Lecture 6:
Sequence alignment

Torgeir R. Hvidsten

Professor
Norwegian University of Life Sciences

Guest lecturer
Umeå Plant Science Centre
Computational Life Science Cluster (CLiC)

This lecture

- Dynamic programming and sequence alignment
- Scoring methods
- Multiple alignments, profiles
- Hidden Markov models

DNA sequence comparison: First success story

- In 1984 Russell Doolittle and colleagues found similarities between a cancer-causing gene and a normal growth factor (PDGF) gene using a database search
- Finding sequence similarities with genes of known function is a common approach to infer the function of a newly sequenced gene

Longest common subsequence (LCS) – alignment without mismatches

i coords:	0	0	1	2	3	4	5	5	6	6	7
elements of ***v***		**–**	**T**	**G**	**C**	**A**	**T**	**–**	**A**	**–**	**C**
elements of ***w***		**A**	**T**	**–**	**C**	**–**	**T**	**G**	**A**	**T**	**C**
j coords:	0	1	2	2	3	3	4	5	6	7	8

Matches shown in red

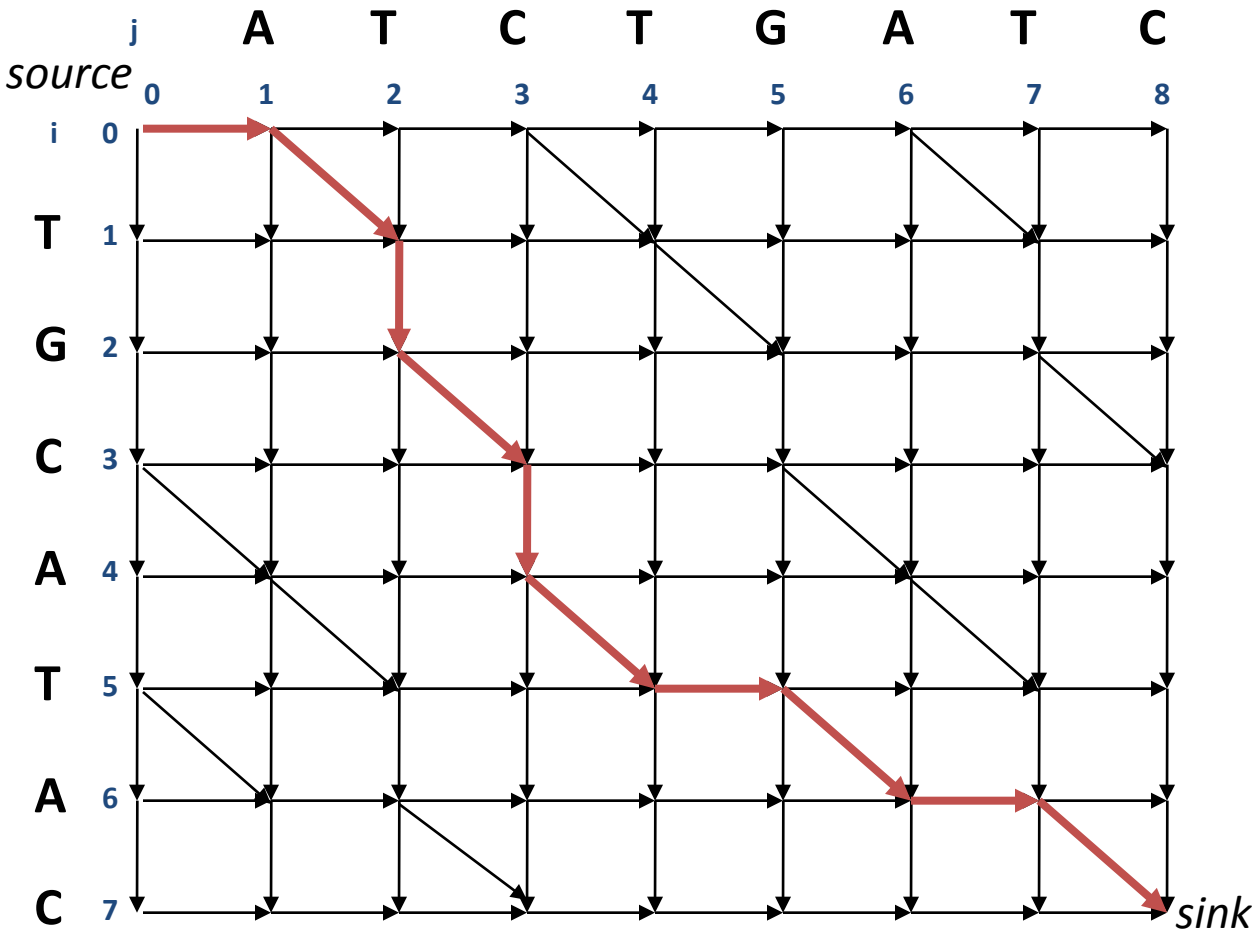
positions in ***v*** : 1 < 3 < 5 < 6 < 7

positions in ***w*** : 2 < 3 < 4 < 6 < 8

TCTAC is a common subsequence of ***v*** and ***w***

Every common subsequence is a path in a 2-D grid

Edit graph for the longest common substring (LCS) problem

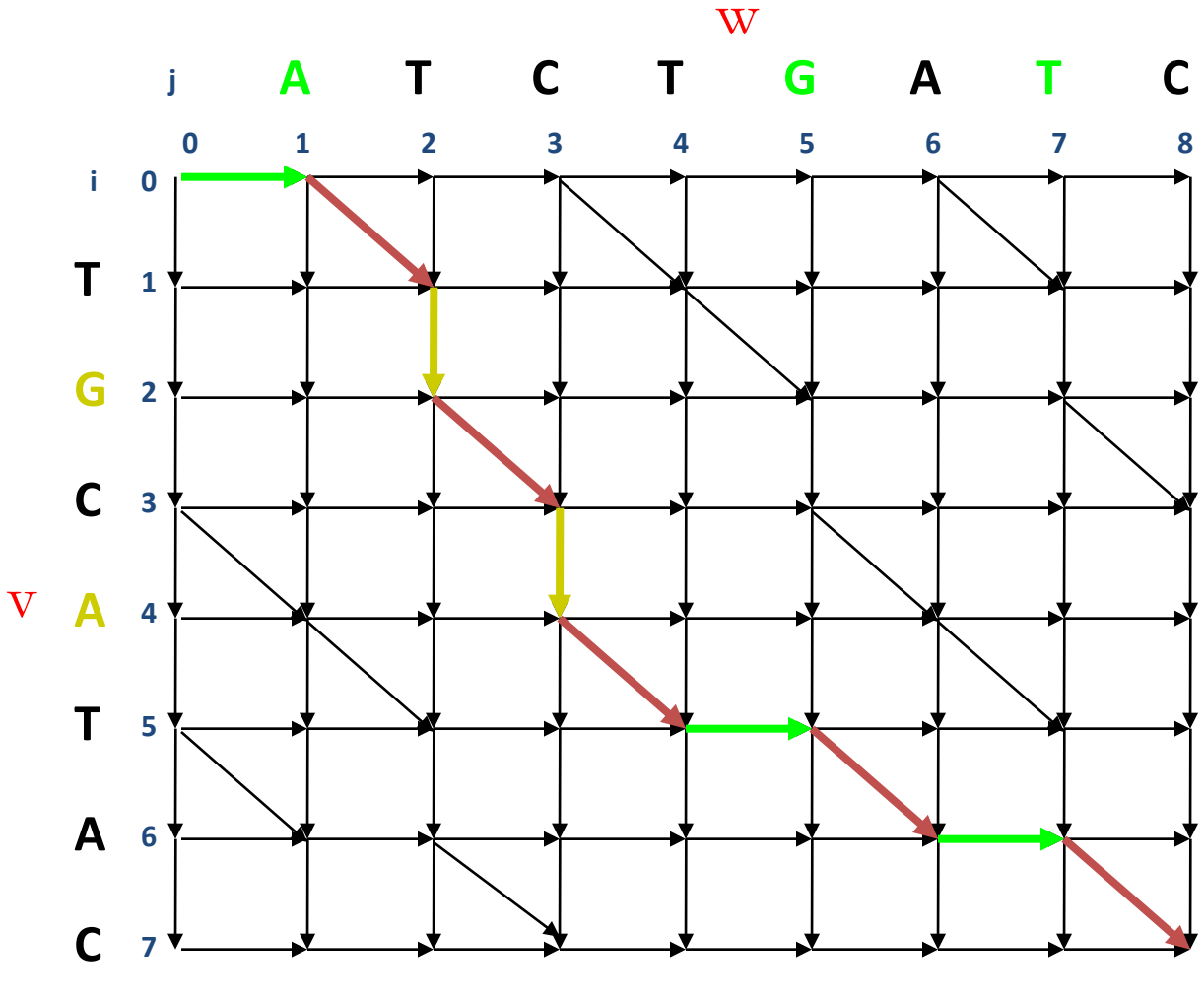


Every path from source to sink is a common subsequence (CS)

Every diagonal edge adds an extra element to the CS

LCS Problem: Find the path with the maximum number of diagonal edges

Edit graph for the LCS problem



Deletion

Matches

Insertion

-TGCAT-A-C
AT-C-TGATC

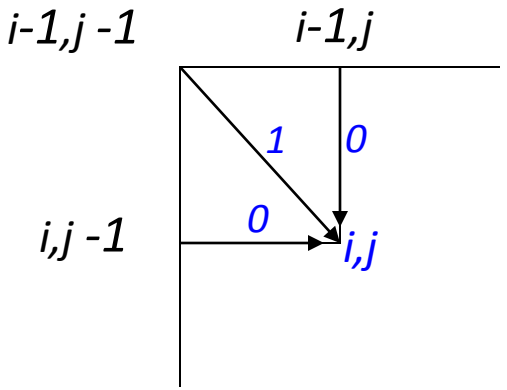
Computing LCS (I)

Let $\boldsymbol{v}_i$ = prefix of $\boldsymbol{v}$ of length i: $v_1 \ldots v_i$

and $\boldsymbol{w}_j$ = prefix of $\boldsymbol{w}$ of length j: $w_1 \ldots w_j$

The length of LCS($\boldsymbol{v}_i$,$\boldsymbol{w}_j$) is computed by:

$$s_{i,j} = \max \begin{cases} s_{i-1,j} & \text{Insertion} \\ s_{i,j-1} & \text{Deletion} \\ s_{i-1,j-1} + 1 \text{ if } v_i = w_j & \text{Match} \end{cases}$$



LCS algorithm

LCS($\mathbf{v}$, n, $\mathbf{w}$, m)

1 **for** $i \leftarrow 1$ **to** n

2 $\quad s_{i,\,0} \leftarrow 0$

3 **for** $j \leftarrow 1$ **to** m

4 $\quad s_{0,\,j} \leftarrow 0$

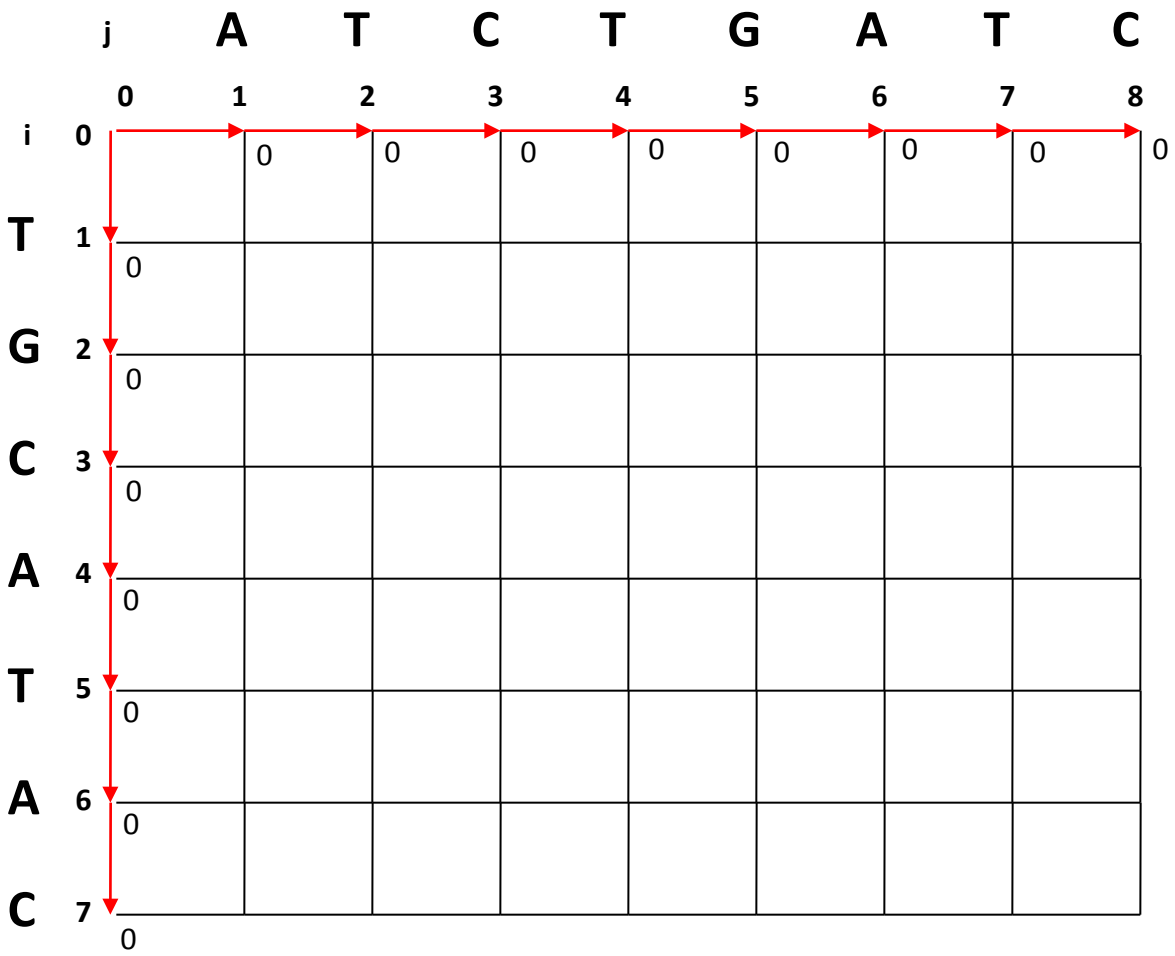
5 **for** $i \leftarrow 1$ **to** n

6 $\quad$ **for** $j \leftarrow 1$ **to** m

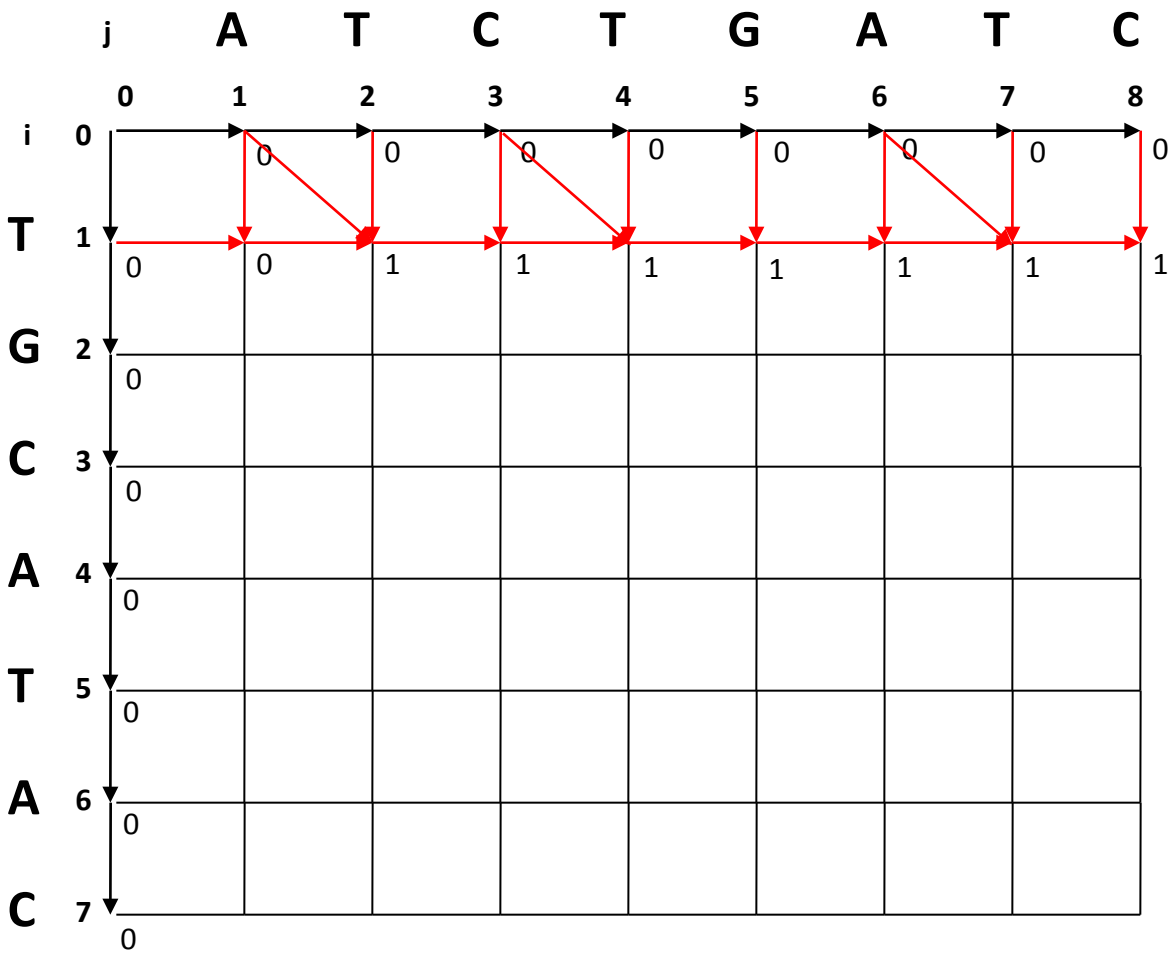
8 $$s_{i,\,j} \leftarrow \max \begin{cases} s_{i-1,\,j} \\ s_{i,\,j-1} \\ s_{i-1,\,j-1} + 1, \text{ if } v_i = w_j \end{cases}$$

10 **return** $s_{n,\,m}$

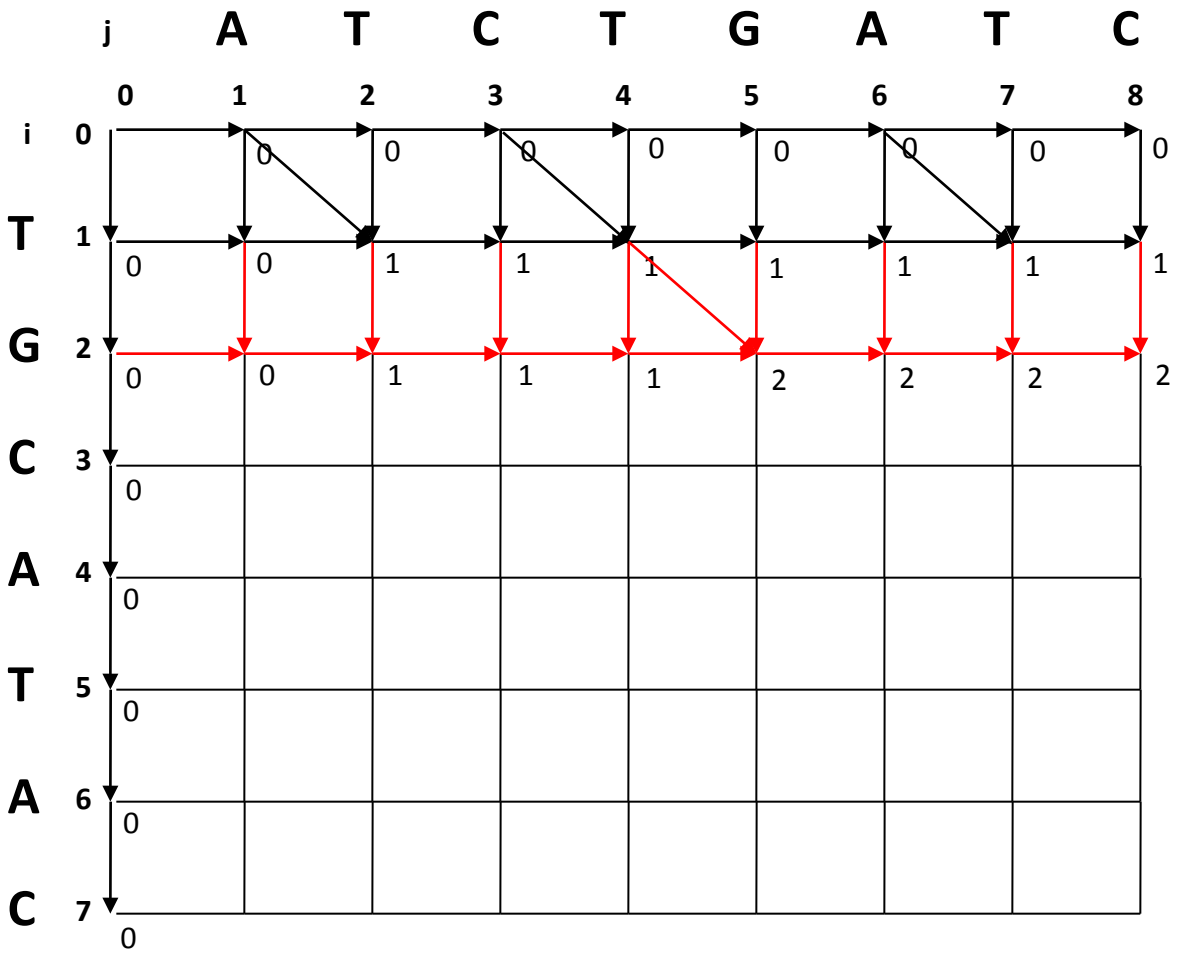
Example: initiation



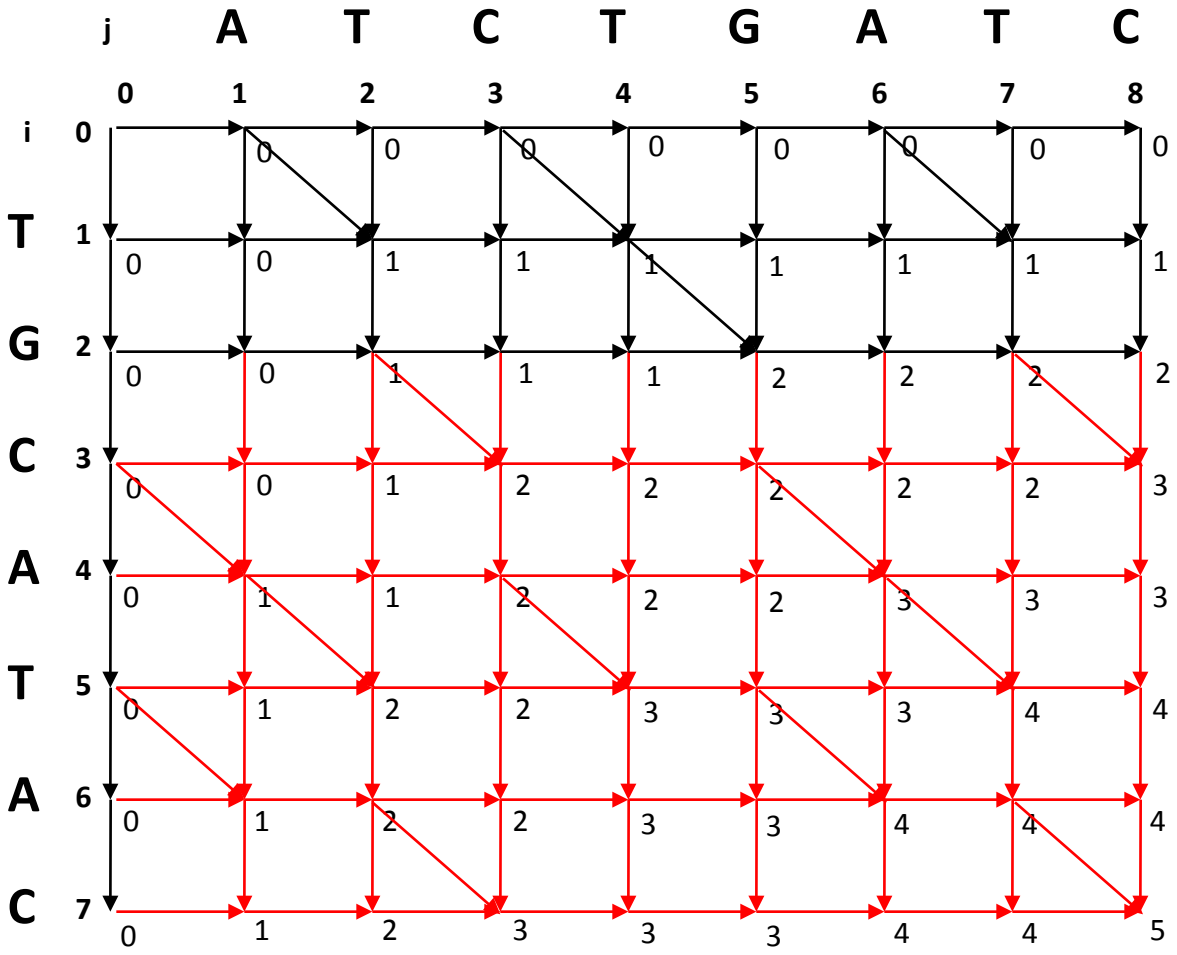
Example: For $i = 1, j = 1 \ldots m$



Example: For $i = 2, j = 1 \ldots m$



Example: For $i = 3 \ldots n$, $j = 1 \ldots m$



LCS Runtime

- It takes O(nm) time to fill in the $n \times m$ dynamic programming matrix

- The pseudocode consists of a nested "**for**" loop inside of another "**for**" loop to set up a $n \times m$ matrix

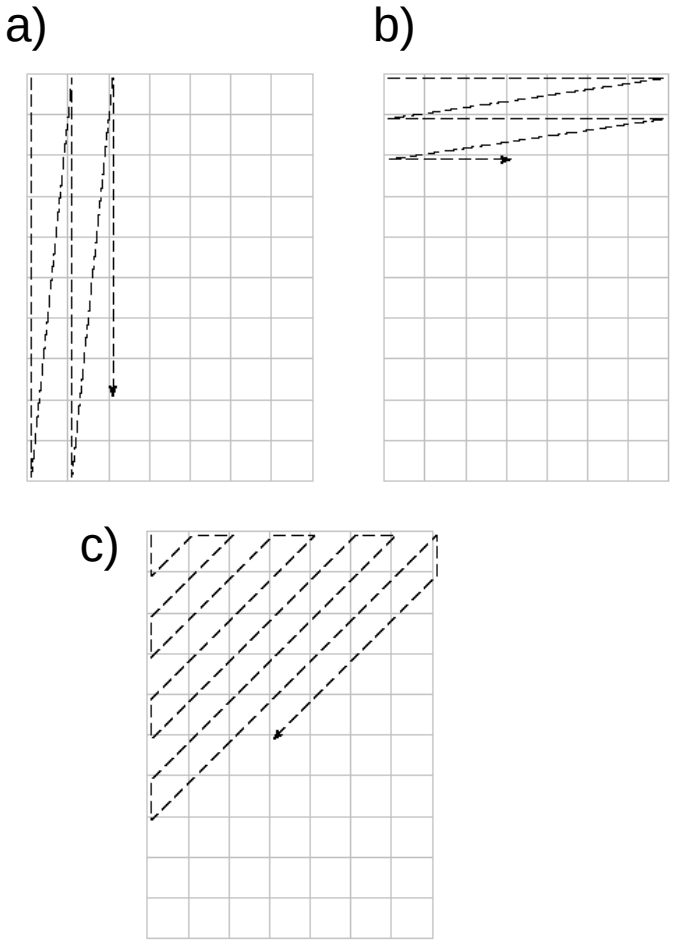
What's so great about dynamic programming?

- A naive exhaustive search would have the running time $O(3^{f(n,m)})$
- An exhaustive search would recompute the same subpaths several times
- Dynamic programming takes advantage of the rich computational structure in the search space, and reuse already computed subpaths

Traversing the edit graph

3 different strategies:

- a) Column by column
- b) Row by row
- c) Along diagonals



Making a scoring matrix

- Scoring matrices are created based on biological evidence
- Alignments can be thought of as two sequences that differ due to mutations
- Some of these mutations have little effect on the protein's function, therefore some penalties, $\delta(i, j)$, will be less harsh than others
- $\delta(i, j) \approx$ how often do amino acid i substitutes amino acid j in alignments of related proteins

Scoring matrix: Example

	A	R	N	K
A	5	-2	-1	-1
R	-	7	-1	3
N	-	-	7	0
K	-	-	-	6

- Notice that although **R** and **K** are different amino acids, they have a positive score
- Why? They are both positively charged amino acids and will not greatly change the function of protein

Scoring matrices

- Amino acid substitution matrices
  - PAM
  - BLOSUM

- DNA substitution matrices
  - DNA is less conserved than protein sequences
  - Less effective to compare coding regions at nucleotide level

PAM

- **P**oint **A**ccepted **M**utation
- 1 PAM = PAM_1 = 1% average change of all amino acid positions
- After 100 PAMs of evolution, not every residue will have changed
  - some residues may have mutated several times
  - some residues may have returned to their original state
  - some residues may not changed at all

PAM_X

- $PAM_x = PAM_1^x$
  - $PAM_{250} = PAM_1^{250}$
- PAM_{250} is a widely used scoring matrix:

	Ala A	Arg R	Asn N	Asp D	Cys C	Gln Q	Glu E	Gly G	His H	Ile I	Leu L	Lys K	... ...
Ala A	13	6	9	9	5	8	9	12	6	8	6	7	...
Arg R	3	17	4	3	2	5	3	2	6	3	2	9	
Asn N	4	4	6	7	2	5	6	4	6	3	2	5	
Asp D	5	4	8	11	1	7	10	5	6	3	2	5	
Cys C	2	1	1	1	52	1	1	2	2	2	1	1	
Gln Q	3	5	5	6	1	10	7	3	7	2	3	5	
...													
Trp W	0	2	0	0	0	0	0	0	1	0	1	0	
Tyr Y	1	1	2	1	3	1	1	1	3	2	2	1	
Val V	7	4	4	4	4	4	4	4	5	4	15	10	

BLOSUM

- **Blo**cks **Su**bstitution **M**atrix
- Scores derived by observing the frequencies of substitutions in blocks of local alignments in related proteins
- Matrix name indicates evolutionary distance
  - BLOSUM62 was created using sequences sharing no more than 62% sequence identity

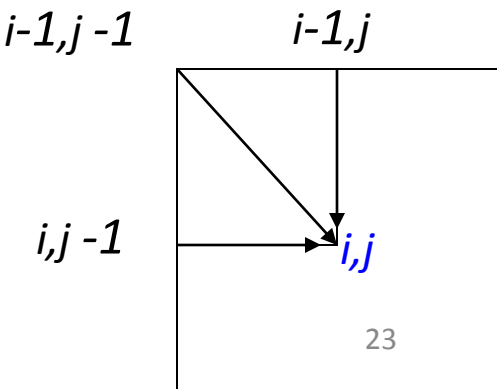
BLOSUM50

	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V	B	Z	X	*
A	5	-2	-1	-2	-1	-1	-1	0	-2	-1	-2	-1	-1	-3	-1	1	0	-3	-2	0	-2	-1	-1	-5
R	-2	7	-1	-2	-4	1	0	-3	0	-4	-3	3	-2	-3	-3	-1	-1	-3	-1	-3	-1	0	-1	-5
N	-1	-1	7	2	-2	0	0	0	1	-3	-4	0	-2	-4	-2	1	0	-4	-2	-3	4	0	-1	-5
D	-2	-2	2	8	-4	0	2	-1	-1	-4	-4	-1	-4	-5	-1	0	-1	-5	-3	-4	5	1	-1	-5
C	-1	-4	-2	-4	13	-3	-3	-3	-3	-2	-2	-3	-2	-2	-4	-1	-1	-5	-3	-1	-3	-3	-2	-5
Q	-1	1	0	0	-3	7	2	-2	1	-3	-2	2	0	-4	-1	0	-1	-1	-1	-3	0	4	-1	-5
E	-1	0	0	2	-3	2	6	-3	0	-4	-3	1	-2	-3	-1	-1	-1	-3	-2	-3	1	5	-1	-5
G	0	-3	0	-1	-3	-2	-3	8	-2	-4	-4	-2	-3	-4	-2	0	-2	-3	-3	-4	-1	-2	-2	-5
H	-2	0	1	-1	-3	1	0	-2	10	-4	-3	0	-1	-1	-2	-1	-2	-3	2	-4	0	0	-1	-5
I	-1	-4	-3	-4	-2	-3	-4	-4	-4	5	2	-3	2	0	-3	-3	-1	-3	-1	4	-4	-3	-1	-5
L	-2	-3	-4	-4	-2	-2	-3	-4	-3	2	5	-3	3	1	-4	-3	-1	-2	-1	1	-4	-3	-1	-5
K	-1	3	0	-1	-3	2	1	-2	0	-3	-3	6	-2	-4	-1	0	-1	-3	-2	-3	0	1	-1	-5
M	-1	-2	-2	-4	-2	0	-2	-3	-1	2	3	-2	7	0	-3	-2	-1	-1	0	1	-3	-1	-1	-5
F	-3	-3	-4	-5	-2	-4	-3	-4	-1	0	1	-4	0	8	-4	-3	-2	1	4	-1	-4	-4	-2	-5
P	-1	-3	-2	-1	-4	-1	-1	-2	-2	-3	-4	-1	-3	-4	10	-1	-1	-4	-3	-3	-2	-1	-2	-5
S	1	-1	1	0	-1	0	-1	0	-1	-3	-3	0	-2	-3	-1	5	2	-4	-2	-2	0	0	-1	-5
T	0	-1	0	-1	-1	-1	-1	-2	-2	-1	-1	-1	-1	-2	-1	2	5	-3	-2	0	0	-1	0	-5
W	-3	-3	-4	-5	-5	-1	-3	-3	-3	-3	-2	-3	-1	1	-4	-4	-3	15	2	-3	-5	-2	-3	-5
Y	-2	-1	-2	-3	-3	-1	-2	-3	2	-1	-1	-2	0	4	-3	-2	-2	2	8	-1	-3	-2	-1	-5
V	0	-3	-3	-4	-1	-3	-3	-4	-4	4	1	-3	1	-1	-3	-2	0	-3	-1	5	-4	-3	-1	-5
B	-2	-1	4	5	-3	0	1	-1	0	-4	-4	0	-3	-4	-2	0	0	-5	-3	-4	5	2	-1	-5
Z	-1	0	0	1	-3	4	5	-2	0	-3	-3	1	-1	-4	-1	0	-1	-2	-2	-3	2	5	-1	-5
X	-1	-1	-1	-1	-2	-1	-1	-2	-1	-1	-1	-1	-1	-2	-2	-1	0	-3	-1	-1	-1	-1	-1	-5
*****	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	1

Scoring matrices and the global alignment problem

- To generalize scoring, consider a $(4+1) \times (4+1)$ scoring matrix δ
- In the case of an amino acid sequence alignment, the scoring matrix would be $(20+1) \times (20+1)$
- The addition of *1* is to include the score for comparison of a gap character "-" (indels)

$$s_{i,j} = max \begin{cases} s_{i-1,j} + \delta(v_i, -) & \text{Insertion} \\ s_{i,j-1} + \delta(-, w_j) & \text{Deletion} \\ s_{i-1,j-1} + \delta(v_i, w_j) & \text{(Mis)match} \end{cases}$$



Local vs. global alignment (I)

- The Global alignment problem : find the longest path between vertices (0,0) and (n,m) in the edit graph
- The Local alignment problem tries to find the longest path between arbitrary vertices (i, j) and (i', j') in the edit graph
- In the edit graph with negative scores, local alignment may score higher than global alignment

Local vs. global alignment (II)

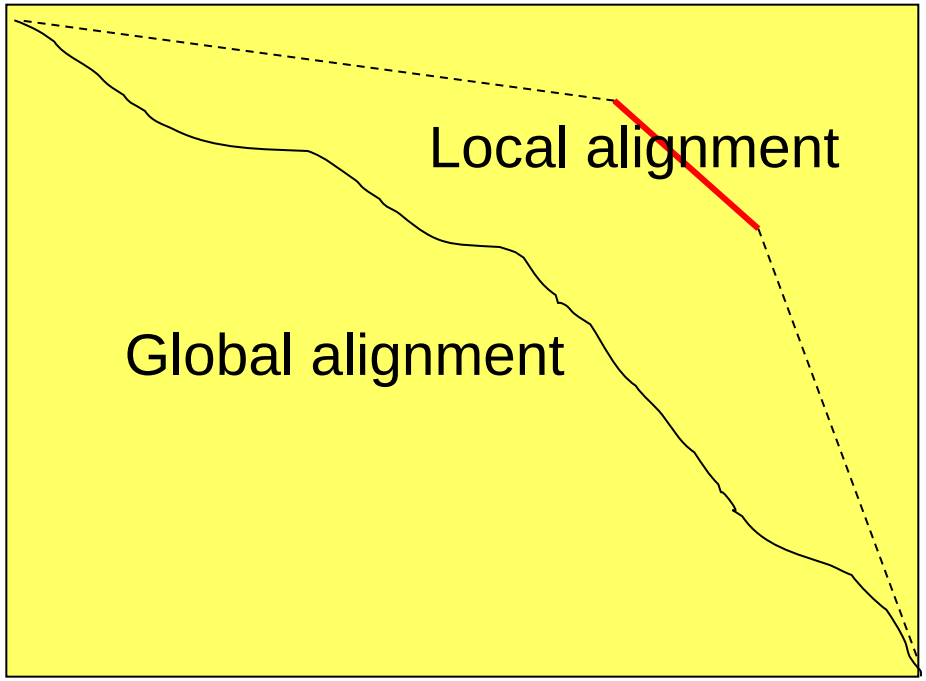
- Global Alignment

```
--T--CC-C-AGT--TATGT-CAGGGGACACG-A-GCATGCAGA-GAC
  |  || |  ||  | | | |||    || | | |  | ||||   |
AATTGCCGCC-GTCGT-T-TTCAG----CA-GTTATG-T-CAGAT--C
```

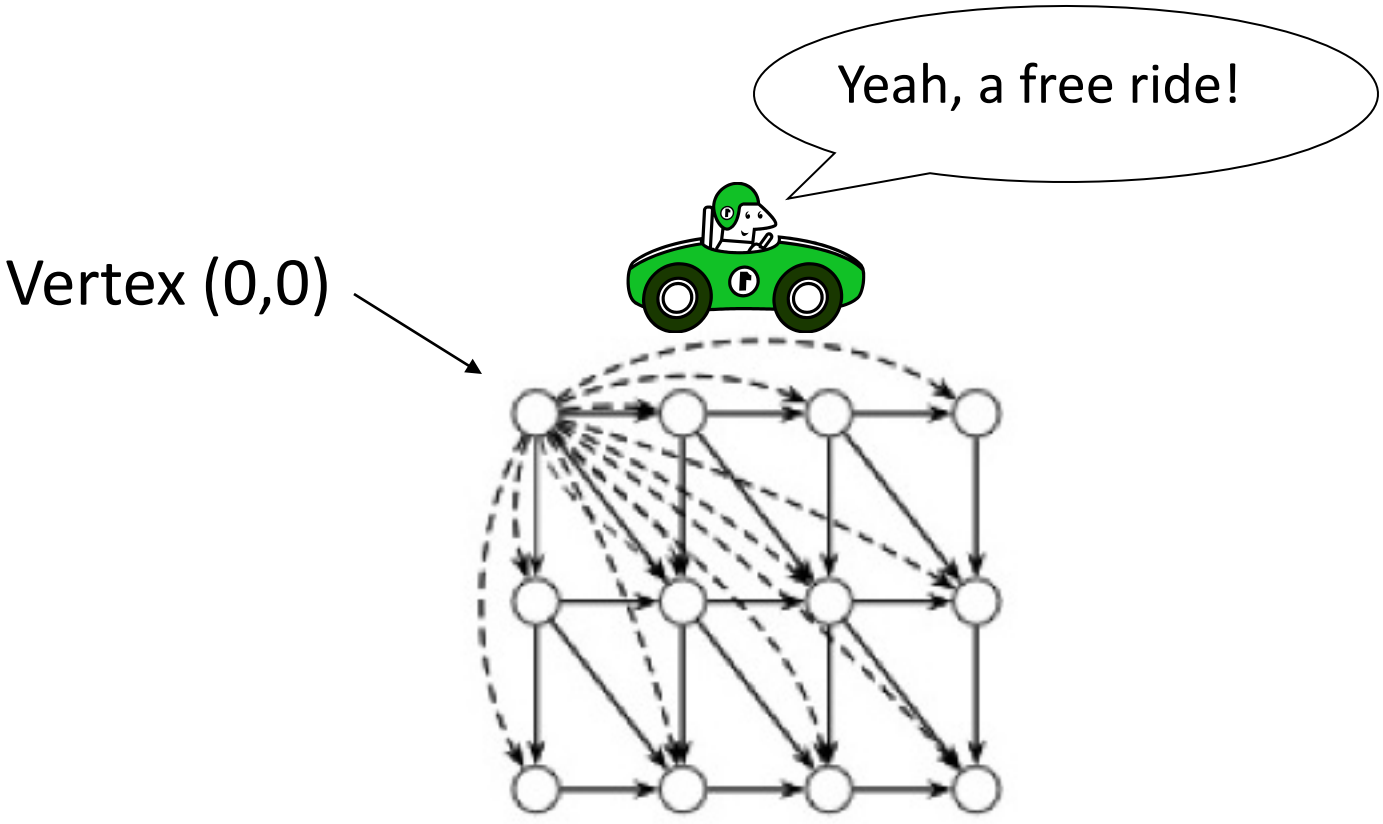
- Local Alignment—better alignment to find conserved segment

```
                  tccCAGTTATGTCAGgggacacgagcatgcagagac
                     ||||||||||||
aattgccgccgtcgttttcagCAGTTATGTCAGatc
```

Local vs. global alignment (III)



Free rides



The dashed edges represent the free rides from (0,0) to every other node.

The local alignment recurrence

- The largest value of $s_{i,j}$ over the whole edit graph is the score of the best local alignment

$$s_{i,j} = \max \begin{cases} 0 \\ s_{i-1,j} + \delta(v_i, -) \\ s_{i,j-1} + \delta(-, w_j) \\ s_{i-1,j-1} + \delta(v_i, w_j) \end{cases}$$

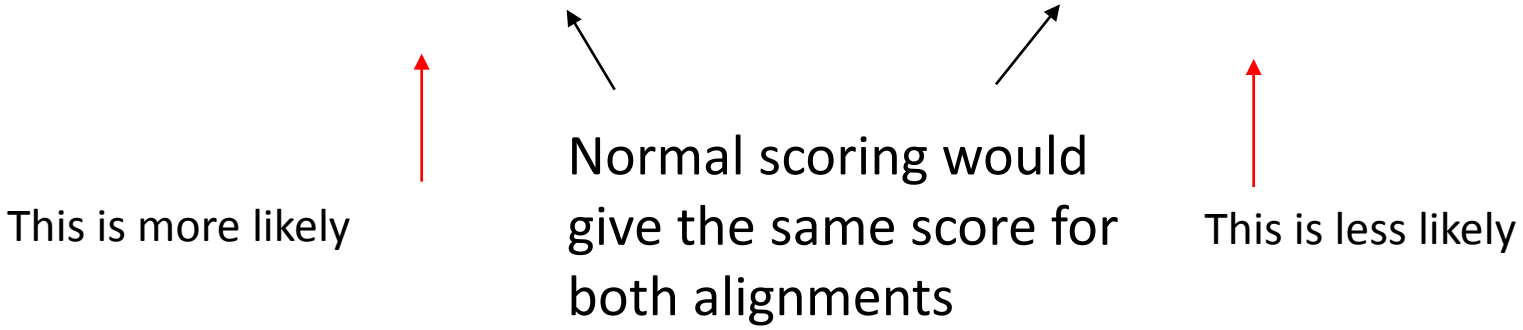
- The 0 is the only difference from the recurrence of the global alignment problem

Gap penalties

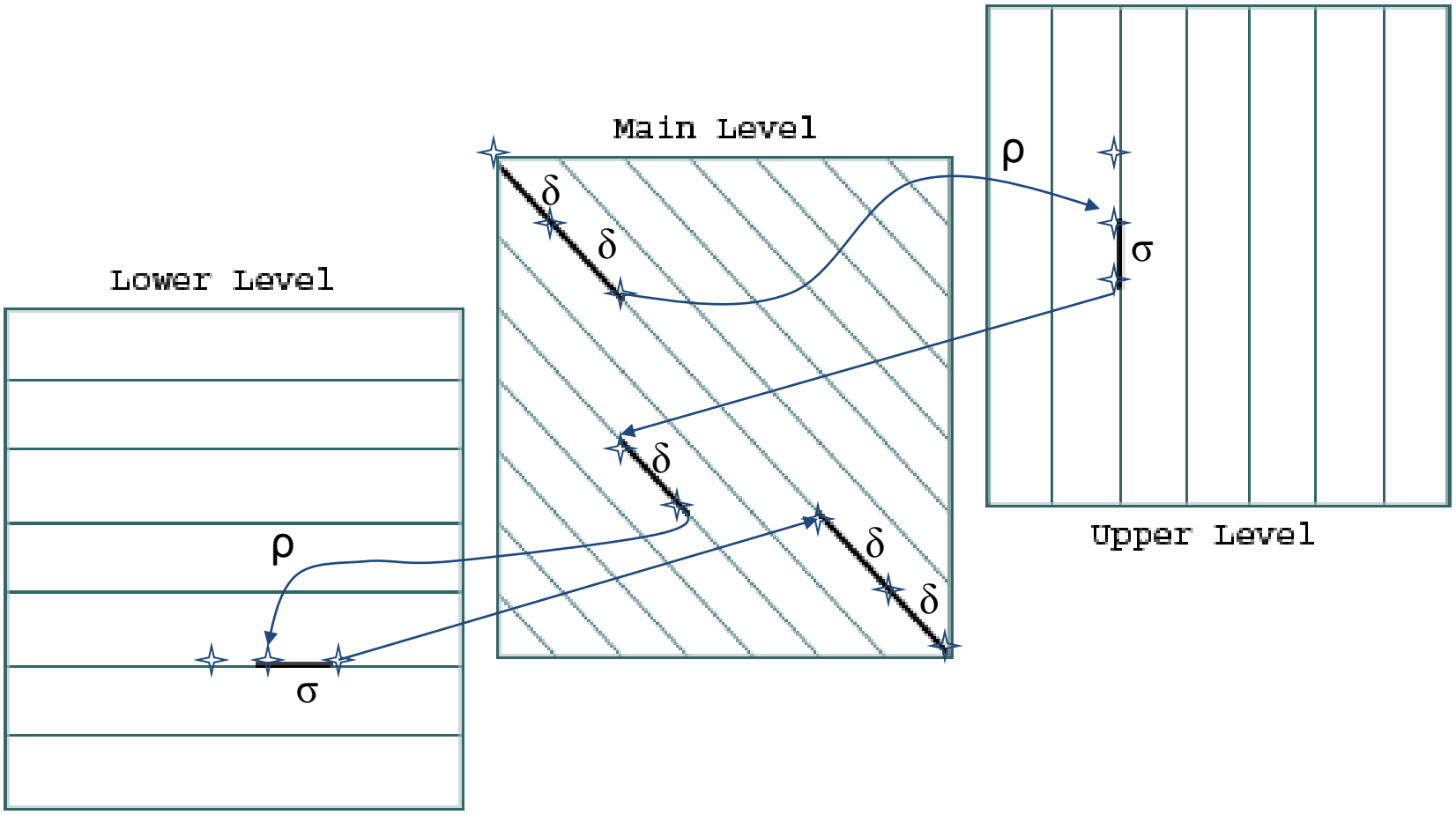
In nature, a series of k indels often come as a single event rather than a series of k single nucleotide events:

ATA– –GC ATAG– GC

ATATTGC AT– GTGC



3 layer edit grap



Gap penalty recurrences

$$\downarrow_{s_{i,j}} = \max \begin{cases} \downarrow_{s_{i-1,j}} - \sigma \\ s_{i-1,j} - (\varrho + \sigma) \end{cases}$$

Continue gap in ***w*** (insertion): upper level

Start gap in ***w*** (insertion): from main level

$$\rightarrow_{s_{i,j}} = \max \begin{cases} \rightarrow_{s_{i,j-1}} - \sigma \\ s_{i,j-1} - (\varrho + \sigma) \end{cases}$$

Continue gap in ***v*** (deletion): lower level

Start gap in ***v*** (deletion): from main level

$$s_{i,j} = \max \begin{cases} s_{i-1,j-1} + \delta(v_i, w_j) \\ \downarrow_{s_{i,j}} \\ \rightarrow_{s_{i,j}} \end{cases}$$

Match or mismatch: main level

End insertion: from upper level

End deletion: from lower level

BLAST (I)

- Basic Local Alignment Search Tool (BLAST) finds regions of local similarity between sequences
- The program compares nucleotide or protein sequences to sequence databases and calculates the statistical significance of matches

BLAST (II)

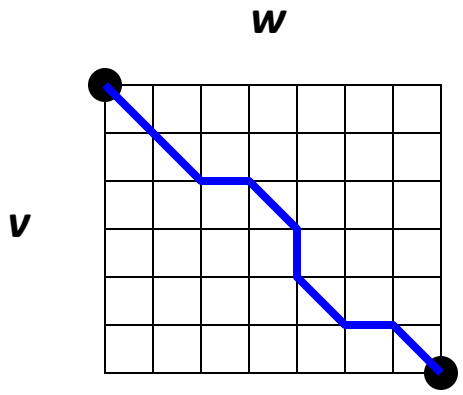
- First stage: Identify exact matches of length W (default W=3) between the query and the sequences in the database
- Second stage: Extend the match in both directions in an attempt to boost the alignment score (insertions and deletions are not considered)
- Third stage: If a high-scoring ungapped alignment is found: Perform a gapped local alignment using dynamic programming

Multiple alignment

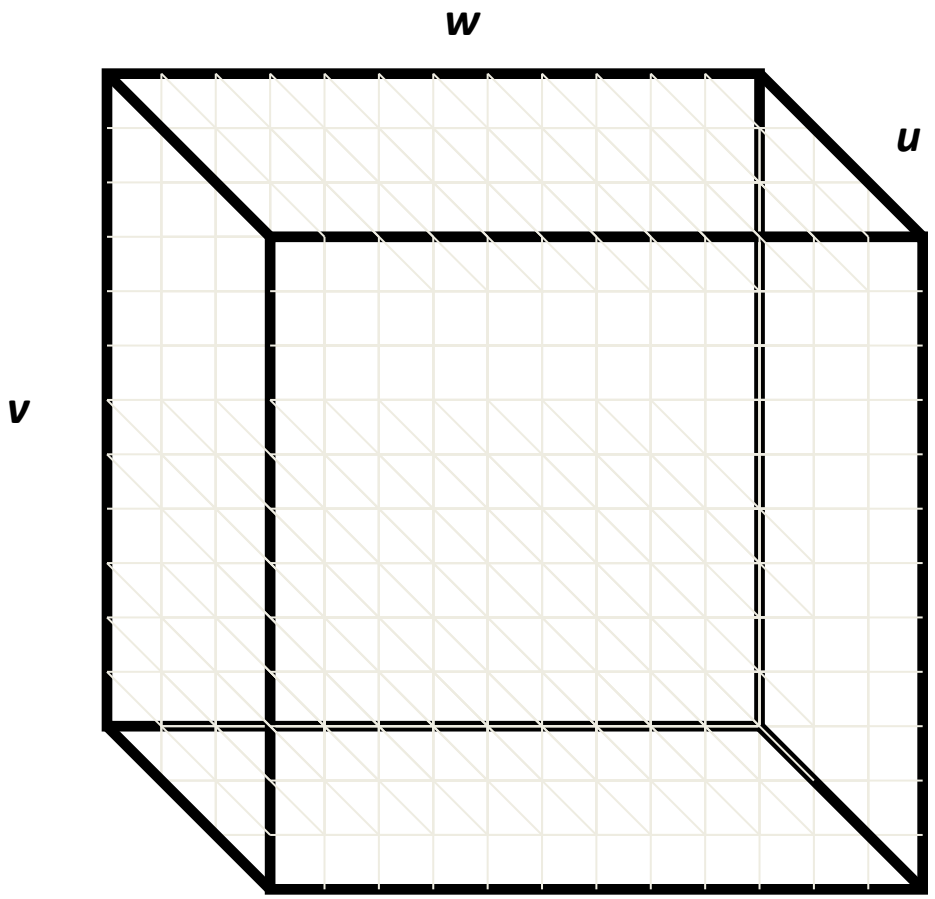
- A faint similarity between two sequences becomes significant if present in many
- Multiple alignments can reveal subtle similarities that pairwise alignments do not reveal

```
A  T  –  G  C  G  –
A  –  C  G  T  –  A
A  T  C  A  C  –  A
```

2D vs 3D edit graph

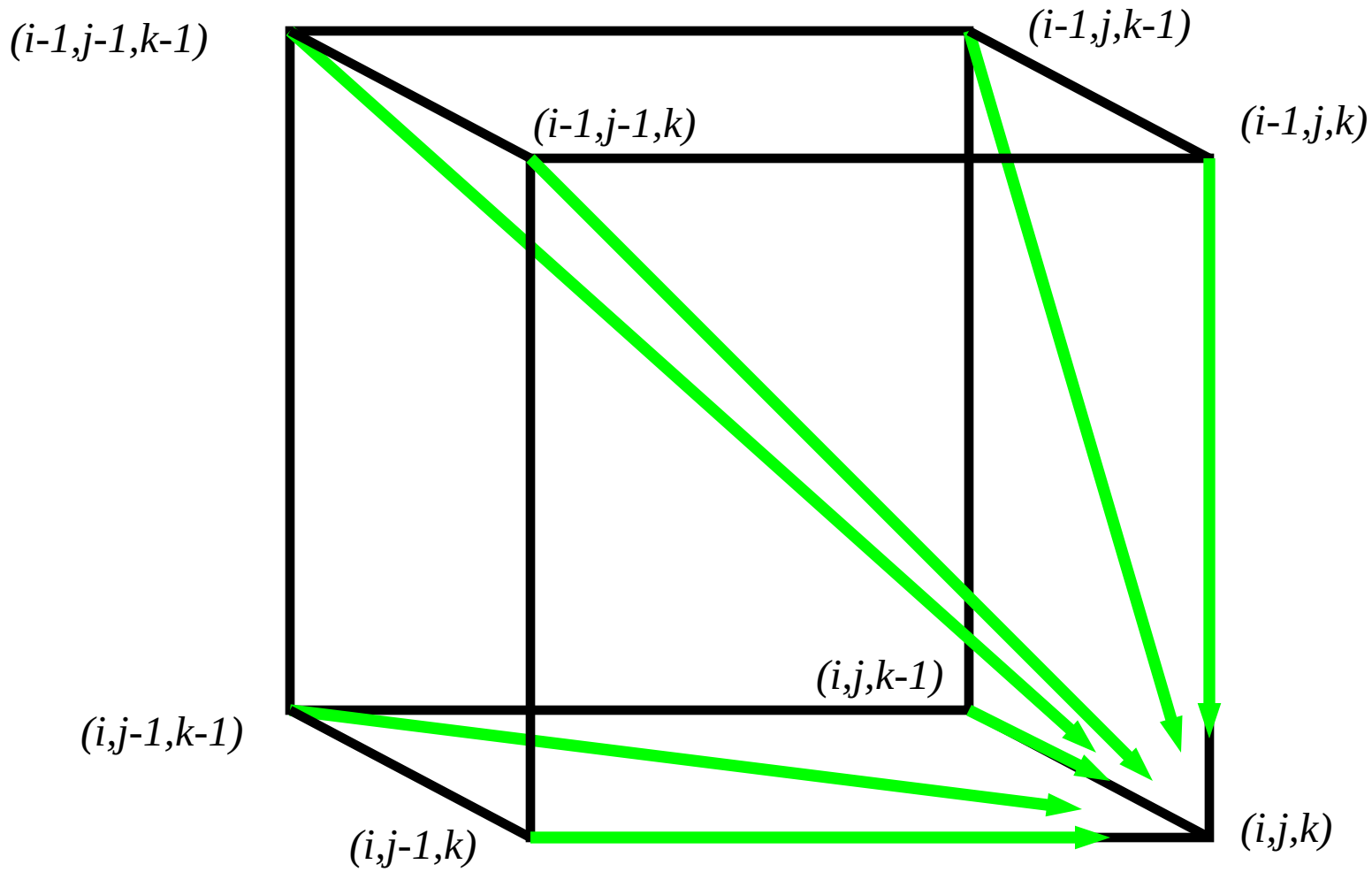


2-D edit graph



3-D edit graph

Architecture of 3D edit graph



Multiple alignment of three sequences: Dynamic programming

$$s_{i,j,k} = \max \begin{cases} s_{i-1,j-1,k-1} + \delta(v_i, w_j, u_k) \\ s_{i-1,j-1,k} + \delta(v_i, w_j, \_) \\ s_{i-1,j,k-1} + \delta(v_i, \_, u_k) \\ s_{i,j-1,k-1} + \delta(\_, w_j, u_k) \\ s_{i-1,j,k} + \delta(v_i, \_, \_) \\ s_{i,j-1,k} + \delta(\_, w_j, \_) \\ s_{i,j,k-1} + \delta(\_, \_, u_k) \end{cases}$$

$\delta(x, y, z)$ is an entry in the 3D scoring matrix

Multiple alignment: Running time

- For three sequences of length n, the run time is $O(n^3)$
- For k sequences, build a k-dimensional edit graph, with run time $O(n^k)$

- Conclusion: dynamic programming approach for alignment between two sequences is easily extended to k sequences, but it is impractical due to exponential running time

Profile representation of multiple alignment

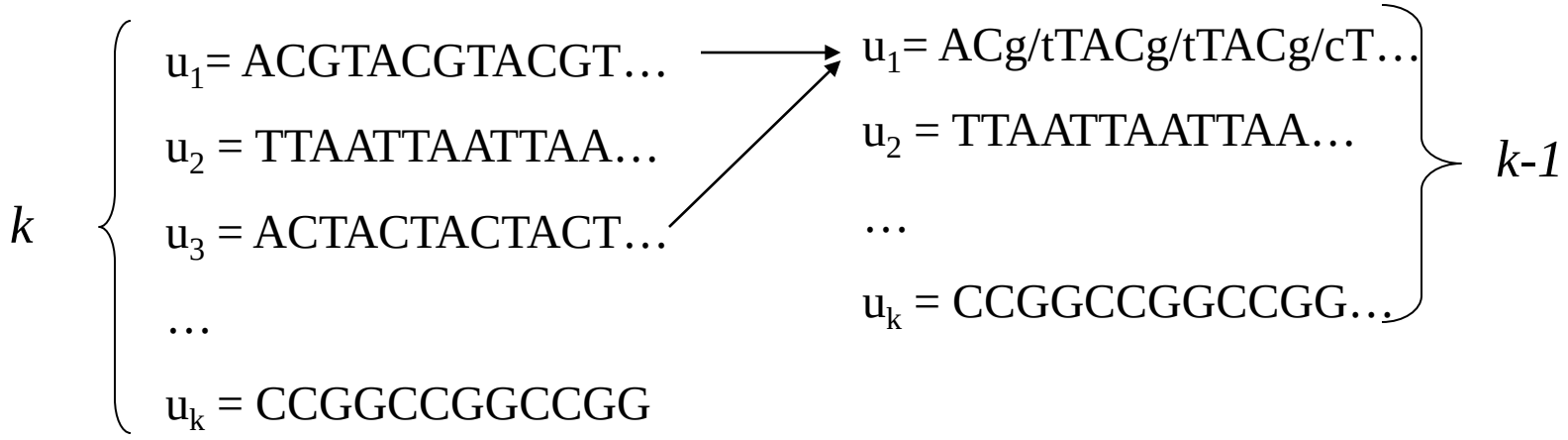
```
 -  A  G  G  C  T  A  T  C  A  C  C  T  G
 T  A  G  –  C  T  A  C  C  A  -  -  -  G
 C  A  G  –  C  T  A  C  C  A  -  -  -  G
 C  A  G  –  C  T  A  T  C  A  C  –  G  G
 C  A  G  –  C  T  A  T  C  G  C  –  G  G

A     1              1        .8
C  .6                 1     .4 1     .6 .2
G        1  .2                    .2        .4  1
T  .2                    1     .6              .2
-  .2        .8                          .4 .8 .4
```

PSSM: Position Specific Scoring Matrix

- In the past we were aligning a sequence against a sequence
- With profiles we can align a sequence against a profile and even a profile against a profile

Multiple alignment: Greedy approach

- Choose most similar pair of strings and combine into a profile, thereby reducing the alignment of *k* sequences to an alignment of of *k-1* sequences/profiles. Repeat!
- This is a heuristic greedy method



CLUSTALW (I)

1. Determine all pairwise alignments between sequences and the degree of similarity between them.

2. Construct a similarity tree.

3. Combine the alignments from 1 in the order specified in 2 using the rule "once a gap always a gap“.

PSI-BLAST

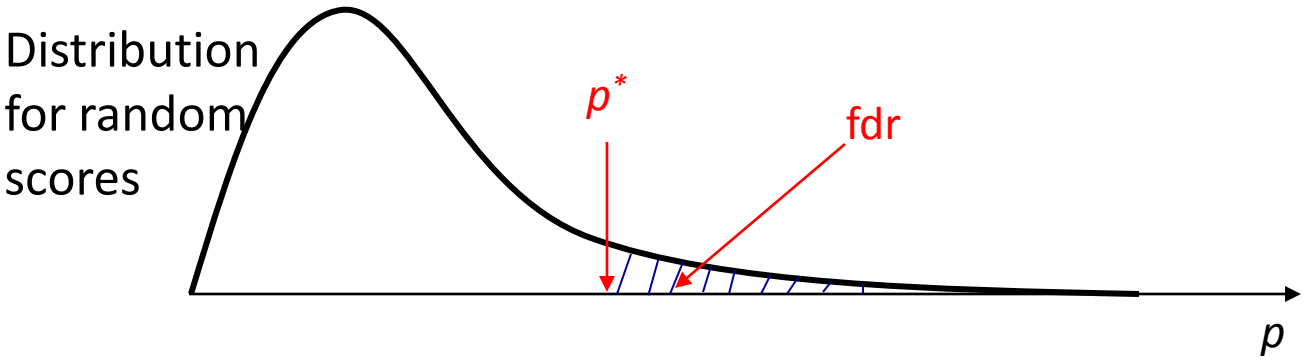
- Position-Specific Iterative (PSI) BLAST detect weak relationships between the query and sequences in the database (higher sensitivity than BLAST)
- PSI-BLAST first constructs a multiple alignment from the highest scoring hits in a initial BLAST search and generate a profile from this alignment i.e. PSSM
- The profile is used to iteratively perform additional BLAST searches (called iterations) and the results of each iteration is used to refine the profile
- The iteration stops when no new matches with a satisfactory score are obtained

Scoring matches

Given a protein sequence ***x*** and an BLAST/PSI-BLAST/HMM, what is a significant score?

- The score for the sequence ***x***: p^*
- Generate 1000 random sequences and score them:

  $p_{rand\ 1}, p_{rand\ 2}, \ldots, p_{rand\ 1000}$
- Fit a distribution to the random scores and calculate the false discover rate (fdr)
- $E\text{-}score = fdr \cdot Size\ of\ query\ database$ (the expected number of false positive hits)

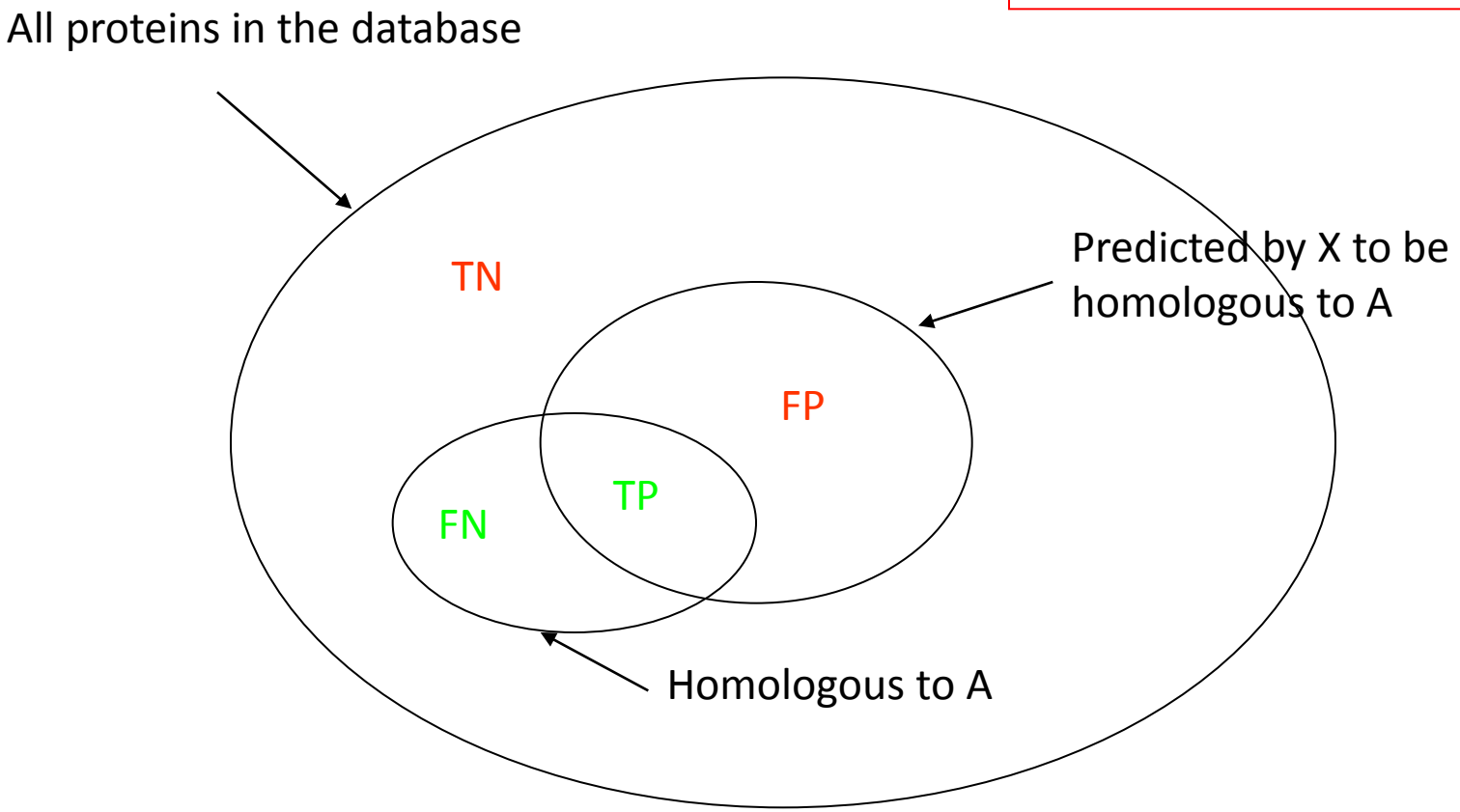


Method power

You want to find homologous proteins to a specific protein A using some computational method X:

Sensitivity: TP/(TP+FN)

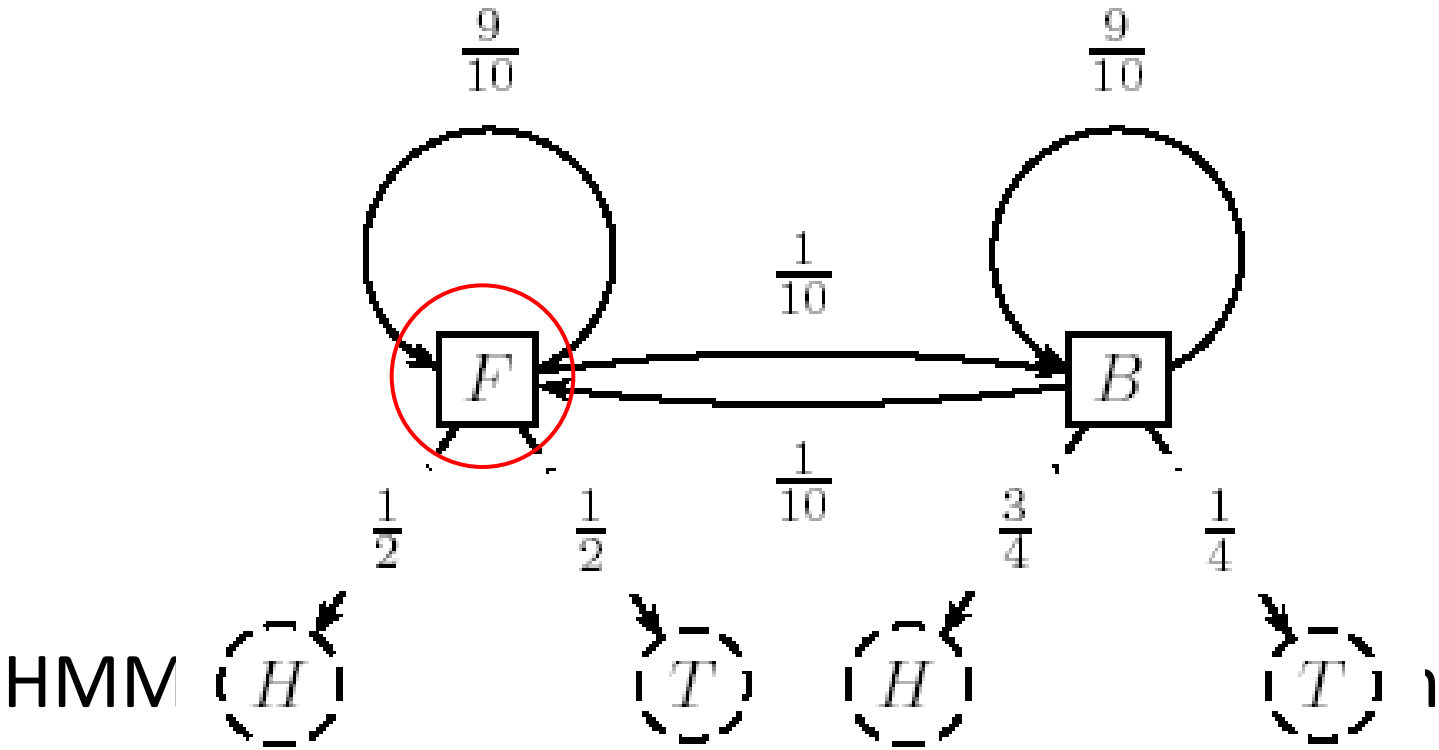
Specificity: TN/(TN+FP)



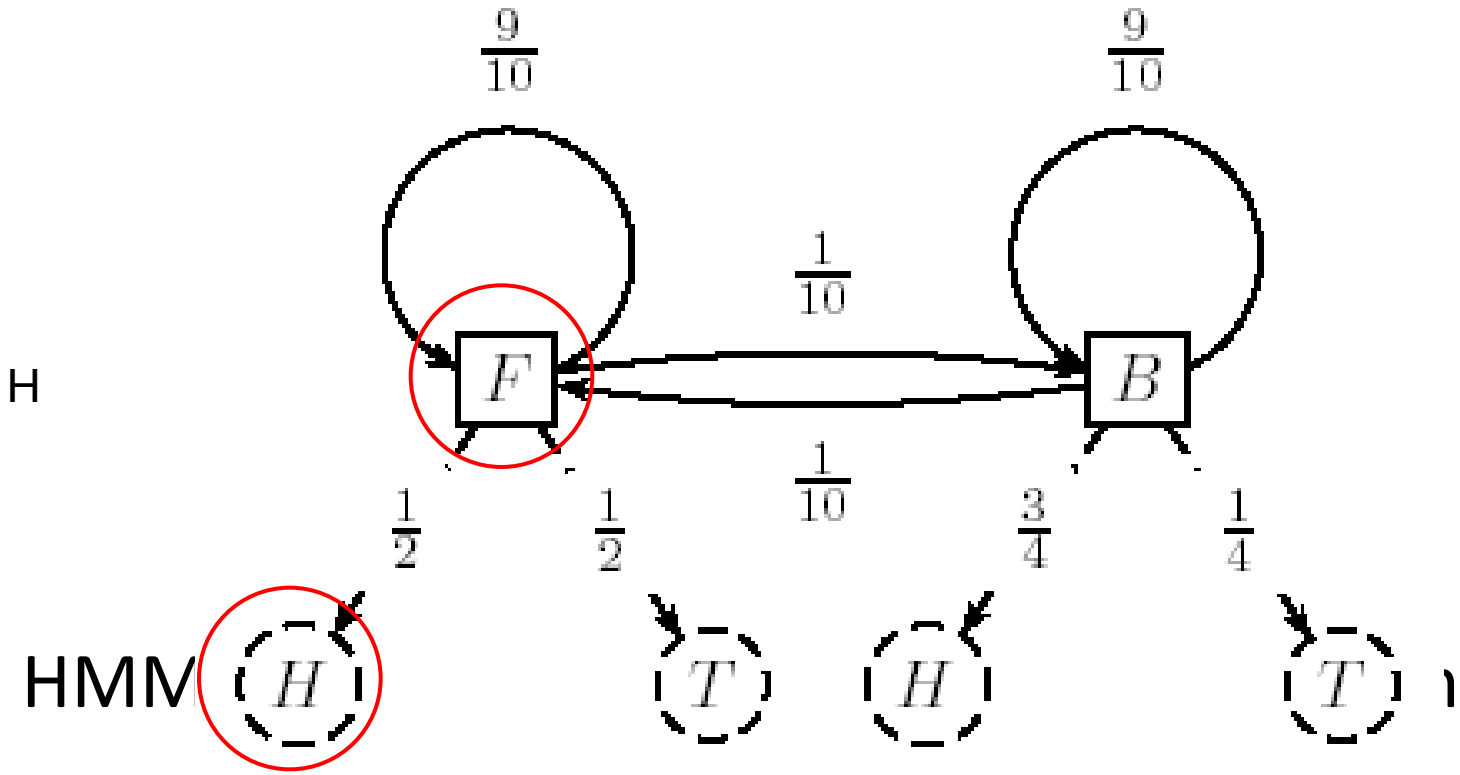
Hidden Markov Model (HMM)

- Can be viewed as an abstract machine with *k hidden* states that emits symbols from an alphabet Σ
- Each state has its own probability distribution, and the machine switches between states according to this probability distribution
- While in a certain state, the machine makes 2 decisions:
  - What state should I move to next?
  - What symbol - from the alphabet Σ - should I emit?

HMM for Fair Bet Casino

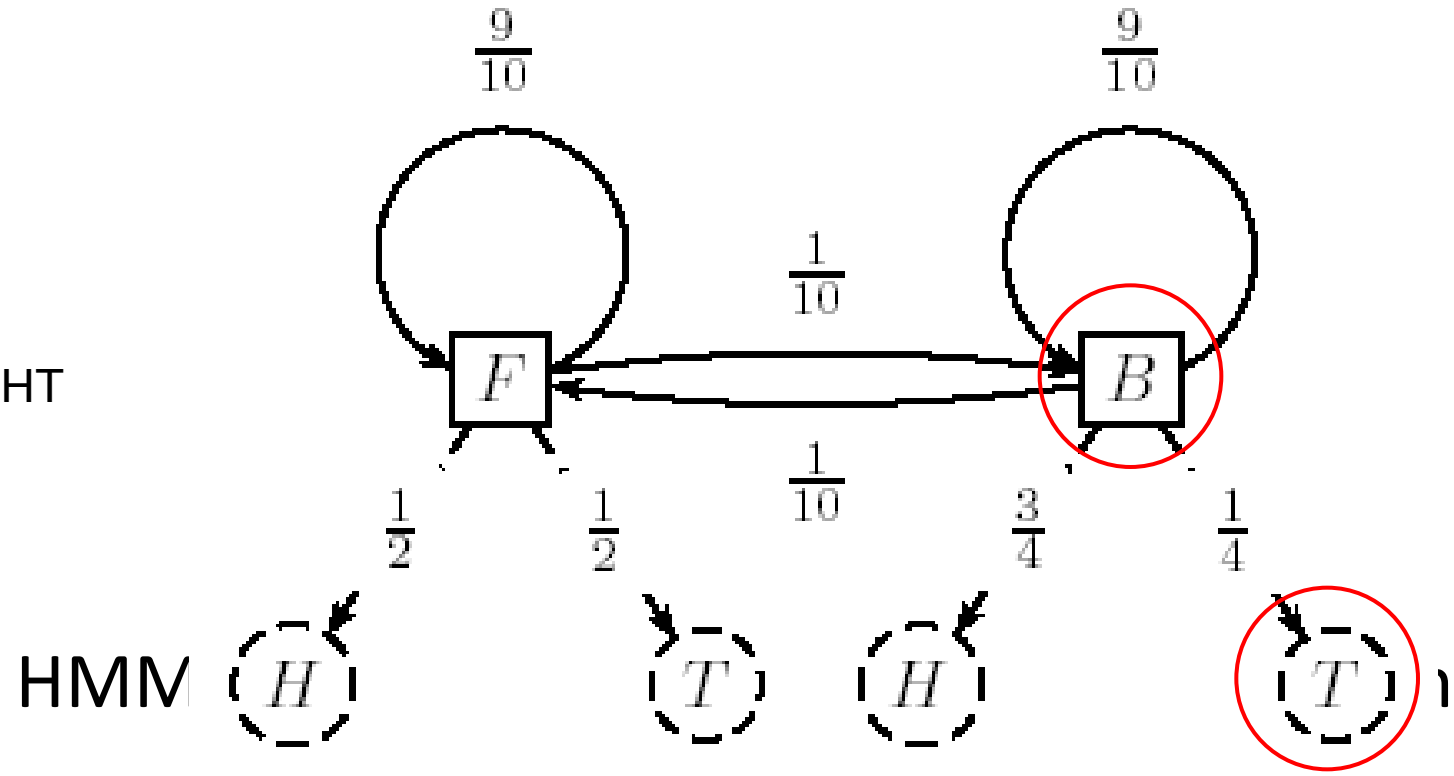


HMM for Fair Bet Casino (cont'd)



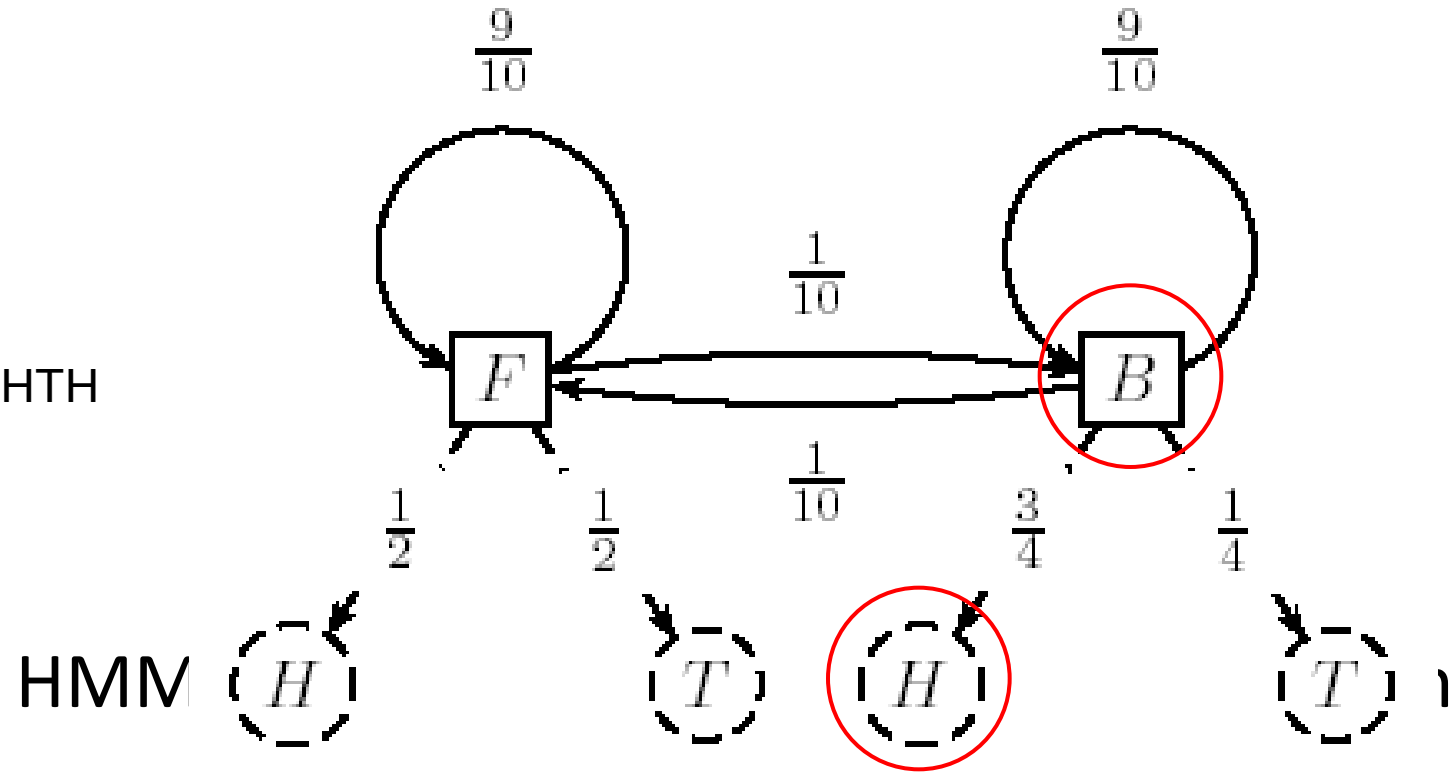
HMM for Fair Bet Casino (cont'd)

HT



HMM for Fair Bet Casino (cont'd)

HTH



Why "Hidden"?

- Observers can see the emitted symbols of an HMM but have no ability to know which state the HMM is currently in
- Thus, the goal is to infer the most likely hidden states of an HMM based on the given sequence of emitted symbols.

HMM parameters

Σ: set of emission characters

$\Sigma = \{H, T\}$ for coin tossing

$\Sigma = \{A, C, G, T\}$ for the CG-island problem

Q: set of hidden states, each emitting symbols from Σ

Q={F,B} for coin tossing

Q={CG-island, not CG-island} for the CG-island problem

HMM Parameters (cont'd)

$A = (a_{kl})$: a $|Q| \times |Q|$ matrix of probability of changing from state k to state l

- *transition probabilities*

$E = (e_k(b))$: a $|Q| \times |\Sigma|$ matrix of probability of emitting symbol b while being in state k

- *emission probabilities*

HMM for Fair Bet Casino

The *Fair Bet Casino* in *HMM* terms:

$\Sigma = \{0, 1\}$ – 0 for ***T***ails and 1 for ***H***eads

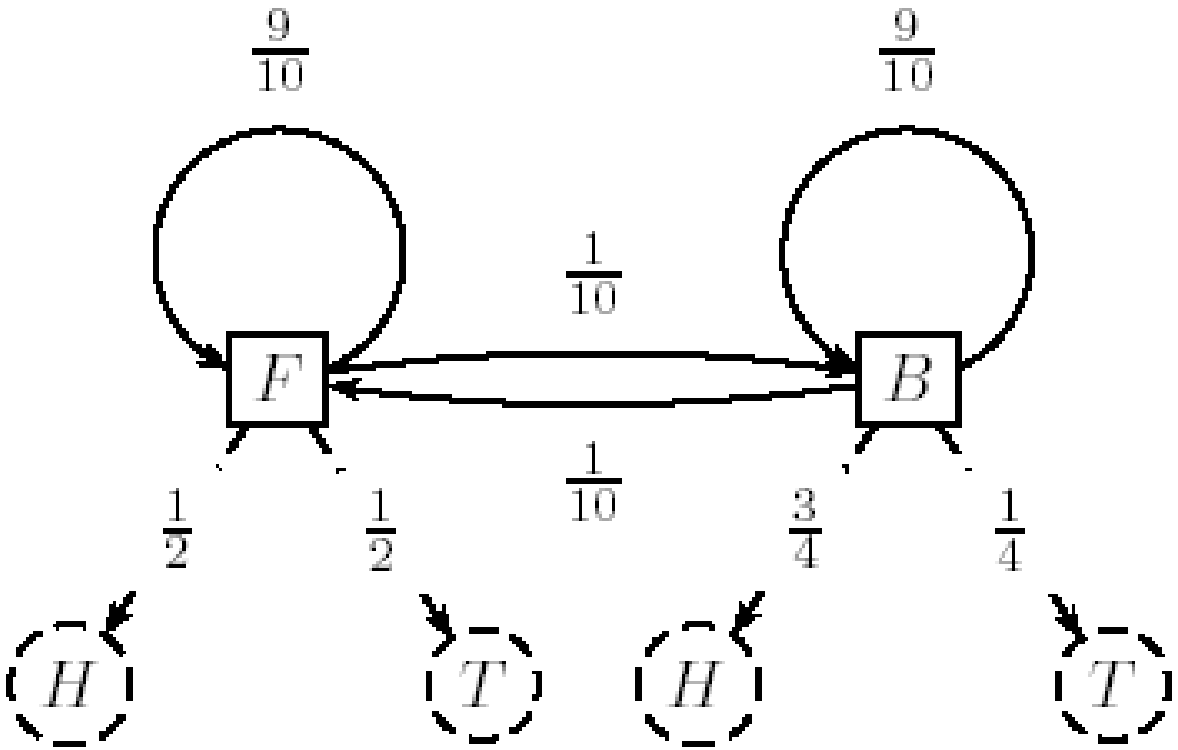
$Q = \{F,B\}$ – F for Fair & B for Biased coin

Transition Probabilities A Emission Probabilities E

	Fair	Biased
Fair	$a_{FF} = 0.9$	$a_{FB} = 0.1$
Biased	$a_{BF} = 0.1$	$a_{BB} = 0.9$

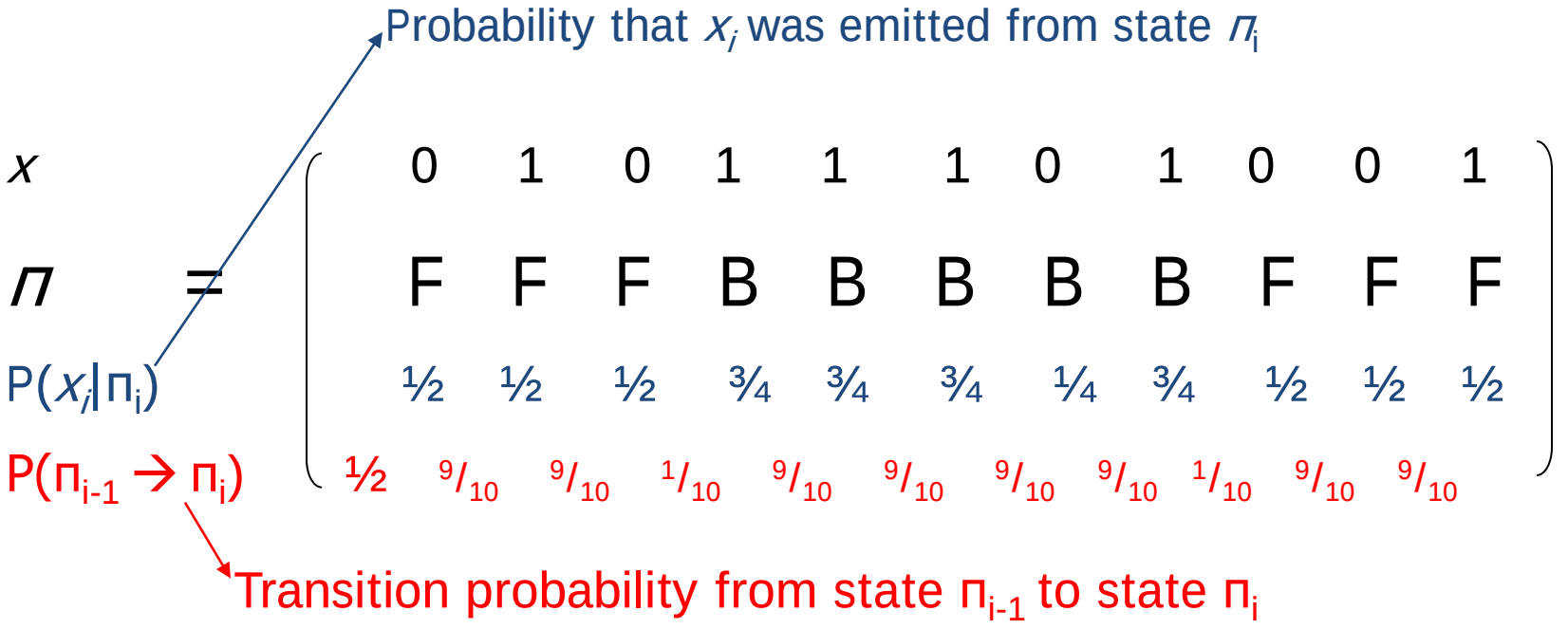
	Tails(0)	Heads(1)
Fair	$e_F(0) = ½$	$e_F(1) = ½$
Biased	$e_B(0) = ¼$	$e_B(1) = ¾$

HMM for Fair Bet Casino (cont'd)



Hidden Paths

- A *path* $\pi = \pi_1 \ldots \pi_n$ in the HMM is defined as a sequence of states
- Consider path π = FFFBBBBBBFFF and sequence $\boldsymbol{x}$ = 01011101001



P(*x*,π) Calculation

P(***x***,***π***)=P(***x***|***π***) P(***π***): Probability that sequence ***x*** was generated by the path ***π***:

$$P(\boldsymbol{x},\boldsymbol{\pi}) = P(\pi_0 \to \pi_1) \cdot \prod_{i=1}^{n} P(x_i \mid \pi_i) \cdot P(\pi_i \to \pi_{i+1})$$

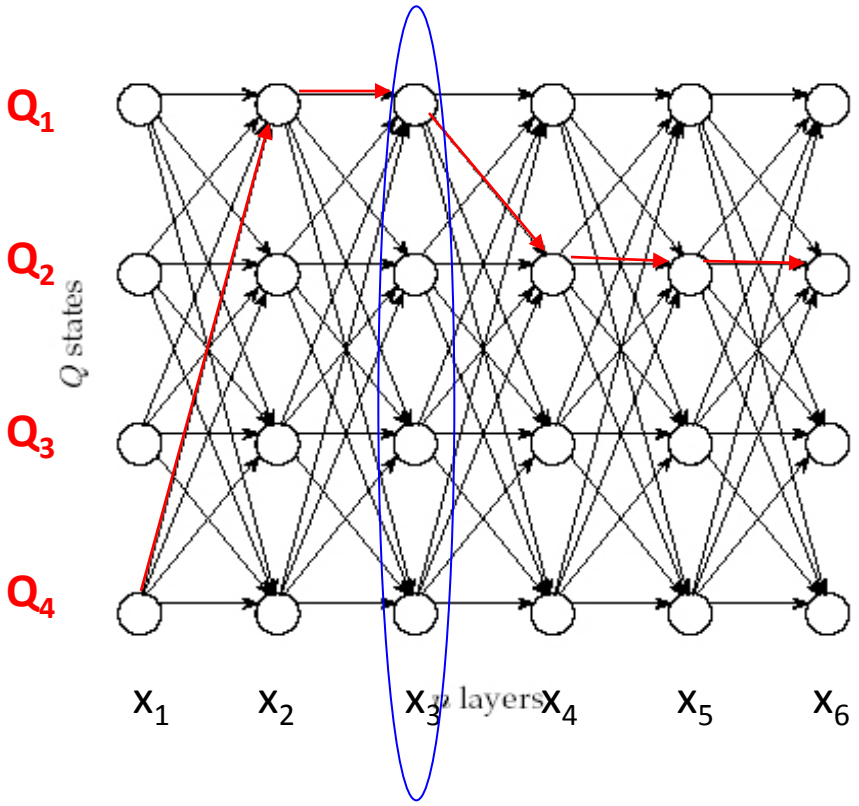
$$= a_{\pi_0, \pi_1} \cdot \prod_{i=1}^{n} e_{\pi_i}(x_i) \cdot a_{\pi_i, \pi_{i+1}}$$

$$= \prod_{i=0}^{n} e_{\pi_{i+1}}(x_{i+1}) \cdot a_{\pi_i, \pi_{i+1}}$$

where π_0 and π_{n+1} are fictitious initial and terminal states *begin* and *end*

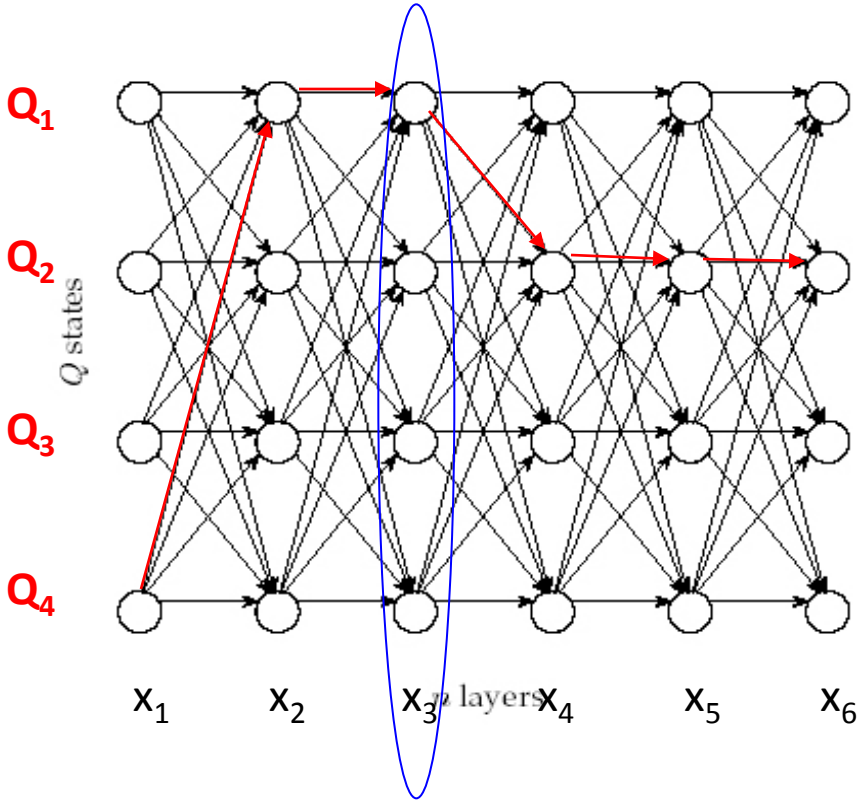
The Viterbi algorithm

- Every layer *i* emit one symbol x_i
- Every path from layer *1* to layer *n* has probability P($\boldsymbol{x},\boldsymbol{\pi}$)
- The path tells us which hidden state in layer *i* that emitted x_i
- The Viterbi algorithm finds the path that maximizes $P(\boldsymbol{x},\boldsymbol{\pi})$ among all possible paths



The Viterbi algorithm

- Dynamic programming
- Define $s_{k,i}$ as the probability of emitting the prefix $x_1 \ldots x_i$ and reaching the state k
- $s_{l,i+1} = e_l(x_{i+1}) \cdot \max_{k \in Q} \{s_{k,i} \cdot a_{kl}\}$
- The Viterbi algorithm runs in $O(n|Q|^2)$ time

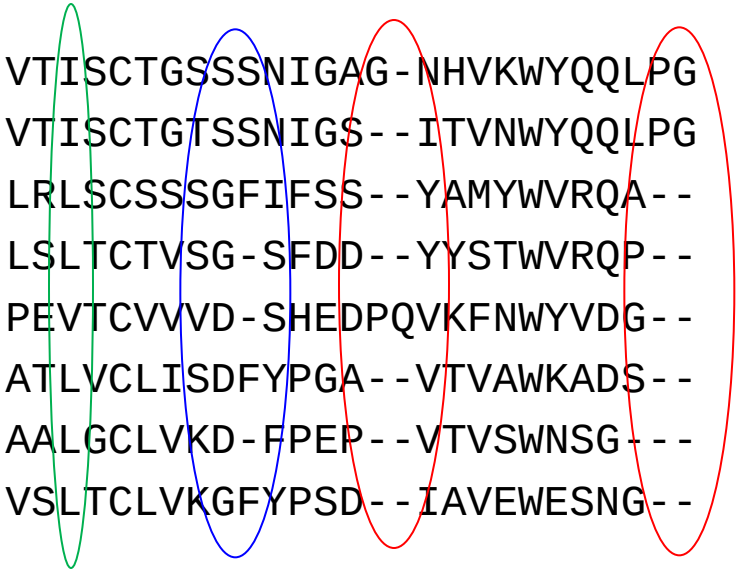


HMMs

- HMMs can be used for aligning a sequence against a protein family
- Conserved positions in the family corresponds to *n* sequentially linked *match* states $M_1,\ldots,M_n$ in the profile HMM
- HMMs handle gaps better than profiles do

Building a profile HMM

- Multiple alignment is used to construct the HMM model
- Assign each column to a *Match* state in HMM. Add I*nsertion* and *Deletion* state
- Estimate the emission probabilities according to amino acid counts in columns
- Estimate the transition probabilities between *Match*, *Deletion* and *Insertion* states



Match state 3:

$e_{M3}(L) = 5/8$

$e_{M3}(I) = 3/8$

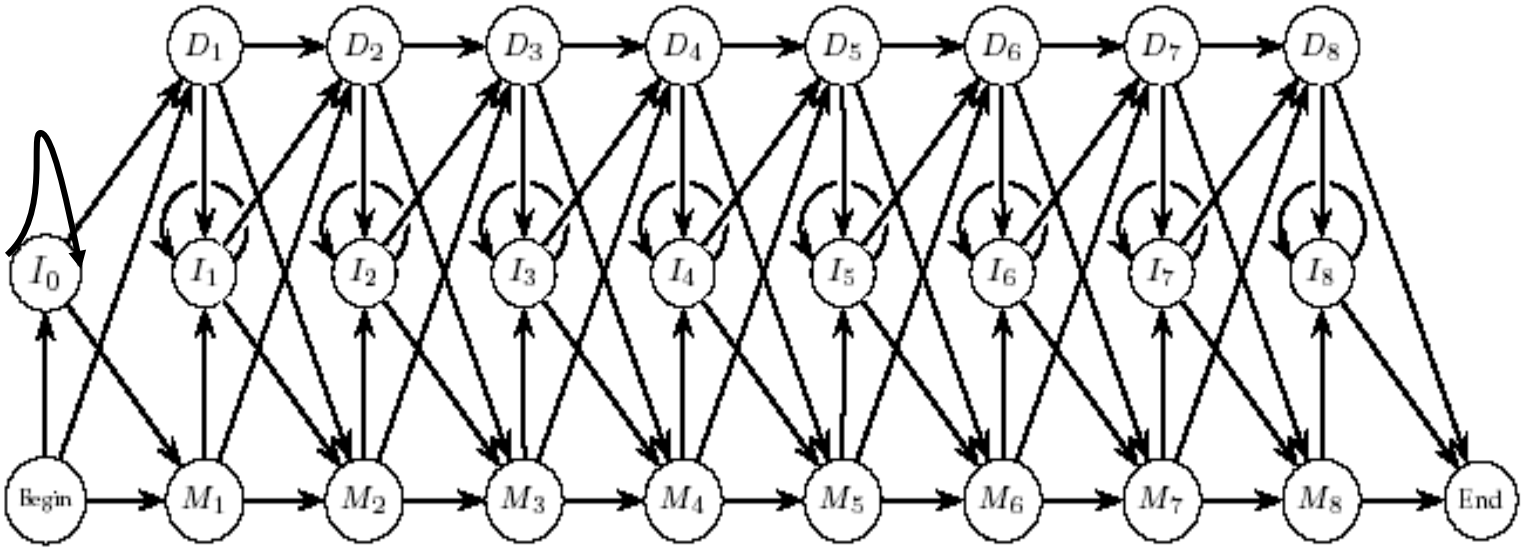
Modeled as insertions

Transitions from match state 9 to 10

$a_{M9,M10} = 5/8$

$a_{M9,D10} = 3/8$

Profile HMM



A profile HMM

Penalties in HMMs

Different penalties for opening a gap and extending the gap is naturally implemented in HMM

- $a_{MI} * a_{IM}$ = gap initiation penalty
- a_{II} = gap extension penalty

Pfam

- Pfam decribes *protein domains*
- Each protein domain family in Pfam has:
  - S*eed alignment*: manually verified multiple alignment of a representative set of sequences
  - *HMM* built from the seed alignment for further database searches
  - *Full alignment* generated automatically from the HMM
- The distinction between seed and full alignments facilitates Pfam updates
  - Seed alignments are stable resources
  - Full alignments can be updated with newly found amino acid sequences

Pfam

Pfam uses a tool called HMMER with the following architecture: