String Algorithms and Data Structures

Semi-Global and Local Alignment

CS 199-225

Brad Solomon

April 24, 2023

UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN

Department of Computer Science

Please fill out ICES Evaluations

Feedback is important for the development of the class

If not enough people fill it out, doesn't actually get recorded

Learning Objectives

Review global alignment dynamic programming

Improve the algorithm to account for local matches

Discuss further optimizations or alternatives

Approximate Pattern Matching

Input: A text *T*, a pattern *P*, and a distance *d*

Output: All positions in *T* where *P* has at most *d* mismatches or edits

P: word

T: There would have been a time for such a word

Alignment 1:
```
would
word
```

~~Not a match!~~

Distance 2 match!

Alignment 2:
```
word
word
```

Match!

Distance 0 match!

Edit Distance

Alignments:

Edit string (w/r/t *x*):

```
x: G C G T A T G C G G C T A - A C G C
   |  | | | | | | | | | | |   | | | |
y: G C - T A T G C G G C T A T A C G C
```

MMDMMMMMMMMMMIMMMM

Distance = 2

replacement (AKA substitution/mismatch)

```
x: G C G T A T G A G G C T A - A C G C
   |  | | | |  | | | | |  | | | |
y: G C - T A T G C G G C T A T A C G C
```

MMDMMMMRMMMMMIMMMM

Distance = 3

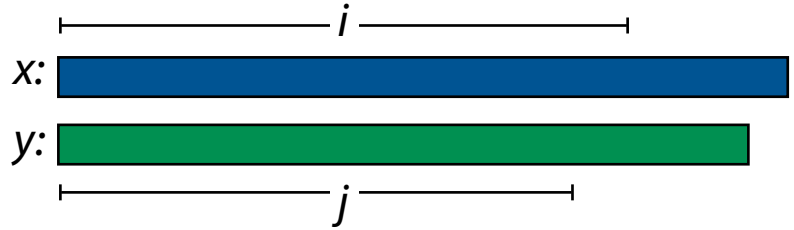
```
x: t h e   l o n g e s t - - - -
         | | | | | | |
y: - - - - l o n g e s t   d a y
```

DDDDMMMMMMMIIII

Distance = 8

Edit Distance

D[*i*, *j*]: edit distance between length-*i* prefix of *x* and length-*j* prefix of *y*



Optimal edit string for D[*i*, *j*] is built by ***extending a shorter optimal string by 1 operation***. 3 options:

Append **D** to transcript for D[*i*-1, *j*]

Append **I** to transcript for D[*i*, *j*-1]

Append **M** or **R** to transcript for D[*i*-1, *j*-1]

We choose based on whichever option has the fewest edits

Edit Distance

X: GTTTAA Y: GGTTTA

D[*5*, *6*]

MIMMMM

G	-	T	T	T	A
G	G	T	T	T	A

MIMMMMD

G	-	T	T	T	A	A
G	G	T	T	T	A	-

D[*6*, *6*][1]

D[*5*, *5*]

MRMMR

G	T	T	T	A
G	G	T	T	T

MRMMRM

G	T	T	T	A	A
G	G	T	T	T	A

D[*6*, *6*][2]

D[*6*, *5*]

MRMMRD

G	T	T	T	A	A
G	G	T	T	T	-

MRMMRDI

G	T	T	T	A	A	-
G	G	T	T	T	-	A

D[*6*, *6*][3]

Edit Distance: dynamic programming

$$D[i,j] = \min \begin{cases} D[i-1,j]+1 \\ D[i,j-1]+1 \\ D[i-1,j-1]+\delta(x[i-1],y[j-1]) \end{cases}$$

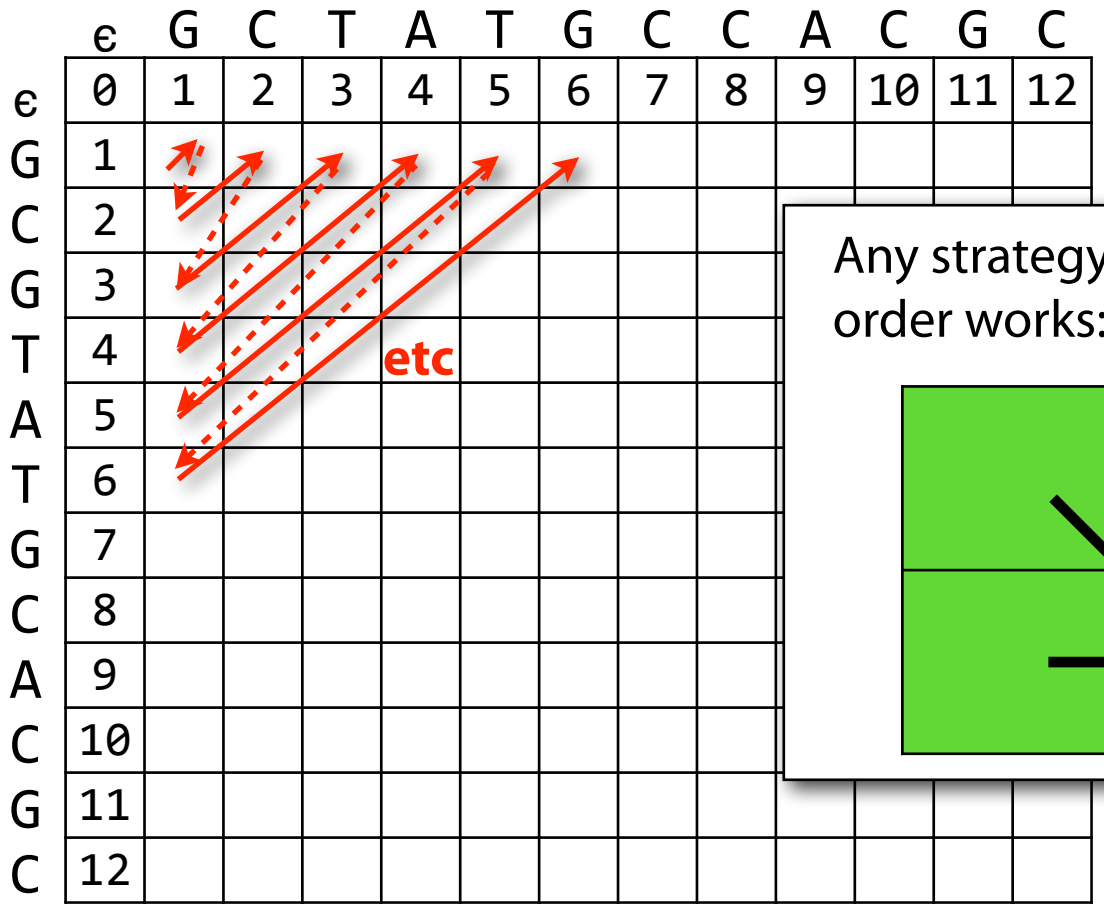
	ϵ	G	C	T	A	T	G	C	C	A	C	G	C
ϵ	0	1	2	3	4	5	6	7	8	9	10	11	12
G	1	0	1	2	3	4	5	6					
C	2	1	0	1	2	3	4	5					
G	3	2	1	1	2	3	3	4					
T	4	3	2	1	2	2	3	4					
A	5												
T	6												
G	7												
C	8												
A	9												
C	10												
G	11												
C	12												

What goes here in **i=4, j=7**?

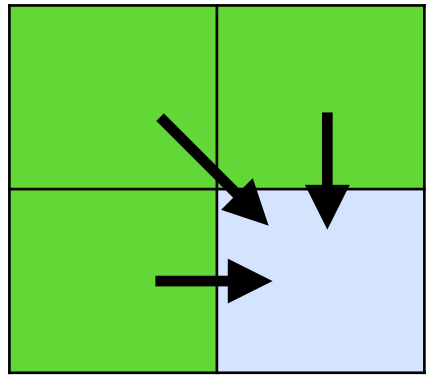
```
= min(D[i-1, j]+1,
      D[i, j-1]+1,
      D[i-1, j-1]+delt)
= min(4 + 1, 3 + 1, 3 + 1)
= 4
```

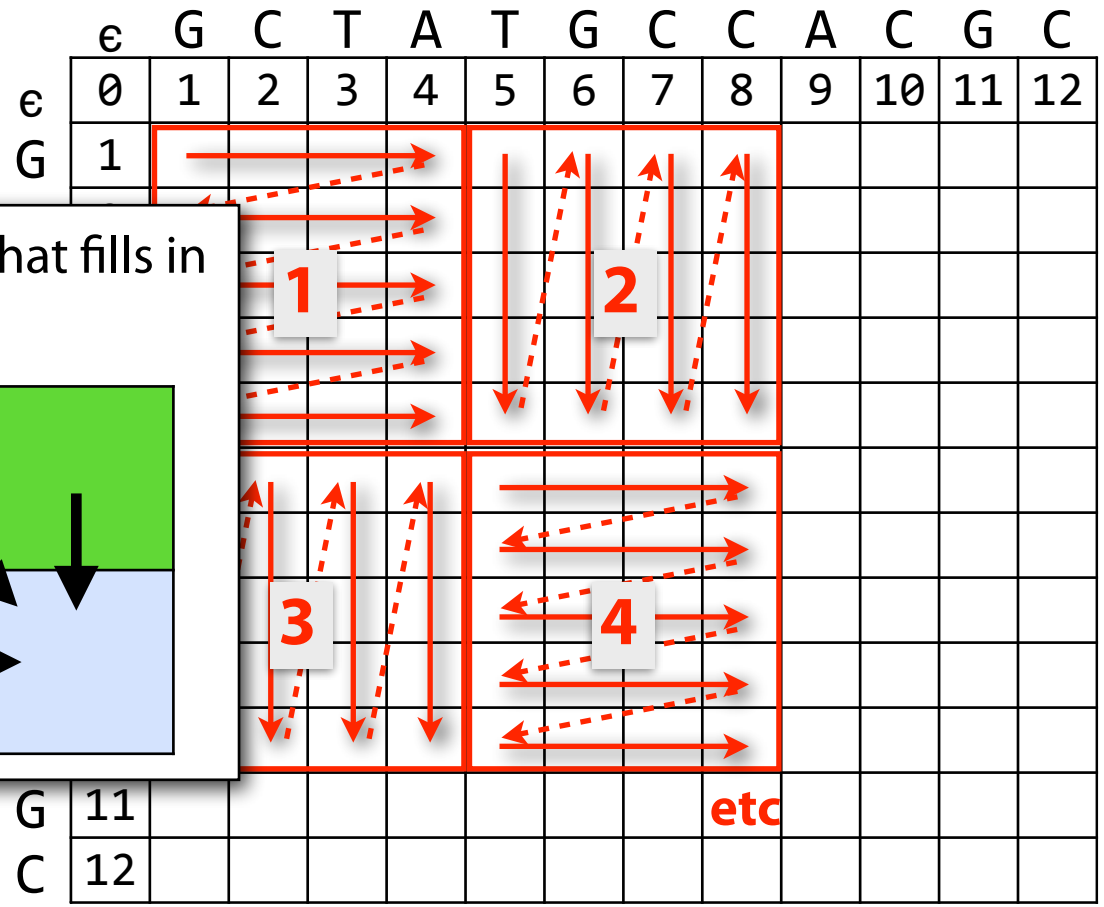
```
GC---GT      GC-GT--
GCTATGC      GCTATGC
MMIIIMR      MMIRMII
```

Edit Distance: dynamic programming



Any strategy that fills in order works:





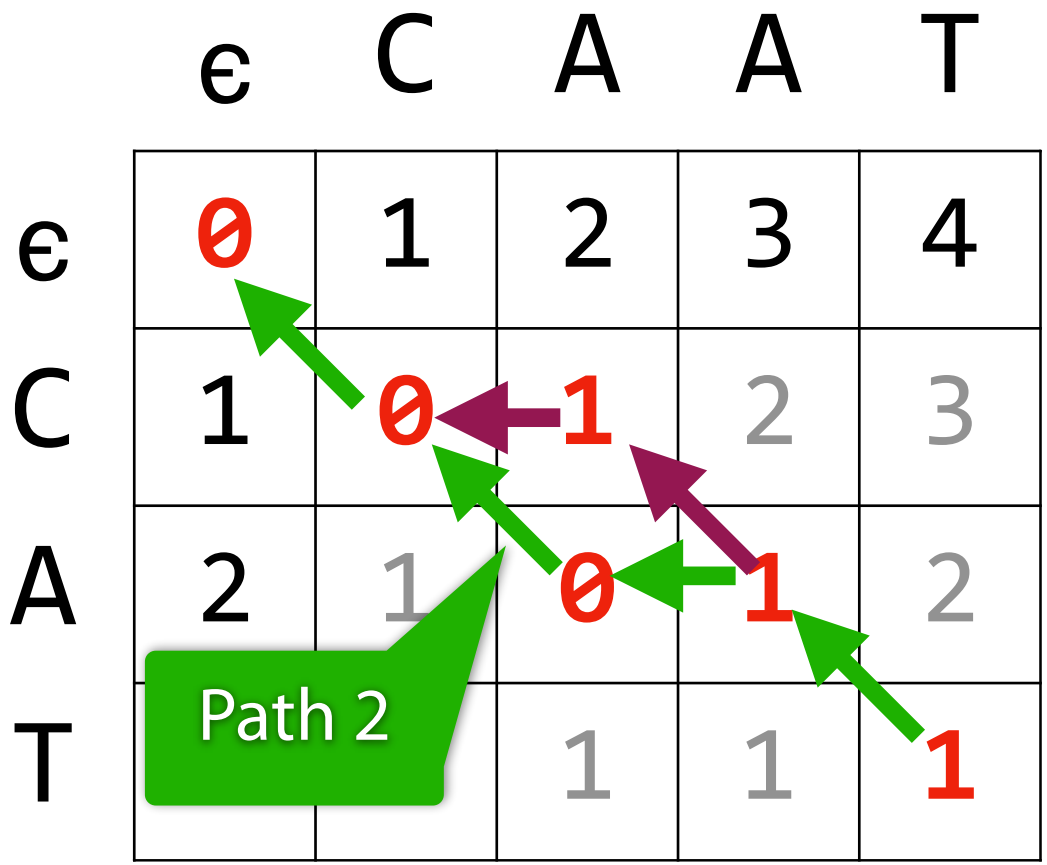
Dynamic Programming

	ϵ	G	C	T	A	T	G	C	C	A	C	G	C
ϵ	0	1	2	3	4	5	6	7	8	9	10	11	12
G	1	0	1	2	3	4	5	6	7	8	9	10	11
C	2	1	0	1	2	3	4	5	6	7	8	9	10
G	3	2	1	1	2	3	3	4	5	6	7	8	9
T	4	3	2	1	2	2	3	4	5	6	7	8	9
A	5	4	3	2	1	2	3	4	5	5	6	7	8
T	6	5	4	3	2	1	2	3	4	5	6	7	8
G	7	6	5	4	3	2	1	2	3	4	5	6	7
C	8	7	6	5	4	3	2	1	2	3	4	5	6
A	9	8	7	6	5	4	3	2	2	2	3	4	5
C	10	9	8	7	6	5	4	3	2	3	2	3	4
G	11	10	9	8	7	6	5	4	3	3	3	2	3
C	12	11	10	9	8	7	6	5	4	4	3	3	2

Edit Distance

Traceback corresponds to an optimal alignment / edit transcript

At each step, ask: which neighbor (⇖, ⇐ or ⇑) gave the minimum?



MIMM

```
C - A T
|   | |
C A A T
```

MMIM

```
C A - T
| |   |
C A A T
```

Dynamic Programming

	ϵ	G	C	T	A	T	G	C	C	A	C	G	C
ϵ	0	1	2	3	4	5	6	7	8	9	10	11	12
G	1	0	1	2	3	4	5	6	7	8	9	10	11
C	2	1	0	1	2	3	4	5	6	7	8	9	10
G	3	2	1	1	2	3	3	4	5	6	7	8	9
T	4	3	2	1	2	2	3	4	5	6	7	8	9
A	5	4	3	2	1	2	3	4	5	5	6	7	8
T	6	5	4	3	2	1	2	3	4	5	6	7	8
G	7	6	5	4	3	2	1	2	3	4	5	6	7
C	8	7	6	5	4	3	2	1	2	3	4	5	6
A	9	8	7	6	5	4	3	2	2	2	3	4	5
C	10	9	8	7	6	5	4	3	2	3	2	3	4
G	11	10	9	8	7	6	5	4	3	3	3	2	3
C	12	11	10	9	8	7	6	5	4	4	3	3	2

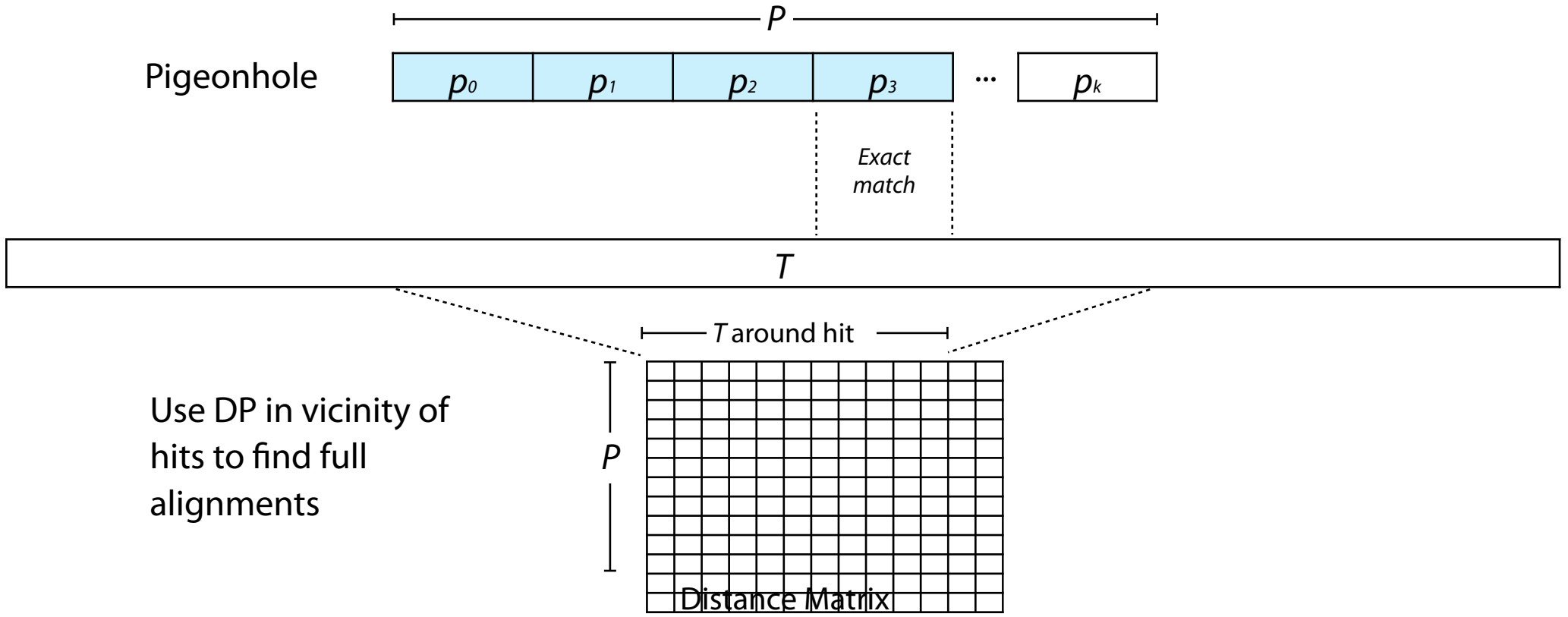
Alignment:

```
G C G T A T G - C A C G C
|| |  |||| |  |||||
G C - T A T G C C A C G C
```

MMDMMMMIMMMMM

Pattern Matching w/ Dynamic Programming

"Seed and extend" works for edit distance too!



Pattern Matching w/ Dynamic Programming

What if I want to match my full pattern but **NOT** my full text?

		T																			
		ϵ	A	A	C	C	C	T	A	T	G	T	C	A	T	G	C	C	T	T	G	G	A
P	ϵ																						
	T																						
	A																						
	C																						
	G																						
	T																						
	C																						
	A																						
	G																						
	C																						

Pattern Matching w/ Dynamic Programming

If I initialize my first row to all 0, I can start my match anywhere…

	ϵ	A	A	C	C	C	T	A	T	G	T	C	A	T	G	C	C	T	T	G	G	A
ϵ	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
T	1																					
A	2																					
C	3																					
G	4																					
T	5																					
C	6																					
A	7																					
G	8																					
C	9																					

Pattern Matching w/ Dynamic Programming

Where is my best match? What is the alignment?

	ϵ	A	A	C	C	C	T	A	T	G	T	C	A	T	G	C	C	T	T	G	G	A
ϵ	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
T	1	1	1	1	1	1	0	1	0	1	0	1	1	0	1	1	1	0	0	1	1	1
A	2	1	1	2	2	2	1	0	1	1	1	1	1	1	1	2	2	1	1	1	2	1
C	3	2	2	1	2	2	2	1	1	2	2	1	2	2	2	1	2	2	2	2	2	2
G	4	3	3	2	2	3	3	2	2	1	2	2	2	3	2	2	2	3	3	2	2	3
T	5	4	4	3	3	3	3	3	2	2	1	2	3	2	3	3	3	2	3	3	3	3
C	6	5	5	4	3	3	4	4	3	3	2	1	2	3	3	3	3	3	3	4	4	4
A	7	6	5	5	4	4	4	4	4	4	3	2	1	2	3	4	4	4	4	4	5	4
G	8	7	6	6	5	5	5	5	5	4	4	3	2	2	2	3	4	5	5	4	4	5
C	9	8	7	6	6	5	6	6	6	5	5	4	3	3	3	2	3	4	5	5	5	5

Pattern Matching w/ Dynamic Programming

Where is my best match? What is the alignment?

	ϵ	A	A	C	C	C	T	A	T	G	T	C	A	T	G	C	C	T	T	G	G	A
ϵ	0	0	0	0	0	0	0	0	0													
T	1	1	1	1	1	1	0	1	0													
A	2	1	1	2	2	2	1	0	1													
C	3	2	2	1	2	2	2	1	1	2	2	1	2	2	2	1	2	2	2	2	2	2
G	4	3	3	2	2	3	3	2	2	1	2	2	2	3	2	2	2	3	3	2	2	3
T	5	4	4	3	3	3	3	3	2	2	1	2	3	2	3	3	3	2	3	3	3	3
C	6	5	5	4	3	3	4	4	3	3	2	1	2	3	3	3	3	3	3	4	4	4
A	7	6	5	5	4	4	4	4	4	4	3	2	1	2	3	4	4	4	4	4	5	4
G	8	7	6	6	5	5	5	5	5	4	4	3	2	2	2	3	4	5	5	4	4	5
C	9	8	7	6	6	5	6	6	6	5	5	4	3	3	3	2	3	4	5	5	5	5

```
T:  AACCCTATGTCATGCCTTGGA
        ||  |||| ||
P:      TACGTCA-GC
```

Pattern Matching w/ Dynamic Programming

How many matches do I have with no more than 3 edits?

	ϵ	A	A	C	C	C	T	A	T	G	T	C	A	T	G	C	C	T	T	G	G	A
ϵ	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
T	1	1	1	1	1	1	0	1	0	1	0	1	1	0	1	1	1	0	0	1	1	1
A	2	1	1	2	2	2	1	0	1	1	1	1	1	1	1	2	2	1	1	1	2	1
C	3	2	2	1	2	2	2	1	1	2	2	1	2	2	2	1	2	2	2	2	2	2
G	4	3	3	2	2	3	3	2	2	1	2	2	2	3	2	2	2	3	3	2	2	3
T	5	4	4	3	3	3	3	3	2	2	1	2	3	2	3	3	3	2	3	3	3	3
C	6	5	5	4	3	3	4	4	3	3	2	1	2	3	3	3	3	3	3	4	4	4
A	7	6	5	5	4	4	4	4	4	4	3	2	1	2	3	4	4	4	4	4	5	4
G	8	7	6	6	5	5	5	5	5	4	4	3	2	2	2	3	4	5	5	4	4	5
C	9	8	7	6	6	5	6	6	6	5	5	4	3	3	3	2	3	4	5	5	5	5

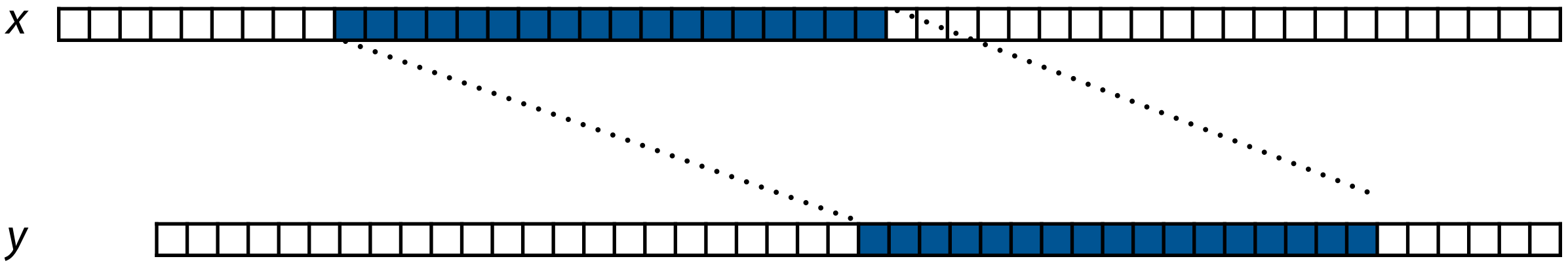
Pattern Matching w/ Dynamic Programming

How many matches do I have with no more than 3 edits?

	ϵ	A	A	C	C	C	T	A	T	G	T	C	A	T	G	C	C	T	T	G	G	A
ϵ	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
T	1	1	1	1	1	1	0	1	0	1	0	1	1	0	1	1	1	0	0	1	1	1
A	2	1	1	2	2	2	1	0	1	1	1	1	1	1	1	2	2	1	1	1	2	1
C	3	2	2	1	2	2	2	1	1	2	2	1	2	2	2	1	2	2	2	2	2	2
G	4	3	3	2	2	3	3	2	2	1	2	2	2	3	2	2	2	3	3	2	2	3
T	5	4	4	3	3	3	3	3	2	2	1	2	3	2	3	3	3	2	3	3	3	3
C	6	5	5	4	3	3	4	4	3	3	2	1	2	3	3	3	3	3	3	4	4	4
A	7	6	5	5	4	4	4	4	4	4	3	2	1	2	3	4	4	4	4	4	5	4
G	8	7	6	6	5	5	5	5	5	4	4	3	2	2	2	3	4	5	5	4	4	5
C	9	8	7	6	6	5	6	6	6	5	5	4	3	3	3	2	3	4	5	5	5	5

Local Alignment

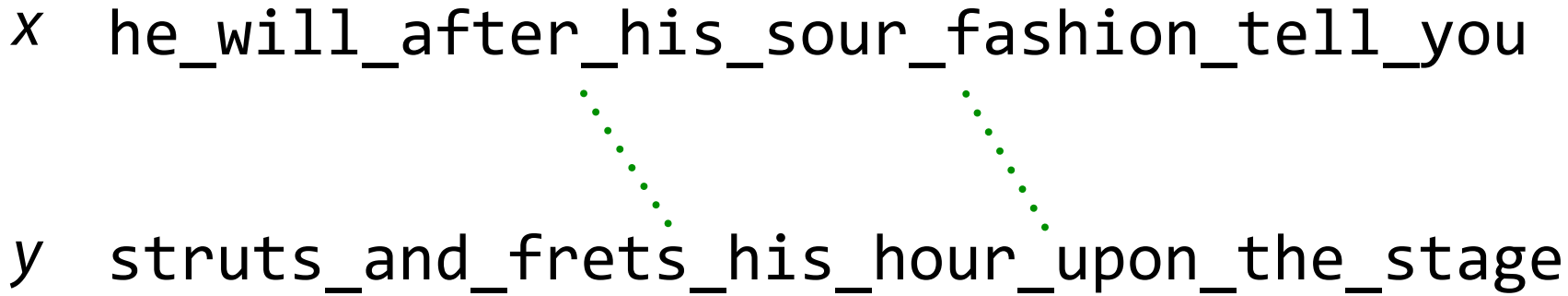
Local alignment is the process of comparing two strings *x* and *y* and finding the optimal alignment value of a *substring* of *x* to a *substring* of *y*.



How does this alter our dynamic programming algorithm?

Local Alignment

Local alignment is the process of comparing two strings *x* and *y* and finding the optimal alignment value of a *substring* of *x* to a *substring* of *y*.

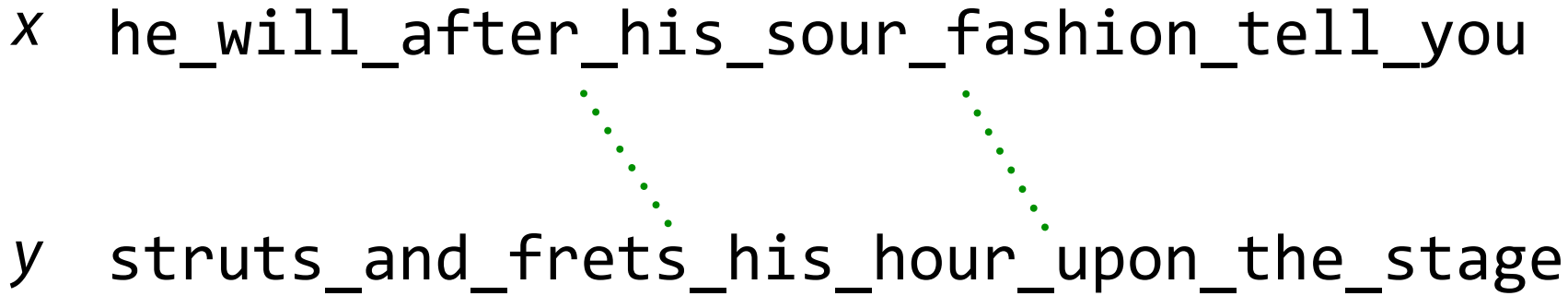



```
_his_sour_
||||| ||||
_his_hour_
```

Naive: Perform a global alignment for every possible pair of substrings.

Local Alignment

Local alignment is the process of comparing two strings *x* and *y* and finding the optimal alignment value of a *substring* of *x* to a *substring* of *y*.




```
_his_sour_
||||| ||||
_his_hour_
```

Better: Lets do this in O(|X| |Y|) — same big O as the global! How?

From global to local alignments

To match TAC with any substring of AAC:

To match AAC with any substring of TAC:

To match any substring of AAC with any substring of TAC:

	ϵ	A	A	C
ϵ				
T				
A				
C				

From global to local alignments

Change #1: Give a scoring function for edits

Change #2: Convert from distance to similarity

	ϵ	A	A	C
ϵ	0	0	0	0
T	0	0	0	0
A	0	1	1	0
C	0	0	0	2

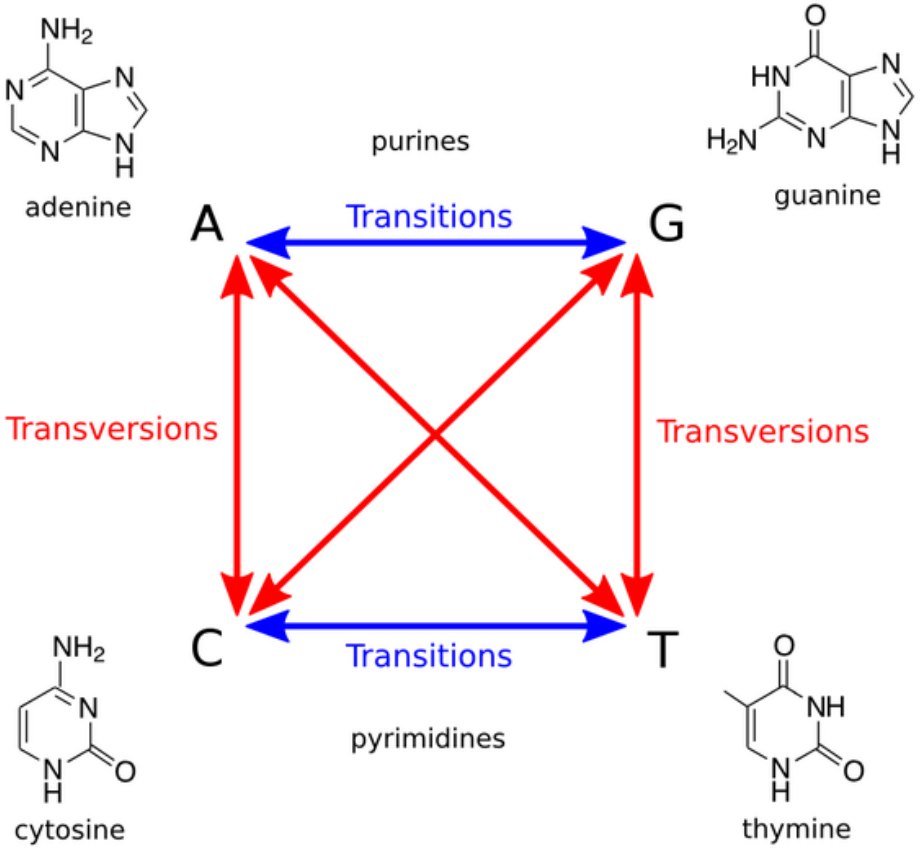
Change #3: Allow our alignment to start from any position in the matrix

Dynamic Programming: Scoring Function

Transitions are A ↔ G and C ↔ T changes

Transversions are A ↔ C, A ↔ T, C ↔ G, G ↔ T

For random mutations, transitions should be half as frequent as transversions...



… **but** the transition to transversion ratio is ~2.1 in humans

Dynamic Programming: Scoring Function

	A	C	G	T	-
A	0	4	2	4	8
C	4	0	4	2	8
G	2	4	0	4	8
T	4	2	4	0	8
-	8	8	8	8	

2 *Transitions* (A ↔ G, C T)

4 *Transversions*

8 *Gaps*

Dynamic Programming: Scoring Function

Let $D[0, j] = \sum_{k=0}^{j-1} s(-, y[k])$, and let $D[i, 0] = \sum_{k=0}^{i-1} s(x[k], -)$

Otherwise, let $D[i, j] = \min \begin{cases} D[i-1, j] + s(x[i-1], -) \\ D[i, j-1] + s(-, y[j-1]) \\ D[i-1, j-1] + s(x[i-1], y[j-1]) \end{cases}$

$s(a, b)$ assigns a cost to a particular gap or substitution

$s(a, b)$:

	A	C	G	T	-
A	0	4	2	4	8
C	4	0	4	2	8
G	2	4	0	4	8
T	4	2	4	0	8
-	8	8	8	8	

2 — *Transitions* (A↔G, C↔T)

4 — *Transversions* (everything else)

8 — *Gaps*

Dynamic Programming: Scoring Function

Adding a scoring function changes little in the method:

	ϵ	T	A	T
ϵ	**0**	**1**	**2**	**3**
T	**1**			
A	**2**			
C	**3**			

	ϵ	T	A	T
ϵ	**0**	**8**	**16**	**24**
T	**8**			
A	**16**			
C	**24**			

$s(a, b)$

	A	C	G	T	-
A	0	1	1	1	1
C	1	0	1	1	1
G	1	1	0	1	1
T	1	1	1	0	1
-	1	1	1	1	

	A	C	G	T	-
A	0	4	2	4	8
C	4	0	4	2	8
G	2	4	0	4	8
T	4	2	4	0	8
-	8	8	8	8	

Dynamic Programming: Scoring Function

Adding a scoring function changes little in the method:

	ϵ	T	A	T
ϵ	0	1	2	3
T	1	0	1	2
A	2	1	0	1
C	3	2	1	1

	ϵ	T	A	T
ϵ	0	8	16	24
T	8	0	8	16
A	16	8	0	8
C	24	16	8	2

$s(a, b)$

	A	C	G	T	-
A	0	1	1	1	1
C	1	0	1	1	1
G	1	1	0	1	1
T	1	1	1	0	1
-	1	1	1	1	

	A	C	G	T	-
A	0	4	2	4	8
C	4	0	4	2	8
G	2	4	0	4	8
T	4	2	4	0	8
-	8	8	8	8	

Dynamic Programming: Scoring Function

Adding a scoring function changes little in the method:

	ϵ	T	A	T	G	T	C	A	T	G	C
ϵ	**0**	**8**	**16**	**24**	**32**	**40**	**48**	**56**	**64**	**72**	**80**
T	**8**	**0**	**8**	**16**	**24**	**32**	**40**	**48**	**56**	**64**	**72**
A	**16**	**8**	**0**	**8**	**16**	**24**	**32**	**40**	**48**	**56**	**64**
C	**24**	**16**	**8**	**2**	**10**	**18**	**24**	**32**	**40**	**48**	**56**
G	**32**	**24**	**16**	**10**	**2**	**10**	**18**	**26**	**34**	**40**	**48**
T	**40**	**32**	**24**	**16**	**10**	**2**	**10**	**18**	**26**	**34**	**42**
C	**48**	**40**	**32**	**24**	**18**	**10**	**2**	**10**	**18**	**26**	**34**
A	**56**	**48**	**40**	**32**	**26**	**18**	**10**	**2**	**10**	**18**	**26**
G	**64**	**56**	**48**	**40**	**32**	**26**	**18**	**10**	**6**	**10**	**18**
C	**72**	**64**	**56**	**48**	**40**	**34**	**26**	**18**	**12**	**10**	**10**

← Optimal global alignment

$s(a, b)$

	A	C	G	T	-
A	0	4	2	4	8
C	4	0	4	2	8
G	2	4	0	4	8
T	4	2	4	0	8
-	8	8	8	8	

Dynamic Programming: Scoring Function

Adding a scoring function changes little in the method:

	ϵ	T	A	T	G	T	C	A	T	G	C
ϵ	0	8	16	24	32	40	48	56	64	72	80
T	8	0	8	16	24	32	40	48	56	64	72
A	16	8	0	8	16	24	32	40	48	56	64
C	24	16	8	2	10	18	24	32	40	48	56
G	32	24	16	10	2	10	18	26	34	40	48
T	40	32	24	16	10	2	10	18	26	34	42
C	48	40	32	24	18	10	2	10	18	26	34
A	56	48	40	32	26	18	10	2	10	18	26
G	64	56	48	40	32	26	18	10	6	10	18
C	72	64	56	48	40	34	26	18	12	10	10

$s(a, b)$

	A	C	G	T	-
A	0	4	2	4	8
C	4	0	4	2	8
G	2	4	0	4	8
T	4	2	4	0	8
-	8	8	8	8	

```
T A C G T C A - G C
|| |  | | | |  | |
T A T G T C A T G C
```

+2 (transition)

+8 (gap)

Dynamic Programming: Similarity

Similarity changes the scoring function but also our algorithm

	ϵ	T	A	T
ϵ	0	8	16	24
T	8	0	8	16
A	16	8	0	8
C	24	16	8	2

	ϵ	T	A	T
ϵ	0	8	16	24
T	8			
A	16			
C	24			

	A	C	G	T	-
A	0	4	2	4	8
C	4	0	4	2	8
G	2	4	0	4	8
T	4	2	4	0	8
-	8	8	8	8	

	A	C	G	T	-
A	1	-3	-1	-3	-7
C	-3	1	-3	-1	-7
G	-1	-3	1	-3	-7
T	-3	-1	-3	1	-7
-	-7	-7	-7	-7	

Dynamic Programming: Similarity

Similarity changes the scoring function but also our algorithm

	ϵ	T	A	T
ϵ	0	8	16	24
T	8	0	8	16
A	16	8	0	8
C	24	16	8	2

	ϵ	T	A	T
ϵ	0	-7	-14	-21
T	-7	1	-6	-13
A	-14	-6	2	-5
C	-21	-13	-5	1

	A	C	G	T	-
A	0	4	2	4	8
C	4	0	4	2	8
G	2	4	0	4	8
T	4	2	4	0	8
-	8	8	8	8	

	A	C	G	T	-
A	1	-3	-1	-3	-7
C	-3	1	-3	-1	-7
G	-1	-3	1	-3	-7
T	-3	-1	-3	1	-7
-	-7	-7	-7	-7	

Dynamic Programming: Similarity

Similarity changes the scoring function but also our algorithm

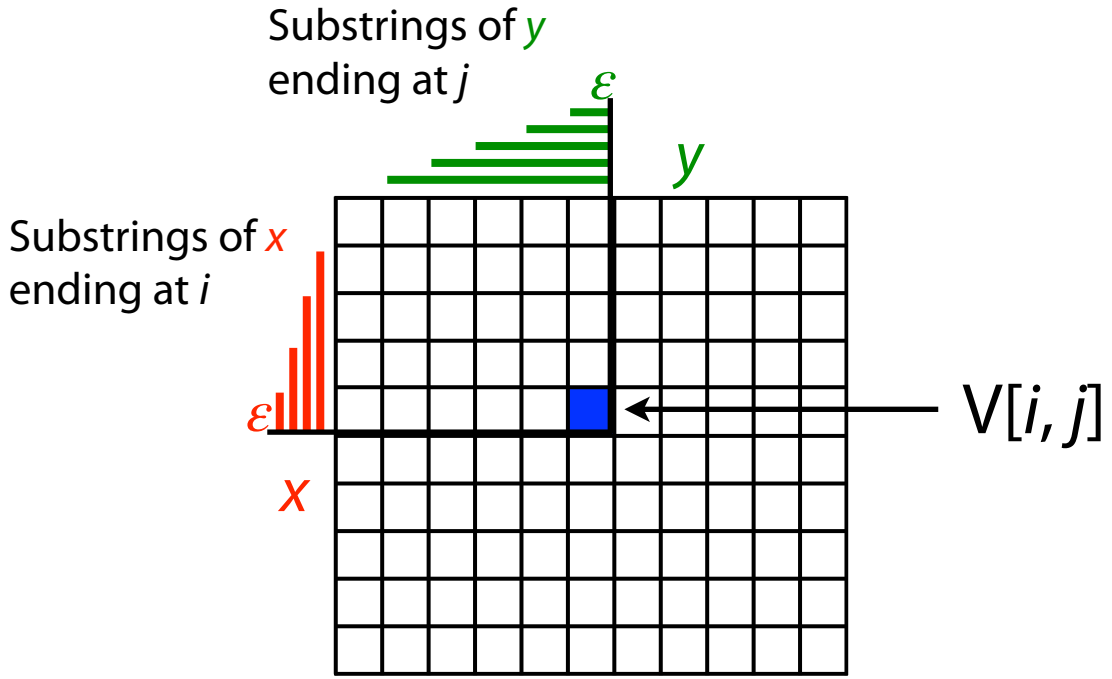
Given a scoring function s :

$$D[i,j] = \min \begin{cases} D[i-1,j] + s(x[i-1],-) \\ D[i,j-1] + s(-,y[j-1]) \\ D[i-1,j-1] + s(x[i-1],y[j-1]) \end{cases}$$

$$V[i,j] = \max \begin{cases} V[i-1,j] + s(x[i-1],-) \\ V[i,j-1] + s(-,y[j-1]) \\ V[i-1,j-1] + s(x[i-1],y[j-1]) \end{cases}$$

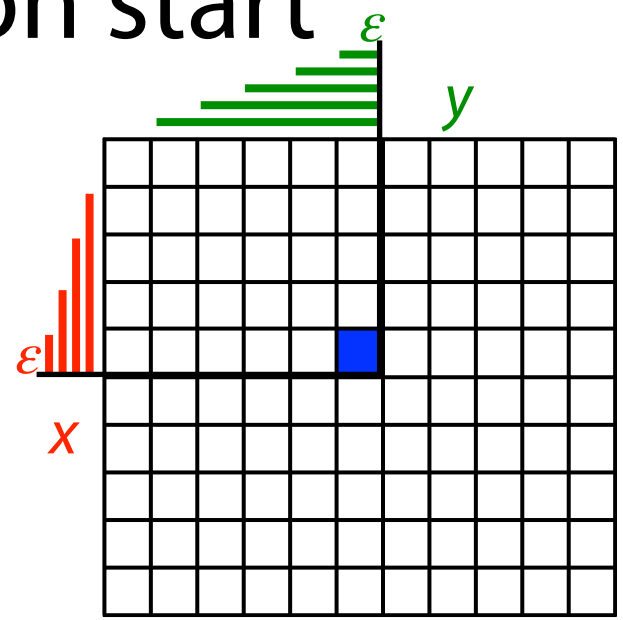
Dynamic Programming: Any position start

Let V[*i*, *j*] be the optimal global alignment among substrings of *x* ending at *i* and substrings of *y* ending at *j*. The substrings may be empty.



How do we calculate V[*i*, *j*]?

Dynamic Programming: Any position start



Let V[*i*, *j*] be the optimal global alignment among substrings of *x* ending at *i* and substrings of *y* ending at *j*. The substrings may be empty.

There are only so many possibilities:

Empty: let both substrings be empty, global alignment value = 0

Vertical: append **D** to transcript for V[*i*-1, *j*], add penalty

Horizontal: append **I** to transcript for V[*i*, *j*-1], add penalty

Diagonal: append **M** or **R** to transcript for V[*i*-1, *j*-1], add match bonus or replacement penalty as appropriate

Dynamic Programming: Any position start

Let $V[0, j] = 0$, and let $V[i, 0] = 0$

$$\text{Otherwise, let } V[i,j] = \max \begin{cases} V[i-1,j] + s(x[i-1], -) \\ V[i, j-1] + s(-, y[j-1]) \\ V[i-1, j-1] + s(x[i-1], y[j-1]) \\ 0 \end{cases}$$

	ϵ	T	C	A	G
ϵ	0	0	0	0	0
C	0				
A	0				
C	0				

	A	C	G	T	-
A	1	-1	-1	-1	-1
C	-1	1	-1	-1	-1
G	-1	-1	1	-1	-1
T	-1	-1	-1	1	-1
-	-1	-1	-1	-1	

Dynamic Programming: Any position start

Let $V[0, j] = 0$, and let $V[i, 0] = 0$

$$\text{Otherwise, let } V[i,j] = \max \begin{cases} V[i-1,j] + s(x[i-1], -) \\ V[i, j-1] + s(-, y[j-1]) \\ V[i-1, j-1] + s(x[i-1], y[j-1]) \\ 0 \end{cases}$$

	ϵ	T	C	A	G
ϵ	0	0	0	0	0
C	0	0	1	0	0
A	0	0	0	2	1
C	0	0	1	1	1

	A	C	G	T	-
A	1	-1	-1	-1	-1
C	-1	1	-1	-1	-1
G	-1	-1	1	-1	-1
T	-1	-1	-1	1	-1
-	-1	-1	-1	-1	

Local Alignment: Putting it all together

A local alignment scoring function must:

- Assign negative weights to edits
- Assign positive weights to matches
- Not shown: There are constraints on what our scores can be

	A	C	G	T	-
A	1	-3	-1	-3	-1
C	-3	1	-3	-1	-1
G	-1	-3	1	-3	-1
T	-3	-1	-3	1	-1
-	-1	-1	-1	-1	

Local Alignment dynamic programming adds a fourth option to each step:

$$V[i,j] = \max \begin{cases} V[i-1,j] + s(x[i-1], -) \\ V[i,j-1] + s(-, y[j-1]) \\ V[i-1,j-1] + s(x[i-1], y[j-1]) \\ 0 \end{cases}$$

Local Alignment: Example

	ϵ	T	A	T	A	T	G	C	G	G	C	G	T
ϵ	0	0	0	0	0	0	0	0	0	0	0	0	0
G	0												
G	0												
T	0												
A	0												
T	0												
G	0												
C	0												
T	0												
G	0												
G	0												
C	0												
G	0												

$$s(a, b)$$

	A	C	G	T	-
A	2	-4	-4	-4	-6
C	-4	2	-4	-4	-6
G	-4	-4	2	-4	-6
T	-4	-4	-4	2	-6
-	-6	-6	-6	-6	

Local Alignment: Example

	ϵ	T	A	T	A	T	G	C	G	G	C	G	T
ϵ	0	0	0	0	0	0	0	0	0	0	0	0	0
G	0	0	0	0	0	0	2	0	2	2	0	2	0
G	0	0	0	0	0	0	2	0	2	4	0	2	0
T	0	2	0	2	0	2	0	0	0	0	0	0	4
A	0	0	4	0									
T	0												
G	0												
C	0												
T	0												
G	0												
G	0												
C	0												
G	0												

$s(a, b)$

	A	C	G	T	-
A	2	-4	-4	-4	-6
C	-4	2	-4	-4	-6
G	-4	-4	2	-4	-6
T	-4	-4	-4	2	-6
-	-6	-6	-6	-6	

Local Alignment: Example

	ϵ	T	A	T	A	T	G	C	G	G	C	G	T
ϵ	0	0	0	0	0	0	0	0	0	0	0	0	0
G	0	0	0	0	0	0	2	0	2	2	0	2	0
G	0	0	0	0	0	0	2	0	2	4	0	2	0
T	0	2	0	2	0	2	0	0	0	0	0	0	4
A	0	0	4	0	4	0	0	0	0	0	0	0	0
T	0	2	0	6	0	6	0	0	0	0	0	0	2
G	0	0	0	0	2	0	8	2	2	2	0	2	0
C	0	0	0	0	0	0	2	10	4	0	4	0	0
T	0	2	0	2	0	2	0	4	6	0	0	0	2
G	0	0	0	0	0	0	4	0	6	8	2	2	0
G	0	0	0	0	0	0	2	0	2	8	4	4	0
C	0	0	0	0	0	0	0	4	0	2	10	4	0
G	0	0	0	0	0	0	2	0	6	2	4	12	6

$$s(a, b)$$

	A	C	G	T	-
A	2	-4	-4	-4	-6
C	-4	2	-4	-4	-6
G	-4	-4	2	-4	-6
T	-4	-4	-4	2	-6
-	-6	-6	-6	-6	

Local Alignment: Example

	ϵ	T	A	T	A	T	G	C	G	G	C	G	T
ϵ	0	0	0	0	0	0	0	0	0	0	0	0	0
G	0	0	0	0	0	0	2	0	2	2	0	2	0
G	0	0	0	0	0	0	2	0	2	4	0	2	0
T	0	2	0	2	0	2	0	0	0	0	0	0	4
A	0	0	4	0	4	0	0	0	0	0	0	0	0
T	0	2	0	6	0	6	0	0	0	0	0	0	2
G	0	0	0	0	2	0	8	2	2	2	0	2	0
C	0	0	0	0	0	0	2	10	4	0	4	0	0
T	0	2	0	2	0	2	0	4	6	0	0	0	2
G	0	0	0	0	0	0	4	0	6	8	2	2	0
G	0	0	0	0	0	0	2	0	2	8	4	4	0
C	0	0	0	0	0	0	0	4	0	2	10	4	0
G	0	0	0	0	0	0	2	0	6	2	4	12	6

Where is my best match?

When do I stop?

Local Alignment: Example

	ϵ	T	A	T	A	T	G	C	G	G	C	G	T
ϵ	0	0	0	0	0	0	0	0	0	0	0	0	0
G	0	0	0	0	0	0	2	0	2	2	0	2	0
G	0	0	0	0	0	0	2	0	2	4	0	2	0
T	0	2	0	2	0	2	0	0	0	0	0	0	4
A	0	0	4	0	4	0	0	0	0	0	0	0	0
T	0	2	0	6	0	6	0	0	0	0	0	0	2
G	0	0	0	0	2	0	8	2	2	2	0	2	0
C	0	0	0	0	0	0	2	10	4	0	4	0	0
T	0	2	0	2	0	2	0	4	6	0	0	0	0
G	0	0	0	0	0	0	4	0	6	8	2	2	0
G	0	0	0	0	0	0	2	0	2	8	4	4	0
C	0	0	0	0	0	0	0	4	0	2	10	4	0
G	0	0	0	0	0	0	2	0	6	2	4	12	6

Traceback from largest value

Stop when 0

```
x : G G T A T G C T G G C G C T A
        | | | | |   | | | | |
y : T A T A T G C - G G C G T T T
```

	ϵ	T	A	T	A	T	G	C	G	G	C	G	T	T	T
ϵ	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
G	0	0	0	0	0	0	2	0	2	2	0				
G	0	0	0	0	0	0	2	0	2	4	0				
T	0	2	0	2	0	2	0	0	0	0	0				
A	0	0	4	0	4	0	0	0	0	0	0				
T	0	2	0	6	0	6	0	0	0	0	0	0	2	2	2
G	0	0	0	0	2	0	8	2	2	2	0	2	0	0	0
C	0	0	0	0	0	0	2	10	4	0	4	0	0	0	0
T	0	2	0	2	0	2	0	4	6	0	0	0	2	2	2
G	0	0	0	0	0	0	4	0	6	8	2	2	0	0	0
G	0	0	0	0	0	0	2	0	2	8	4	4	0	0	0
C	0	0	0	0	0	0	0	4	0	2	10	4	0	0	0
G	0	0	0	0	0	0	2	0	6	2	4	12	6	0	0
C	0	0	0	0	0	0	0	4	0	2	4	6	8	2	0
T	0	2	0	2	0	2	0	0	0	0	0	0	8	10	4
A	0	0	4	0	4	0	0	0	0	0	0	0	2	4	6

```
x : G G T A T G C T G G C G C T A
        | | | | |   | | | |
y : T A T A T G C - G G C G T T T
```

BLOSUM62

Some amino acid substitutions have a smaller impact on structure & function than others. BLOSUM62 elements are, roughly speaking, log-odds of observing these substitutions between two related proteins

	Ala	Arg	Asn	Asp	Cys	Gln	Glu	Gly	His	Ile	Leu	Lys	Met	Phe	Pro	Ser	Thr	Trp	Tyr	Val
Ala	4																			
Arg	-1	5																		
Asn	-2	0	6																	
Asp	-2	-2	1	6																
Cys	0	-3	-3	-3	9															
Gln	-1	1	0	0	-3	5														
Glu	-1	0	0	2	-4	2	5													
Gly	0	-2	0	-1	-3	-2	-2	6												
His	-2	0	1	-1	-3	0	0	-2	8											
Ile	-1	-3	-3	-3	-1	-3	-3	-4	-3	4										
Leu	-1	-2	-3	-4	-1	-2	-3	-4	-3	2	4									
Lys	-1	2	0	-1	-3	1	1	-2	-1	-3	-2	5								
Met	-1	-1	-2	-3	-1	0	-2	-3	-2	1	2	-1	5							
Phe	-2	-3	-3	-3	-2	-3	-3	-3	-1	0	0	-3	0	6						
Pro	-1	-2	-2	-1	-3	-1	-1	-2	-2	-3	-3	-1	-2	-4	7					
Ser	1	-1	1	0	-1	0	0	0	-1	-2	-2	0	-1	-2	-1	4				
Thr	0	-1	0	-1	-1	-1	-1	-2	-2	-1	-1	-1	-1	-2	-1	1	5			
Trp	-3	-3	-4	-4	-2	-2	-3	-2	-2	-3	-2	-3	-1	1	-4	-3	-2	11		
Tyr	-2	-2	-2	-3	-2	-1	-2	-3	2	-1	-1	-2	-1	3	-3	-2	-2	2	7	
Val	0	-3	-3	-3	-1	-2	-2	-3	-3	3	1	-2	1	-1	-2	-2	0	-3	-1	4

Rare; larger effect on structure/function

Common; modest effect on structure/function

negative

positive

Matrix is symmetric

Amino acids