

String Algorithms and Data Structures

Approximate Pattern Matching

CS 199-225

April 17, 2023

Brad Solomon



UNIVERSITY OF
ILLINOIS
URBANA - CHAMPAIGN

Department of Computer Science

End of Semester Material Votes:

67 votes (in weight) for Dynamic Programming

61 votes (in weight) for Markov Chains

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: word

Alignment 2: word

~~Not a match!~~

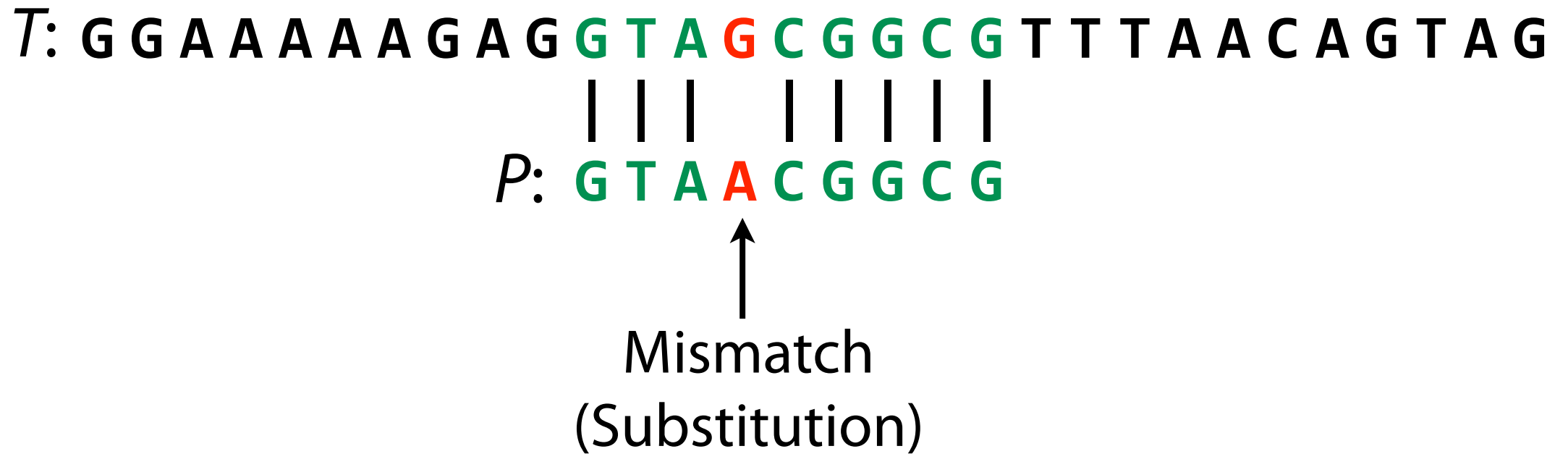
Match!

Distance 2 match!

Distance 0 match!

Hamming Distance

The number of **substitutions** required to turn one string into another



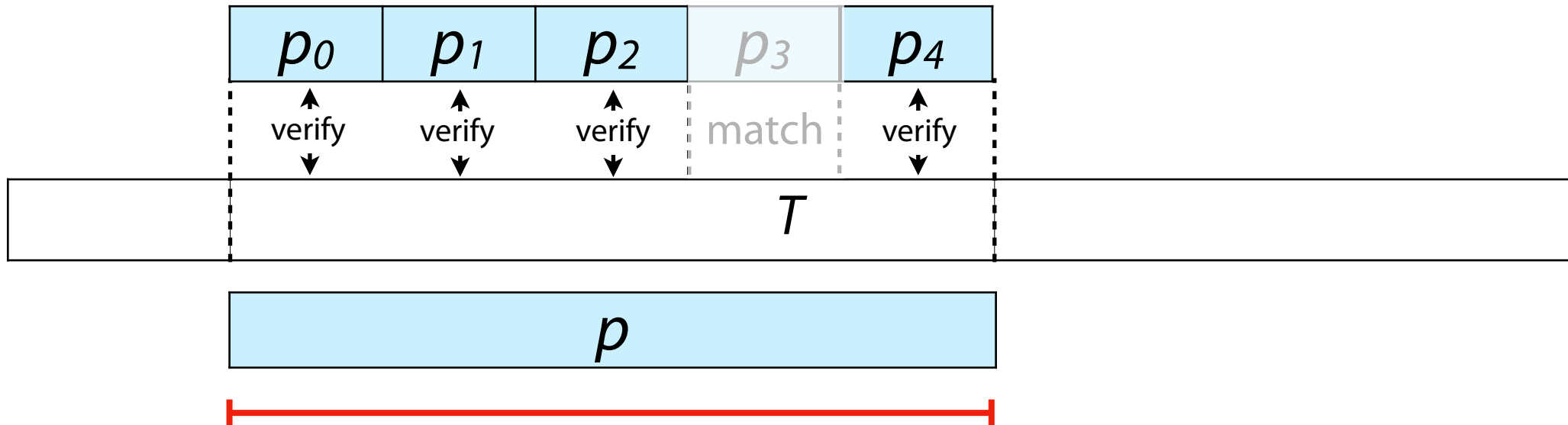
Approximate Pattern Matching

As a *heuristic*, seed and extend reduces the overall search space

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

word

word
word



Only consider mismatches while verifying a seed hit

Edit Distance

Score = 248 bits (129), Expect = 1e-63
Identities = 213/263 (80%), Gaps = 34/263 (12%)
Strand = Plus / Plus

Substitution

Query: 161 atatcaccacgtcaaaggtgactccaactcca---ccact**cc**at~~ttt~~gttcagataatgc 217
|||||
Sbjct: 481 atatcaccacgtcaaaggtgactccaact-tattgatag**gt**tttatgttcagataatgc 539

Query: 218 ccgatgatcatgtcatgcagctccaccgattgtgag**aacgacagcgac**ttccgtcccagc 277
|||||
Sbjct: 540 ccgatgactttgtcatgcagctccaccgattttg-g-----ttccgtcccagc 586

Deletion

Query: 278 c-gtgcc--aggtgctgcctcagattcaggttatgccgctcaattcgctgcgtatatcgc 334
| || | |
Sbjct: 587 caatgacgta-gtgctgcctcagattcaggttatgccgctcaattcgctgggtatatcgc 645

Query: 335 ttgctgattacgtgcagctttcccttcaggcggga-----ccagccatccgtc 382
|||||
Sbjct: 646 ttgctgattacgtgcagctttcccttcaggcggga**ttcatacagcgg**ccagccatccgtc 705

Insertion

Query: 383 ctccatatc-accacgtcaaagg 404
|||||
Sbjct: 706 atccatatcaaccacgtcaaagg 728

Learning Objectives



Review approximate pattern matching

Formalize edit distance storage as an 'edit string'

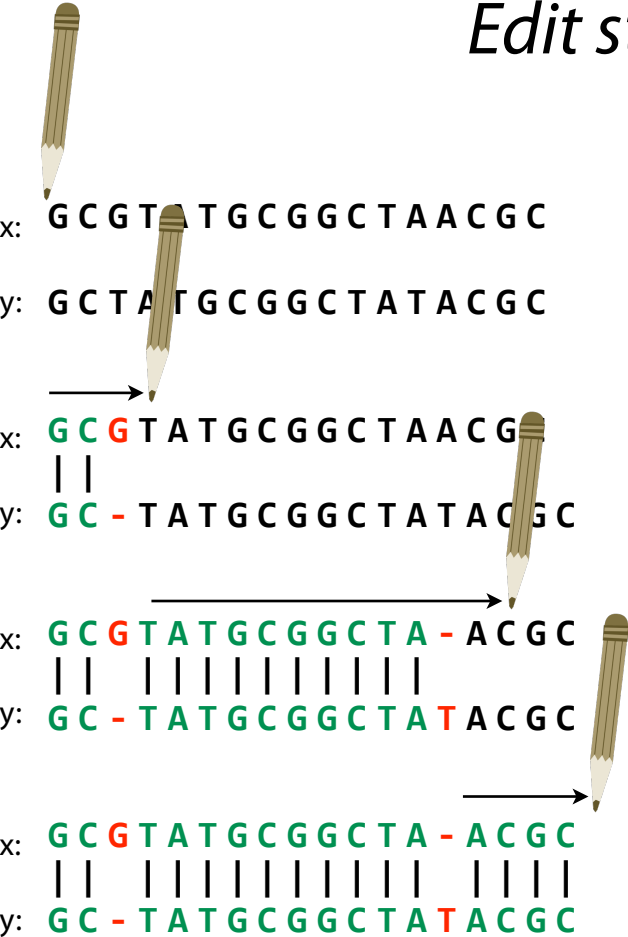
Discuss strategies for efficient APM with edits

Introduce dynamic programming

Edit Distance

Imagine edits are introduced by an *optimal editor* working left-to-right:

Edit string summarizes how editor turns x into y:



Operations:

M = match, **R** = replace (substitute),

I = insert into x, **D** = delete from x

MMD

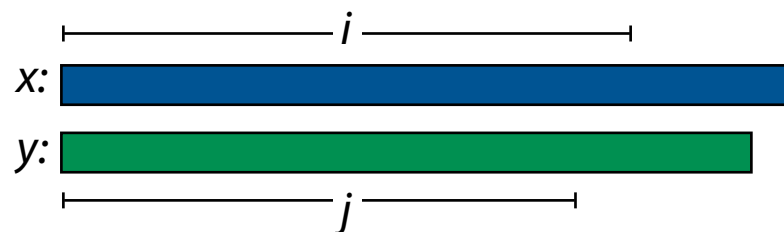
Reminder: this is **D**, not **I**, because we have to delete a character from x to make it more like y

MMDMMMMMMMMMI

MMDMMMMMMMMMIMMMM

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] GTTTA
GGTTTA

D[5, 5] GTTTA
GGTTT

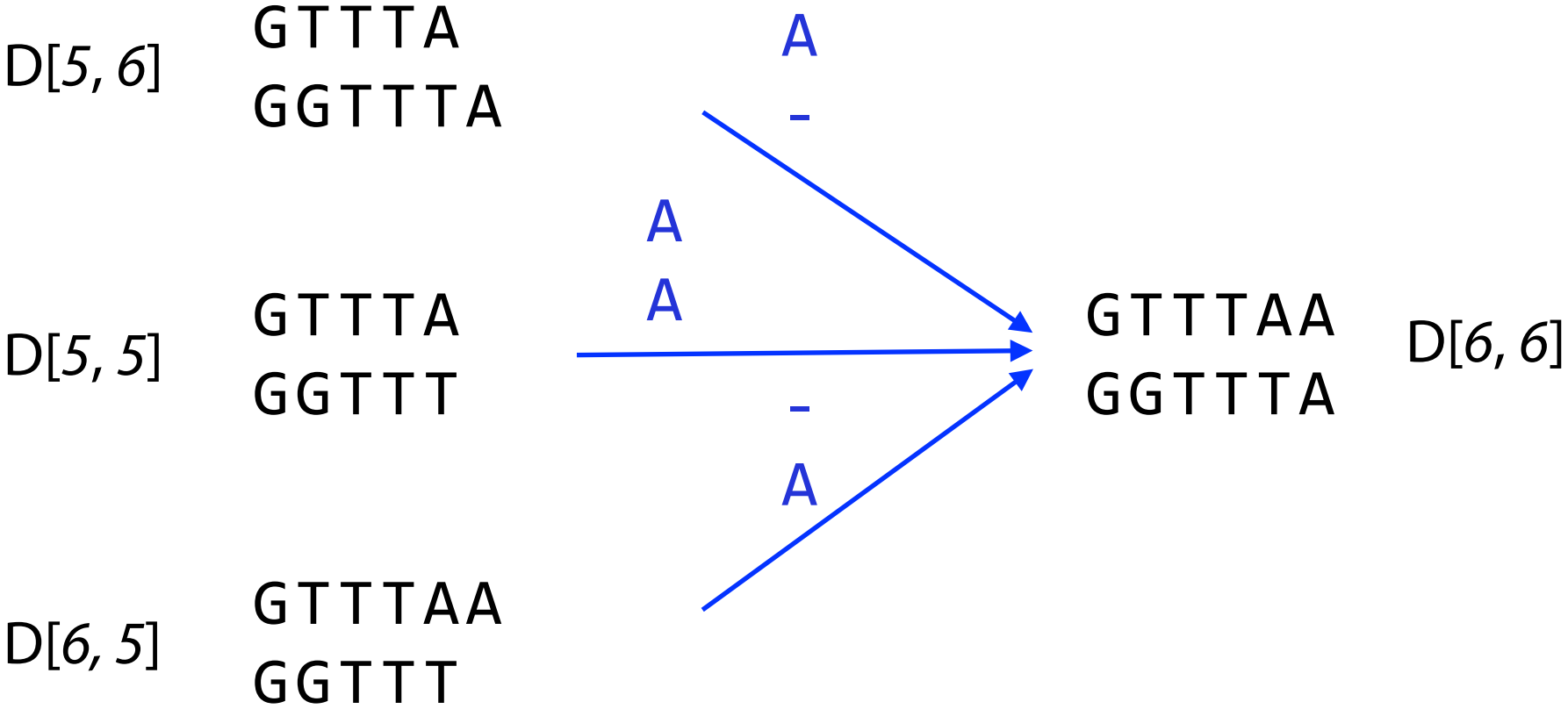
D[6, 5] GTTTAA
GGTTT

GTTTAA
GGTTTA D[6, 6]

Edit Distance

X: GTTTAA

Y: GGTTTA



Edit Distance

X: GTTTAA

Y: GGTTTA

MIMMMM

D[5, 6]

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

MRM**M**R

D[5, 5]

G	T	T	T	A
G	G	T	T	T

MRM**M**R**D**

D[6, 5]

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

Edit Distance

X: GTTTAA

Y: GGTTTA

MIMMMM

D[5, 6]	G	-	T	T	T	A	G	-	T	T	T	A	D[6, 6] ¹
	G	G	T	T	T	A	G	G	T	T	T	A	

MRMMR

D[5, 5]	G	T	T	T	A	G	T	T	T	A	D[6, 6] ²
	G	G	T	T	T	G	G	T	T	T	

MRMMRD

D[6, 5]	G	T	T	T	A	A	G	T	T	T	A	A	D[6, 6] ³
	G	G	T	T	T	-	G	G	T	T	T	-	

Edit Distance

X: GTTTAA

Y: GGTTTA

MIMMMM

D[5, 6]

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

MIMMMM**D**

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

D[6, 6]¹

MRMMR

D[5, 5]

G	T	T	T	A
G	G	T	T	T

MRMMR**M**

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

D[6, 6]²

MRMMRD

D[6, 5]

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

MRMMRD**I**

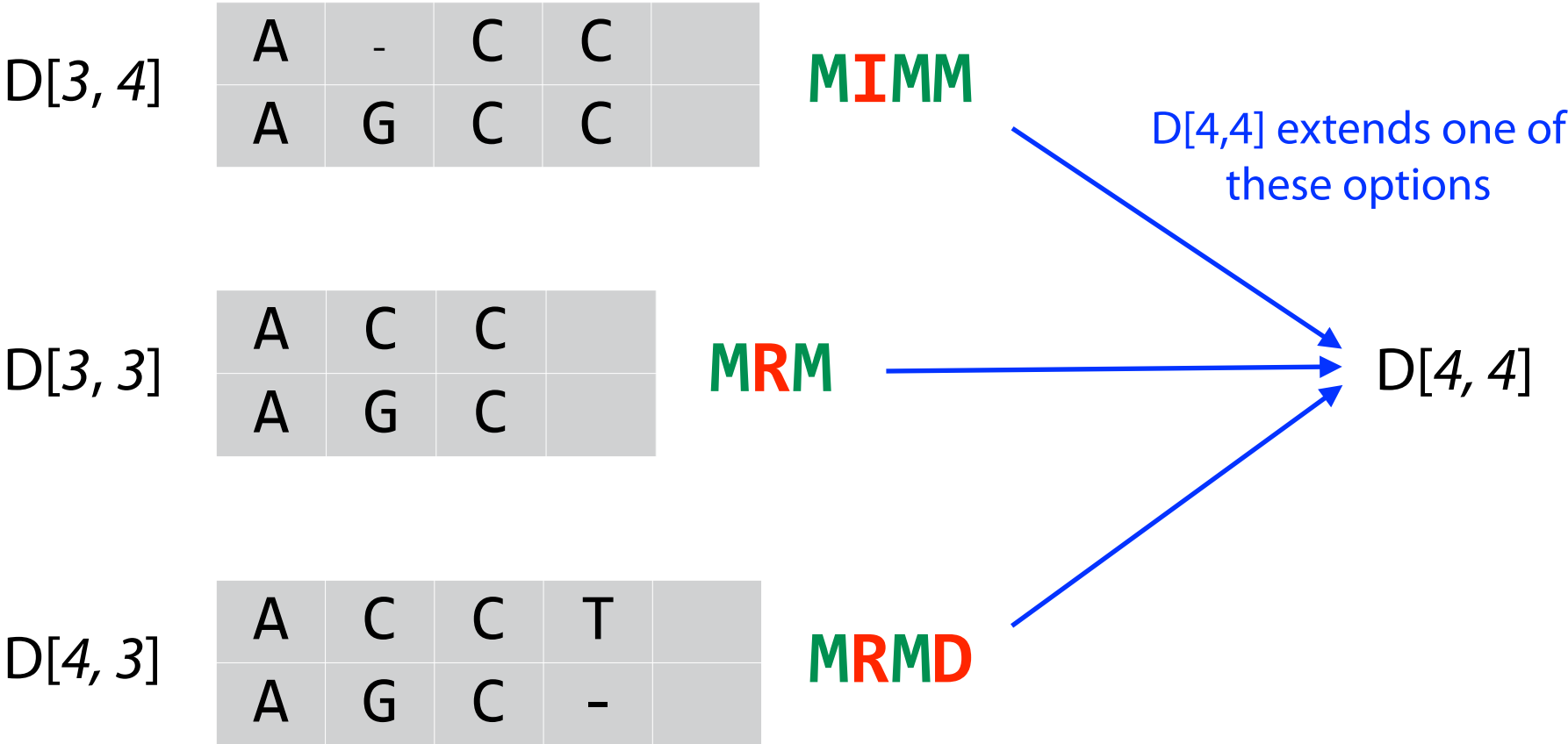
G	T	T	T	A	A	-
G	G	T	T	T	-	A

D[6, 6]³

Edit Distance

X: ACCT

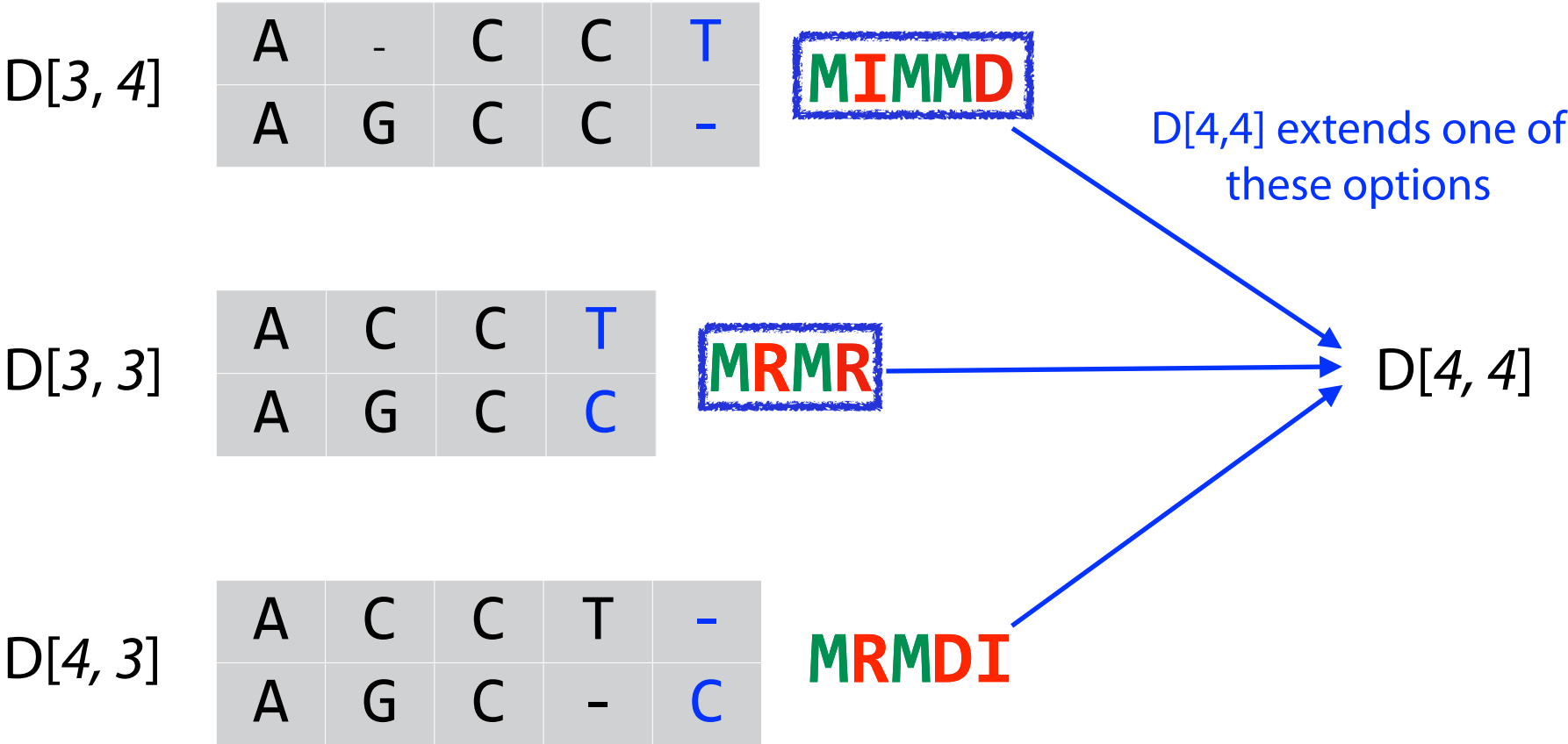
Y: AGCC



Edit Distance

X: ACCT

Y: AGCC



Edit Distance

We can store D as a 2D matrix:

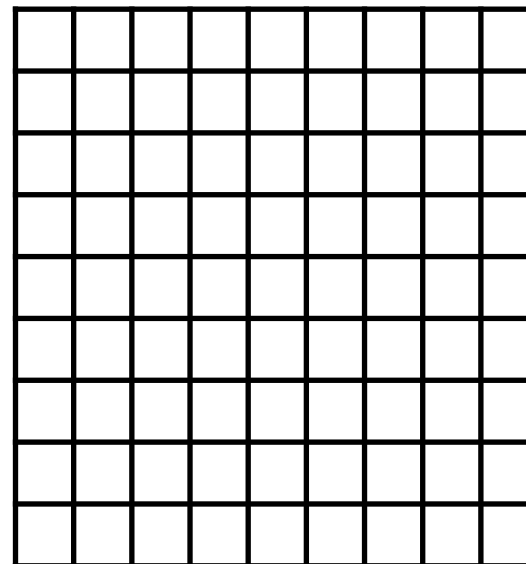
Is at beginning



Ds at beginning



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



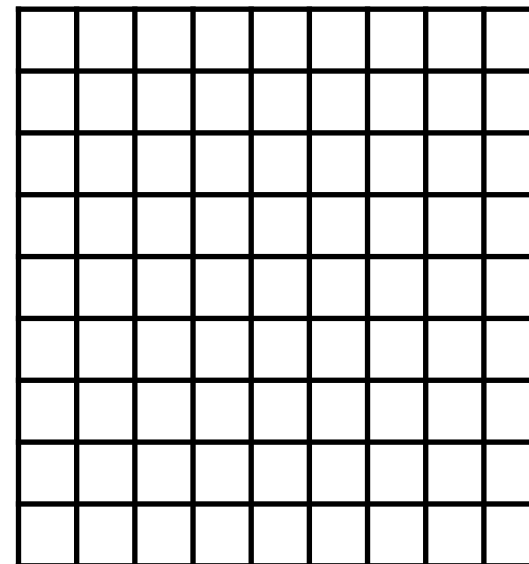
- - - - **B B B B B B B**
 | | | | | | |
A A A A B B B B B B B

t h e **l o n g e s t** - - - -
 | | | | | | |
- - - - **l o n g e s t** **d a y**

Edit Distance



We can store D as a 2D matrix:



Is at beginning



Ds at beginning

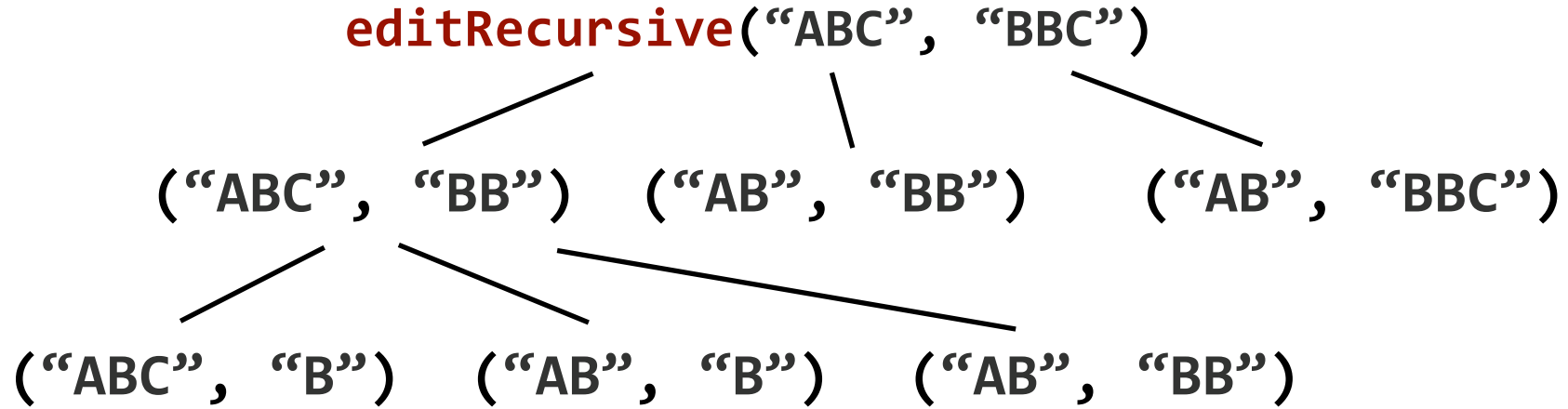


Let $D[0, j] = j$, and let $D[i, 0] = i$

Otherwise, let $D[i, j] = \min \left\{ \begin{array}{l} D[i - 1, j] + 1 \leftarrow \text{vertical (D)} \\ D[i, j - 1] + 1 \leftarrow \text{horizontal (I)} \\ D[i - 1, j - 1] + \delta(x[i - 1], y[j - 1]) \leftarrow \text{diagonal (M or R)} \end{array} \right.$

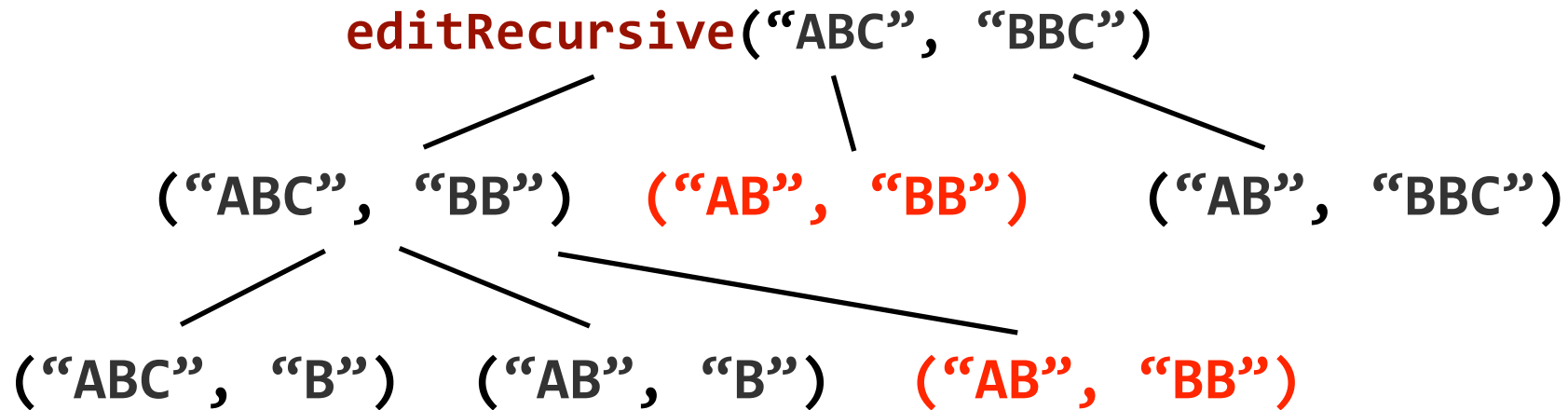
$\delta(a, b)$ is 0 if $a = b$, 1 otherwise

Edit Distance



(only part of recursion tree shown)

Edit Distance



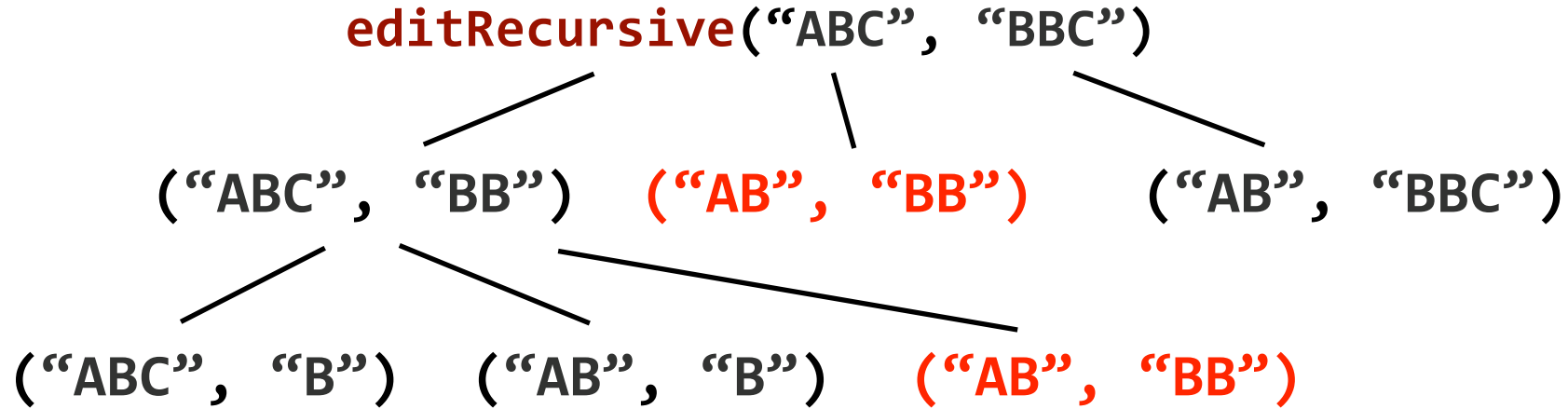
(only part of recursion tree shown)

editRecursive("Shakespeare", "shake spear")

Calculate ("Shake", "shake") 8989 times

How can we address this problem?

Edit Distance

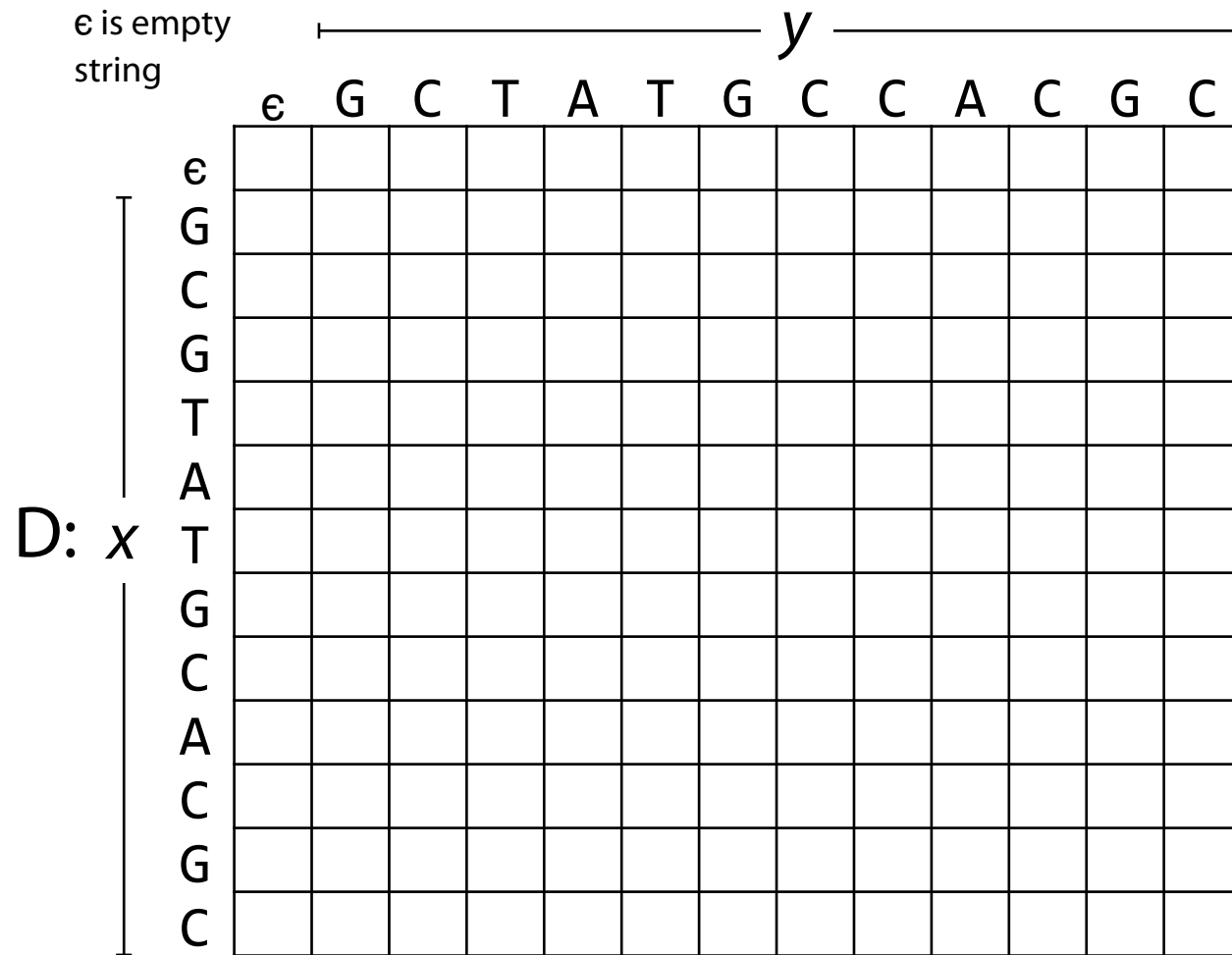


Memoization: Top-down

Dynamic Programming: Bottom-up

Both: Solve individual sub-problems once

Edit Distance: dynamic programming



Let $n = |x|, m = |y|$

D: $(n+1) \times (m+1)$ matrix

$D[i, j]$ = edit distance b/t length- i prefix of x and length- j prefix of y

Edit Distance: dynamic programming

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

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

What is A?

$D[i, j]$ = edit distance b/t length- i prefix of x and length- j prefix of y

Edit Distance: dynamic programming

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

	ε	G	C	T	A	T	G	C	C	A	C	G	C
ε			2										
G													
C													
G													
T													
A													
T													
G													
C													
A													
C													
G													
C													

What is A?

Edit Distance: dynamic programming

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

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

GCT

Edit Distance: dynamic programming

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

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

GCGTAT

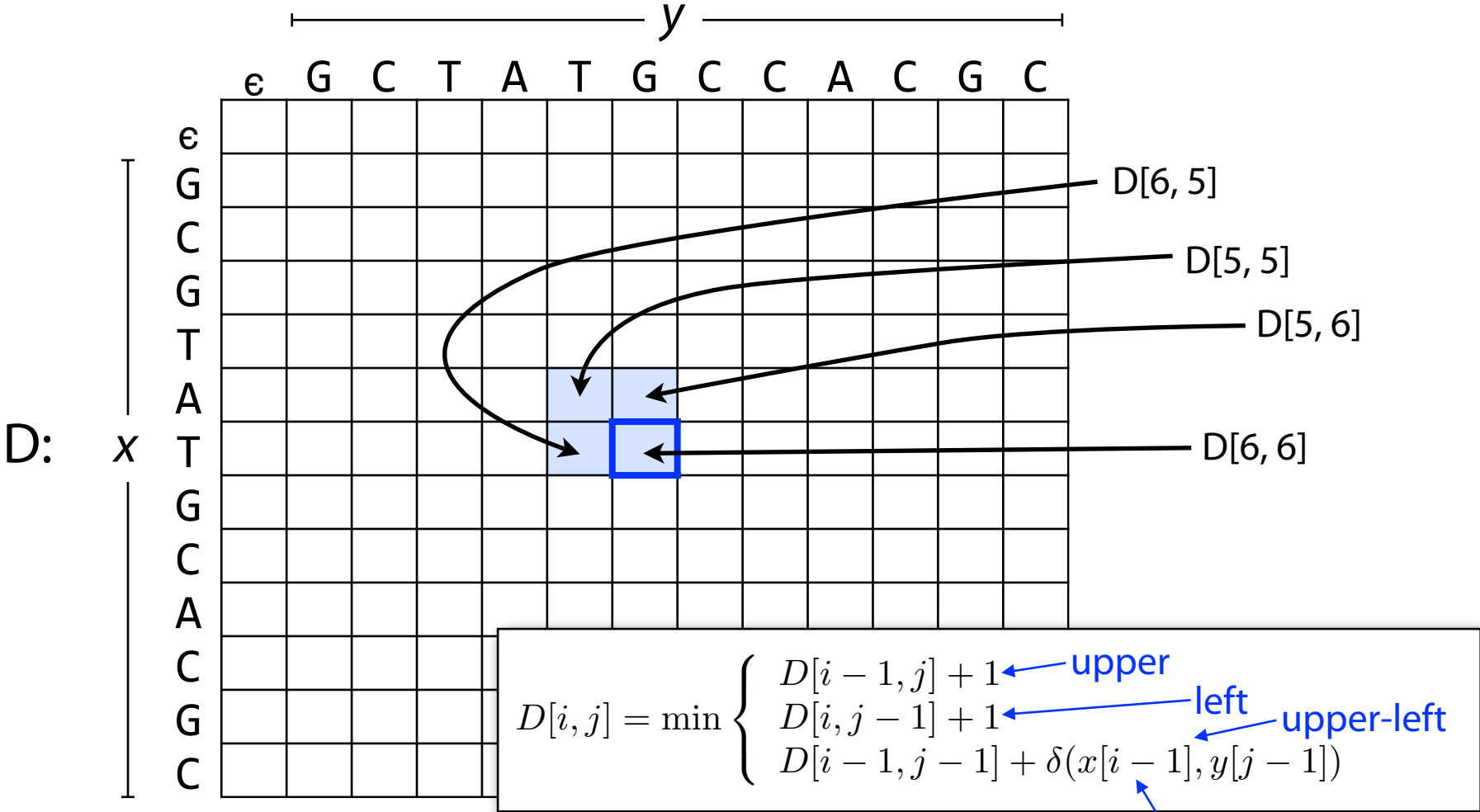
Edit Distance: dynamic programming

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

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

D is one row / column larger than x / y !

Edit Distance: dynamic programming



Cell depends upon its upper, left, and upper-left neighbors

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

Fill remaining cells from top row to bottom and from left to right

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	?											
C	2												
G	3												
T	4												
A	5												
T	6												
G	7												
C	8												
A	9												
C	10												
G	11												
C	12												

- G	G -	G
G -	- G	G
ID	DI	M

What goes here in i=1, j=1?

x[i-1] = 'G',
y[j-1] = 'G',
so delt = 0

$$D[i, j] = \min(D[i-1, j]+1, D[i, j-1]+1, D[i-1, j-1]+delt)$$

$$= \min(1 + 1, 1 + 1, 0 + 0)$$

$$= 0$$

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	?										
C	2												
G	3												
T	4												
A	5												
T	6												
G	7												
C	8												
A	9												
C	10												
G	11												
C	12												

What goes here in $i=1, j=2$?

$x[i-1] = 'G'$;
 $y[j-1] = 'C'$;
 so $\delta = 1$

- - G	G -	- G
G C -	G C	G C
I I D	M I	I R

$$D[i, j] = \min(D[i-1, j]+1, D[i, j-1]+1, D[i-1, j-1]+\delta)$$

$$= \min(2 + 1, 0 + 1, 1 + 1)$$

$$= 1$$

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	?					
A	5												
T	6												
G	7												
C	8												
A	9												
C	10												
G	11												
C	12												

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



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												

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

x[i-1] = 'T';
y[j-1] = 'C';
so delt = 1

8	GC - - - GT	GC - GT - -
9	GCTATG C	GCTATG C
11	MMIIIIMR	MMIRMI I

$$D[i, j] = \min(D[i-1, j]+1, D[i, j-1]+1, D[i-1, j-1]+delt)$$

$$= \min(4 + 1, 3 + 1, 3 + 1)$$

$$= 4$$

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	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

Fill remaining cells from top row to bottom and from left to right

← Edit distance for x, y

Edit Distance

		<i>Y</i>				
		ϵ	C	A	A	T
<i>X</i>	ϵ	0	1	2	3	4
	C	1	0	A	2	3
	A	2	1	0	B	
	T	3	2	C		

Edit Distance

		<i>Y</i>				
		ϵ	C	A	A	T
<i>X</i>	ϵ	\emptyset	1	2	3	4
	C	1	\emptyset	1	2	3
	A	2	1	\emptyset	B	
	T	3	2	C		

Edit Distance

		<i>Y</i>				
		ϵ	C	A	A	T
<i>X</i>	ϵ	0	1	2	3	4
	C	1	0	1	2	3
	A	2	1	0	1	
	T	3	2	C		

Edit Distance

		<i>Y</i>				
		ϵ	C	A	A	T
<i>X</i>	ϵ	0	1	2	3	4
	C	1	0	1	2	3
	A	2	1	0	1	
	T	3	2	1		

Edit Distance

		<i>Y</i>				
		ϵ	C	A	A	T
<i>X</i>	ϵ	0	1	2	3	4
	C	1	0	1	2	3
	A	2	1	0	1	2
	T	3	2	1	1	1

eMatrix buildEditMatrix(X, Y)



Input:

string X: Input string X

string Y: Input string Y

Output:

eMatrix : vector<vector<int>> storing all optimal edit distances

	ε	C	A	A	T
ε	0	1	2	3	4
C	1	0	1	2	3
A	2	1	0	1	2
T	3	2	1	1	1

Edit Distance: 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												
C	2												
G	3												
T	4												
A	5												
T	6												
G	7												
C	8												
A	9												
C	10												
G	11												
C	12												

Red arrows indicate transitions between cells in the DP table. Solid arrows point right, and dashed arrows point left. The word "etc" is written in red in the cell at row A, column 6.

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

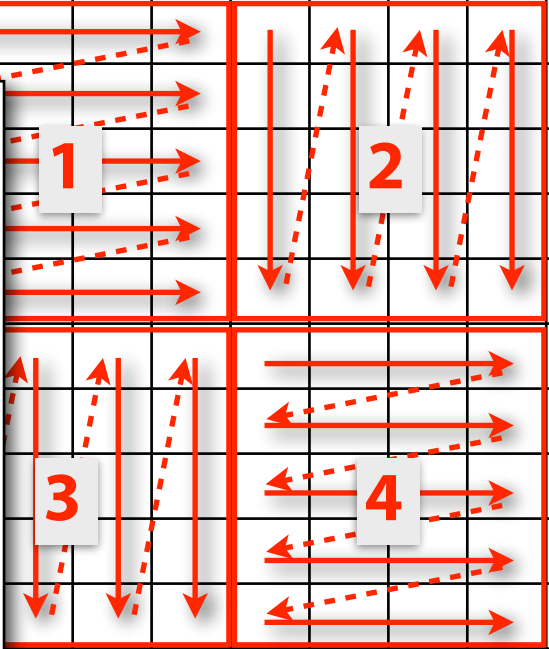
Red arrows indicate transitions between cells in the DP table. Solid arrows point down, and dashed arrows point up. The word "etc" is written in red in the cell at row T, column 6.

Edit Distance: 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												
C	2												
G	3												
T	4												
A	5												
T	6												
G	7												
C	8												
A	9												
C	10												
G	11												
C	12												

	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												
G	11												
C	12												

Any strategy that fills in order works:



etc

etc

Assignment 11: a_edist

Learning Objective:

Use dynamic programming to build an edit distance matrix

Construct an optimal edit string from the edit matrix

Consider: Does substitution, insertion, and deletion need to have the same 'weight' as a penalty? How could you modify the code to take a user-specified input for each?

Edit Distance

	ϵ	C	A	A	T
ϵ	0	1	2	3	4
C	1	0	1	2	3
A	2	1	0	1	2
T	3	2	1	1	1

← Edit distance for x, y
But where and what
is the edit?

Edit Distance

Traceback corresponds to an optimal alignment / edit transcript

At each step, ask: which neighbor (\swarrow , \leftarrow or \uparrow) gave the minimum?

	ϵ	C	A	A	T
ϵ	0	1	2	3	4
C	1	0	1	2	3
A	2	1	0	1	2
T	3	2	1	1	1

← Q: How did I get here?

Edit Distance

Traceback corresponds to an optimal alignment / edit transcript

At each step, ask: which neighbor (\swarrow , \leftarrow or \uparrow) gave the minimum?

	ϵ	C	A	A	T
ϵ	0	1	2	3	4
C	1	0	1	2	3
A	2	1	0	1	2
T	3	2	1	1	1

$D[2, 4] = \underline{\hspace{2cm}}$

← Q: How did I get here?

Edit Distance

Traceback corresponds to an optimal alignment / edit transcript

At each step, ask: which neighbor (\swarrow , \leftarrow or \uparrow) gave the minimum?

	ϵ	C	A	A	T
ϵ	0	1	2	3	4
C	1	0	1	2	3
A	2	1	0	1	2
T	3	2	1	1	1

$D[3, 3] = \underline{\hspace{2cm}}$

← Q: How did I get here?

Edit Distance

Traceback corresponds to an optimal alignment / edit transcript

At each step, ask: which neighbor (\swarrow , \leftarrow or \uparrow) gave the minimum?

	ϵ	C	A	A	T
ϵ	0	1	2	3	4
C	1	0	1	2	3
A	2	1	0	1	2
T	3	2	1	1	1

$D[2, 3] = \underline{\hspace{2cm}}$

← Q: How did I get here?

Edit Distance

Traceback corresponds to an optimal alignment / edit transcript

At each step, ask: which neighbor (\swarrow , \leftarrow or \uparrow) gave the minimum?

	ϵ	C	A	A	T
ϵ	0	1	2	3	4
C	1	0	1	2	3
A	2	1	0	1	2
T	3	2	1	1	1

$$D[1, 3] = \underline{\hspace{2cm}}$$

$$D[1, 2] = \underline{\hspace{2cm}}$$

$$D[2, 2] = \underline{\hspace{2cm}}$$

Q: How did I get here?

Edit Distance

Traceback corresponds to an optimal alignment / edit transcript

At each step, ask: which neighbor (\nwarrow , \leftarrow or \nearrow) gave the minimum?

	ϵ	C	A	A	T
ϵ	0	1	2	3	4
C	1	0	1	2	3
A	2	1	0	1	2
T	3	2	1	1	1

$$D[1, 3] = 2 + 1$$

$$D[1, 2] = 1 + 0$$

$$D[2, 2] = 0 + 1$$

Q: How did I get here?

Edit Distance

Traceback corresponds to an optimal alignment / edit transcript

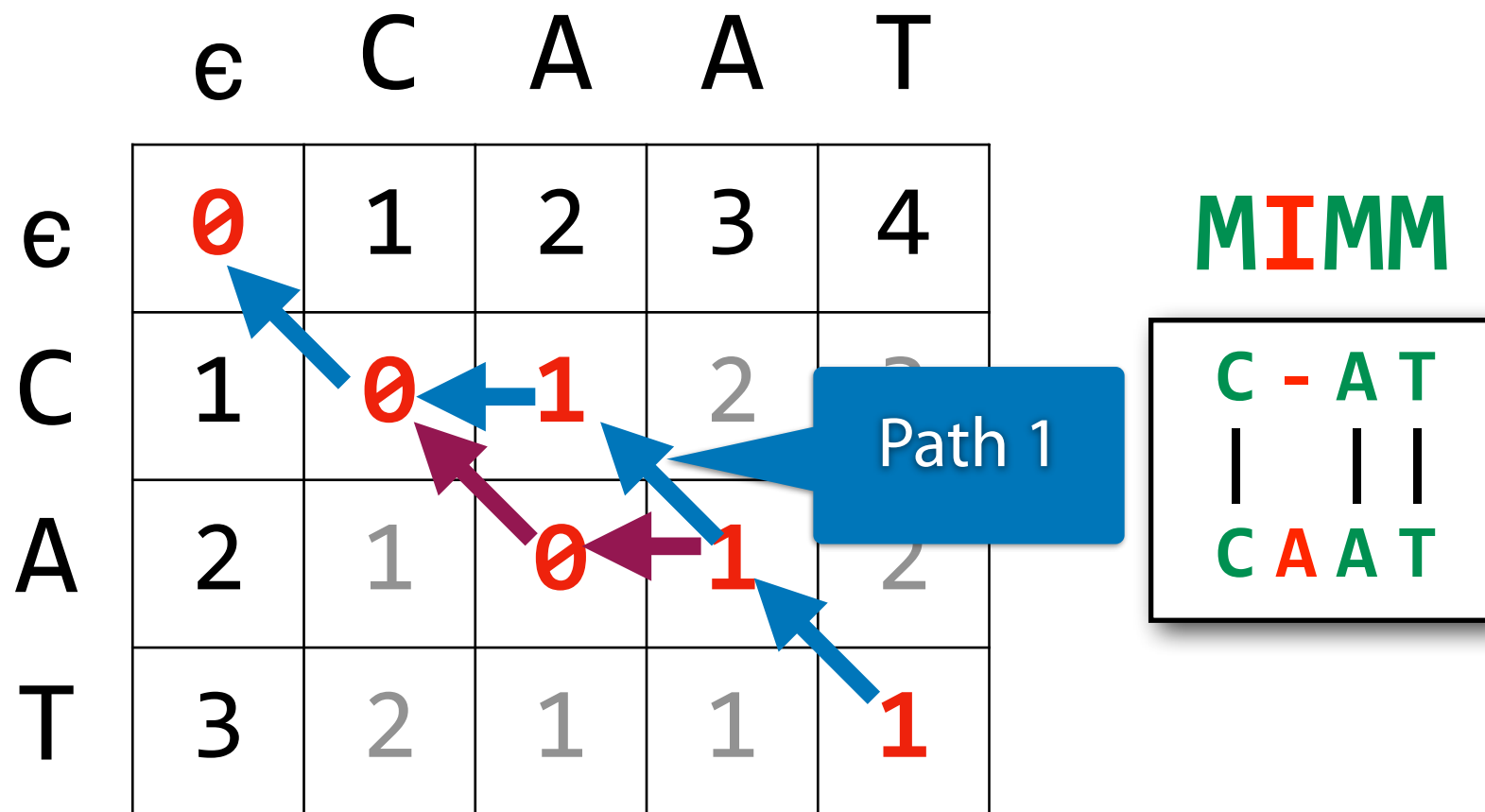
At each step, ask: which neighbor (\swarrow , \leftarrow or \uparrow) gave the minimum?

	ϵ	C	A	A	T
ϵ	0	1	2	3	4
C	1	0	1	2	3
A	2	1	0	1	2
T	3	2	1	1	1

Edit Distance

Traceback corresponds to an optimal alignment / edit transcript

At each step, ask: which neighbor (\swarrow , \leftarrow or \uparrow) gave the minimum?



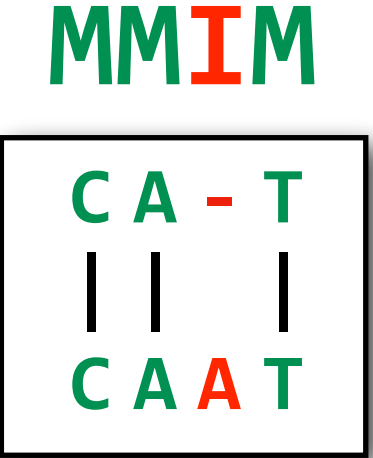
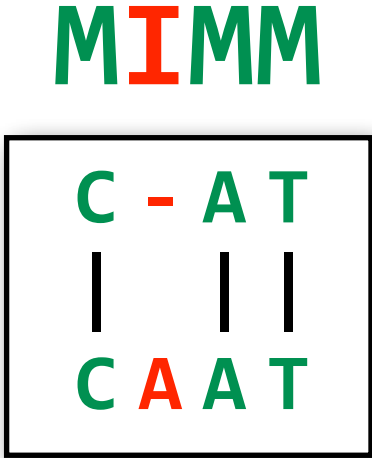
Edit Distance

Traceback corresponds to an optimal alignment / edit transcript

At each step, ask: which neighbor (\swarrow , \leftarrow or \uparrow) gave the minimum?

	ϵ	C	A	A	T
ϵ	0	1	2	3	4
C	1	0	1	2	3
A	2	1	0	1	2
T			1	1	1

Path 2



Edit Distance

	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

← Q: How did I get here?

Edit Distance

	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

$$D[11, 12] = 3 + 1$$

$$D[11, 11] = 2 + 0$$

$$D[12, 11] = 3 + 1$$

A: From here

Q: How did I get here?

Edit Distance

	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

Q: How did I get here?



Edit Distance

	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

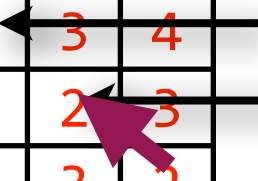
$D[10, 11] = 3 + 1$

$D[10, 10] = 2 + 0$

$D[11, 10] = 3 + 1$

A: From here

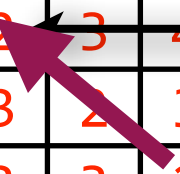
Q: How did I get here?



Edit Distance

	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

Q: How did I get here?



Edit Distance

	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

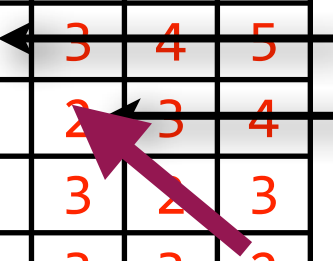
$$D[9, 10] = 3 + 1$$

$$D[9, 9] = 2 + 0$$

$$D[10, 9] = 3 + 1$$

A: From here

Q: How did I get here?



Edit Distance

	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

Edit Distance

Dynamic Programming fills our table with optimal distances

Traceback identifies the optimal *edit string* to convert X to Y

What is our efficiency? ($|X| = m, |Y| = n$)

	e	G	C	T	A	T	G	C	C	A	C	G	C
e	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	3	4	5	6
C	10	9	8	7	6	5	4	3	2	3	3	4	5
G	11	10	9	8	7	6	5	4	3	3	3	3	4
C	12	11	10	9	8	7	6	5	4	4	3	3	3

Edit Distance

Dynamic Programming fills our table with optimal distances

Traceback identifies the optimal *edit string* to convert X to Y

What is our efficiency? ($|X| = m, |Y| = n$)

	e	G	C	T	A	T	G	C	C	A	C	G	C
e	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	3	4	5	6
C	10	9	8	7	6	5	4	3	2	3	3	4	5
G	11	10	9	8	7	6	5	4	3	3	3	3	4
C	12	11	10	9	8	7	6	5	4	4	3	3	3

Table filling: Filling $(m + 1)$
 $(n + 1)$ cells, each requiring
 constant work, so $O(mn)$

Traceback: Each step goes ↖, ⇐ or
 ⤴. Worst case: traceback never
 moves diagonally, requiring m ⤴'s
 and n ⇐'s, so $O(m + n)$



string buildEditString(X, Y)

Input: **string X**: Input string X (edits with respect to X)

string Y: Input string Y (edits turn X into Y)

Output: **string**: An optimal edit string produced by the matrix

	ε	C	A	A	T
ε	0	1	2	3	4
C	1	0	1	2	3
A	2	1	0	1	2
T	3	2	1	1	1

On tie: prioritize diagonal, then vertical, then horizontal



MIMM

C	-	A	T
C	A	A	T

Approximate Pattern Matching

“Seed and extend” works for edit distance too!

