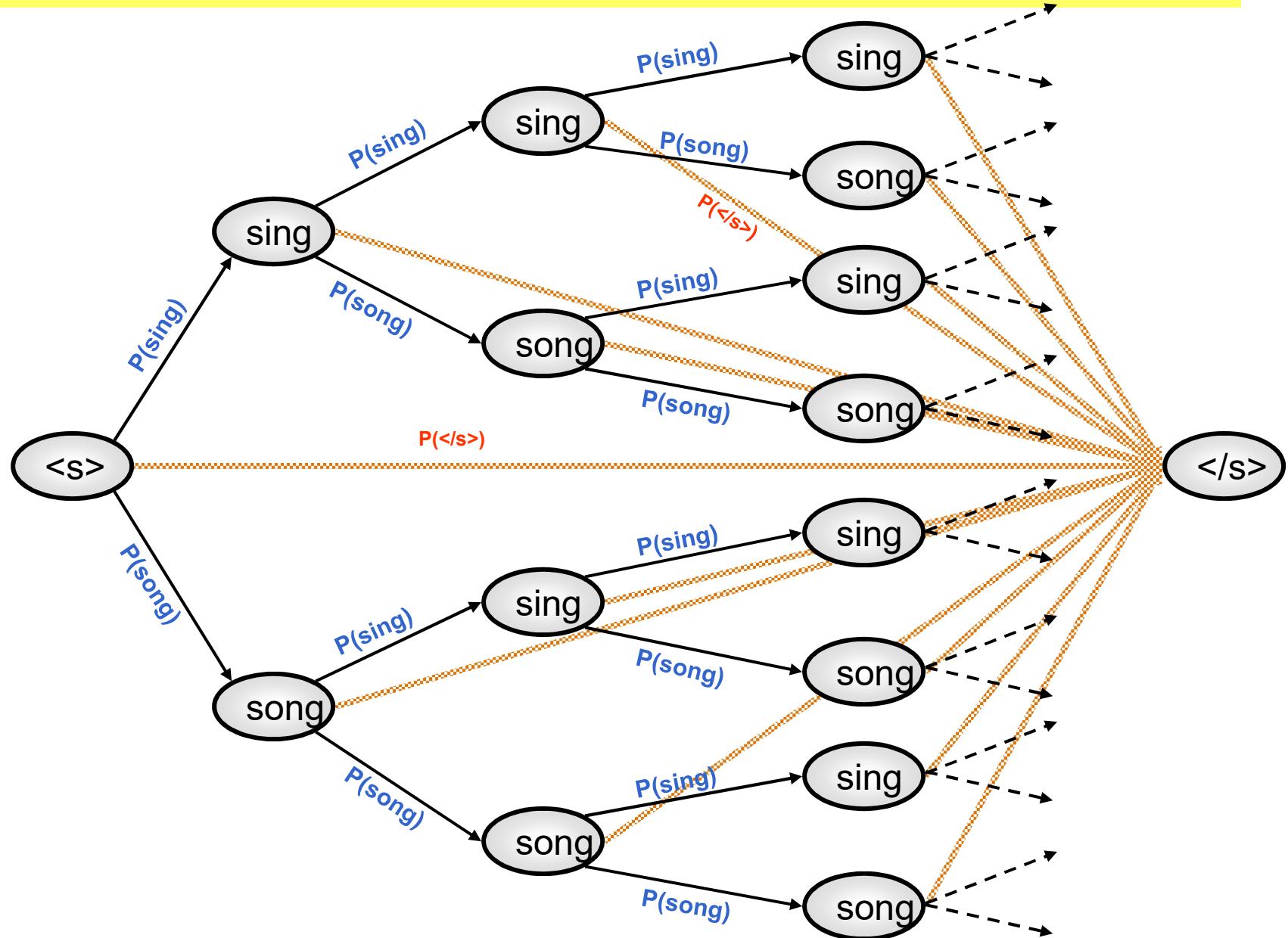


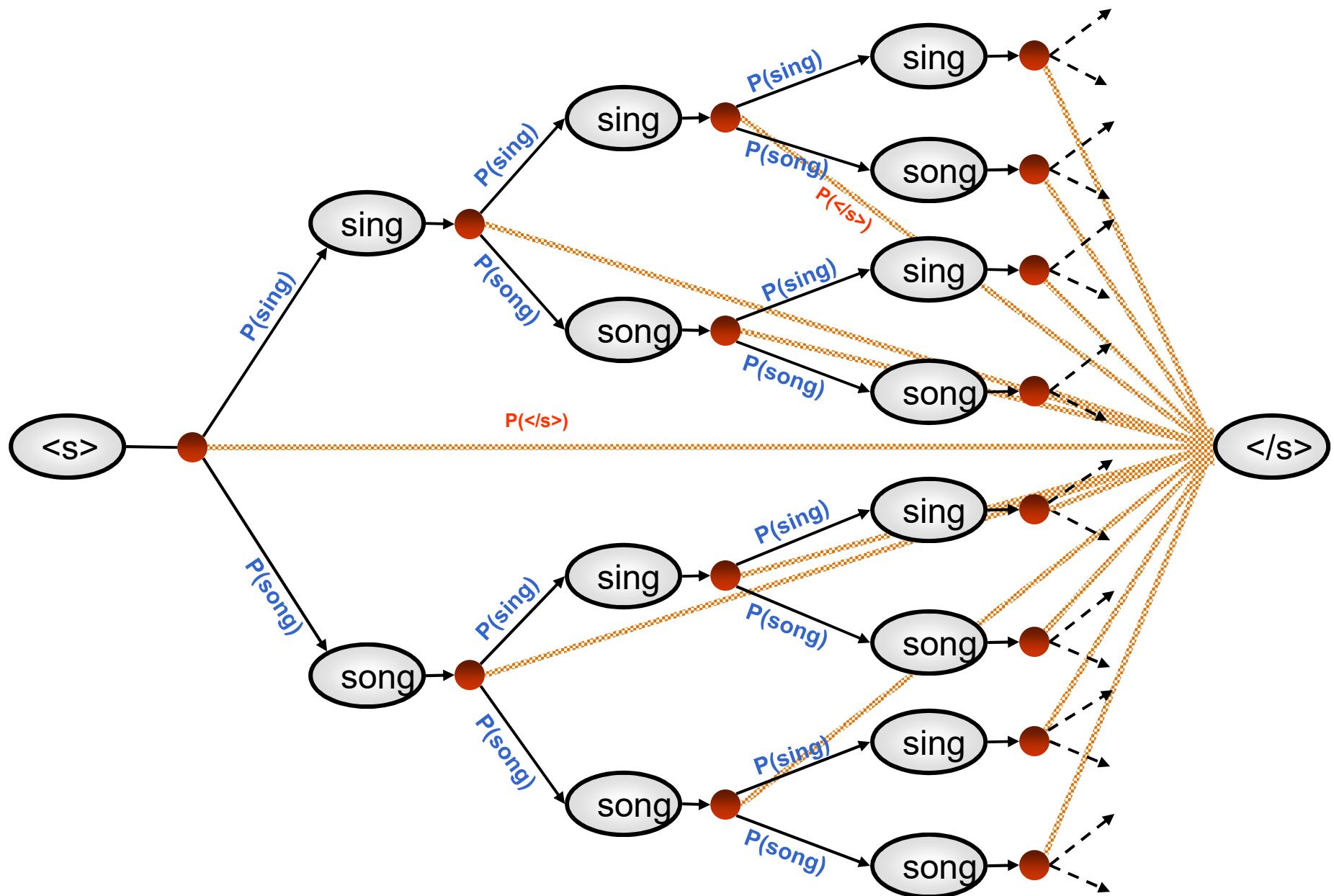
The environment state for
any word is implicit in the position
of the state

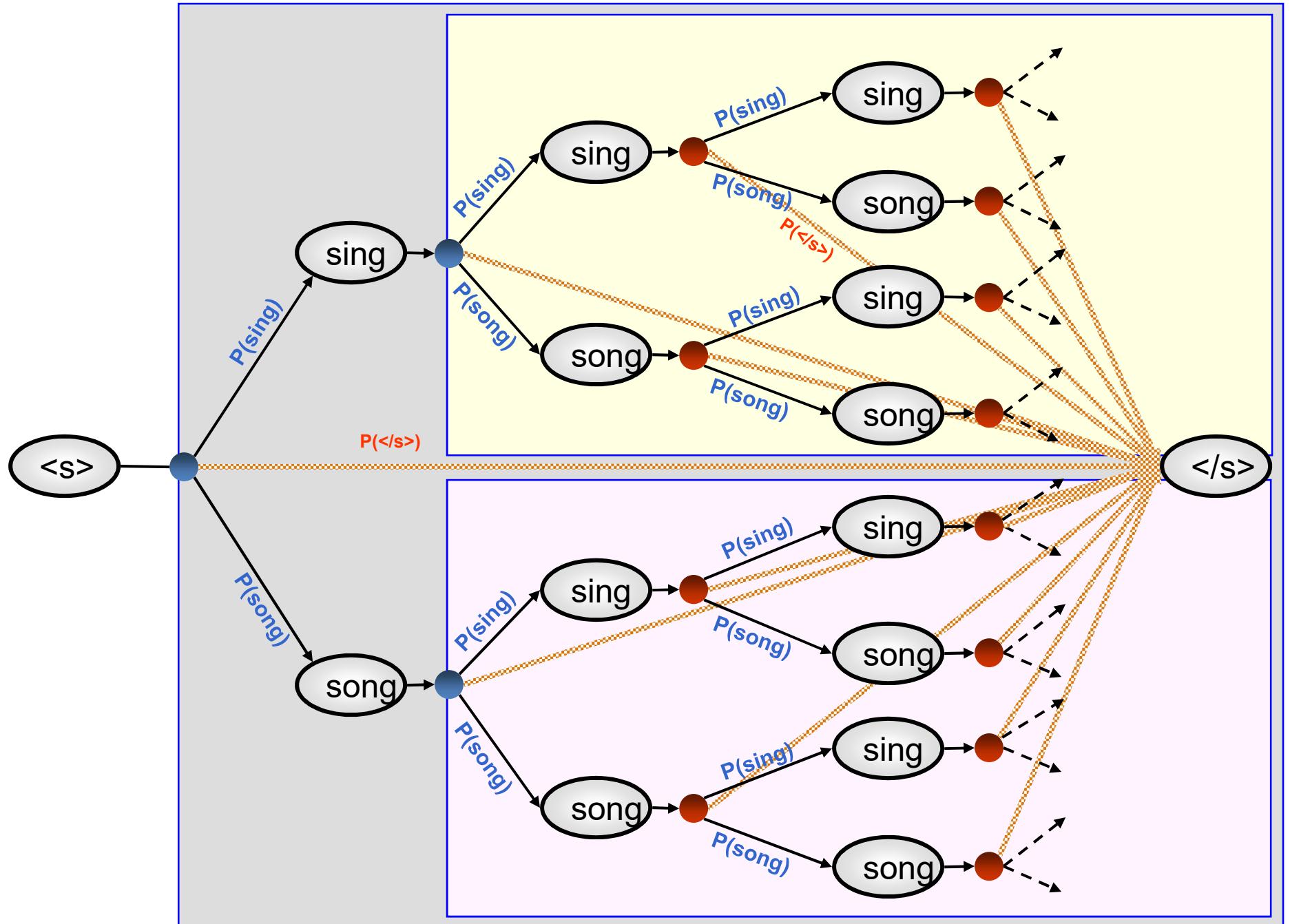
Considering the effect of N-gram assumptions

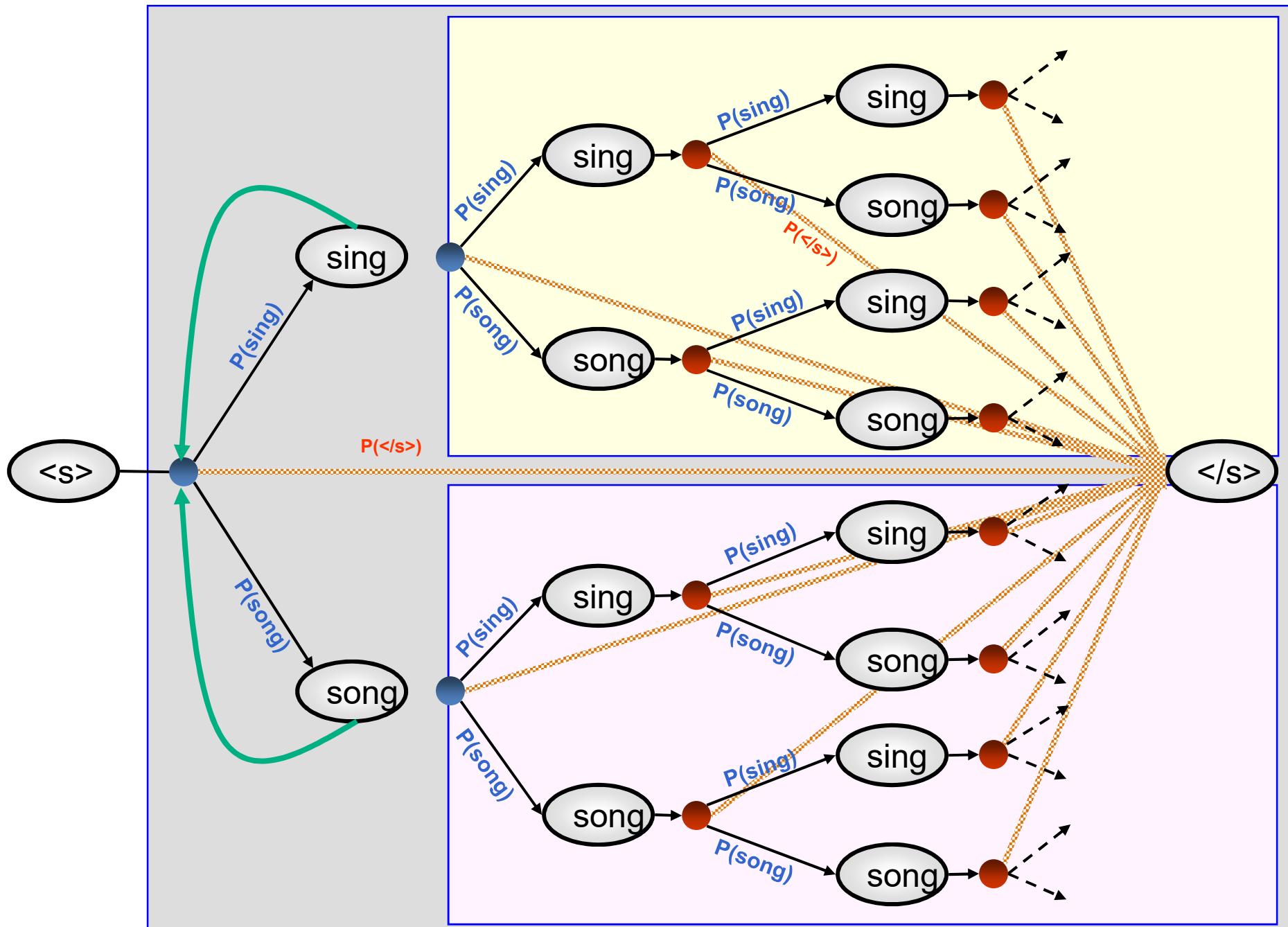
- Lets sequentially consider the effect of the unigram, bigram and higher-order assumptions

The two-word example as a full tree with a unigram LM

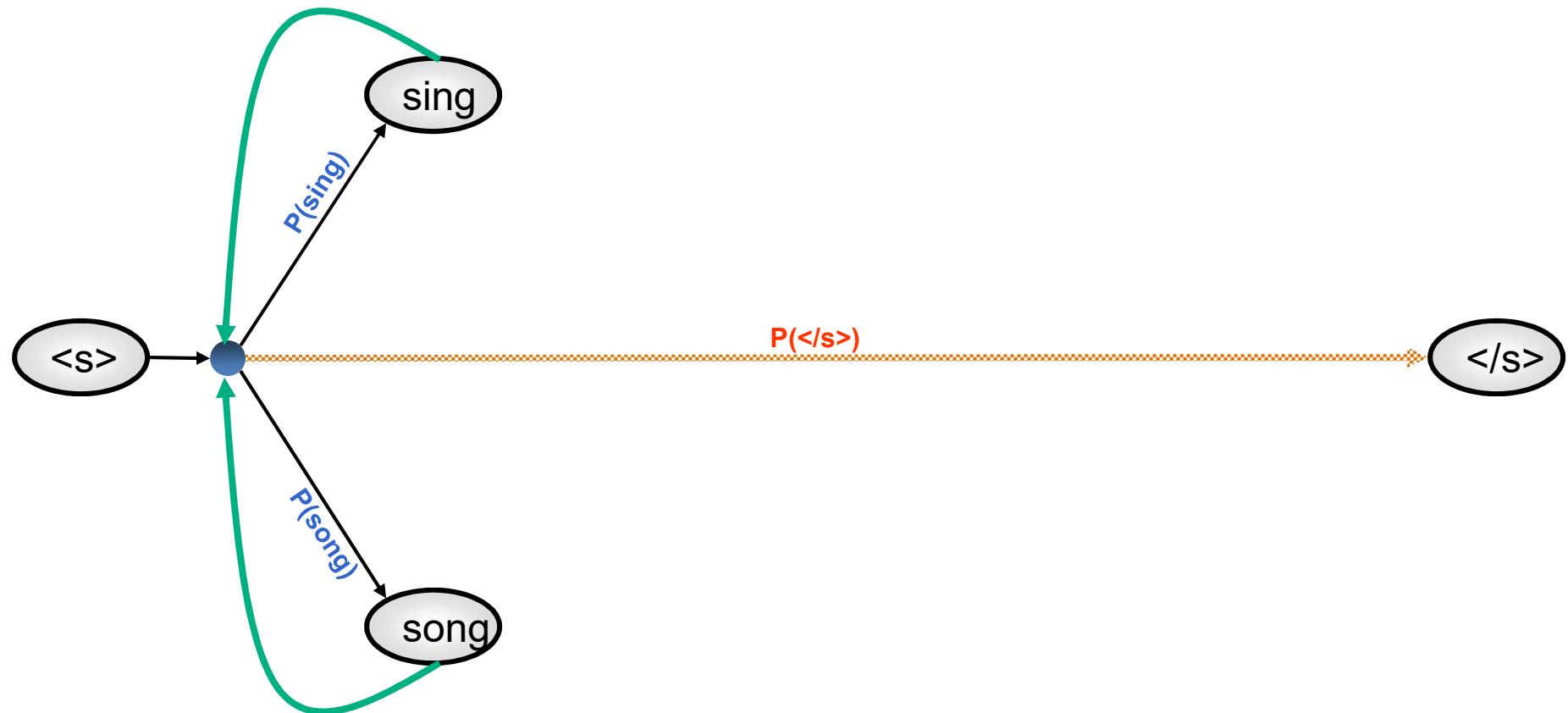




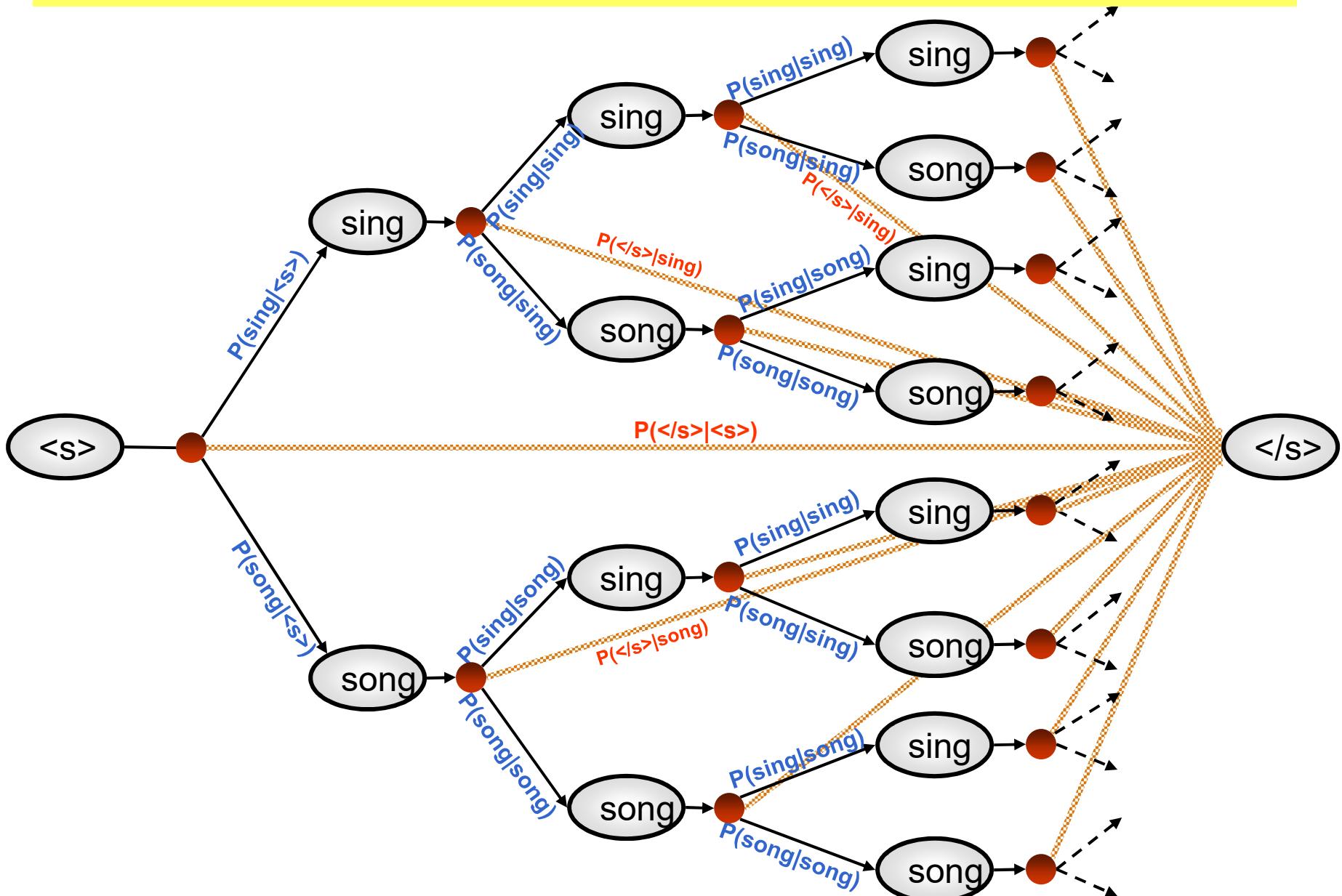




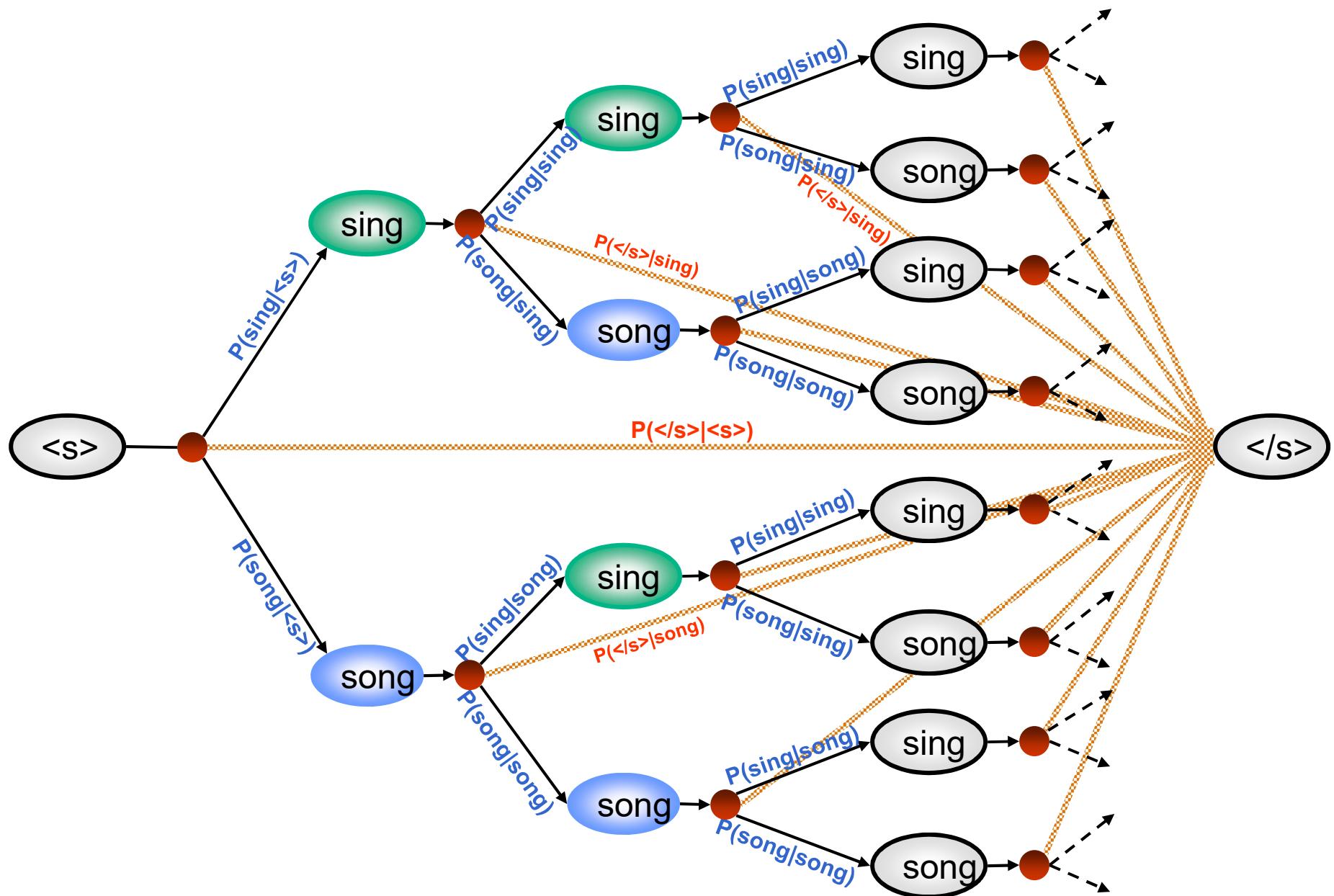
The two-word example with a unigram LM

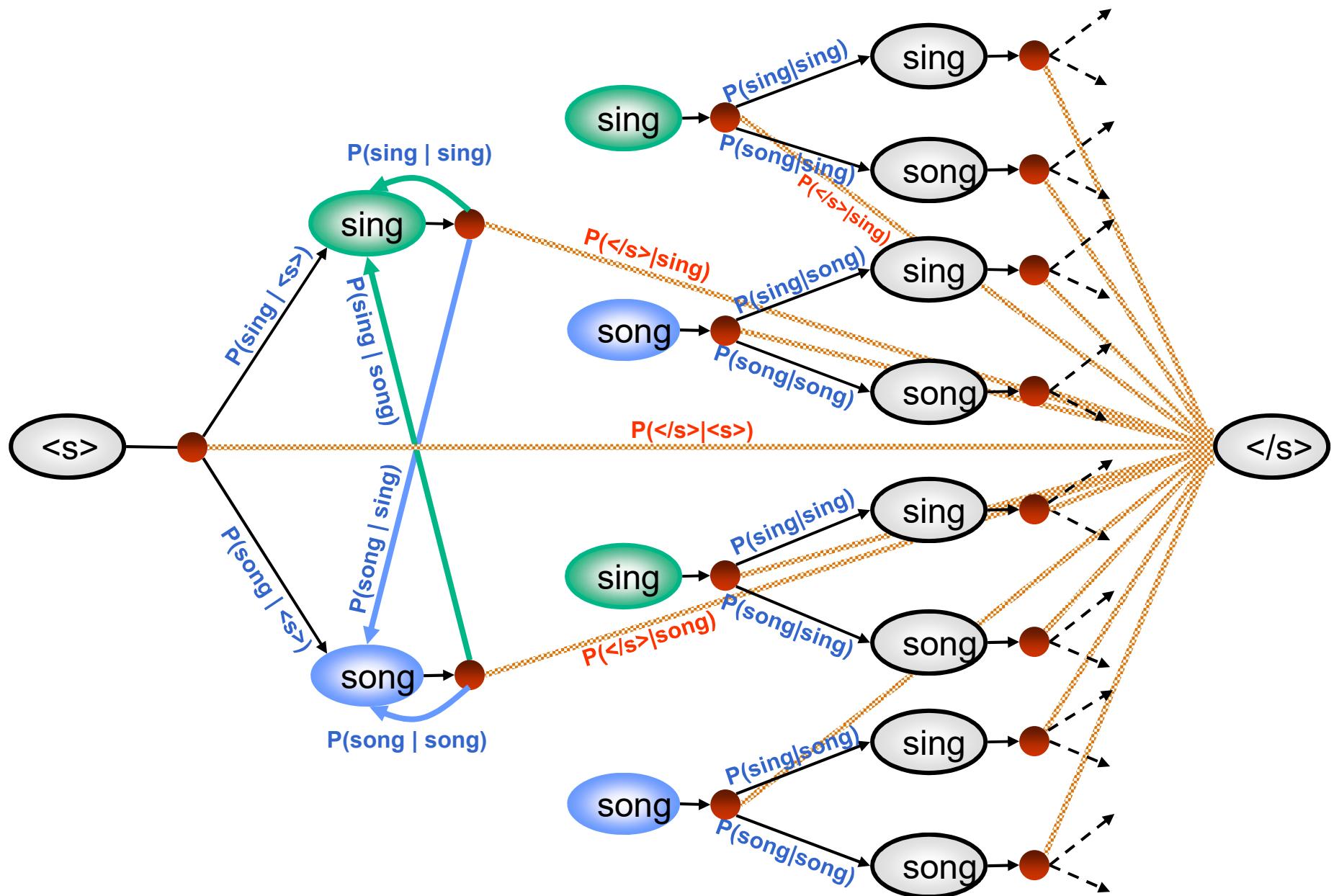


The two-word example as a full tree with a bigram LM

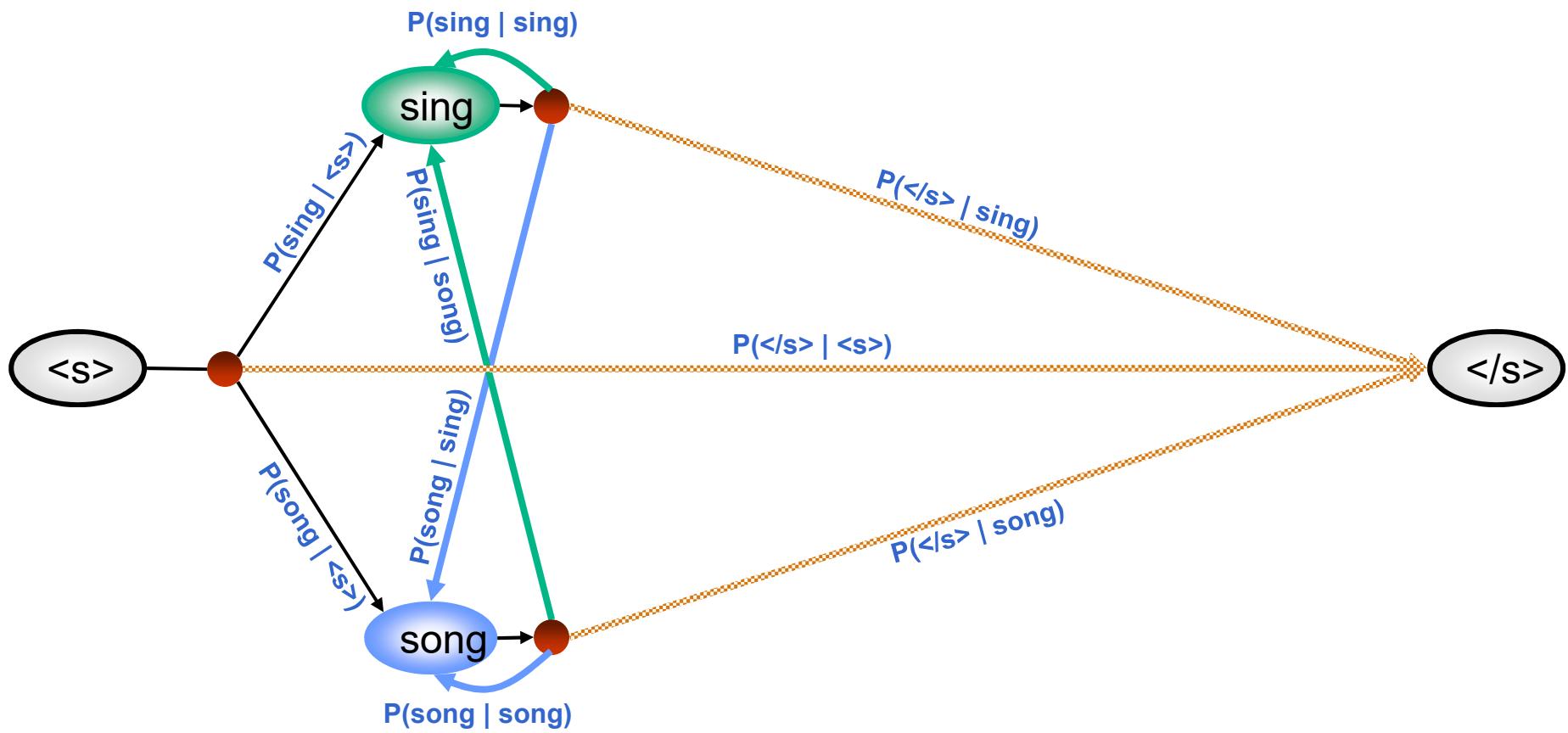


- The structure is recursive and can be collapsed

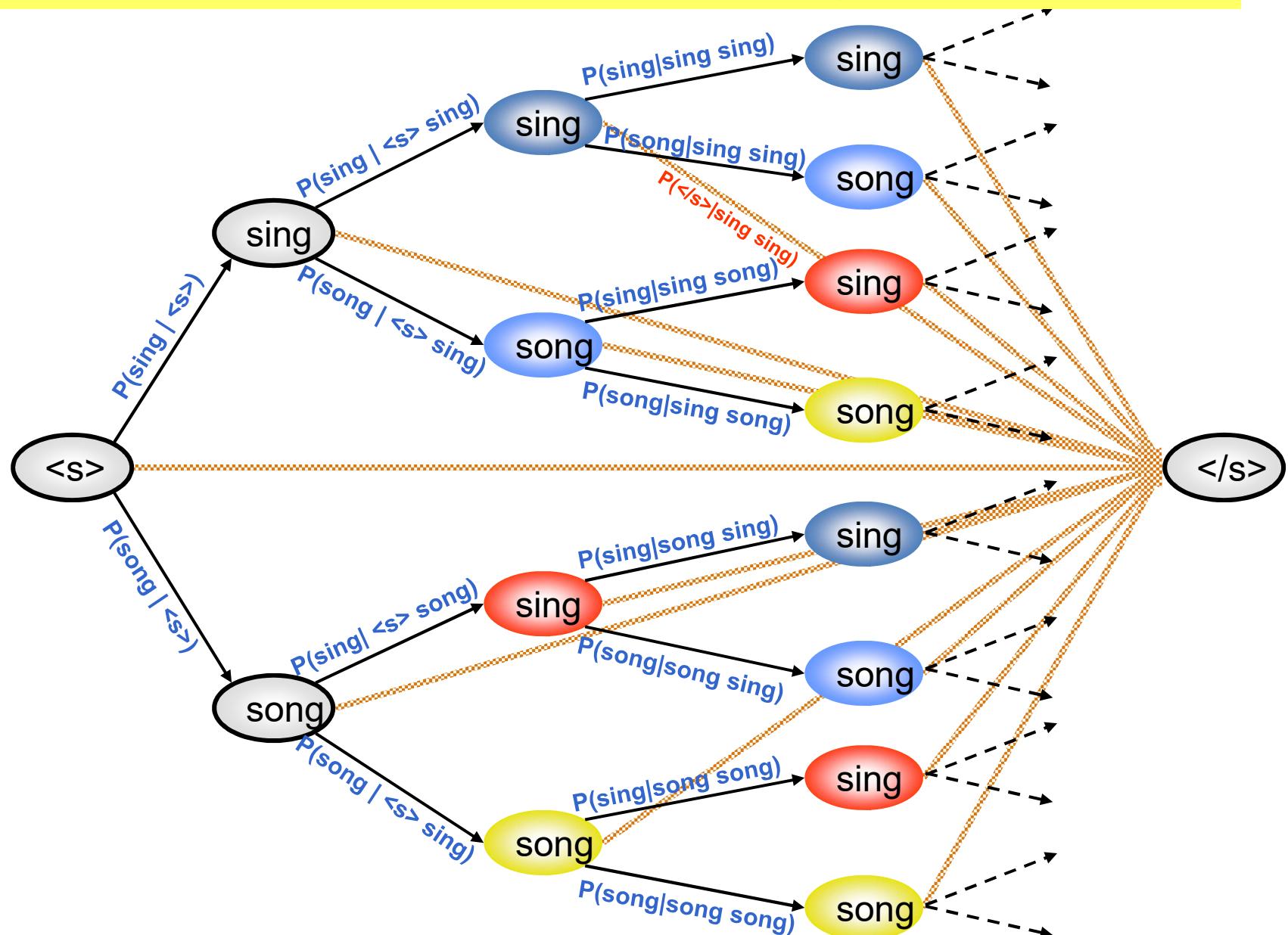




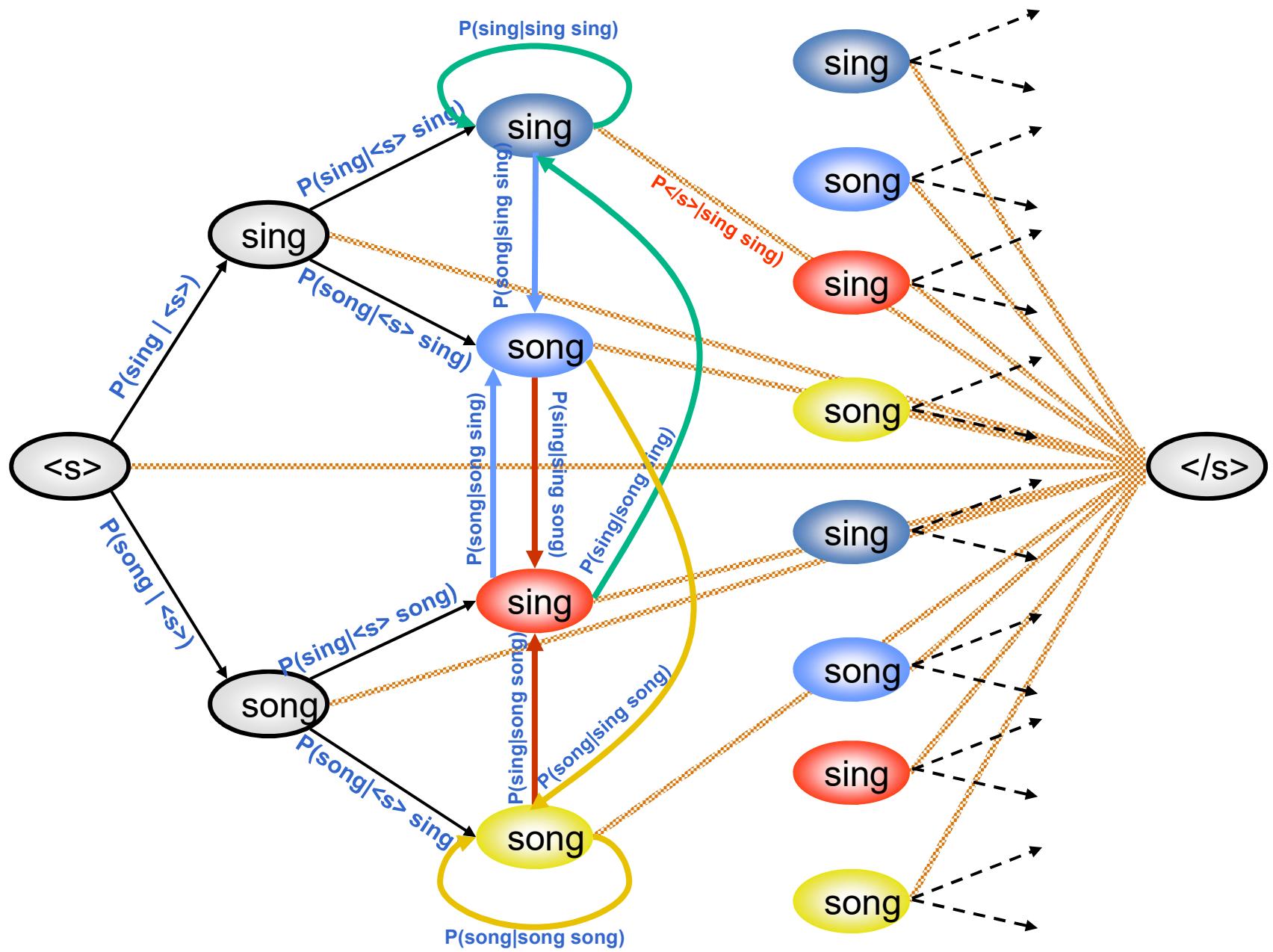
The two-word example with a bigram LM

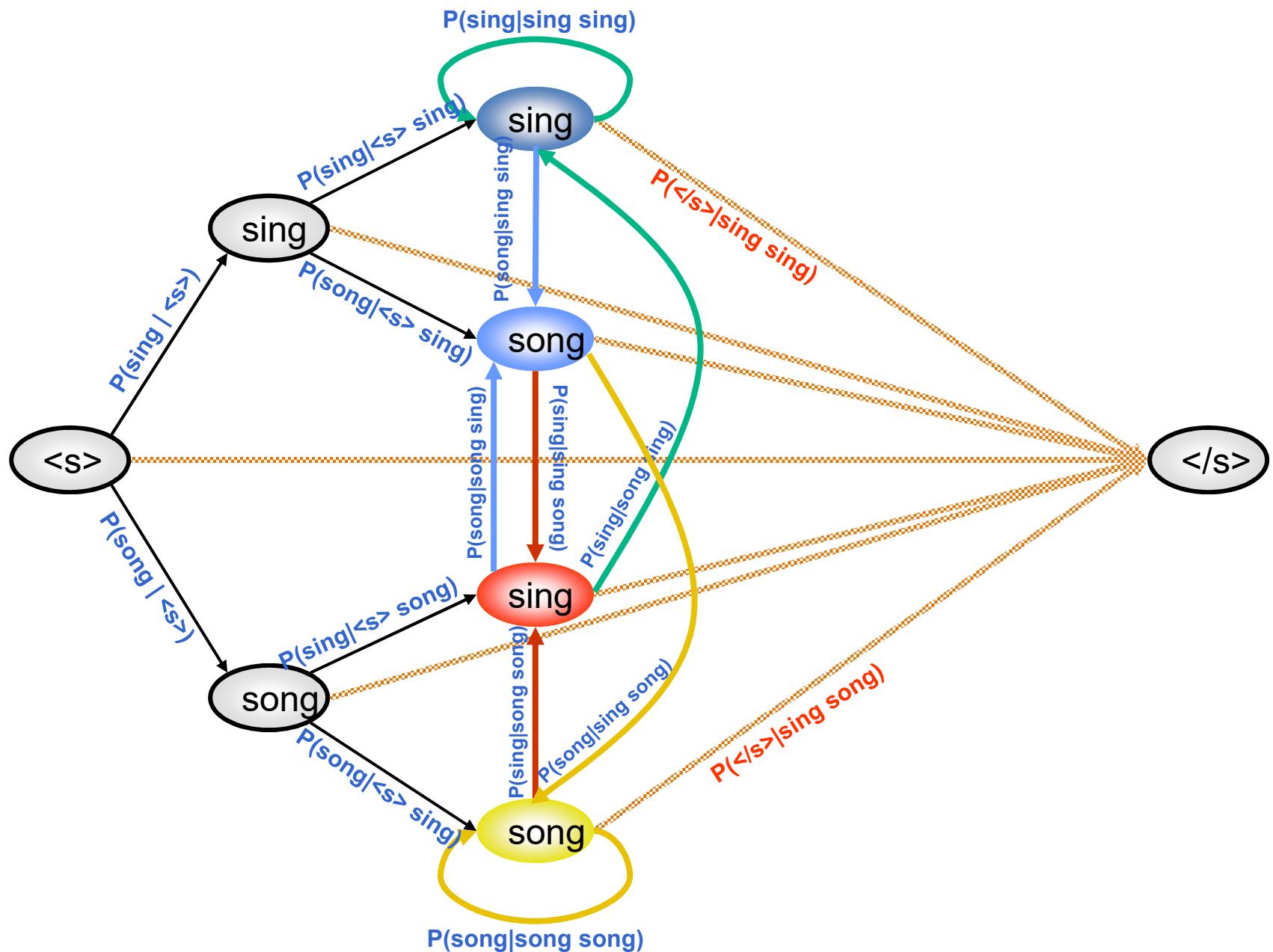


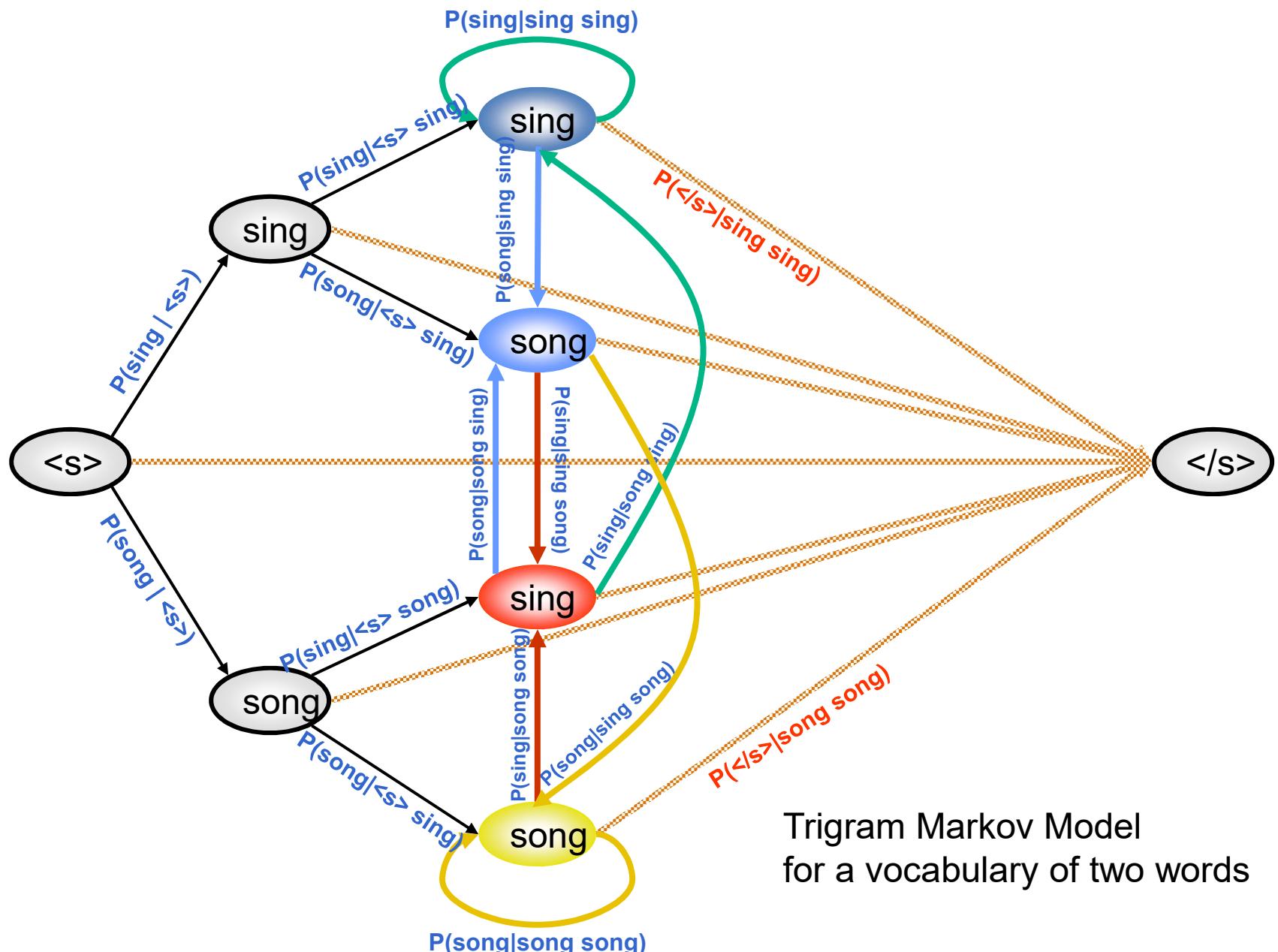
The two-word example as a full tree with a trigram LM



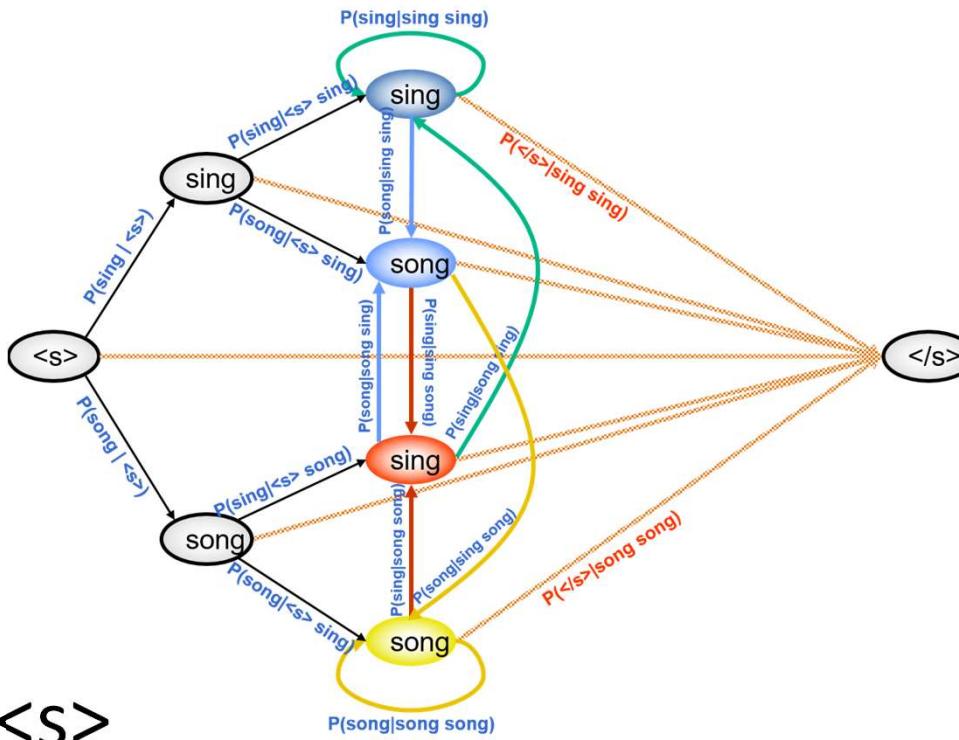
- The structure is recursive and can be collapsed







Generating Text



- Start at $<\text{s}>$
- Randomly choose one of the outgoing edges according to their probability
- Stop producing text when you hit $</\text{s}>$

Example

- Unigram model estimated on 2.8M words of American political blog text.

this trying our putting and funny
and among it herring it obama
but certainly foreign my
c on byron again but from i
i so and i chuck yeah the as but but republicans if
this stay oh so or it mccain bush npr this with what
and they right i while because obama

Example

- Bigram model estimated on 2.8M words of American political blog text.

the lack of the senator mccain hadn t keep this story backwards

while showering praise of the kind of gop weakness it was mistaken for american economist anywhere in the white house press hounded the absence of those he s as a wide variety of this election day after the candidate b richardson was polled ri in hempstead moderated by the convention that he had zero wall street journal argues sounds like you may be the primary but even close the bill told c e to take the obama on the public schools and romney fred flinstone s see how a lick skillet road it s little sexist remarks

Example

- Trigram model estimated on 2.8M words of American political blog text.

as i can pin them all none of them want to bet that
any of the might be

conservatism unleashed into the privacy rule book and
when told about what paul

fans organized another massive fundraising initiative
yesterday and i don t know what the rams supposedly
want ooh

but she did but still victory dinner
alone among republicans there are probably best not
all of the fundamentalist community

asked for an independent maverick now for
crystallizing in one especially embarrassing

Example

- 5-gram model estimated on 2.8M words of American political blog text.

he realizes fully how shallow and insincere conservative behavior has been he realizes that there is little way to change the situation

this recent arianna huffington item about mccain issuing heartfelt denials of things that were actually true or for that matter about the shia sunni split and which side iran was on would get confused about this any more than someone with any knowledge of us politics would get confused about whether neo confederates were likely to be supporting the socialist workers party

at the end of the world and i m not especially discouraged now that newsweek shows obama leading by three now

Example

- 100-gram model estimated on 2.8M words of American political blog text.

and it would be the work of many hands to catalogue all the ridiculous pronouncements made by this man since his long train of predictions about the middle east has been gaudily disastrously stupefyingly misinformed just the buffoon it seems for the new york times to award with a guest column for if you object to the nyt rewarding failure in quite this way then you re intolerant according to the times editorial page editor andrew rosenthal rosenthal doesn t seem to recognize that his choice of adjectives to describe kristol serious respected are in fact precisely what is at issue for those whom he dismisses as having a fear of opposing views

N-Gram Models

Pros

- Easily understood **linguistic formalism.**
- Fully generative.
- Algorithms:
 - calculate probability of a sequence
 - choose a sequence from a set
 - training

Cons

- Obviously inaccurate linguistic formalism.
- As N grows, data sparseness becomes a problem.
 - Smoothing is a black art.
- How to deal with unknown words?

N-Gram Models

Pros

- Easily understood **linguistic formalism.**
- Fully generative.
- Algorithms:
 - calculate probability of a sequence
 - choose a sequence from a set
 - training

Cons

- Obviously inaccurate linguistic formalism.
- As N grows, data sparseness becomes a problem.
 - Smoothing is a black art.
- How to deal with unknown words?

Calculating the Probability of a Sequence

- Let n be the length of the sequence and m be the length of the history.
- For every consecutive $(m+1)$ words $w_i \dots w_{i+m}$, look up $p(w_{i+m} | w_i \dots w_{i+m-1})$.
- Look up $p(\text{stop} | w_{n-m} \dots w_n)$.
- Multiply these quantities together.

Choosing a Sequence from a Set

- Calculate the probability of each sequence in the set.
- Choose the one that has the highest probability.

Training

- Maximum likelihood estimation by relative frequencies:

unigram	$\hat{\gamma}(w) = \frac{\text{freq}(w)}{\#\text{words}}$
bigram	$\hat{\gamma}(w w') = \frac{\text{freq}(w'w)}{\text{freq}(w')}$
trigram	$\hat{\gamma}(w w'w'') = \frac{\text{freq}(w'w''w)}{\text{freq}(w'w'')}$
general	$\hat{\gamma}(w \mathbf{h}) = \frac{\text{freq}(\mathbf{h}w)}{\text{freq}(\mathbf{h})}$

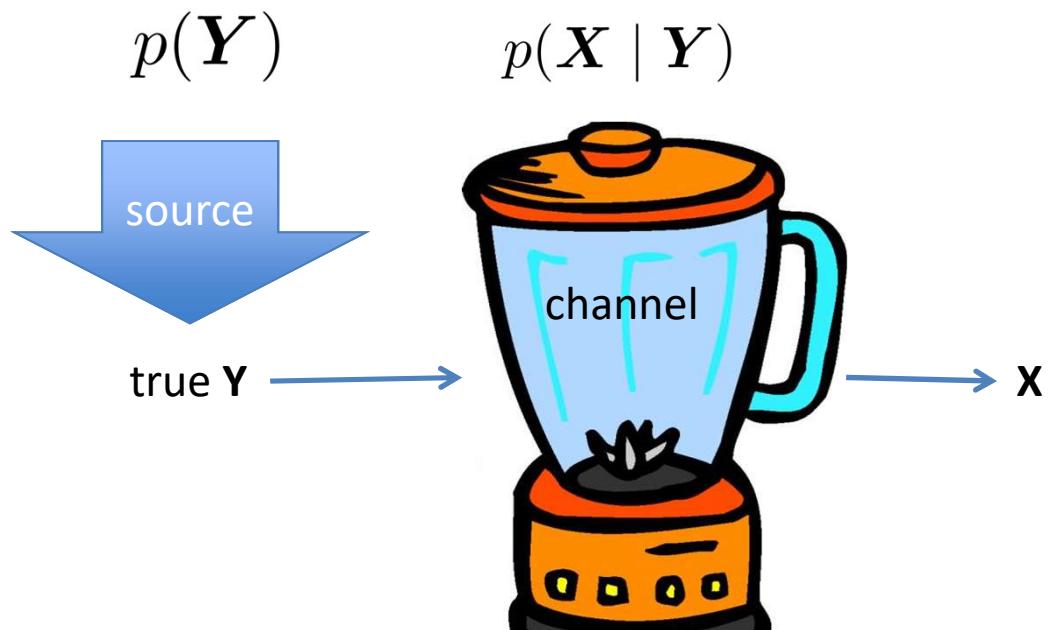
Note about Current Research

- In the past few years, “web-scale” n-gram models have become popular.
 - More data always seem to make language models better (Brants et al., 2007, *inter alia*)
- A number of recent research efforts seek to make the construction and use of language models very efficient.
 - Runtime: MapReduce architectures (e.g., Lin & Dyer, 2010)
 - Memory: compression (e.g., Heafield, 2011)
- Neural language models
 - Too many to cite
 - Not really a “fixed” sized history (n) anymore

Sequence Models as Components

- Typically we care about a sequence together with something else.
 - Analysis: sequence in, predict “something else.”
 - Generation: “something else in,” sequence out.
- Sequence models are useful components in *both* scenarios.

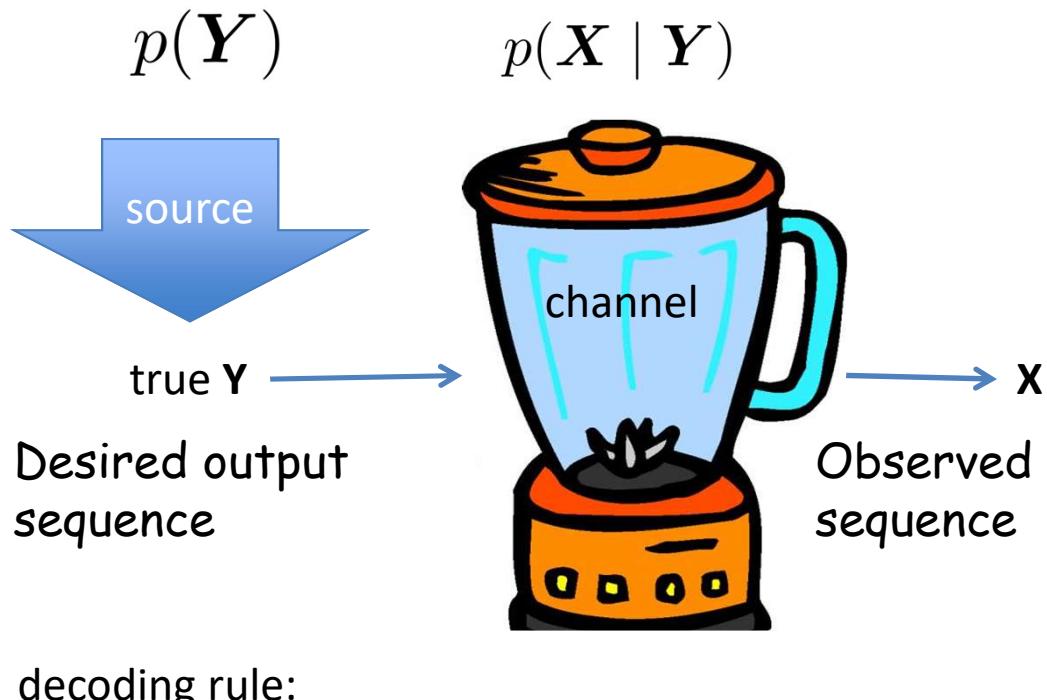
Noisy Channel



decoding rule:

$$\hat{y} = \arg \max_y p(y | x) = \arg \max_y p(x | y) \times p(y)$$

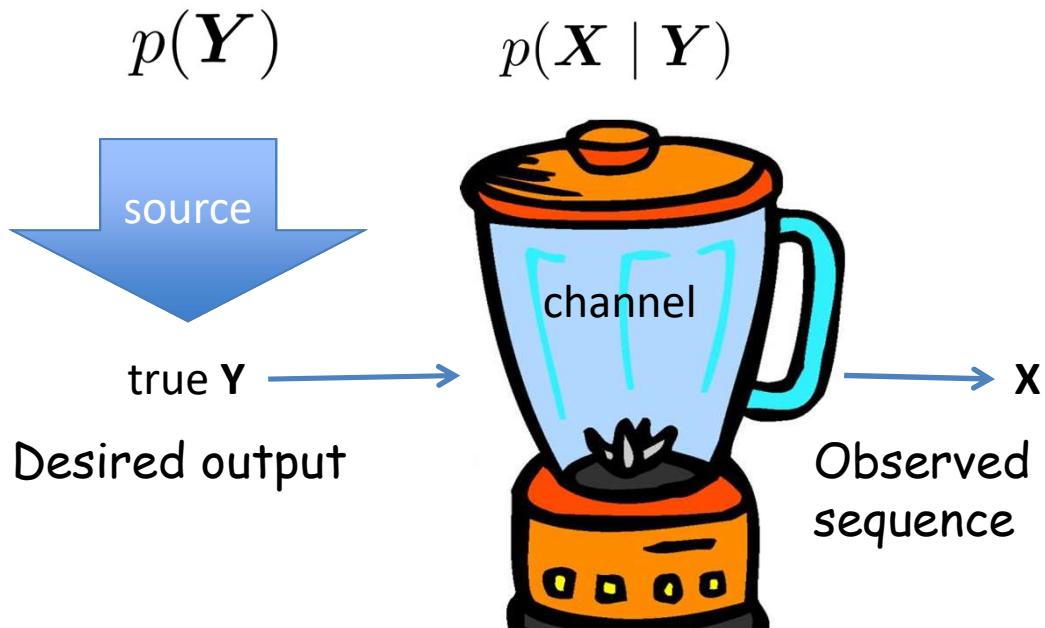
Sequence Model as Source



- speech recognition (Jelinek, 1997)
- machine translation (Brown et al., 1993)
- optical character recognition (Kolak and Resnik, 2002)
- spelling and punctuation correction (Kernighan et al., 1990)

$$\hat{y} = \arg \max_y p(y | x) = \arg \max_y p(x | y) \times p(y)$$

Sequence Model as Channel



decoding rule:

$$\hat{y} = \arg \max_y p(y | x) = \arg \max_y p(x | y) \times p(y)$$

- text categorization
- language identification
- information retrieval
(Ponte and Croft, 1998; Berger and Lafferty, 1999)
- Sentence compression
(Knight and Marcu, 2002)
- Question to search query (Radev et al., 2001)

It's Hard to Beat N-Grams!

- They are very fast to work with. They fit the data really, really well.
- Improvements for *some specific problems* follow from:
 - task-specific knowledge
 - domain knowledge (e.g., linguistics)

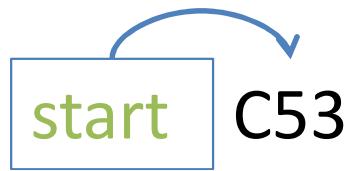
Class-Based Sequence Models

- From Brown et al. (1990):

$$p(\text{start}, w_1, w_2, \dots, w_n, \text{stop}) = \prod_{i=1}^{n+1} \gamma(w_i \mid \text{cl}(w_i)) \times \eta(\text{cl}(w_i) \mid \text{cl}(w_{i-1}))$$

- “cl” is a deterministic function from words to a smaller set of classes.
 - Each word only gets one class; known in advance.
 - Discovered from data using a clustering algorithm.

start

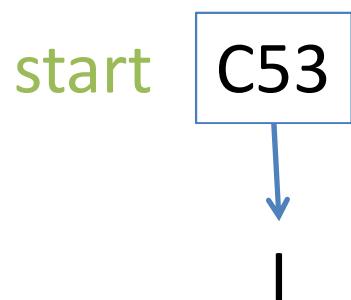


one “next class” die per class:

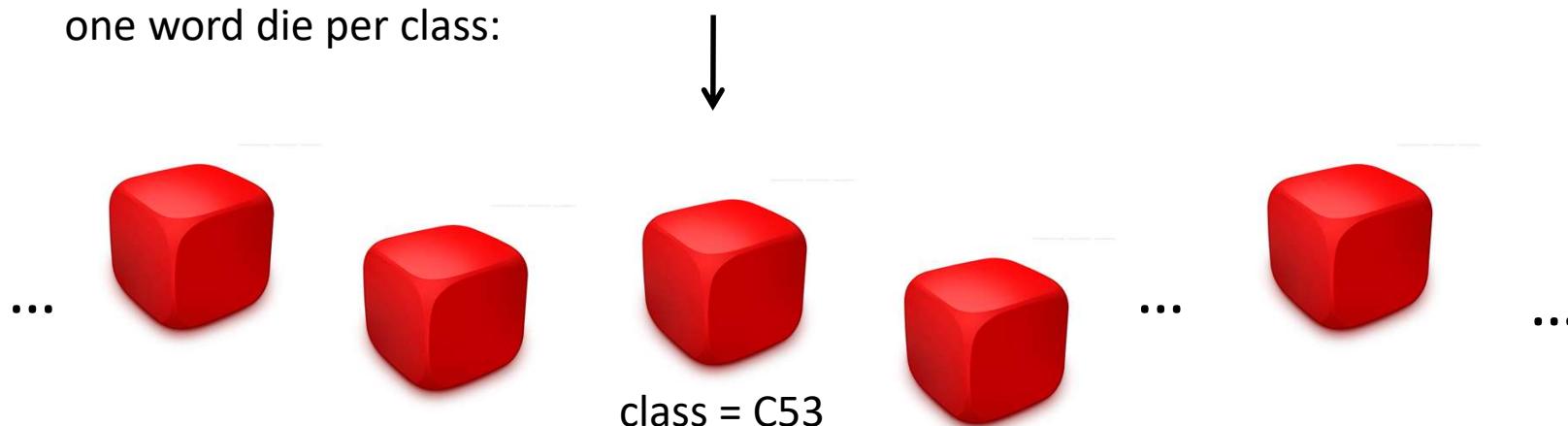


history = **start**

Each word appears on
only one of the word dice.



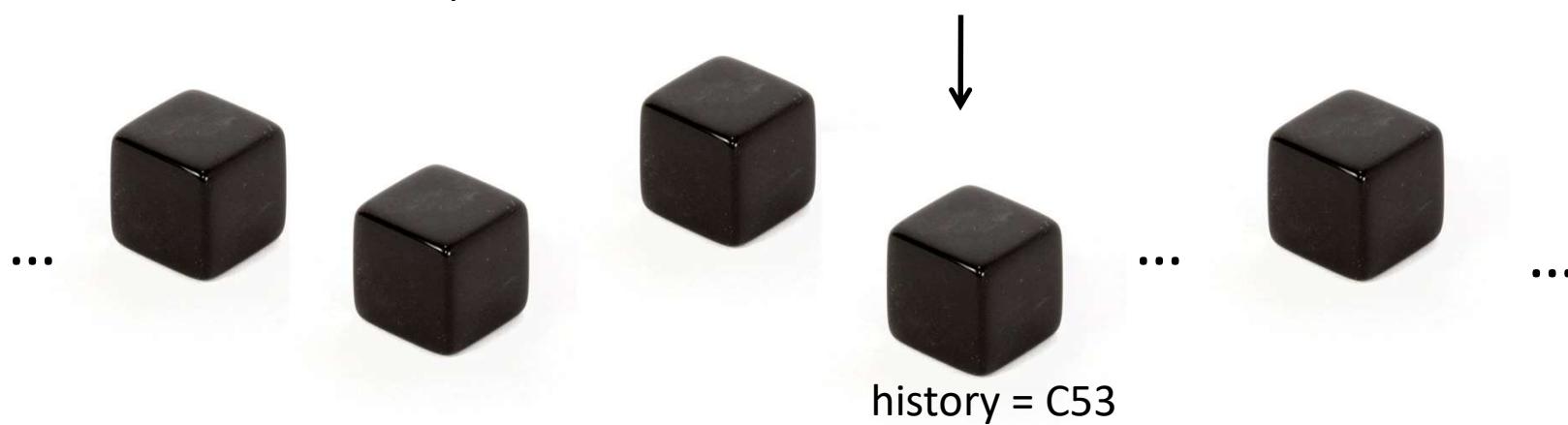
one word die per class:





|

one “next class” die per class:

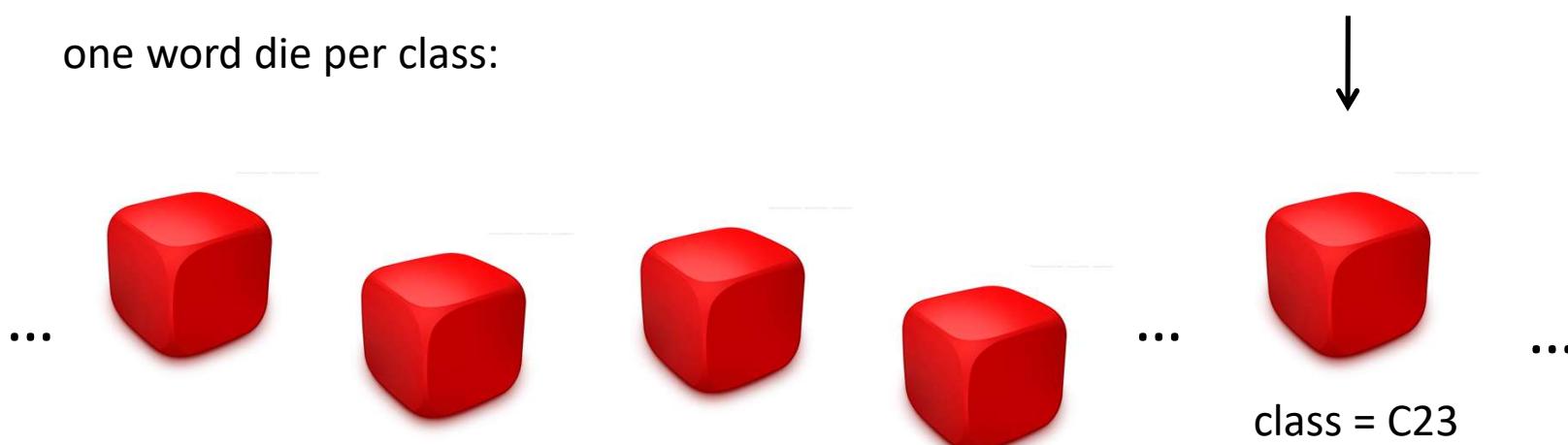


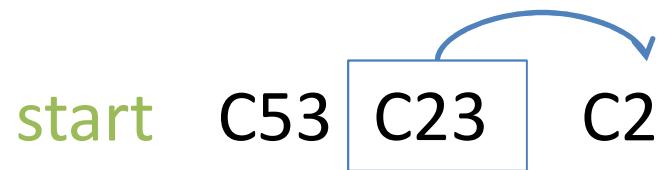
start C53 C23



I want

one word die per class:

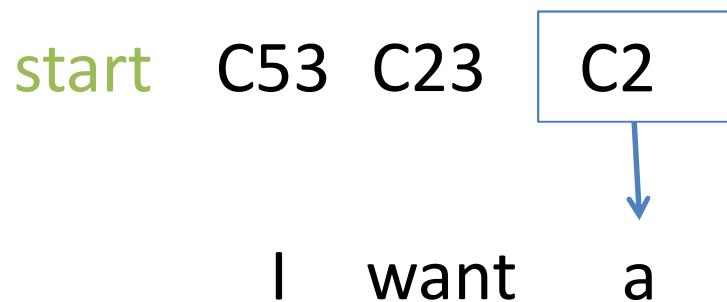




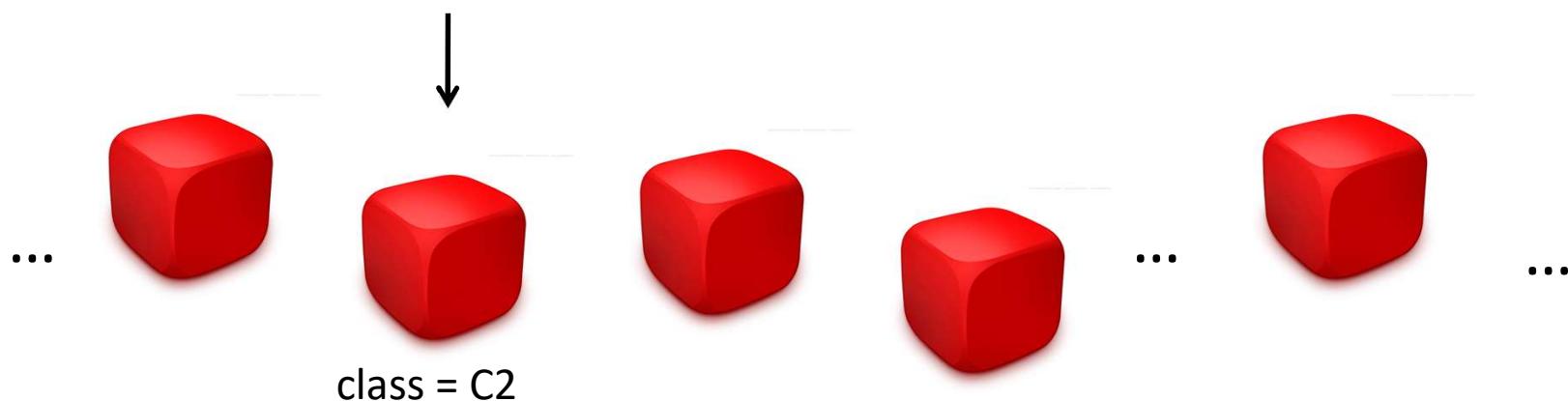
I want

one “next class” die per class:





one word die per class:

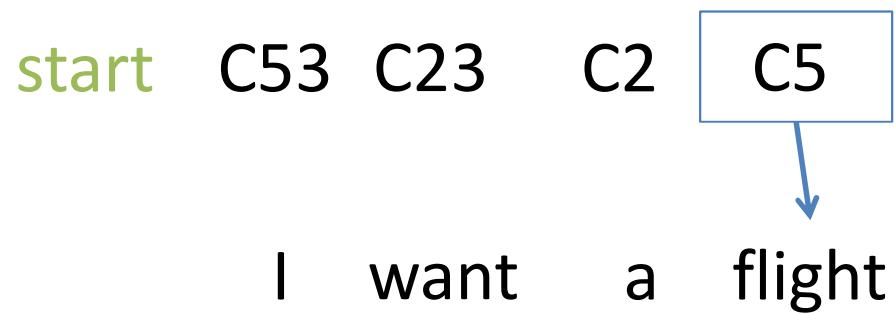




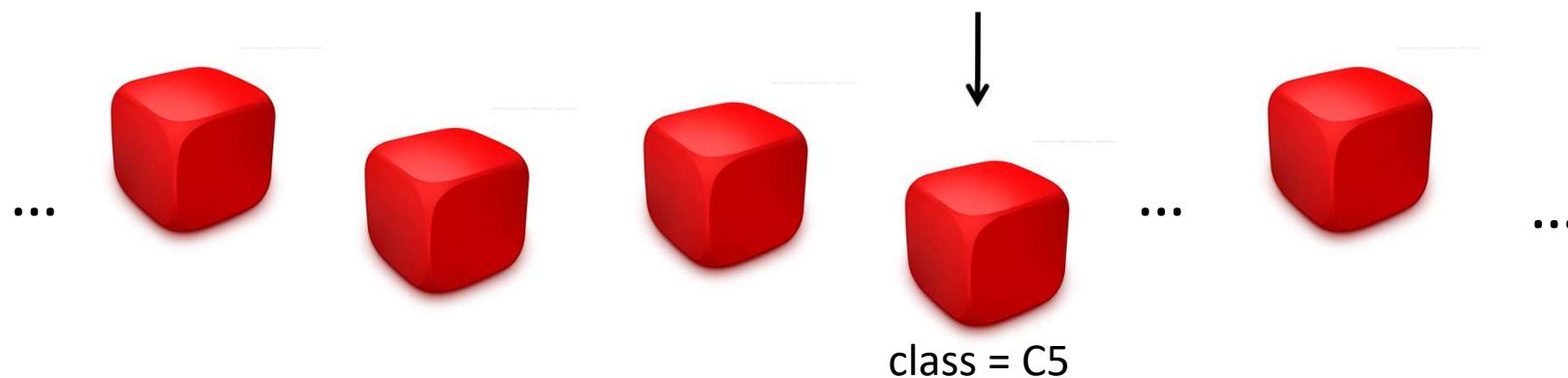
I want a

one “next class” die per class:





one word die per class:



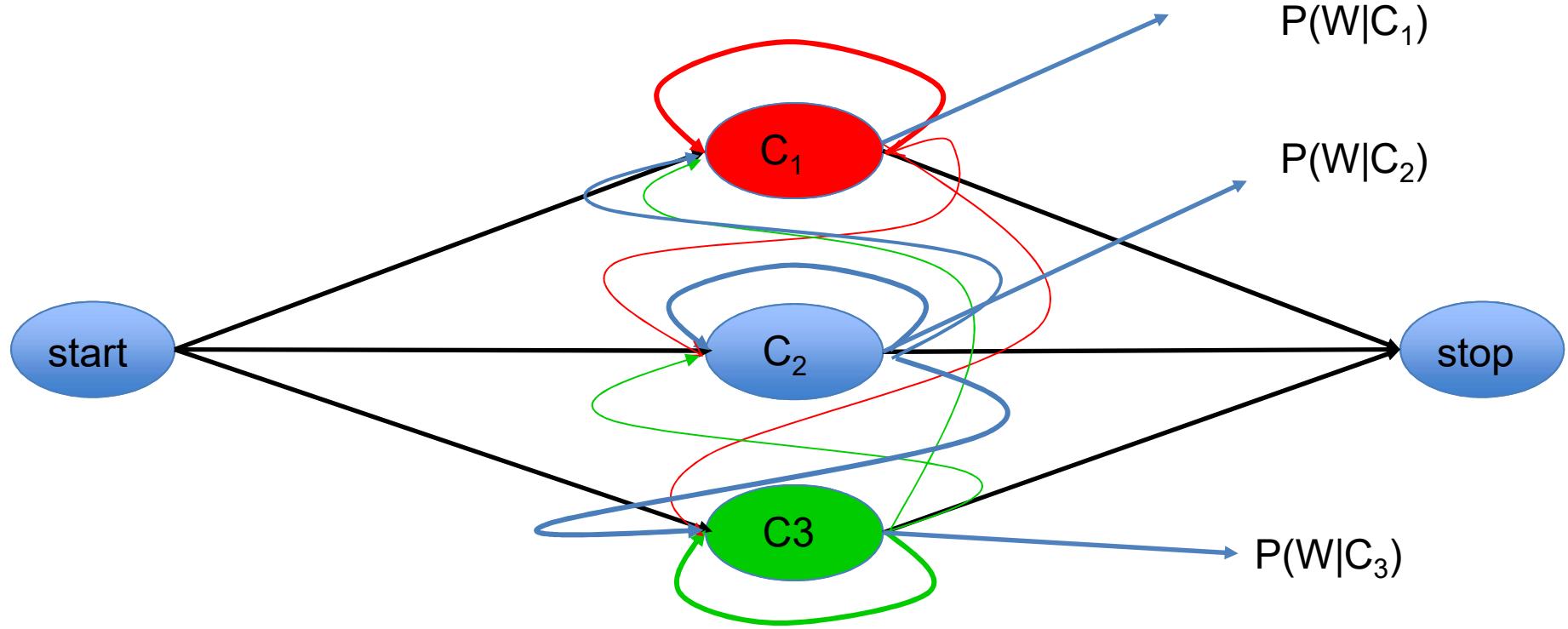
Class-Based Sequence Models

- From Brown et al. (1990):

$$p(\text{start}, w_1, w_2, \dots, w_n \text{stop}) = \prod_{i=1}^{n+1} \eta(w_i \mid \text{cl}(w_i)) \times \gamma(\text{cl}(w_i) \mid \text{cl}(w_{i-1}))$$

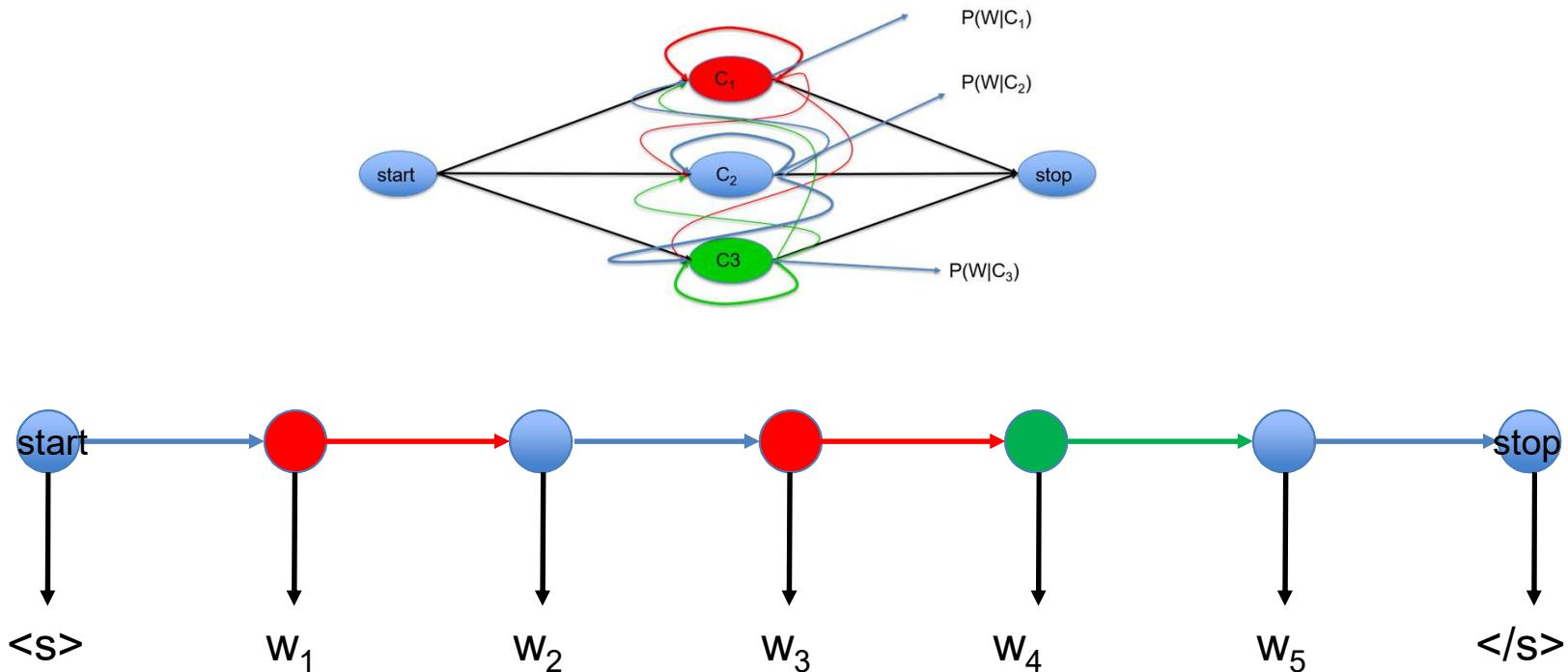
- Independence assumptions?
- Number of parameters?
- Generalization ability?

Markov Model



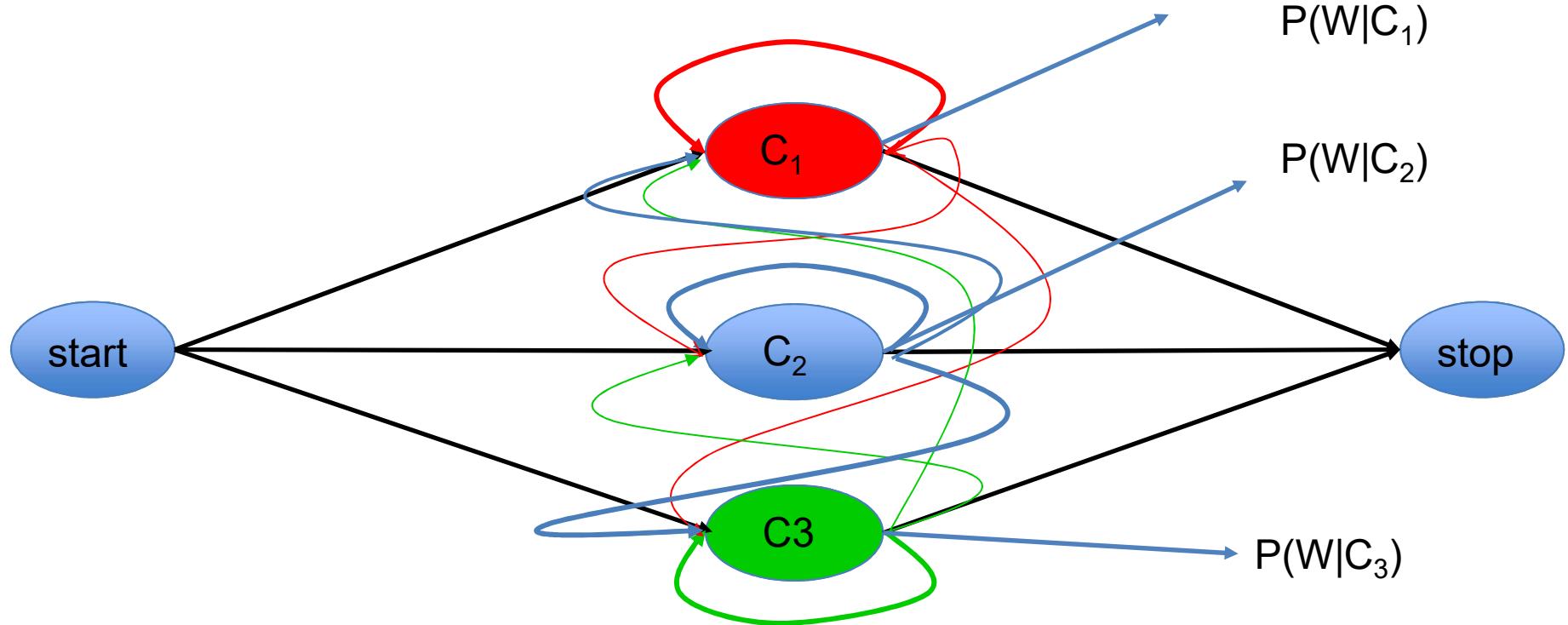
- Each time a class is visited, draw a word from the class

Generating Text



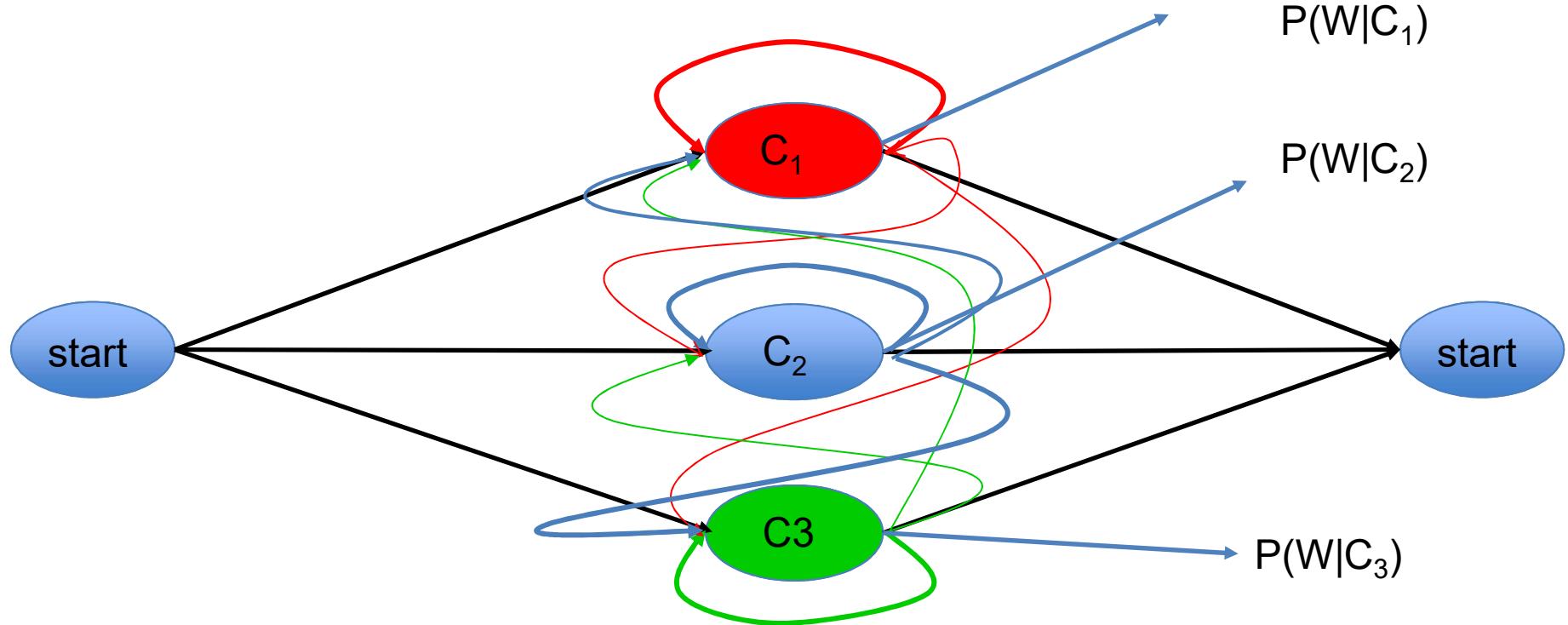
- Note the distinction between wandering through the states and producing the text

Hidden Markov Model



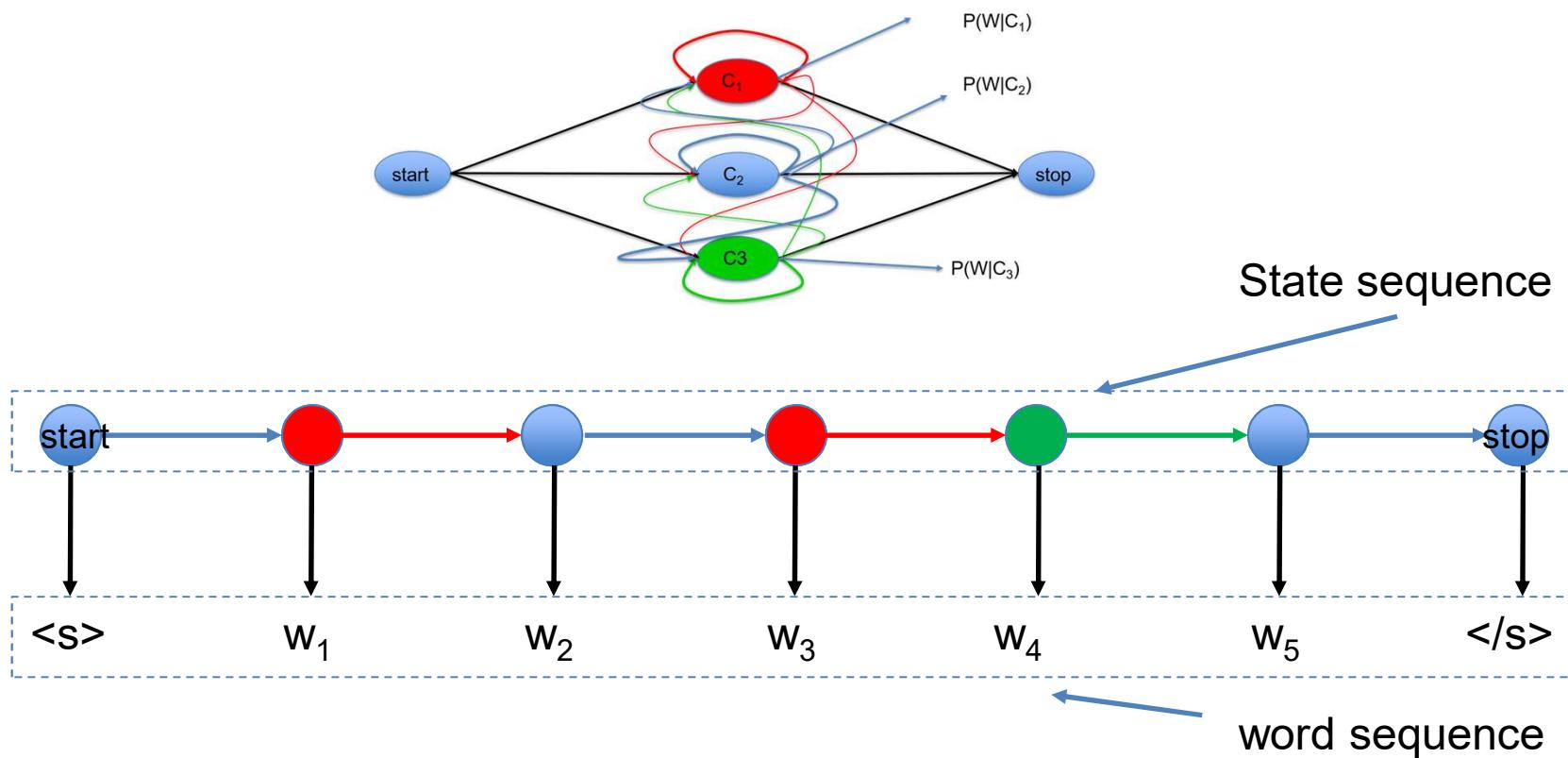
- Consider now the situation where a word may belong to multiple classes
 - E.g. if classes are Penn Bank labels, “I” may be a cardinal number, noun, or personal pronoun
 - In the limit $P(W | C)$ is a distribution over *all* words
 - All classes include all words, but differ in the probability with which they produce them

Hidden Markov Model



- This is a *hidden* Markov model
 - Also called a “stochastic function of a Markov chain”
 - The process transitions through a state sequence
 - From each state, it produces an observation
 - We only see the observation but do not directly see the state

HMM: Generating Text

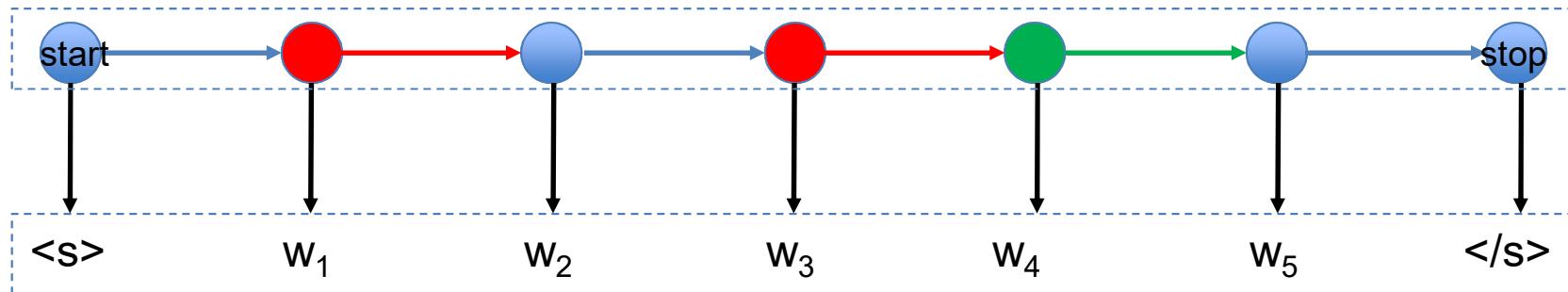


- We only get to observe the word sequence, but the actual state sequence is hidden to us

Parameters of the HMM

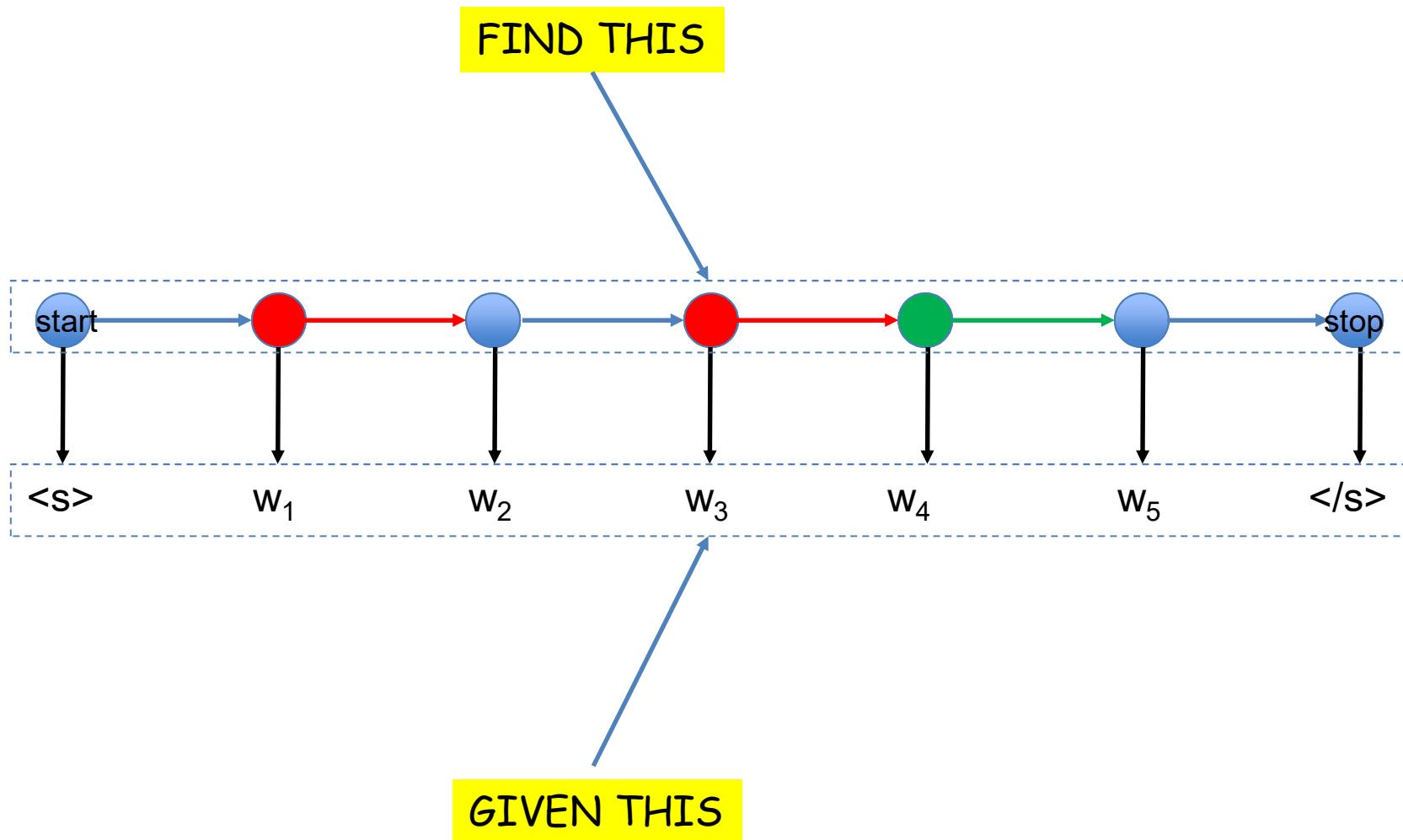
- The **state transition probabilities** of the underlying Markov chain
 - $P(S_i|S_j)$
- **Initial state probabilities**
 - What is the probability that *at the very first instant*, the process will be in state S_j
 - Often denoted by $\pi(S_j)$
- **Emission probabilities**
 - $P(w_i|s_j)$
- Note: we've switched from calling it a "class" to calling it a "state" since these are states of the Markov chain
 - To conform to Markov Chain terminology

Decoding the state sequence

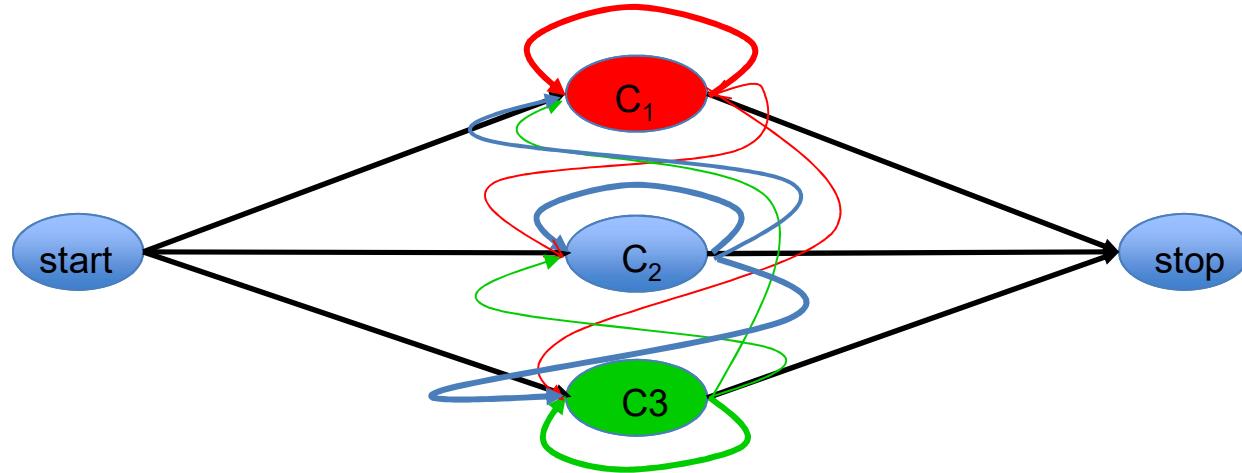


- *Preliminary:* Given all parameters of the HMM
 - Transition probabilities, initial state probabilities, emission probabilities
- *Problem:* Given a word sequence $< s > w_1 w_2 \dots < /s >$, find the underlying state sequence

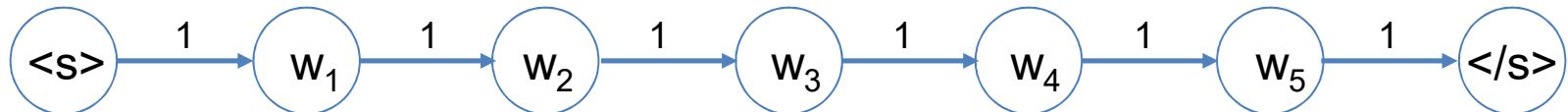
Decoding the state sequence



The graph view of the problem

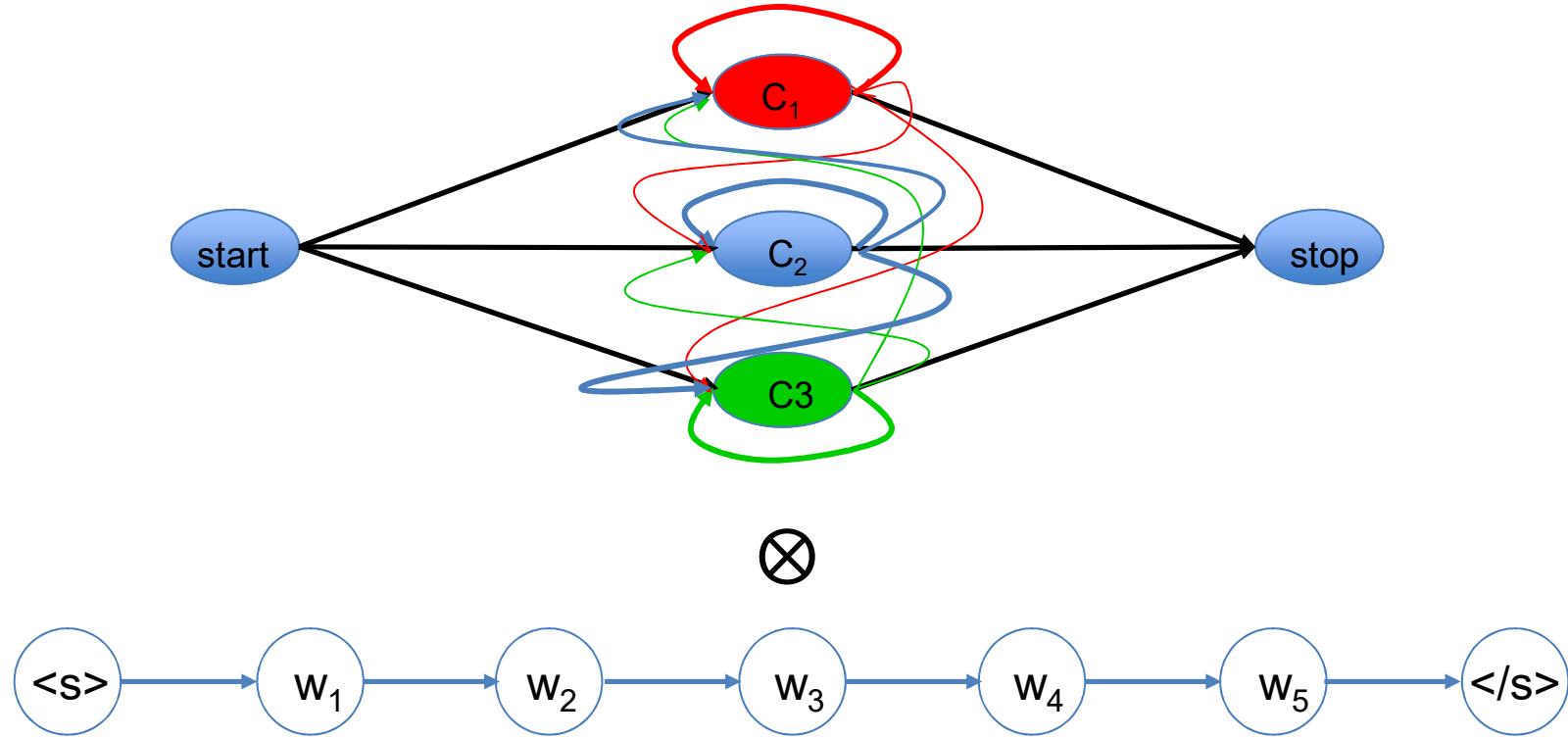


- Any valid state sequence *must* conform to the transition structure imposed by the Markov model
 - It *must* be a valid path through the Markov graph



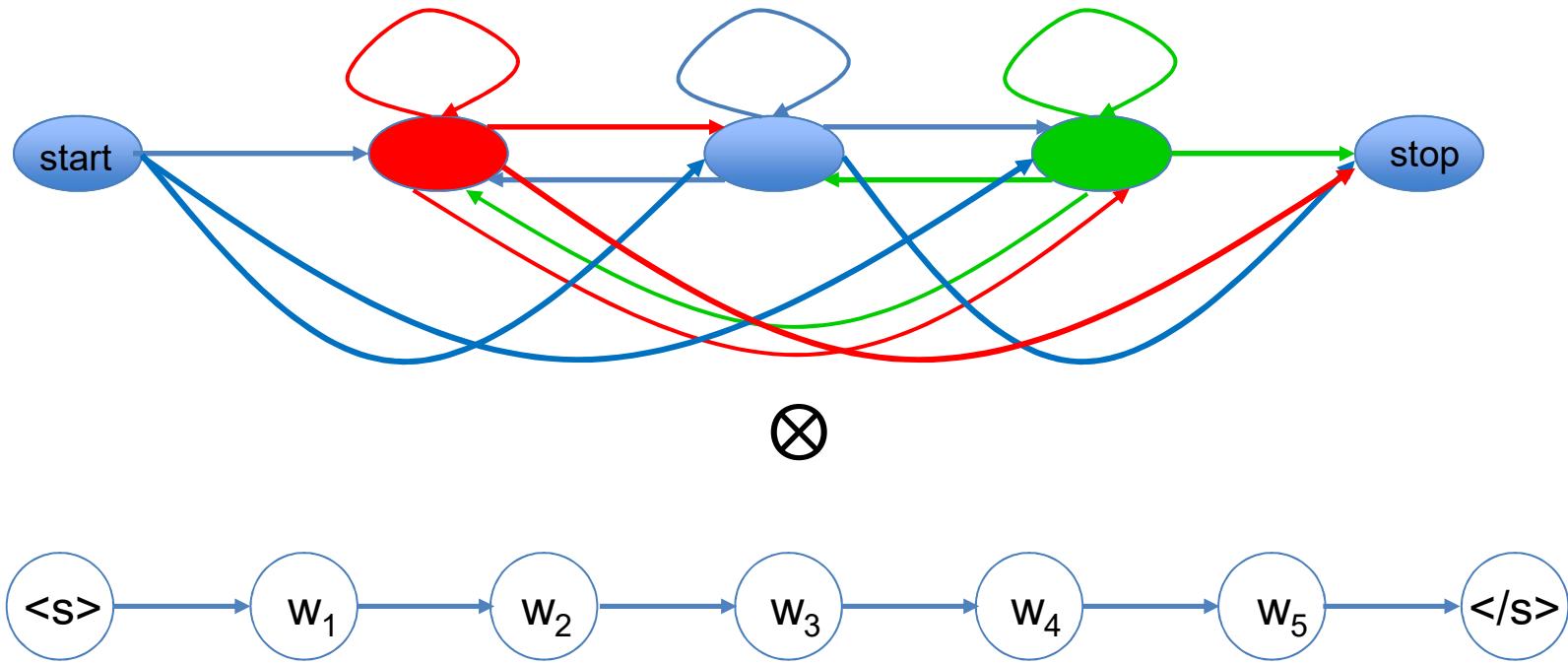
- At the same time, the *productions* from the state sequence must conform to the structure of the observation
 - I.e. w_i *must* be followed by w_{i+1} with probability 1

The graph view of the problem



- The set of all combination of states and words can be represented as a combined graph that conforms to the restrictions of *both* graphs
 - I.e. the composition of both graphs, which is a trellis..

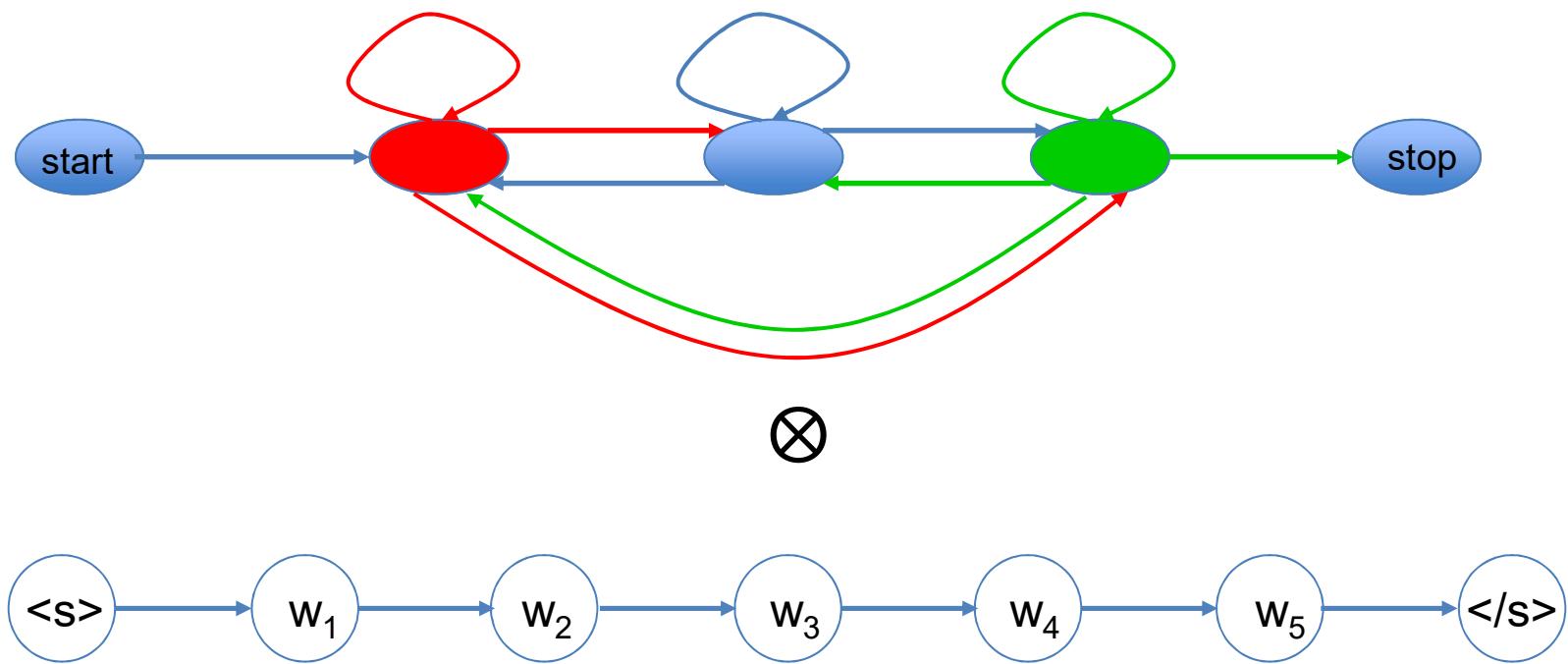
The graph view of the problem



- The set of all combination of states and words can be represented as a combined graph that conforms to the restrictions of *both* graphs
 - I.e. the composition of both graphs, which is a trellis..

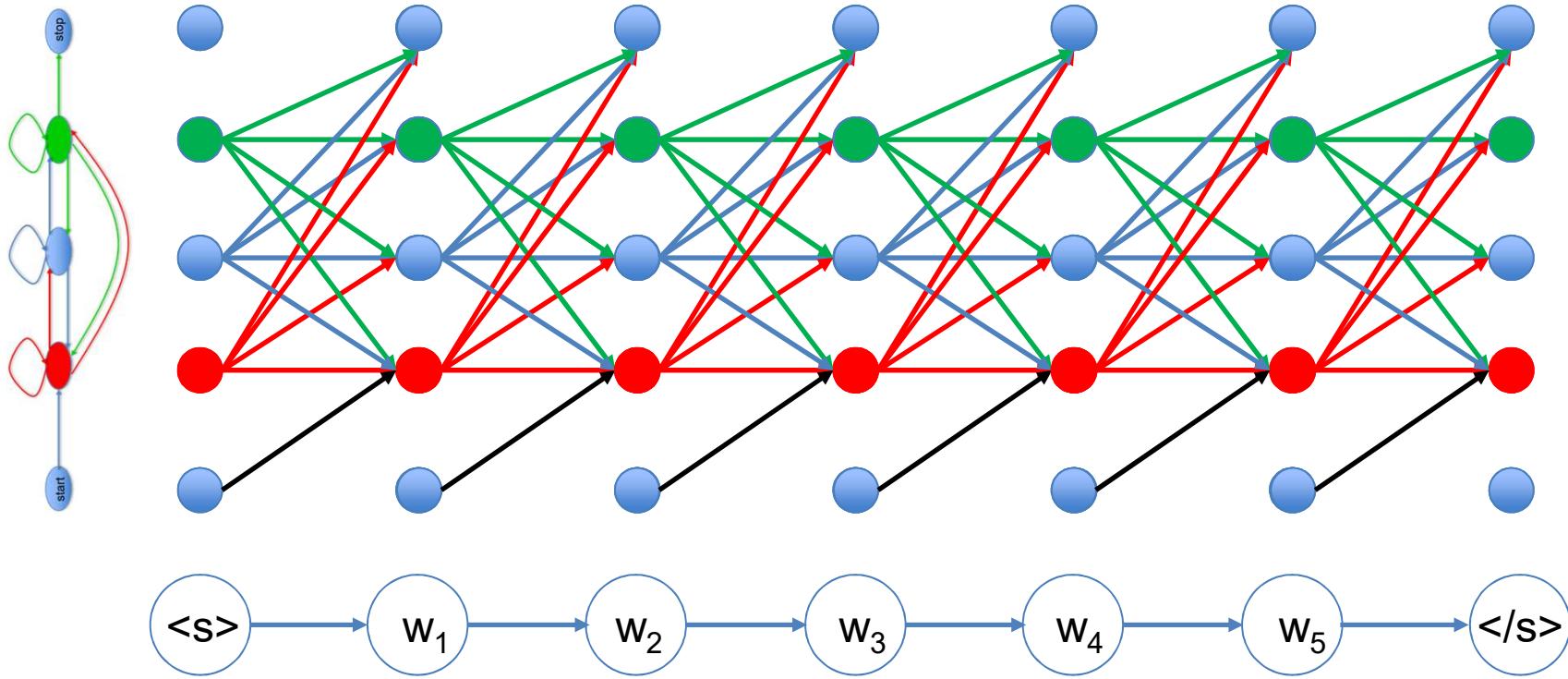
The graph view of the problem

Assuming a simpler model for clarity of illustration (first word must be from red state, last word must be from green state)



- The set of all combination of states and words can be represented as a combined graph that conforms to the restrictions of *both* graphs
 - I.e. the composition of both graphs, which is a trellis..

The graph view of the problem

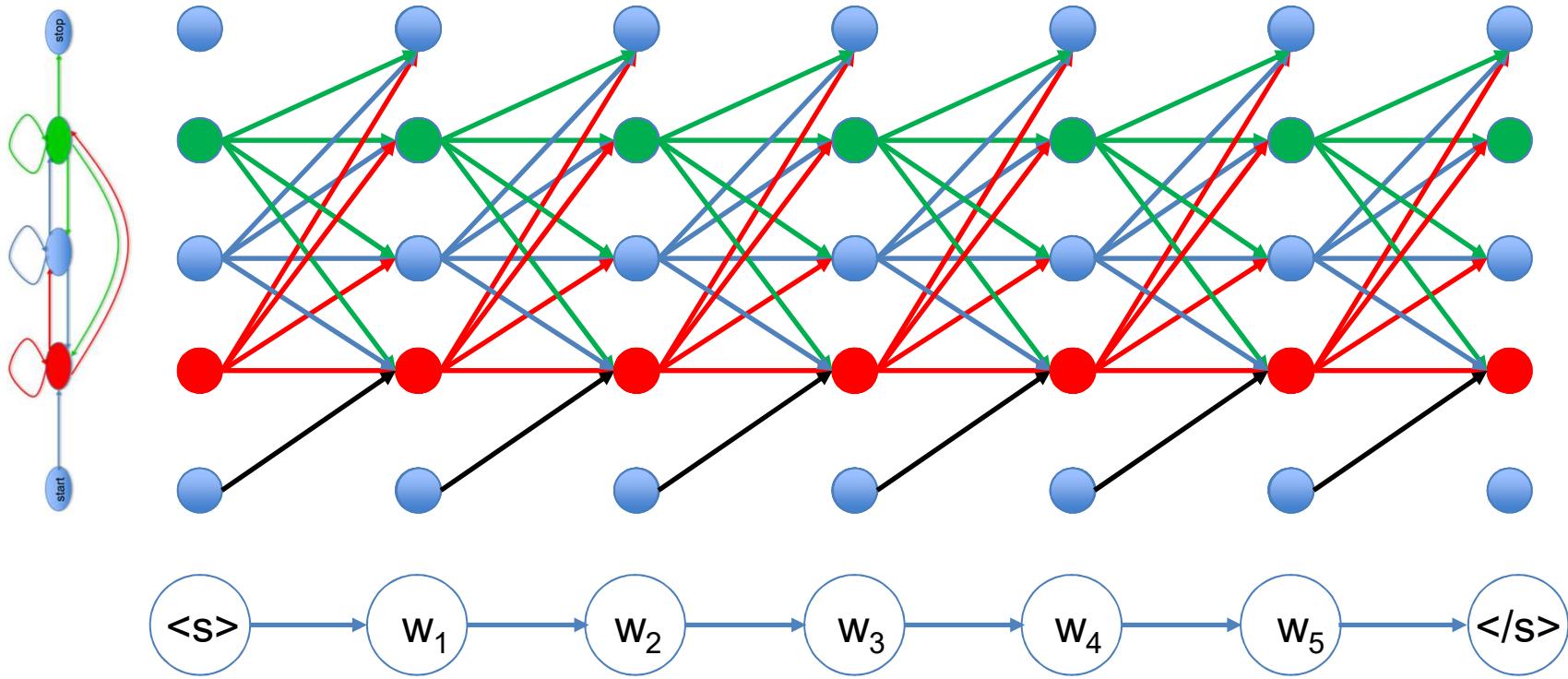


- The Trellis that composes the state graph and the observation graph
- Every state sequence through this trellis conforms to both, the Markov graph over states and the linear ordering of words

Probabilities on the Trellis

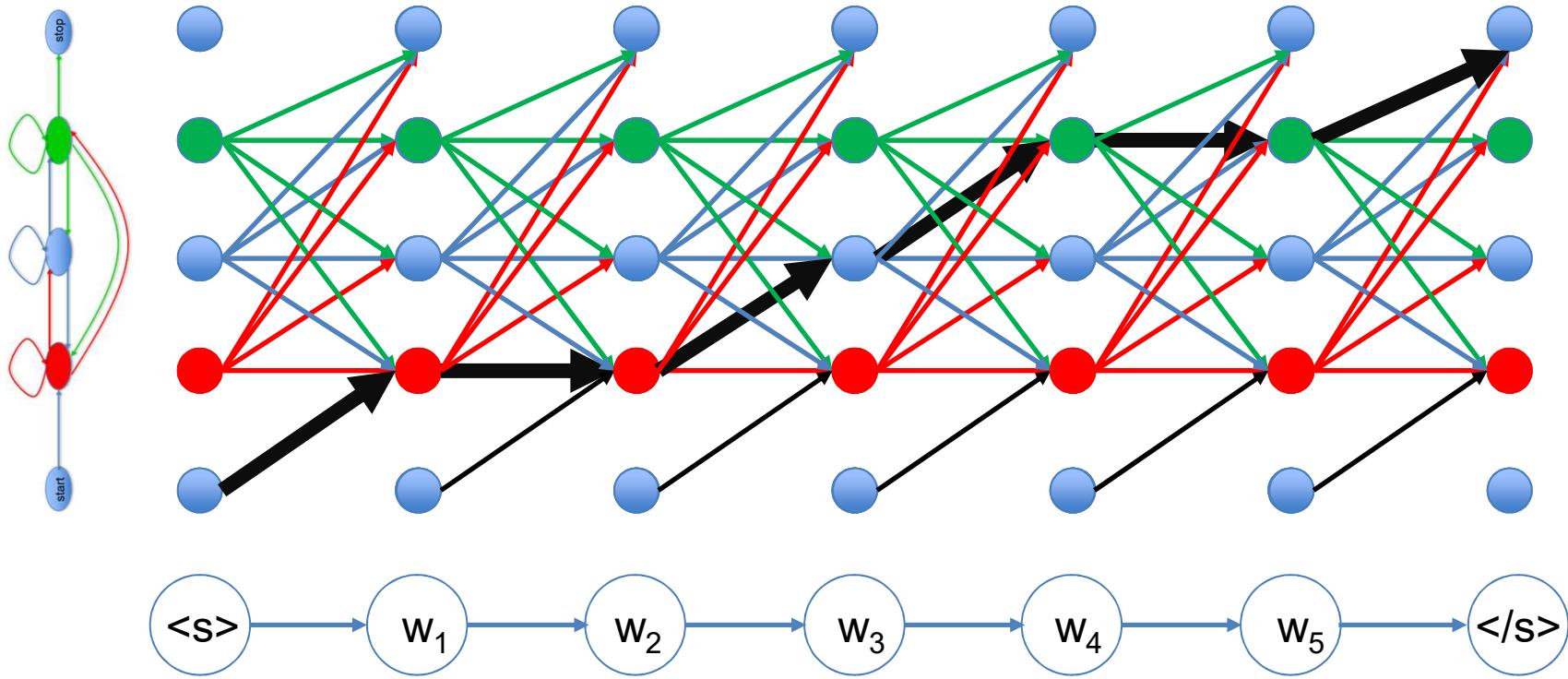
- The “score” for combining a state and a word is the probability of emitting that word from the state
- The “score” for an edge is the product of the probabilities associated with edges in both graphs
- The “score” for a path through the trellis is now obtained by *multiplying* component node and edge probabilities

The graph view of the problem



- Trellis: $\text{NodeScore}(s, w) = P(w|s)$
- Trellis $\text{Edgescore}((s_i, s_j), (w_k, w_l)) = P(s_j|s_i) P(w_l|w_k)$
 $= P(s_j|s_i)$ for $l = k + 1$; 0 otherwise

The graph view of the problem

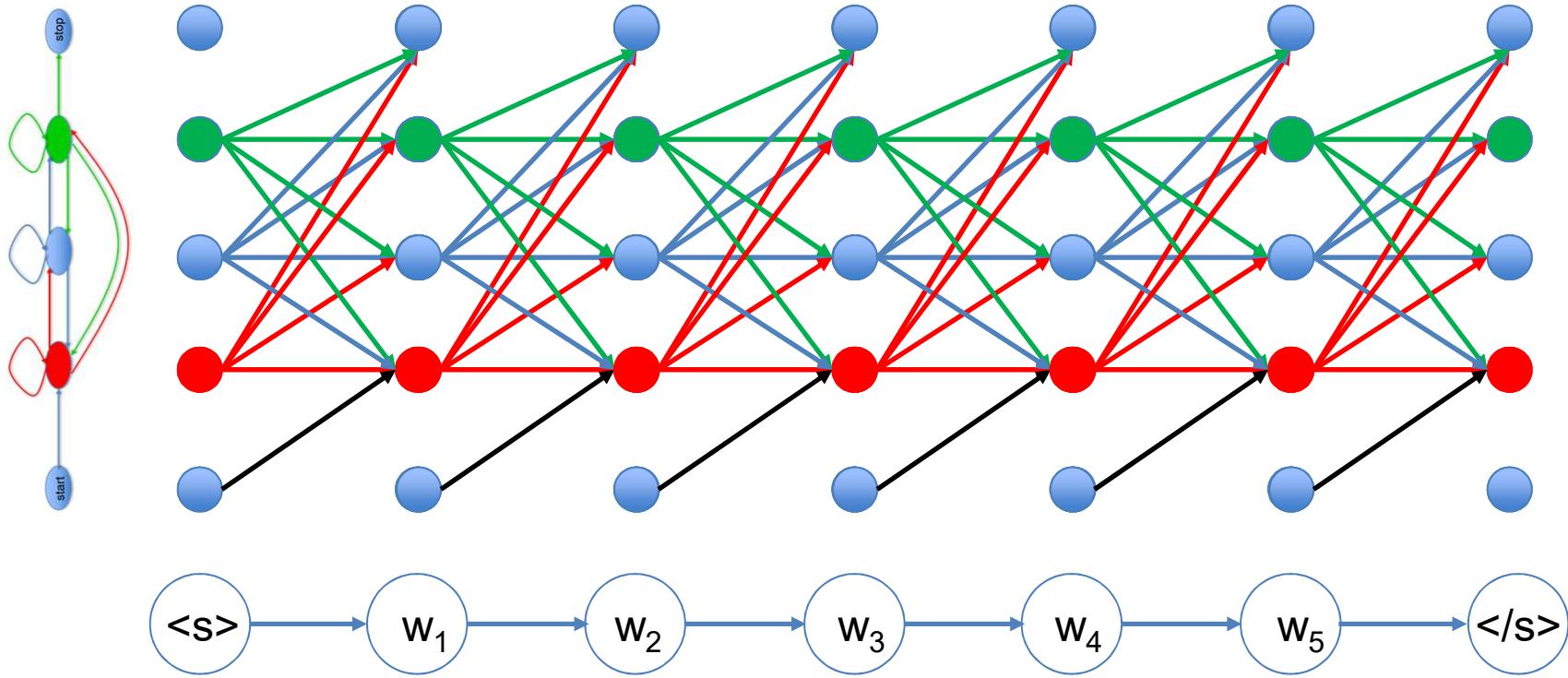


- The score associated with any path is trivially seen to be $P(start, < s >, s_1, w_1, s_2, w_2, \dots, stop, </s >)$, where s_i is the i-th state in the state sequence

Probabilities on the Trellis

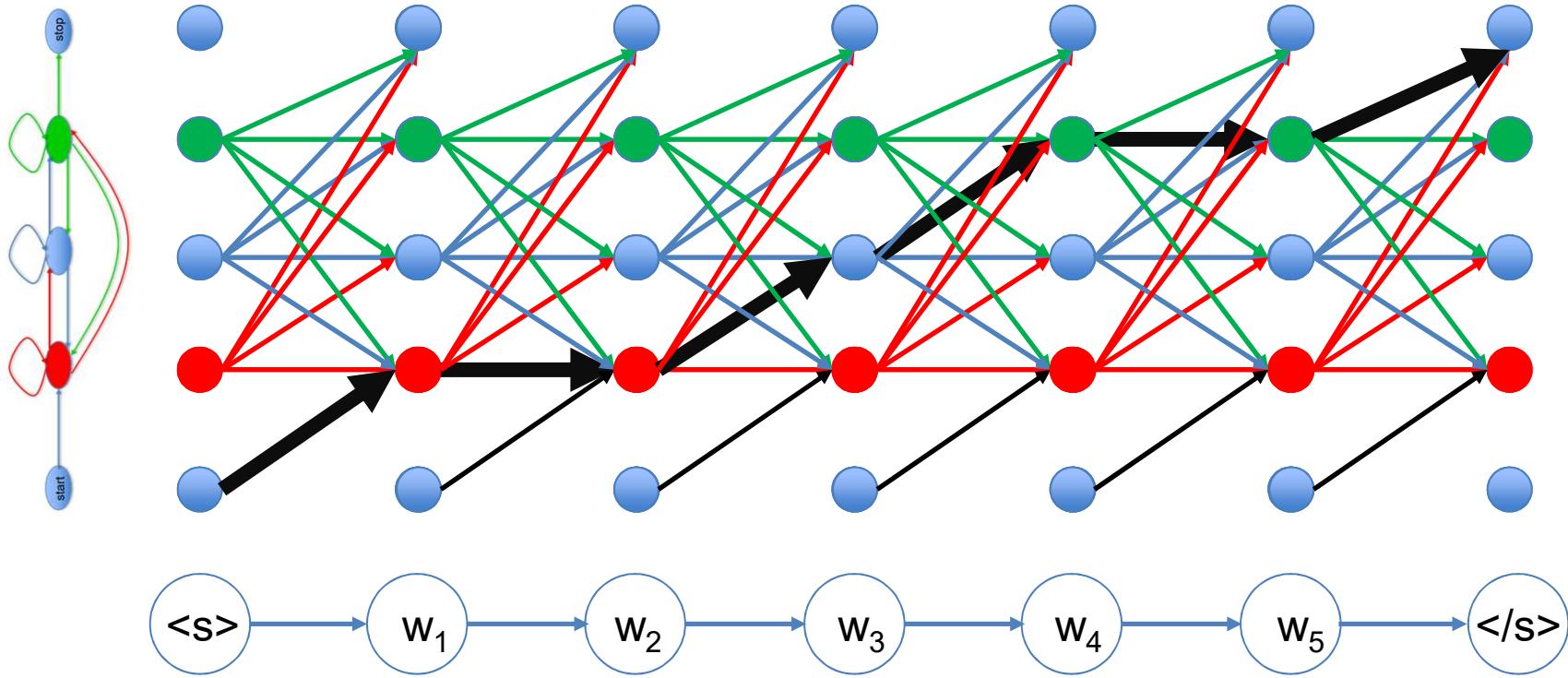
- Instead of probabilities, we will often work with *log* probabilities
- Instead of multiplying components along the paths, we can now add them as usual

The graph view of the problem



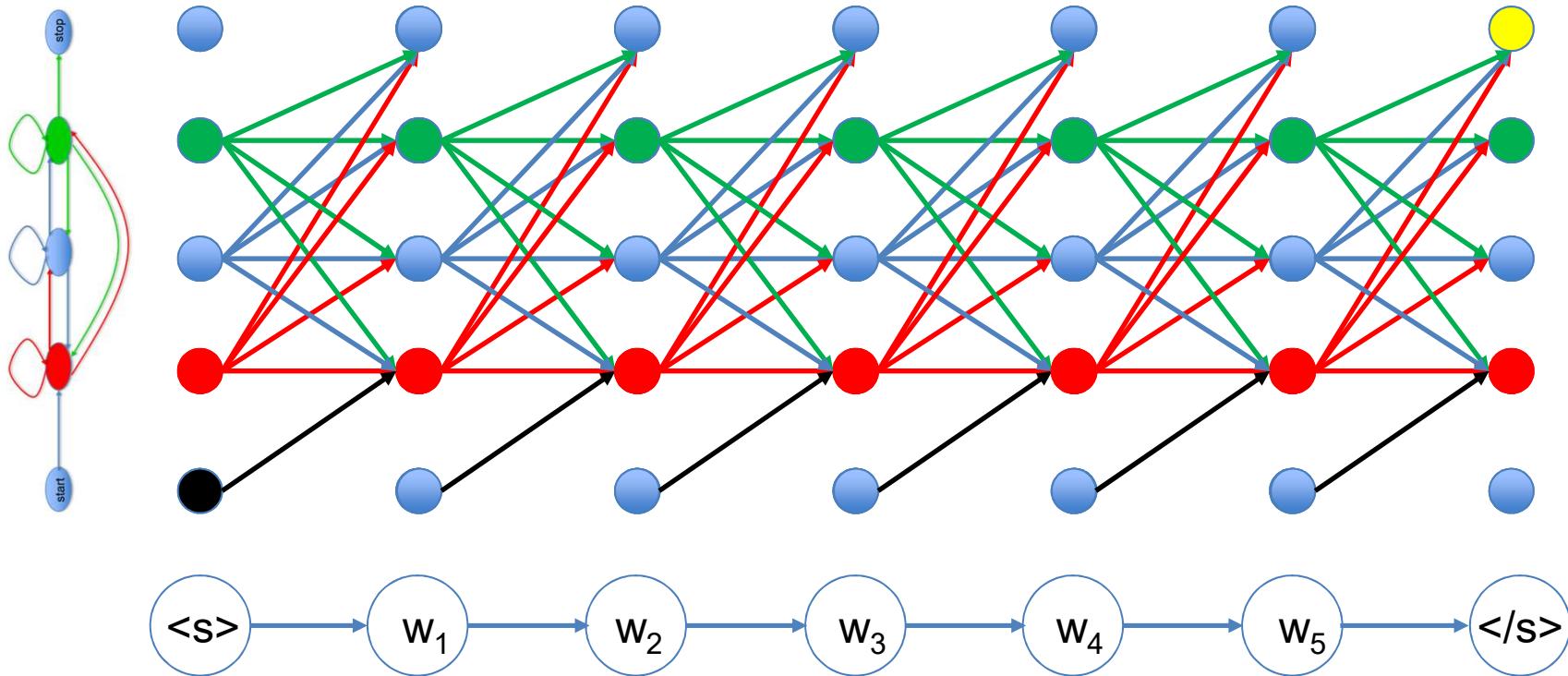
- $NodeScore(s, w) = \log P(w|s)$
- $Edgescore((s_i, s_j), (w_k, w_l)) = \begin{cases} \log P(s_j|s_i), & l = k + 1 \\ -\infty, & otherwise \end{cases}$

The graph view of the problem



- The score associated with any path is trivially seen to be $\log P(start, < s >, s_1, w_1, s_2, w_2, \dots, stop, </s >)$
- This is the sum of the node and edge scores along the path

Finding the state sequence



- Problem: Find the state sequence that best explains the word sequence
- Equivalent problem: Find the highest scoring path from the start (black) node to the final (yellow) node
- For this we can now use the Viterbi algorithm with one modification

Viterbi algorithm

- **Initialize:**

```
Score[1:M, 1:N] = -infinity  
Bestpredecessor[1:M, 1:N] = null
```

Note: scores instead of cost
(Score = log probability)

- **Algorithm:**

```
Score[1,1] = nodescore(node(1,1))  
for i = 2:M  
    for j = 1:N  
        BP = argmax_k(Score[i-1,k] + edgescore((i-1,k),(i,j)))  
        Score[i,j] = Score[i-1,BP] + edgescore((i-1,BP),(i,j))  
                    + nodescore(i,j)  
        Bestpredecessor[i,j] = BP
```

- **Final overall cost:**

```
BestScore = Score[M,N]
```

- **Actual sequence of states (from parent 1):**

```
State[M] = N  
for i = M downto 2  
    State[i-1] = Bestpredecessor(i, State[i])
```

Viterbi algorithm

- **Initialize:**

```
Score[1:M, 1:N] = -infinity  
Bestpredecessor[1:M, 1:N] = null
```

Note: scores instead of cost
(Score = log probability)

- **Algorithm:**

```
Score[1,1] = nodescore(node(1,1))  
for i = 2:M  
    for j = 1:N  
        BP = argmax_k(Score[i-1,k] + edgescore((i-1,k),(i,j)))  
        Score[i,j] = Score[i-1,BP] + edgescore((i-1,BP),(i,j))  
                    + nodescore(i,j)  
        Bestpredecessor[i,j] = BP
```

Note: argmax instead of argmin

- **Final overall cost:**

```
BestScore = Score[M,N]
```

- **Actual sequence of states (from parent 1):**

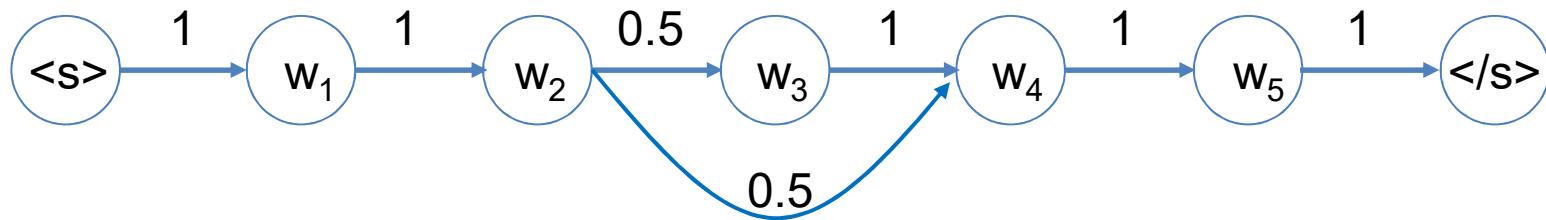
```
State[M] = N  
for i = M downto 2  
    State[i-1] = Bestpredecessor(i, State[i])
```

Generalizing the approach

$\langle s \rangle w_1 w_2 [w_3] w_4 w_5 \langle /s \rangle$

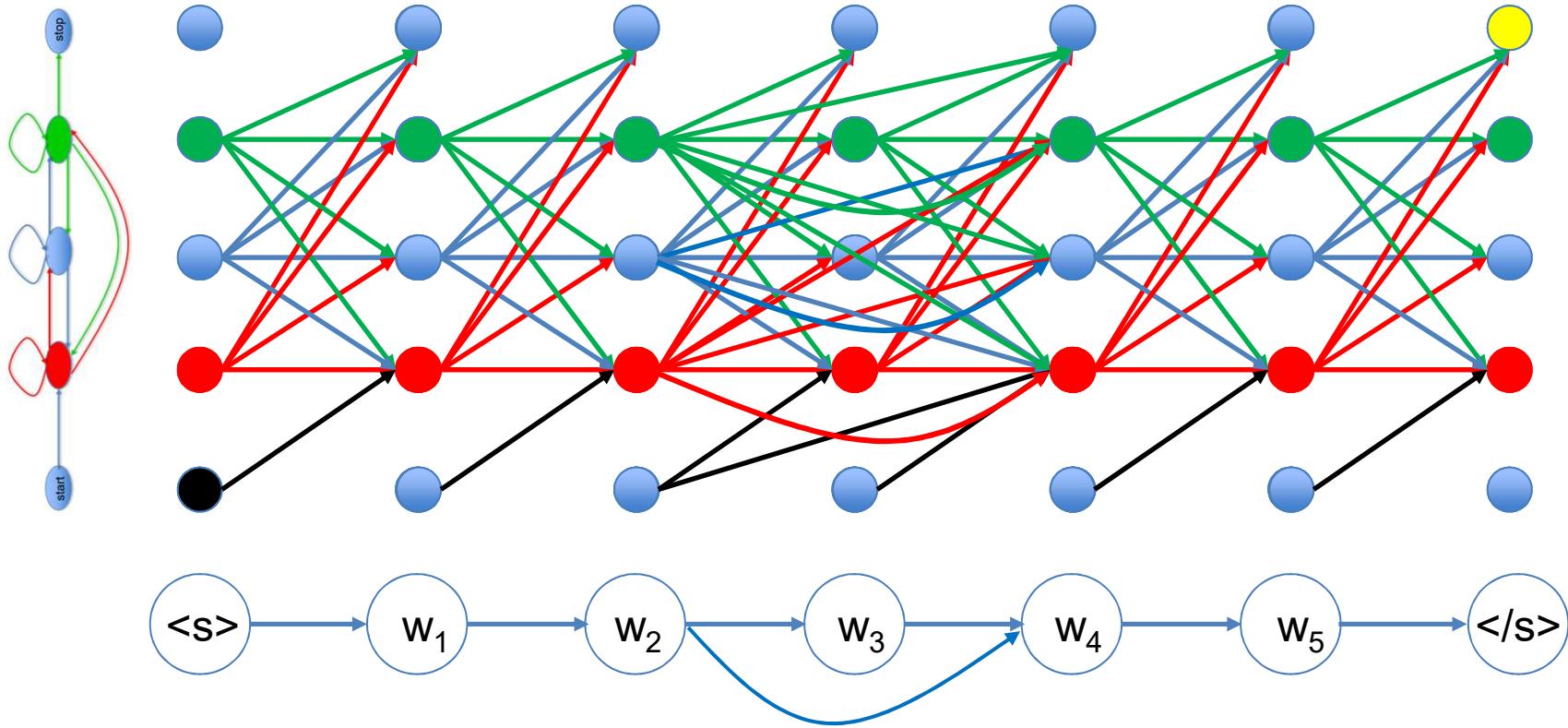
- Consider the case where the observed word sequence is uncertain
 - Uncertain whether w_3 was said or not
 - But the presence or absence of w_3 changes the interpretation of the sentence
 - How to find the most likely state sequence

The uncertain observation graph



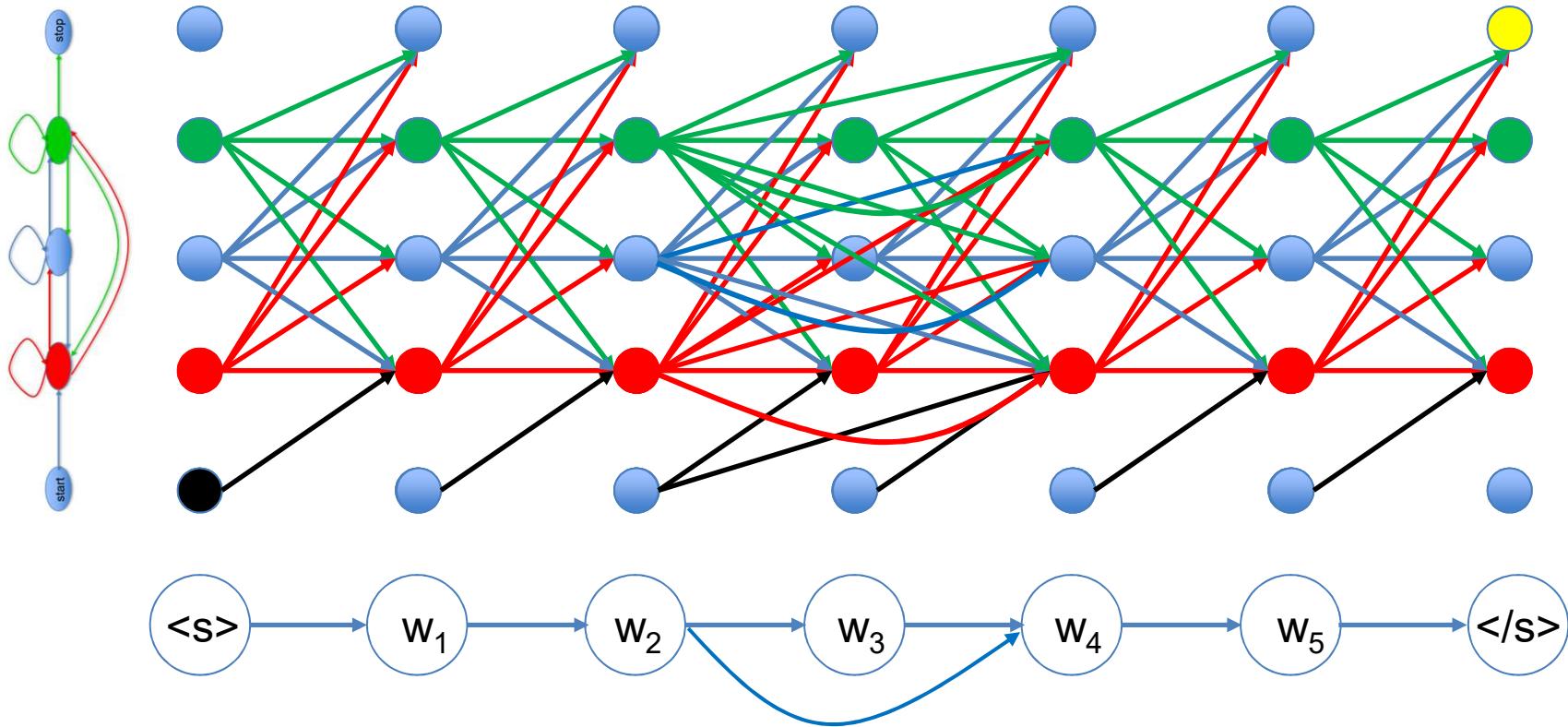
- The observation sequence can now be modeled by this modified graph
 - Note the probabilities
 - The 0.5 may be replaced by any other value indicative of our certainty in the occurrence of the word

The modified trellis



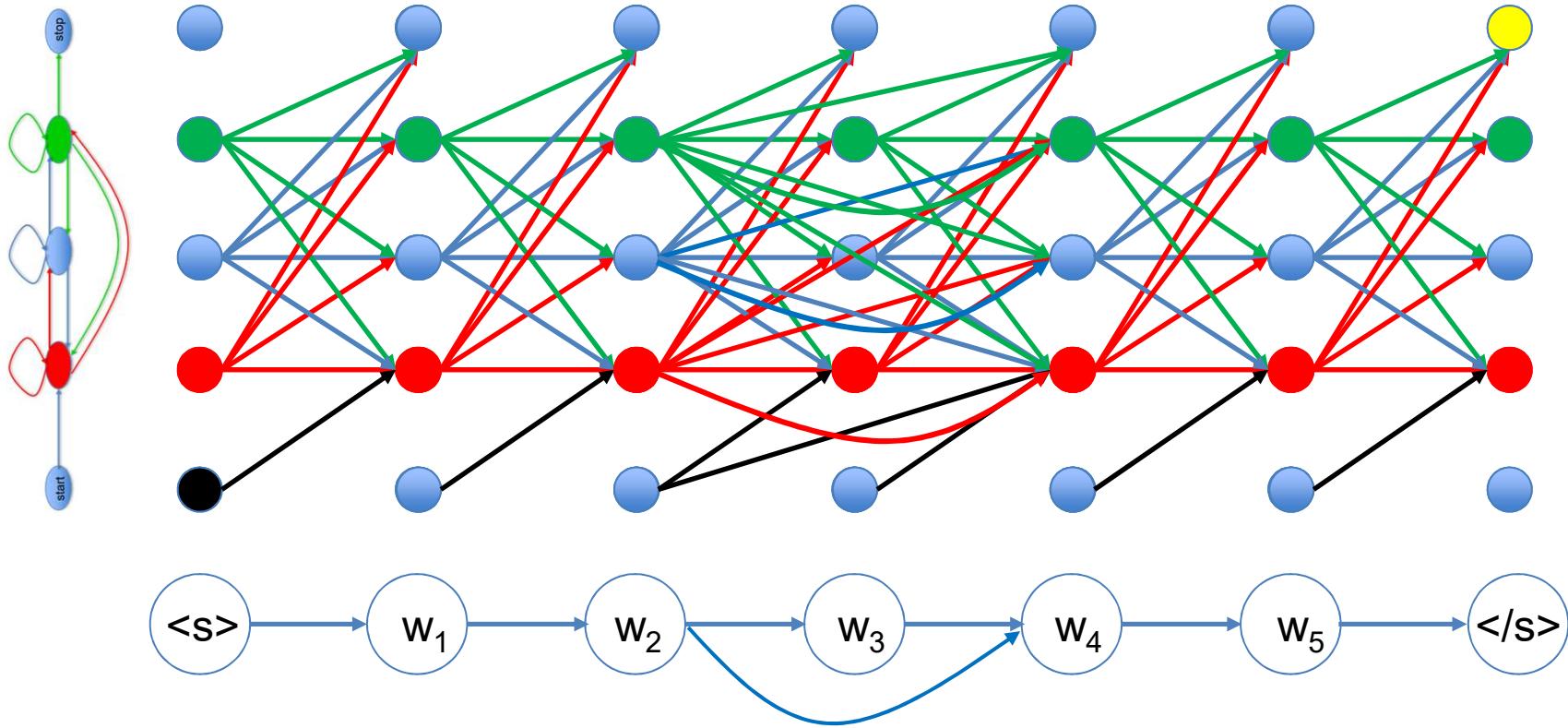
- Trellis obtained by composing Markov graph and observation graph
- Permits state sequences that skip the uncertain word

The modified trellis



- $NodeScore(s, w) = \log P(w|s)$
- $Edgescore((s_i, s_j), (w_k, w_l)) = \log P(s_j|s_i) + \log P(w_k|w_l)$
 - Note: $\log P(w_k|w_l) = -\infty$ for words that are not connected

The modified trellis



- *The Viterbi algorithm must be modified for this problem*
 - How?

Generalizing the approach

Spare him not , kill him **OR** Spare him , not kill him

- What is the word graph for this problem?

Lecture Outline

- ✓ Markov models
- 2. Hidden Markov models
- 3. Viterbi algorithm

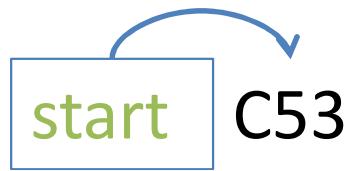
HIDDEN MARKOV MODELS

Hidden Markov Model

- A model over sequences of symbols, but there is missing information associated with each symbol: its “state.”
 - Assume a finite set of possible states, Λ .

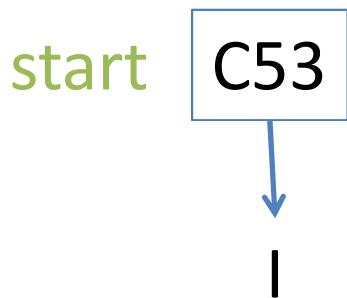
$$p(\text{start}, s_1, w_1, s_2, w_2, \dots, s_n, w_n \text{stop}) = \prod_{i=1}^{n+1} \eta(w_i \mid s_i) \times \gamma(s_i \mid s_{i-1})$$

- A *joint* model over the observable symbols and their hidden/latent/unknown classes.



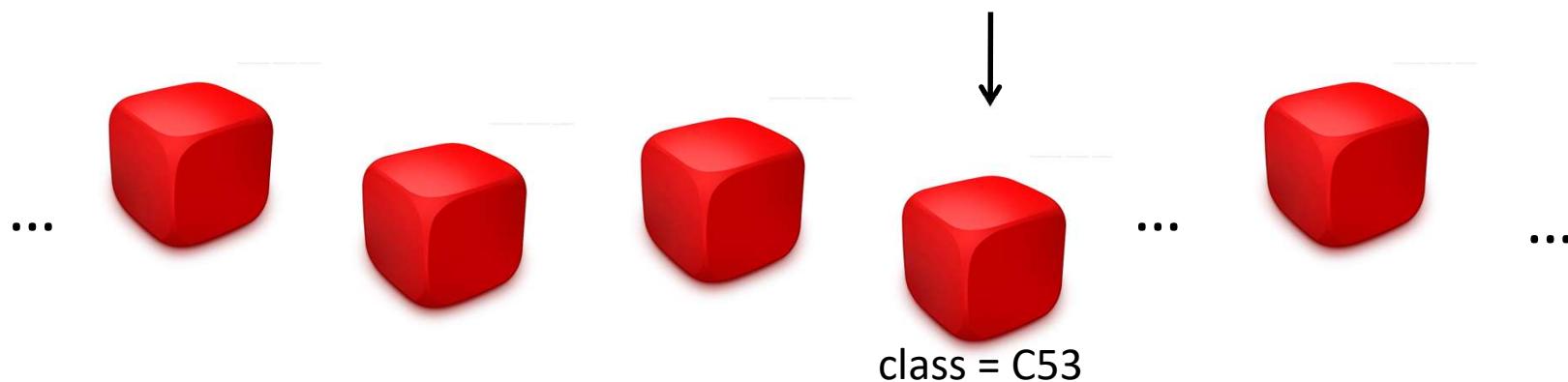
one “next class” die per class:





The only change to the class-based model is that now, the different word dice can *share words*!

one word die per class:





|

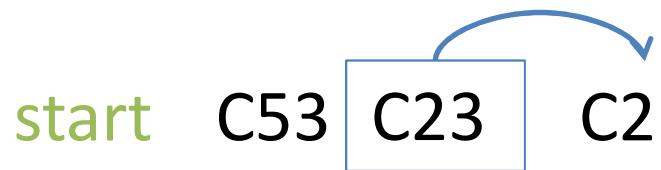
one “next class” die per class:



start C53 C23
I want

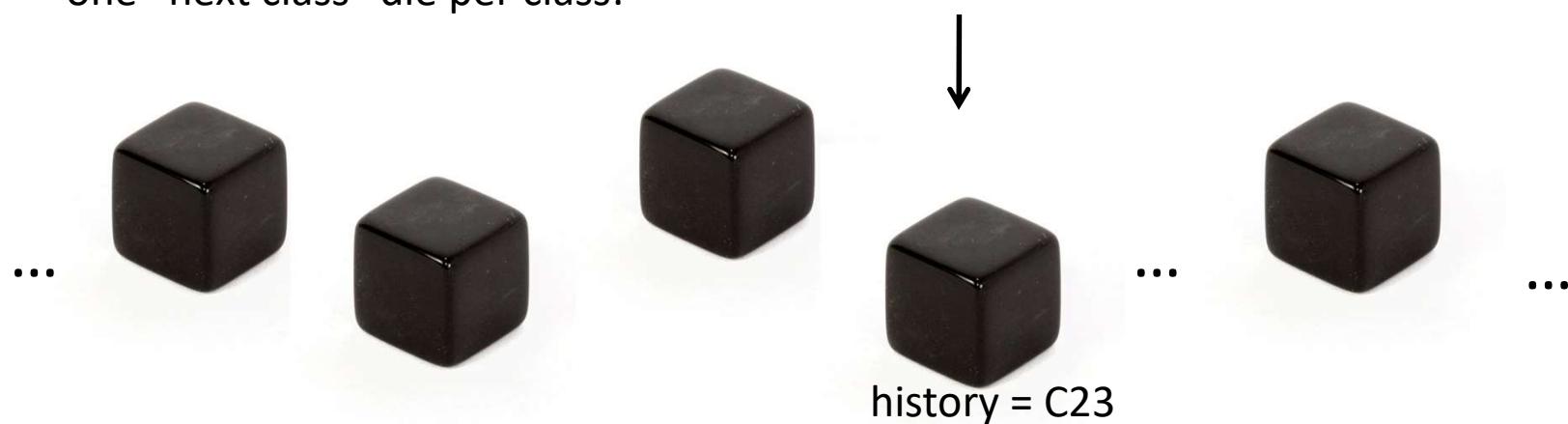
one word die per class:
...

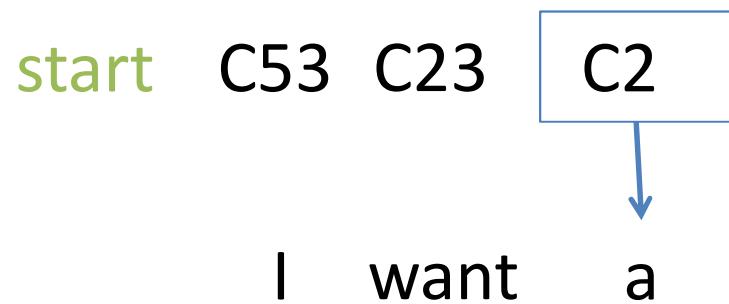
class = C23



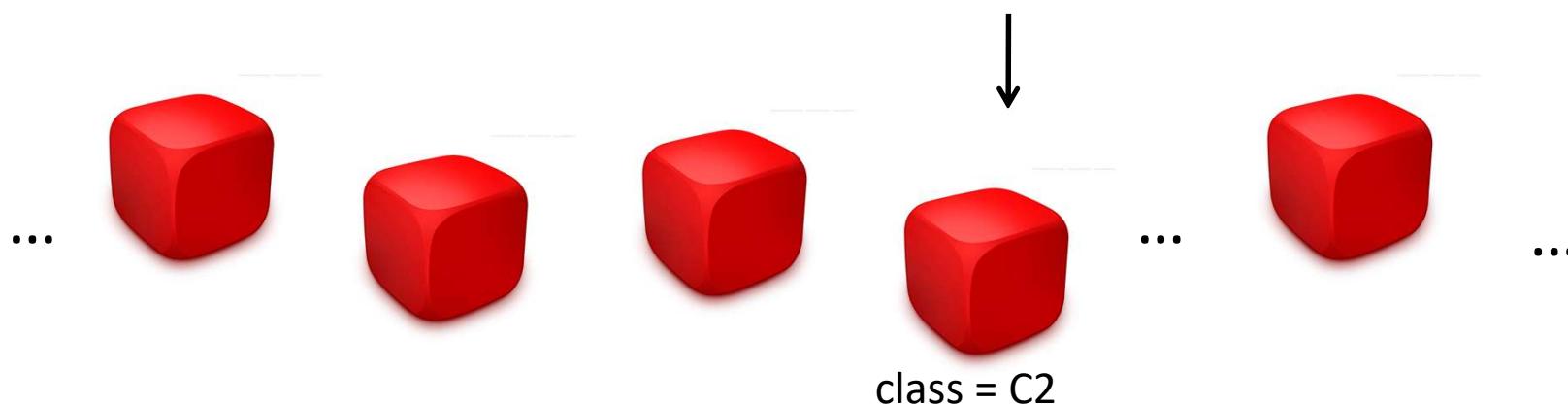
I want

one “next class” die per class:





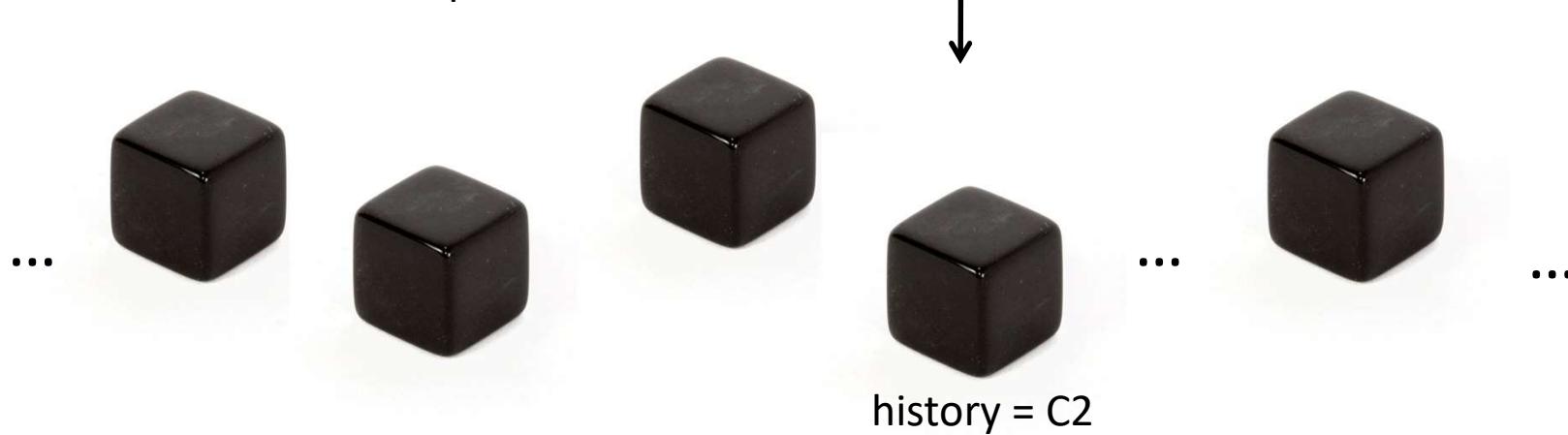
one word die per class:

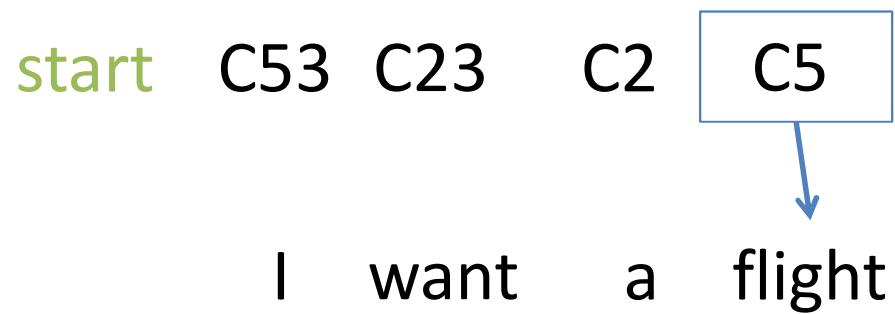




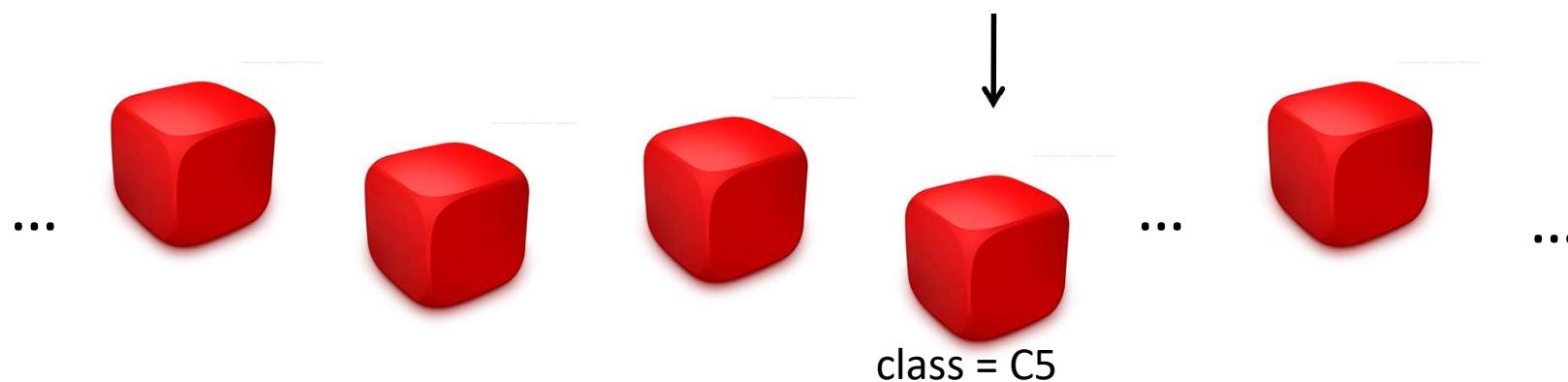
I want a

one “next class” die per class:





one word die per class:



Two Equivalent Stories

- First, as shown: transition, emit, transition, emit, transition, emit.



- Second:
 - Generate the sequence of transitions. Essentially, a Markov model on classes.
 - Stochastically replace each class with a word.



Uses of HMMs in NLP

- Part-of-speech tagging (Church, 1988; Brants, 2000)
- Named entity recognition (Bikel et al., 1999) and other information extraction tasks
- Text chunking and shallow parsing (Ramshaw and Marcus, 1995)
- Word alignment in parallel text (Vogel et al., 1996)
- Also popular in computational biology and central to speech recognition.

Part of Speech Tagging

After paying the medical bills , Frances was nearly broke .

RB VBG DT JJ NNS , NNP VBZ RB JJ .

- Adverb (RB)
- Verb (VBG, VBZ, and others)
- Determiner (DT)
- Adjective (JJ)
- Noun (NN, NNS, NNP, and others)
- Punctuation (., ,, and others)

Named Entity Recognition

With Commander Chris Ferguson at the helm ,

Atlantis touched down at Kennedy Space Center .

Named Entity Recognition

O B-person I-person I-person O O O O

With Commander Chris Ferguson at the helm ,

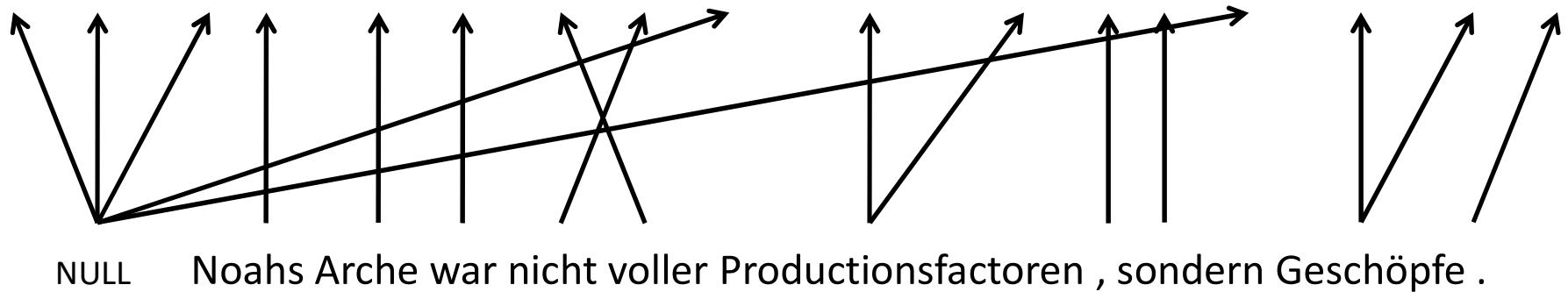
B-space-shuttle O O O B-place I-place I-place O

Atlantis touched down at Kennedy Space Center .

- What makes this hard?

Word Alignment

Mr. President , Noah's ark was filled not with production factors , but with living creatures.



Word Alignment

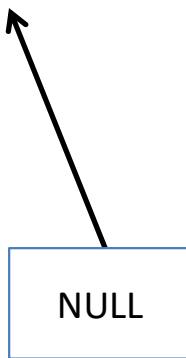
Mr. President , Noah's ark was filled not with production factors , but with living creatures.

NULL

Noahs Arche war nicht voller Productionsfactoren , sondern Geschöpfe .

Word Alignment

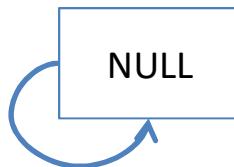
Mr. President , Noah's ark was filled not with production factors , but with living creatures.



Noahs Arche war nicht voller Productionsfactoren , sondern Geschöpfe .

Word Alignment

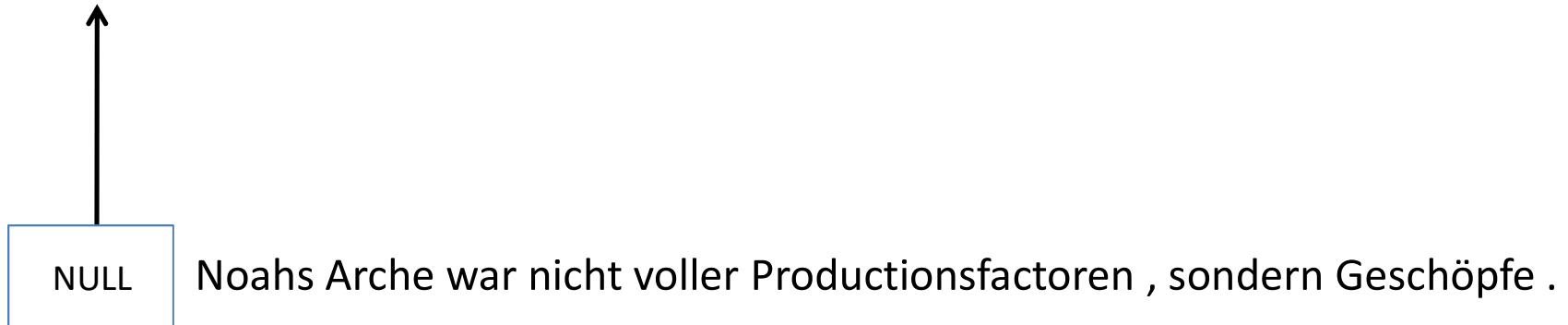
Mr. President , Noah's ark was filled not with production factors , but with living creatures.



Noahs Arche war nicht voller Productionsfactoren , sondern Geschöpfe .

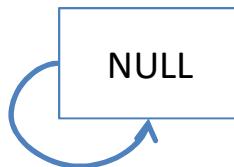
Word Alignment

Mr. President , Noah's ark was filled not with production factors , but with living creatures.



Word Alignment

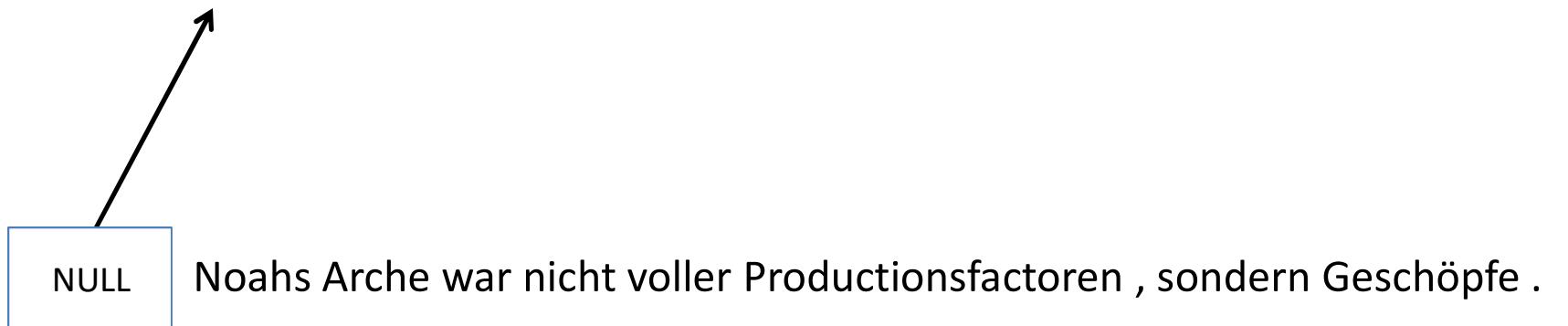
Mr. President , Noah's ark was filled not with production factors , but with living creatures.



Noahs Arche war nicht voller Productionsfactoren , sondern Geschöpfe .

Word Alignment

Mr. President , Noah's ark was filled not with production factors , but with living creatures.



Word Alignment

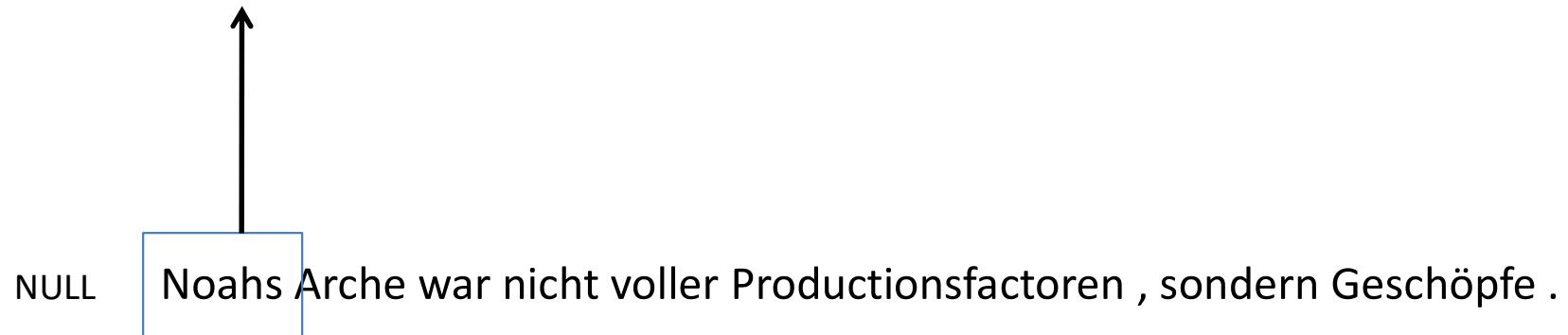
Mr. President , Noah's ark was filled not with production factors , but with living creatures.

NULL Noahs Arche war nicht voller Productionsfactoren , sondern Geschöpfe .



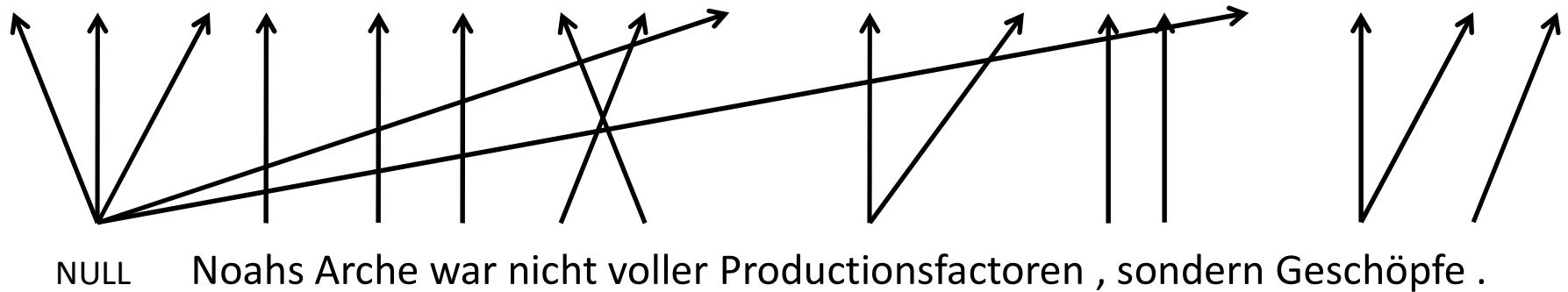
Word Alignment

Mr. President , Noah's ark was filled not with production factors , but with living creatures.



Word Alignment

Mr. President , Noah's ark was filled not with production factors , but with living creatures.



Hidden Markov Model

- A model over sequences of symbols, but there is missing information associated with each symbol: its “state.”
 - Assume a finite set of possible states, Λ .

$$p(\text{start}, s_1, w_1, s_2, w_2, \dots, s_n, w_n \text{stop}) = \prod_{i=1}^{n+1} \eta(w_i \mid s_i) \times \gamma(s_i \mid s_{i-1})$$

- A *joint* model over the observable symbols and their hidden/latent/unknown classes.

Lecture Outline

- ✓ Markov models
- ✓ Hidden Markov models
- 3. Viterbi algorithm

VITERBI ALGORITHM

Algorithms for HMMs

Given the HMM and a sequence:

1. The most probable state sequence?
2. The probability of the word sequence?
3. The probability distribution over states, for each word?
4. Minimum risk sequence

Given states and sequences, or just states:

5. The parameters of the HMM (γ and η)?

Problem 1:

Most Likely State Sequence

- Input: HMM (γ and η) and symbol sequence w .
- Output: $\arg \max_s p(s \mid w, \gamma, \eta)$
- Statistics view: maximum *a posteriori* inference
- Computational view: discrete, combinatorial optimization

Example

I suspect the present forecast is pessimistic .

CD JJ DT JJ NN NNS JJ .

NN NN JJ NN VB VBZ

NNP VB NN RB VBD

PRP VBP NNP VB VBN

VBP VBP VBP

4 4 5 5 5 2 1 1

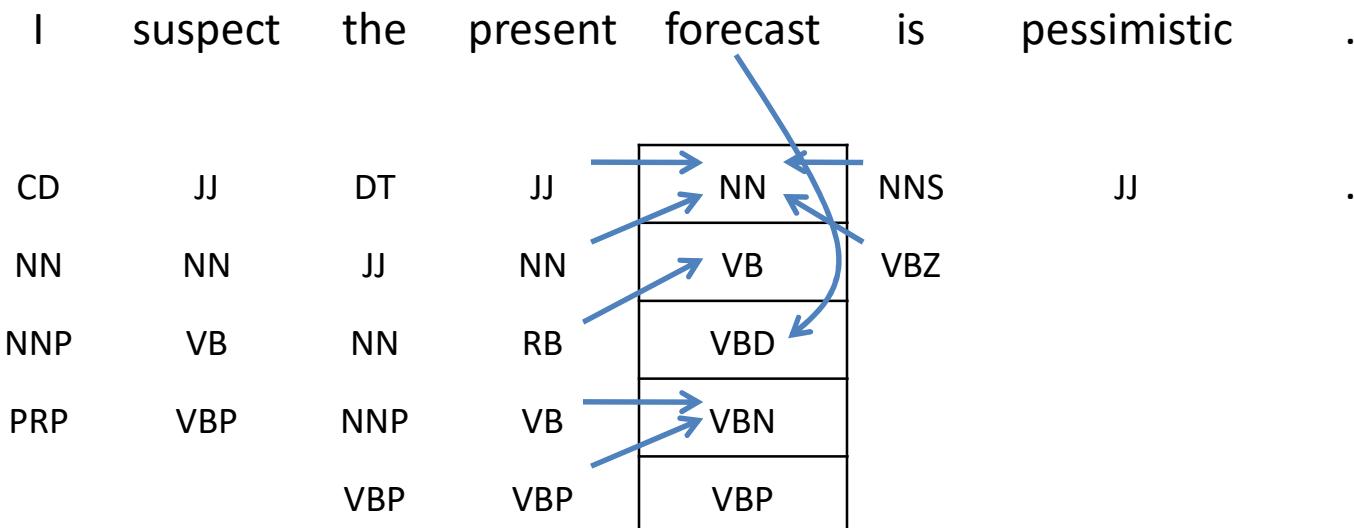
4,000 possible state sequences!

Naïve Solutions

- List all the possibilities in Λ^n .
 - Correct.
 - Inefficient.
- Work left to right and greedily pick the best s_i at each point, based on s_{i-1} and w_i .
 - Not correct; solution may not be equal to:
$$\arg \max_s p(s | w, \gamma, \eta)$$
 - But fast!

Interactions

- Each word's label depends on the word, and nearby labels.
- But given *adjacent* labels, others do not matter.



Base Case: Last Label

	start	w_1	w_2	w_3	...	w_{n-1}	$w_n \downarrow$	stop
σ_1								
σ_2								
σ_3							✓ →	
σ_4								
⋮								
$\sigma_{ \Lambda }$								

$$\text{score}_n(\sigma) = \gamma(\text{stop} \mid \sigma) \times \eta(w_n \mid \sigma) \times \gamma(\sigma \mid s_{n-1})$$

Of course, we do not actually know s_{n-1} !

Recurrence

- If I knew the score of every sequence $s_1 \dots s_{n-1}$, I could reason easily about s_n .
 - But my decision about s_n would only depend on s_{n-1} !
- So I really only need to know the score of the best sequence ending in **each** s_{n-1} .
- Think of that as some “precalculation” that happens before I think about s_n .

Recurrence

- Assume we have the scores for all prefixes of the current state sequence.
 - One score for each possible last-state of the prefix.

$$\text{score}_n(\sigma) = \gamma(\text{stop} \mid \sigma) \times \eta(w_n \mid \sigma) \times \max_{\sigma'} \gamma(\sigma \mid \sigma') \times \text{score}_{n-1}(\sigma')$$

Recurrence

- The recurrence “bottoms out” at start.
- This leads to a simple algorithm for calculating all the scores.

$$\text{score}_n(\sigma) = \gamma(\text{stop} \mid \sigma) \times \eta(w_n \mid \sigma) \times \max_{\sigma'} \gamma(\sigma \mid \sigma') \times \text{score}_{n-1}(\sigma')$$

$$\text{score}_{n-1}(\sigma) = \eta(w_{n-1} \mid \sigma) \times \max_{\sigma'} \gamma(\sigma \mid \sigma') \times \text{score}_{n-2}(\sigma')$$

$$\text{score}_{n-2}(\sigma) = \eta(w_{n-2} \mid \sigma) \times \max_{\sigma'} \gamma(\sigma \mid \sigma') \times \text{score}_{n-3}(\sigma')$$

$$\vdots \quad \vdots$$

$$\text{score}_1(\sigma) = \eta(w_1 \mid \sigma) \times \gamma(\sigma \mid \text{start})$$

Viterbi Algorithm (Scores Only)

- For every σ in Λ , let:

$$\text{score}_1(\sigma) = \eta(w_1 | \sigma) \times \gamma(\sigma | \text{start})$$

- For $i = 2$ to $n - 1$, for every σ in Λ :

$$\text{score}_i(\sigma) = \eta(w_i | \sigma) \times \max_{\sigma' \in \Lambda} \gamma(\sigma | \sigma') \times \text{score}_{i-1}(\sigma')$$

- For every σ in Λ :

$$\text{score}_n(\sigma) = \gamma(\text{stop} | \sigma) \times \eta(w_n | \sigma) \times \max_{\sigma' \in \Lambda} \gamma(\sigma | \sigma') \times \text{score}_{n-1}(\sigma')$$

- **Claim:** $\max_s p(s, w | \gamma, \eta) = \max_{\sigma \in \Lambda} \text{score}_n(\sigma)$

Exploiting Distributivity

$$\begin{aligned}
\max_{\sigma \in \Lambda} \text{score}_n(\sigma) &= \max_{\sigma \in \Lambda} \gamma(\text{stop} \mid \sigma) \times \eta(w_n \mid \sigma) \times \max_{\sigma' \in \Lambda} \gamma(\sigma \mid \sigma') \times \text{score}_{n-1}(\sigma') \\
&= \max_{\sigma \in \Lambda} \gamma(\text{stop} \mid \sigma) \times \eta(w_n \mid \sigma) \times \max_{\sigma' \in \Lambda} \gamma(\sigma \mid \sigma') \\
&\quad \times \eta(w_{n-1} \mid \sigma') \times \max_{\sigma'' \in \Lambda} \gamma(\sigma' \mid \sigma'') \times \text{score}_{n-2}(\sigma'') \\
&= \max_{\sigma \in \Lambda} \gamma(\text{stop} \mid \sigma) \times \eta(w_n \mid \sigma) \times \max_{\sigma' \in \Lambda} \gamma(\sigma \mid \sigma') \\
&\quad \times \eta(w_{n-1} \mid \sigma') \times \max_{\sigma'' \in \Lambda} \gamma(\sigma' \mid \sigma'') \\
&\quad \times \eta(w_{n-2} \mid \sigma'') \times \max_{\sigma''' \in \Lambda} \gamma(\sigma'' \mid \sigma''') \times \text{score}_{n-3}(\sigma''') \\
&= \max_{\sigma, \sigma', \sigma'', \sigma'''} \gamma(\text{stop} \mid \sigma) \times \eta(w_n \mid \sigma) \times \gamma(\sigma \mid \sigma') \\
&\quad \times \eta(w_{n-1} \mid \sigma') \times \gamma(\sigma' \mid \sigma'') \\
&\quad \times \eta(w_{n-2} \mid \sigma'') \times \gamma(\sigma'' \mid \sigma''') \times \text{score}_{n-3}(\sigma''') \\
&= \max_{\mathbf{s} \in \Lambda^n} \prod_{i=1}^{n+1} \gamma(s_i \mid s_{i-1}) \times \eta(w_i \mid s_i)
\end{aligned}$$

$$\max_{\mathbf{s}} p(\mathbf{s}, \mathbf{w} \mid \boldsymbol{\gamma}, \boldsymbol{\eta}) = \max_{\sigma \in \Lambda} \text{score}_n(\sigma)$$

I suspect the present forecast is pessimistic .

CD	3E-7							
DT			3E-8					
JJ		1E-9	1E-12	3E-12			7E-23	
NN	4E-6	2E-10	1E-13	6E-13	4E-16			
NNP	1E-5		4E-13					
NNS						1E-21		
PRP	4E-3							
RB				2E-14				
VB		6E-9		3E-15	2E-19			
VBD					6E-18			
VBN					4E-18			
VBP		5E-7	4E-14	4E-15	9E-19			

Not Quite There

- As described, this algorithm only lets us calculate the *probability* of the best label sequence.
- It does not recover the best sequence!

Understanding the Scores

- $\text{score}_i(\sigma)$ is the score of the best sequence labeling up through w_i , ignoring what comes later.

$$\text{score}_i(\sigma) = \max_{s_1, \dots, s_{i-1}} p(s_1, w_1, s_2, w_2, \dots, s_i = \sigma, w_i)$$

- Similar trick as before: if I know what s_{i+1} is, then I can use the scores to choose s_i .
- Solution: keep backpointers.

I suspect the present forecast is pessimistic .

CD	3E-7								
DT			3E-8						
JJ		1E-9		1E-12	3E-12				7E-23
NN	4E-6	2E-10		1E-13	6E-13	4E-16			
NNP	1E-5			4E-13					
NNS							1E-21		
PRP	4E-3								
RB				2E-14					
VB		6E-9			3E-15	2E-19			
VBD					6E-18				
VBN					4E-18				
VBP		5E-7	4E-14	4E-15	9E-19				

I suspect the present forecast is pessimistic .

CD	3E-7								
DT			3E-8						
JJ		1E-9		1E-12	3E-12				7E-23
NN	4E-6	2E-10		1E-13	6E-13	4E-16			
NNP	1E-5			4E-13					
NNS							1E-21		
PRP	4E-3								
RB					2E-14				
VB		6E-9			3E-15	2E-19			
VBD						6E-18			
VBN						4E-18			
VBP		5E-7	4E-14	4E-15	9E-19				

Viterbi Algorithm

- For every σ in Λ , let:

$$\text{score}_1(\sigma) = \eta(w_1 | \sigma) \times \gamma(\sigma | \text{start})$$

- For $i = 2$ to $n - 1$, for every σ in Λ :

$$\text{score}_i(\sigma) = \eta(w_i | \sigma) \times \max_{\sigma' \in \Lambda} \gamma(\sigma | \sigma') \times \text{score}_{i-1}(\sigma')$$

$$\text{bp}_i(\sigma) = \arg \max_{\sigma' \in \Lambda} \gamma(\sigma | \sigma') \times \text{score}_{i-1}(\sigma')$$

- For every σ in Λ :

$$\text{score}_n(\sigma) = \gamma(\text{stop} | \sigma) \times \eta(w_n | \sigma) \times \max_{\sigma' \in \Lambda} \gamma(\sigma | \sigma') \times \text{score}_{n-1}(\sigma')$$

$$\text{bp}_n(\sigma) = \arg \max_{\sigma' \in \Lambda} \gamma(\sigma | \sigma') \times \text{score}_{n-1}(\sigma')$$

Viterbi Algorithm: Backtrace

- After calculating all score and bp values, start by choosing s_n to maximize score_n .
- Then let $s_{n-1} = \text{bp}_n(s_n)$.
- In general, $s_{i-1} = \text{bp}_i(s_i)$.

Another Example

	time	flies	like	an	arrow	.
DT				10e-15	6e-21	
IN			8e-13		1e-19	
JJ			6e-14		2e-16	
NN	2e-4				3e-16	
NNP					1e-16	
VB	2e-7		1e-14		1e-19	
VBP			8e-16		4e-19	
VBZ		2e-9			3e-18	
.					1e-21	3e-17
,				4e-20	5e-22	

Another Example

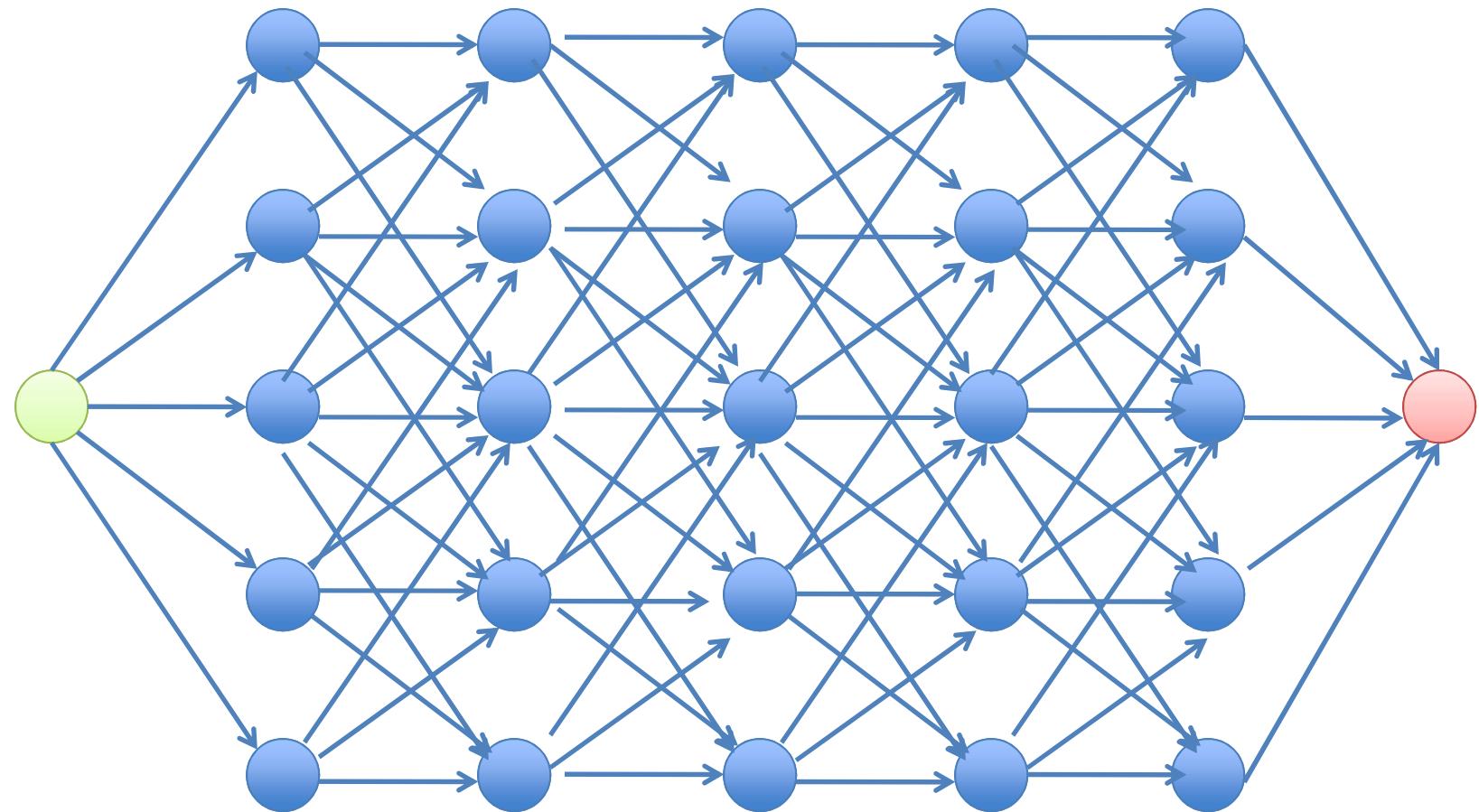
	time	flies	like	an	arrow	.
DT				10e-15	6e-21	
IN			8e-13		1e-19	
JJ			6e-14		2e-16	
NN	2e-4				3e-16	
NNP					1e-16	
VB	2e-7		1e-14		1e-19	
VBP			8e-16		4e-19	
VBZ		2e-9			3e-18	
.					1e-21	3e-17
,				4e-20	5e-22	

The diagram illustrates the mapping of words to rows in the table. Blue arrows point from the words 'time', 'flies', 'like', 'an', 'arrow', and '.' to their respective rows. The word 'time' points to the first row (DT). The word 'flies' points to the second row (IN). The word 'like' points to the third row (JJ). The word 'an' points to the fourth row (NN). The word 'arrow' points to the fifth row (NNP). The word '.' points to the sixth row (VB). The word ',' points to the seventh row (VBP).

General Idea: Dynamic Programming

- Use a table data structure to store partial quantities that will be reused many times.
 - Optimal substructure: best solution to a problem relies on best solutions to its (similar-looking) subproblems.
 - Overlapping subproblems: reuse a small number of quantities many times
- Examples: Viterbi, minimum Levenshtein distance, Dijkstra's shortest path algorithm, ...

A Different View: Best Path



Asymptotic Analysis

Memory:

- The table is $n \times |\Lambda|$.

Runtime:

- Each cell in the table requires $O(|\Lambda|)$ operations.
- Total runtime is $O(n|\Lambda|^2)$.

Lecture Outline

- ✓ Markov models
- ✓ Hidden Markov models
- ✓ Viterbi algorithm