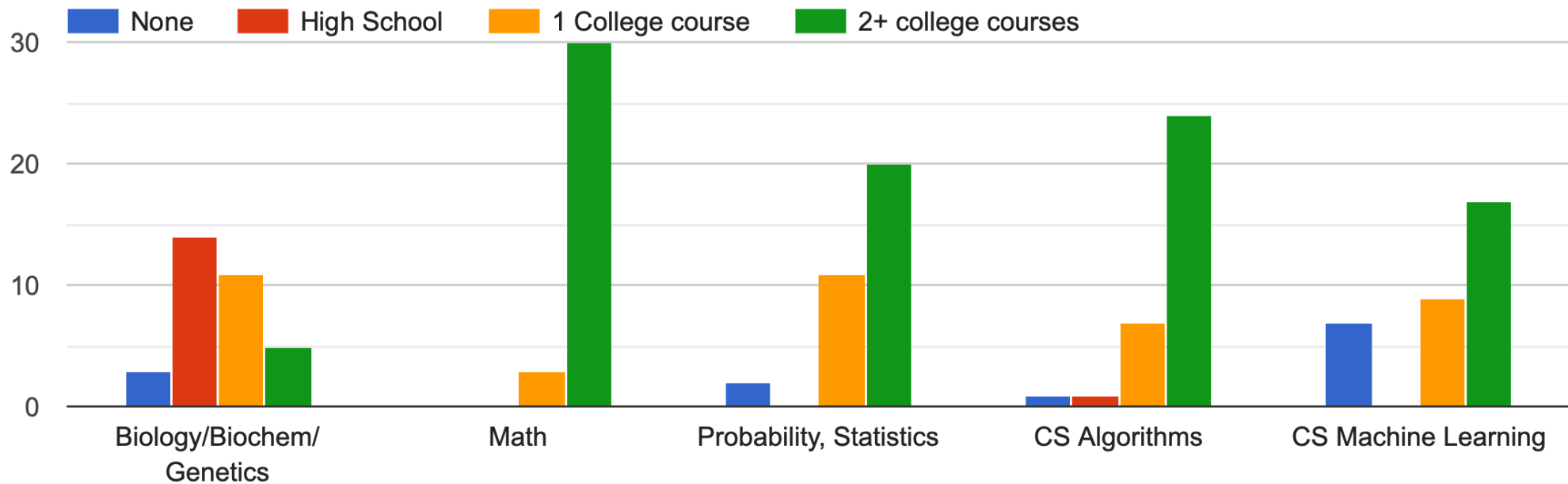


CSEP 527
Computational Biology
Autumn 2020

Lectures 2-4
Sequence Alignment

20au Survey Results

Describe your experience in each of the following areas. ("None" is a ...



This week

Sequence alignment

Weekly “bio” interlude - DNA replication

More sequence alignment

What is an alignment?

Compare two strings to see how “similar” they are
E.g., maximize the # of identical chars that line up

But we'll see more
subtle measures

ATGTTAT vs
ATCGTAC

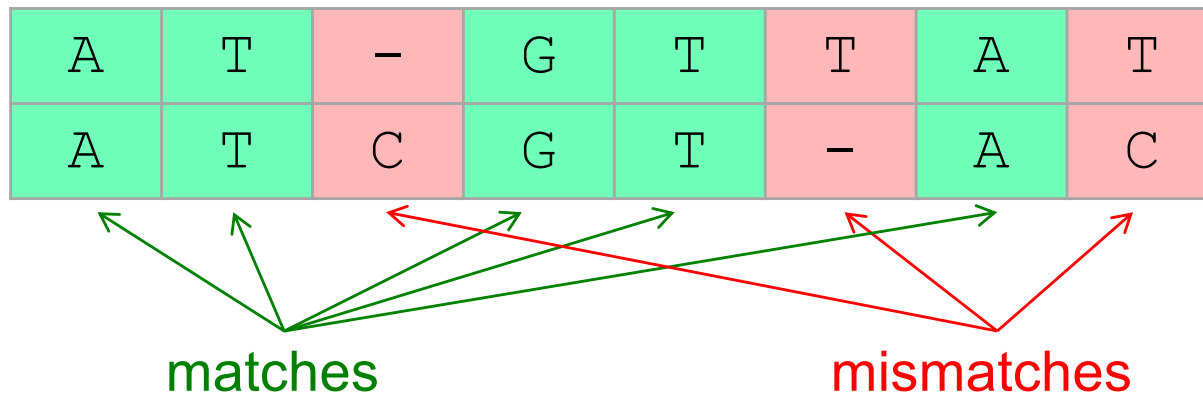
A	T	-	G	T	T	A	T
A	T	C	G	T	-	A	C

What is an alignment?

Compare two strings to see how “similar” they are
E.g., maximize the # of identical chars that line up

But we’ll see more
subtle measures

ATGTTAT vs
ATCGTAC



Sequence Alignment: Why

Biology

Among most widely used comp. tools in biology

DNA sequencing & assembly

New sequence always compared to data bases

Similar sequences often have similar origin and/or function

Recognizable similarity after $10^8 - 10^9$ yr

Other

spell check/correct, diff, svn/git/..., plagiarism, ...

BLAST Demo

<http://www.ncbi.nlm.nih.gov/blast/>

click “protein blast”

[get seq from: http://www.uniprot.org/uniprot/P15172](http://www.uniprot.org/uniprot/P15172) i.e.:

```
>sp|P15172|MYOD1_HUMAN Myoblast determination protein 1 OS=Homo sapiens GN=MYOD1 PE=1 SV=3
MELLSPPLRDVDLTAPDGSLCSFATDDDFYDDPCFDSPDLRFFEDLDPRLMHVVGALLKPE
EHSHFPAAVHPAPGAREDEHVRAPSGHHQAGRCLLWACKACKRKTNNADRRKAATMRERR
RLSKVNEAFETLKRCTSSNPQRLPKVEILRNAIRYIEGLQALLRDQDAAPPAAAAFYA
PGPLPPGRGGEHYSGDSDASSPRSNCSGMMMDYSGPPSGARRRNCYEGAYYNEAPSEPRP
GKSAAVSSLDCLSSIVERISTESPAAPALLLADVPSSEPPRRQEAAAPSEGESSGDPTQS
PDAAPQCPAGANPNPIYQVL
```

in blast window, paste “P15172” (or fasta above) in “query” box

adjust “alg parameters” to get top 1000 hits (or more)

click “BLAST” button.

scrolling to bottom of hits list & click may download all alignments, (otherwise it seems to download only 4 or 5 more when you scroll to bottom)

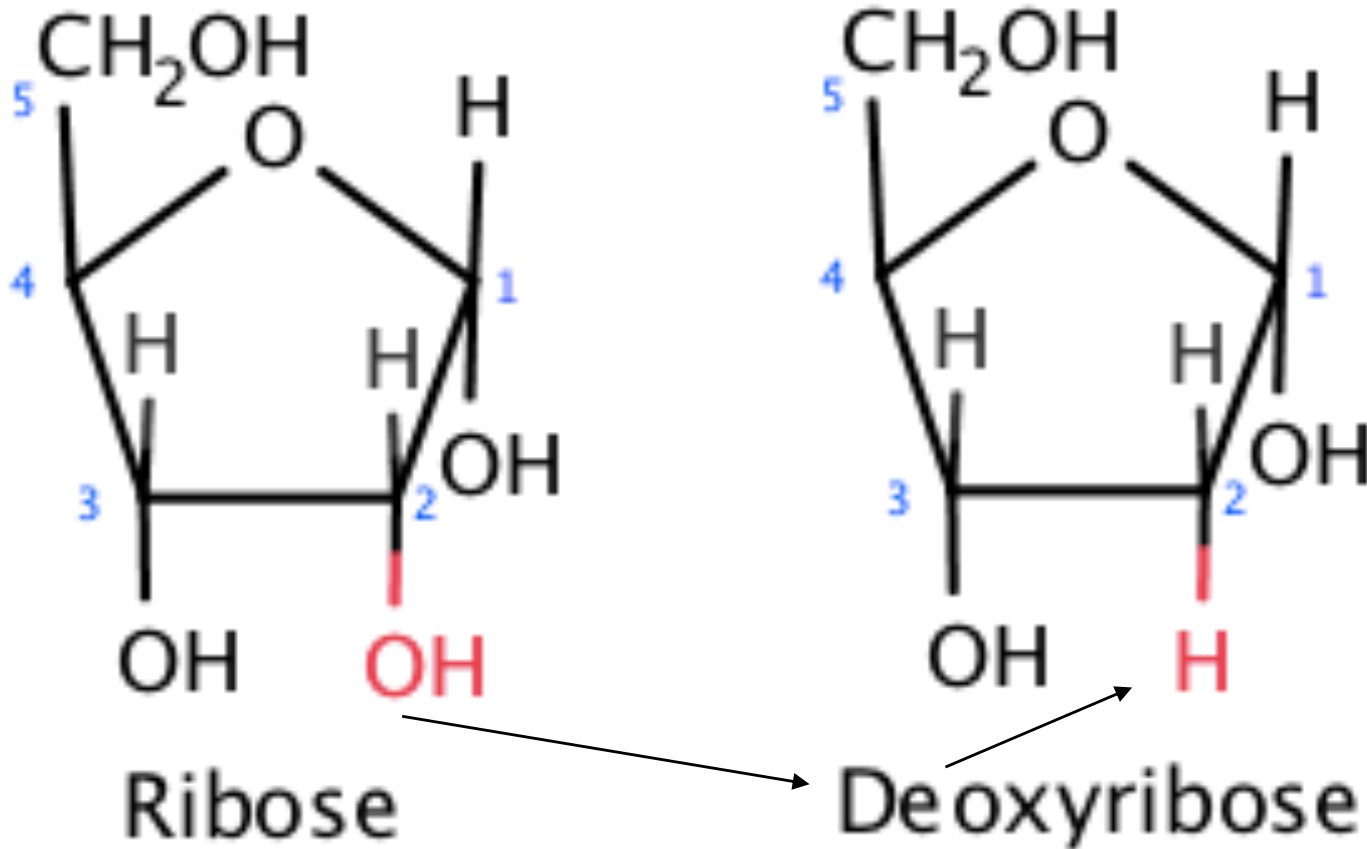
Also check out the “Graphic Summary”, “Alignments”, “Taxonomy” tabs, + “Distance tree of results”

CSEP 527
Computational Biology
Autumn 2020

Lecture 3
Sequence Alignment

More on DNA/RNA

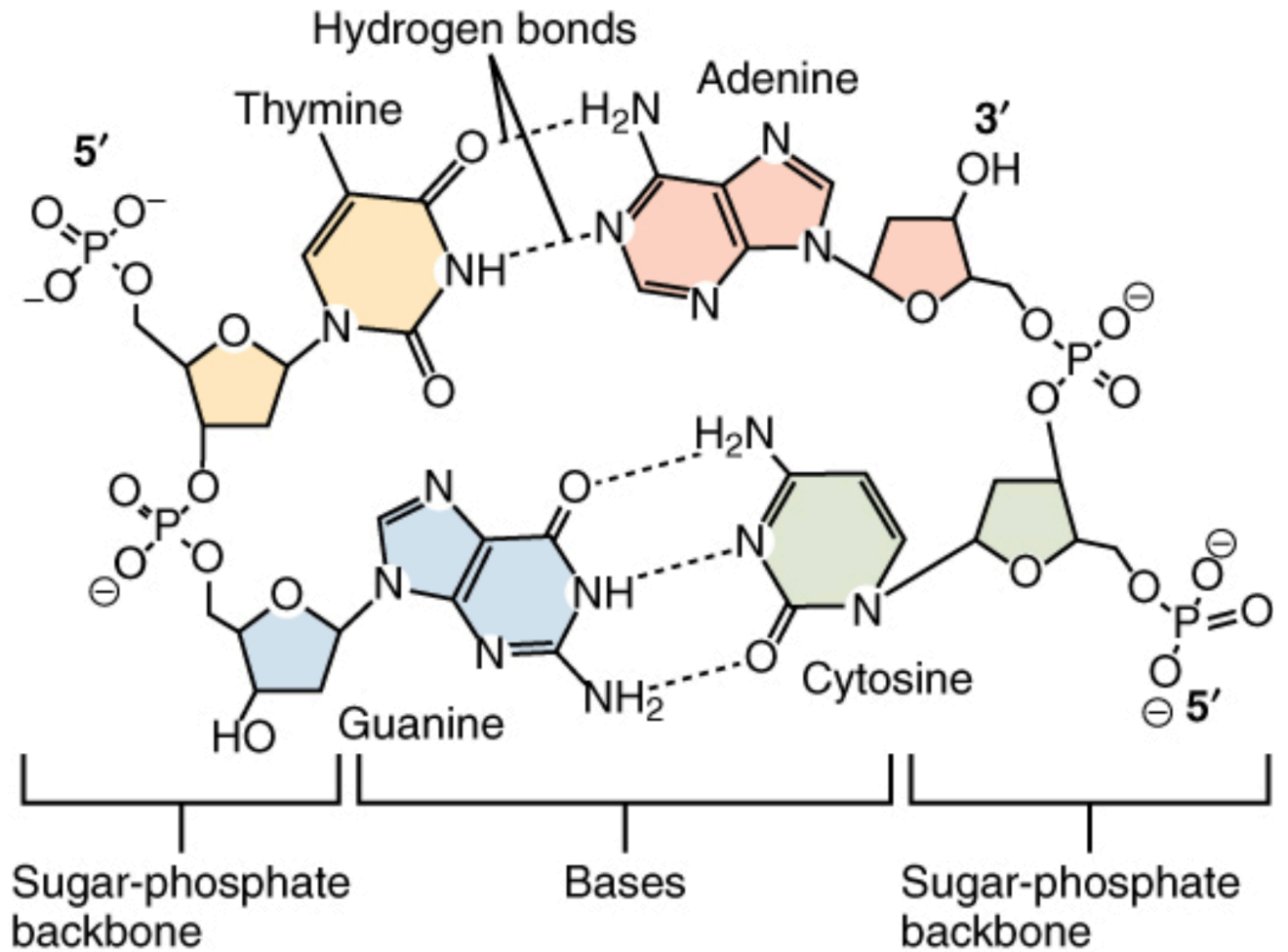
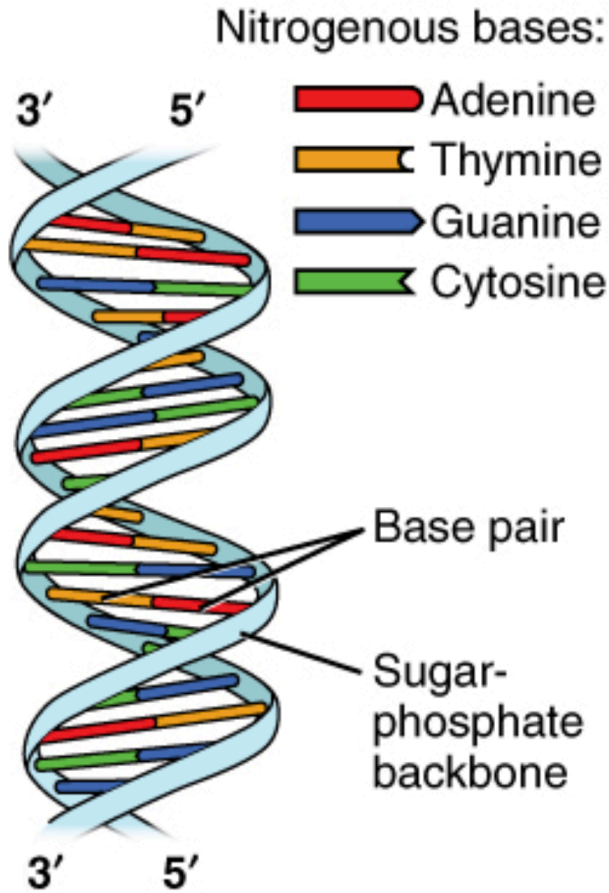
DNA, RNA, 3', 5', ...



<https://en.wikipedia.org/wiki/Ribose>

More on DNA/RNA

Nucleotides

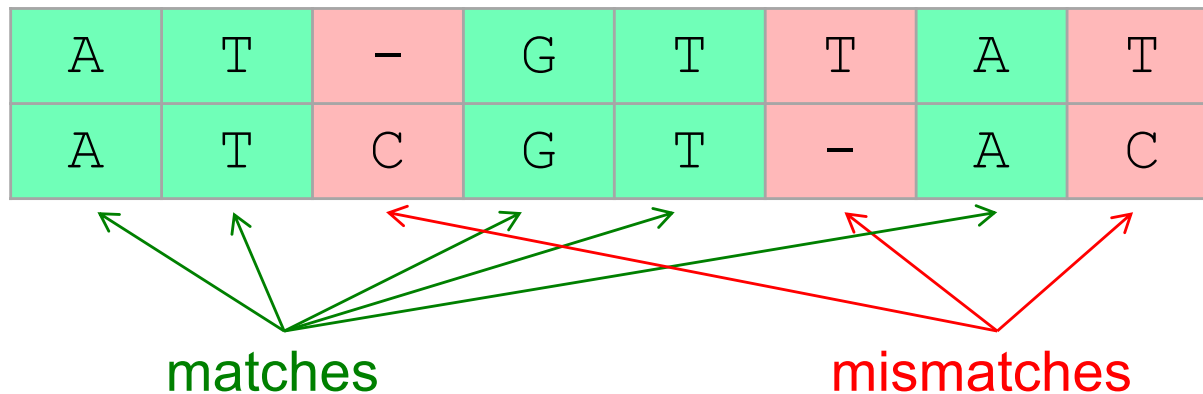


What is an alignment?

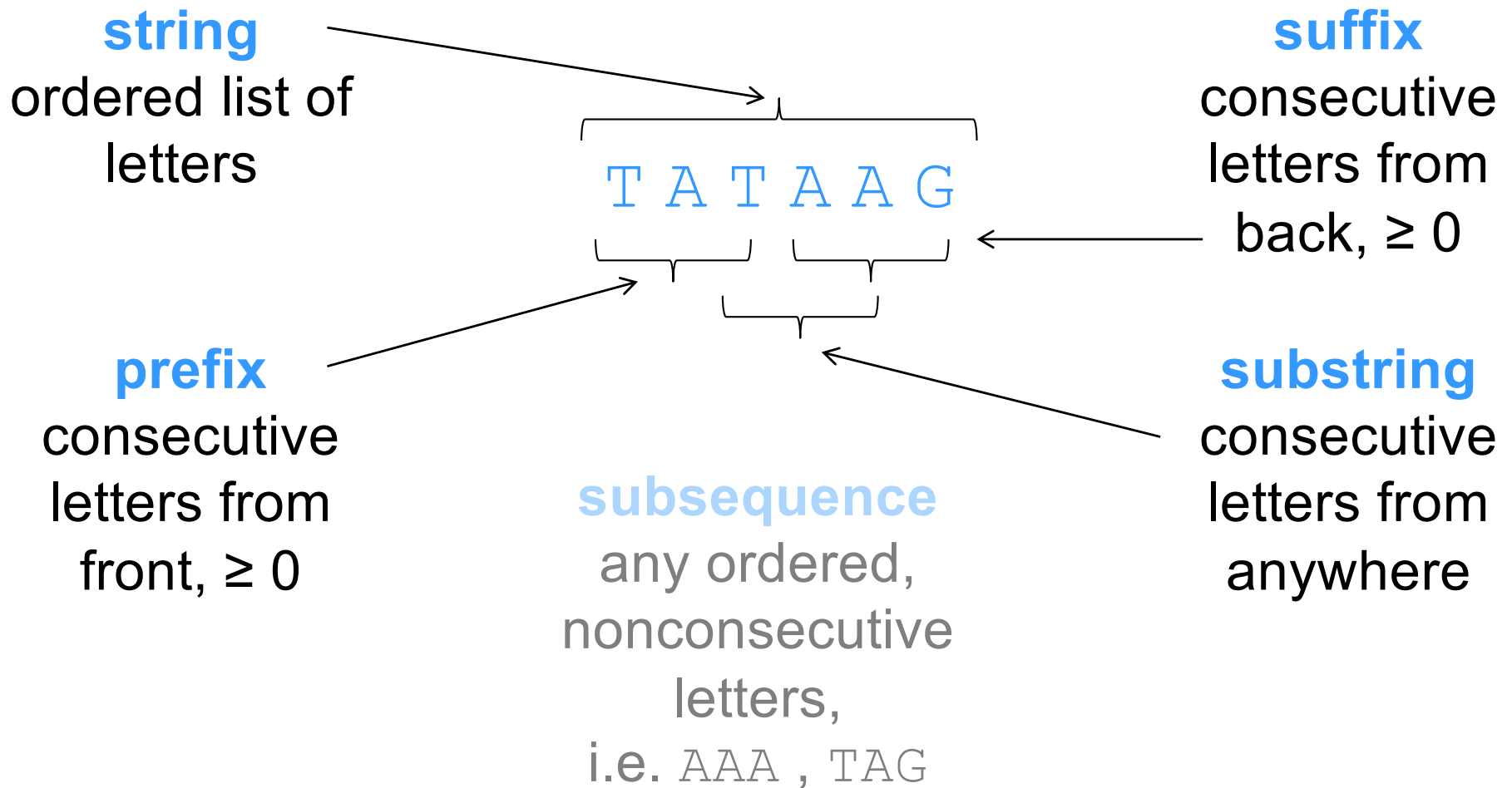
Compare two strings to see how “similar” they are
E.g., maximize the # of identical chars that line up

But we'll see more
subtle measures

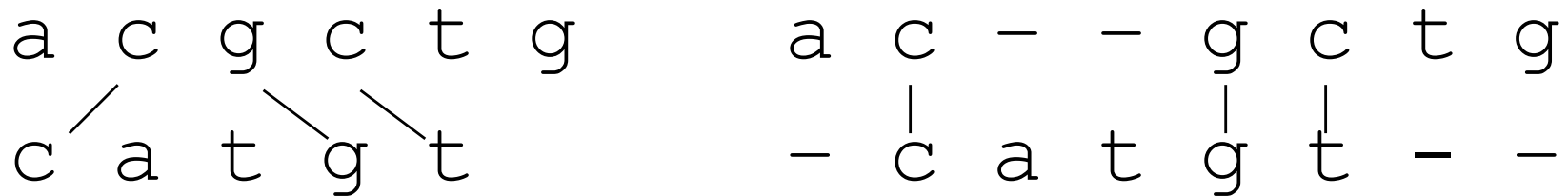
ATGTTAT vs
ATCGTAC



Terminology



Formal definition of an alignment



An **alignment** of strings S , T is a pair of strings S' , T' with dash characters “-” inserted, so that

1. $|S'| = |T'|$, and $(|S| = \text{“length of } S\text{”})$
2. Removing dashes leaves S , T

Consecutive dashes are called “**a gap.**”

(NB: this is a defn for a general alignment, not necessarily optimal.)

Scoring an arbitrary alignment

Define a score for *pairs* of aligned chars, e.g.

$$\sigma(x, y) = \begin{cases} \text{match} & 2 \\ \text{mismatch} & -1 \end{cases}$$

(Toy scores for examples in slides)

Apply that *per column*, then *add*.

a	c	-	-	g	c	t	g
-	c	a	t	g	t	-	-
-1	+2	-1	-1	+2	-1	-1	-1

Total Score = -2

More Realistic Scores: BLOSUM 62

(the “ σ ” scores)

	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V
A	4	-1	-2	-2	0	-1	-1	0	-2	-1	-1	-1	-1	-2	-1	1	0	-3	-2	0
R	-1	5	0	-2	-3	1	0	-2	0	-3	-2	2	-1	-3	-2	-1	-1	-3	-2	-3
N	-2	0	6	1	-3	0	0	0	1	-3	-3	0	-2	-3	-2	1	0	-4	-2	-3
D	-2	-2	1	6	-3	0	2	-1	-1	-3	-4	-1	-3	-3	-1	0	-1	-4	-3	-3
C	0	-3	-3	-3	9	-3	-4	-3	-3	-1	-1	-3	-1	-2	-3	-1	-1	-2	-2	-1
Q	-1	1	0	0	-3	5	2	-2	0	-3	-2	1	0	-3	-1	0	-1	-2	-1	-2
E	-1	0	0	2	-4	2	5	-2	0	-3	-3	1	-2	-3	-1	0	-1	-3	-2	-2
G	0	-2	0	-1	-3	-2	-2	6	-2	-4	-4	-2	-3	-3	-2	0	-2	-2	-3	-3
H	-2	0	1	-1	-3	0	0	-2	8	-3	-3	-1	-2	-1	-2	-1	-2	-2	2	-3
I	-1	-3	-3	-3	-1	-3	-3	-4	-3	4	2	-3	1	0	-3	-2	-1	-3	-1	3
L	-1	-2	-3	-4	-1	-2	-3	-4	-3	2	4	-2	2	0	-3	-2	-1	-2	-1	1
K	-1	2	0	-1	-3	1	1	-2	-1	-3	-2	5	-1	-3	-1	0	-1	-3	-2	-2
M	-1	-1	-2	-3	-1	0	-2	-3	-2	1	2	-1	5	0	-2	-1	-1	-1	-1	1
F	-2	-3	-3	-3	-2	-3	-3	-3	-1	0	0	-3	0	6	-4	-2	-2	1	3	-1
P	-1	-2	-2	-1	-3	-1	-1	-2	-2	-3	-3	-1	-2	-4	7	-1	-1	-4	-3	-2
S	1	-1	1	0	-1	0	0	0	-1	-2	-2	0	-1	-2	-1	4	1	-3	-2	-2
T	0	-1	0	-1	-1	-1	-1	-2	-2	-1	-1	-1	-1	-2	-1	1	5	-2	-2	0
W	-3	-3	-4	-4	-2	-2	-3	-2	-2	-3	-2	-3	-1	1	-4	-3	-2	11	2	-3
Y	-2	-2	-2	-3	-2	-1	-2	-3	2	-1	-1	-2	-1	3	-3	-2	-2	2	7	-1
V	0	-3	-3	-3	-1	-2	-2	-3	-3	3	1	-2	1	-1	-2	-2	0	-3	-1	4

Optimal Alignment: A Simple Algorithm

for all subseqs A of S , B of T s.t. $|A| = |B|$ **do**
 align $A[i]$ with $B[i]$, $1 \leq i \leq |A|$
 align all other chars to spaces
 compute its value
 retain the max
end
output the retained alignment

$S = agct$	$A = ct$
$T = wxyz$	$B = xz$
$-agc-t$	$a-gc-t$
$w--xyz$	$-w-xyz$

Analysis

Assume $|S| = |T| = n$

Cost of evaluating one alignment: $\geq n$

How many alignments are there: $\geq \binom{2n}{n}$

pick n chars of S, T together

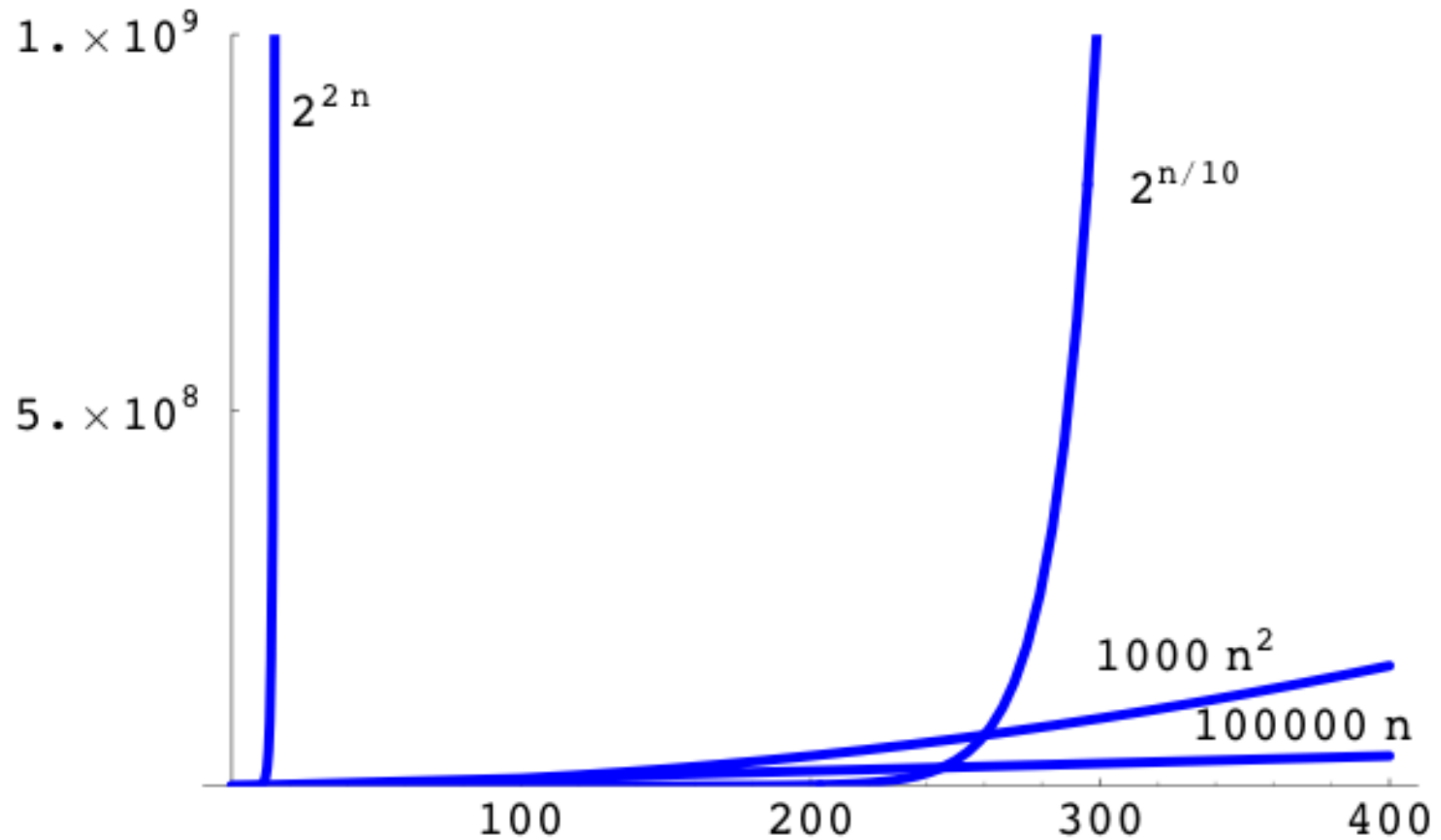
say k of them are in S

match these k to the k unpicked chars of T , rest to -

Total time: $\geq n \binom{2n}{n} > 2^{2n}$, for $n > 3$

E.g., for $n = 20$, time is $> 2^{40}$ operations

Polynomial vs Exponential Growth



Asymptotic Analysis

How does run time grow as a function of problem size?

$$n^2 \text{ or } 100n^2 + 100n + 100 \text{ vs } 2^{2n}$$

Defn: $f(n) = O(g(n))$ iff there is a constant c s.t. $|f(n)| \leq cg(n)$ for all sufficiently large n .

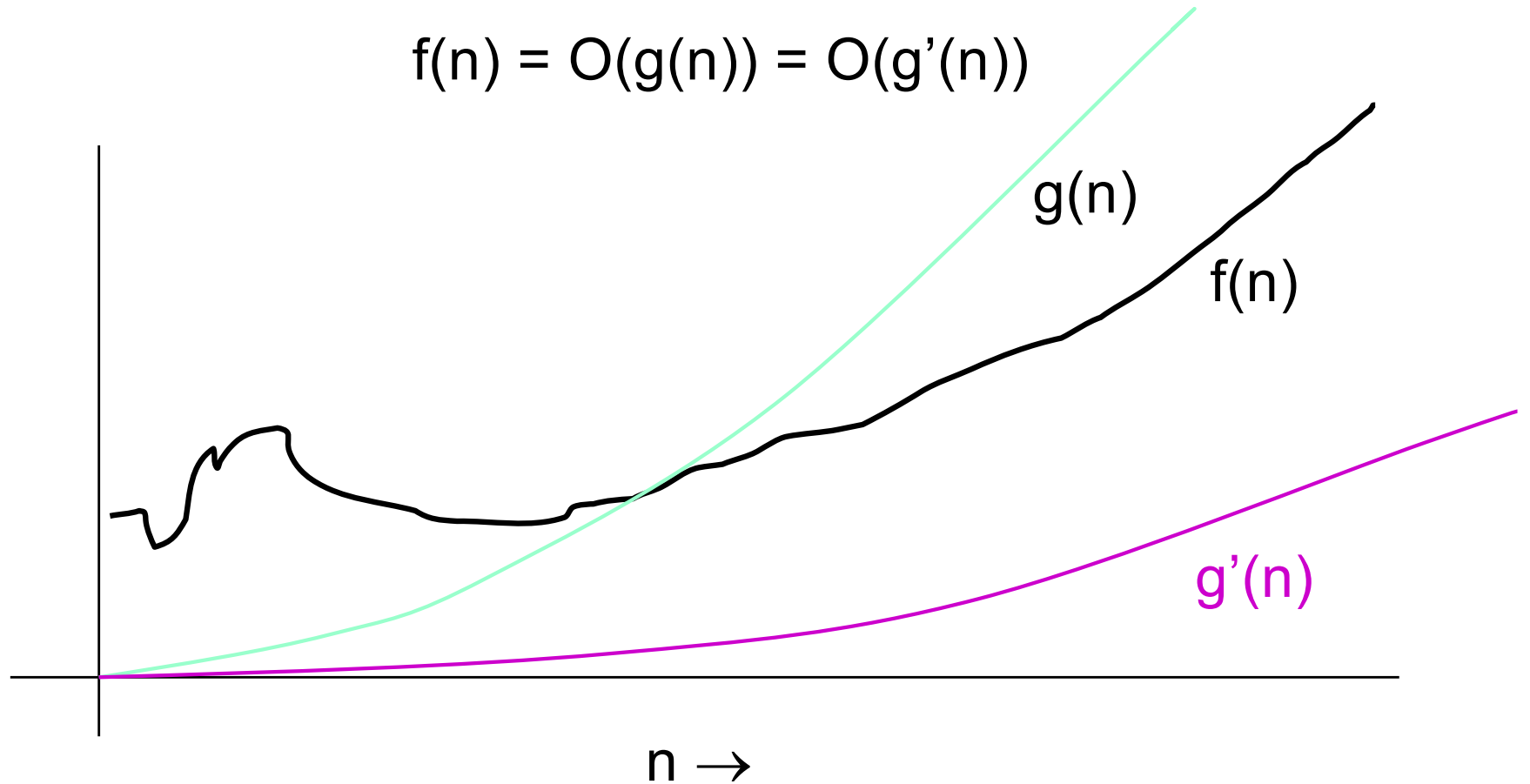
$$100n^2 + 100n + 100 = O(n^2) \quad [\text{e.g. } c = 101]$$

$$n^2 = O(2^{2n})$$

$$2^{2n} \text{ is } \textit{not} O(n^2)$$

Big-O Example

$$f(n) = O(g(n)) = O(g'(n))$$



Utility of Asymptotics

“All things being equal,” smaller asymptotic growth rate is better

All things are never equal

Even so, big-O bounds often let you quickly pick most promising candidates among competing algorithms

Poly time algs often practical; non-poly algs seldom are.

(Yes, there are exceptions.)

Fibonacci Numbers (recursion)

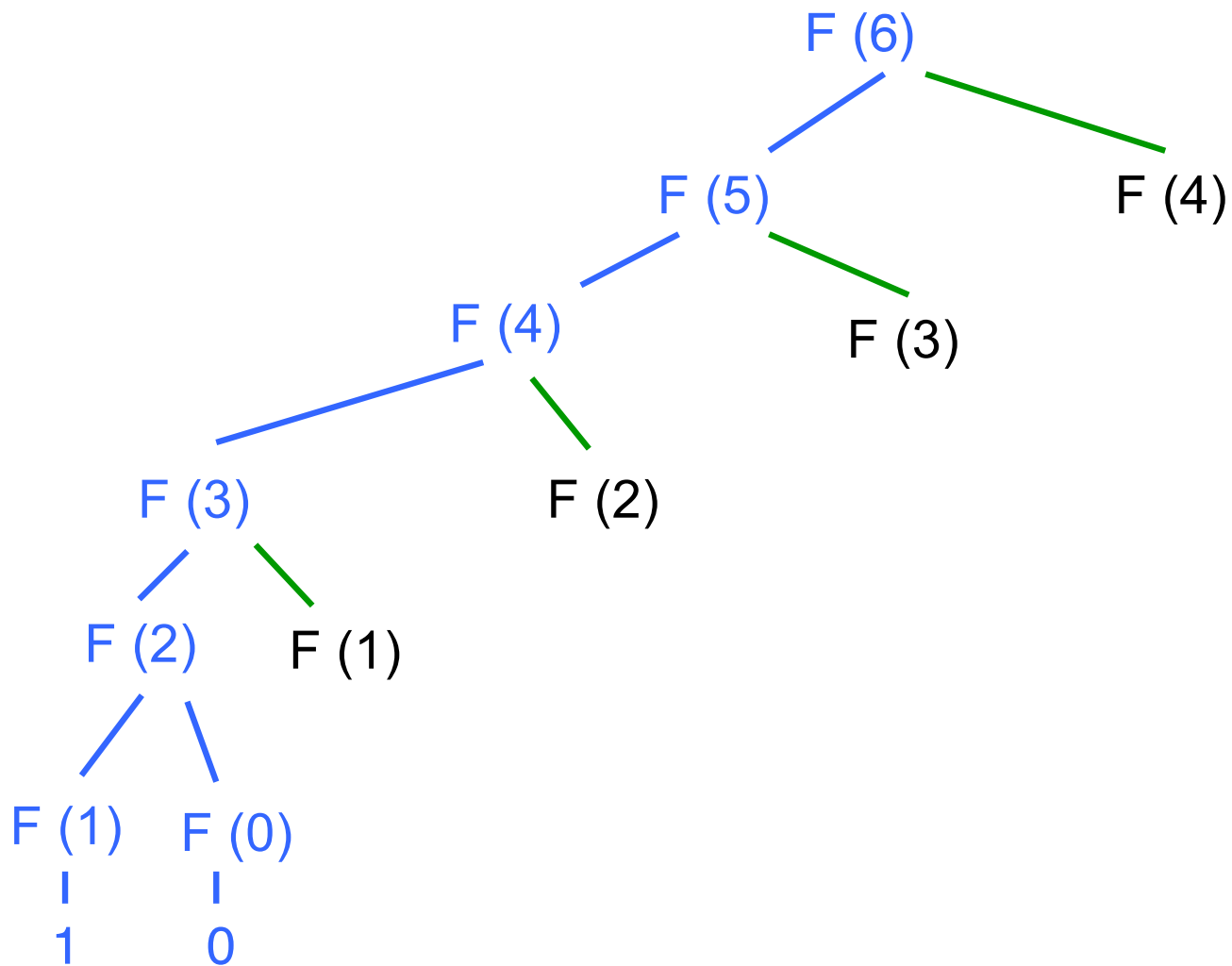
```
fibr(n) {  
  if (n <= 1) {  
    return n;  
  } else {  
    return fibr(n-1) + fibr(n-2);  
  }  
}
```

Simple recursion,
but many
repeated
subproblems!!

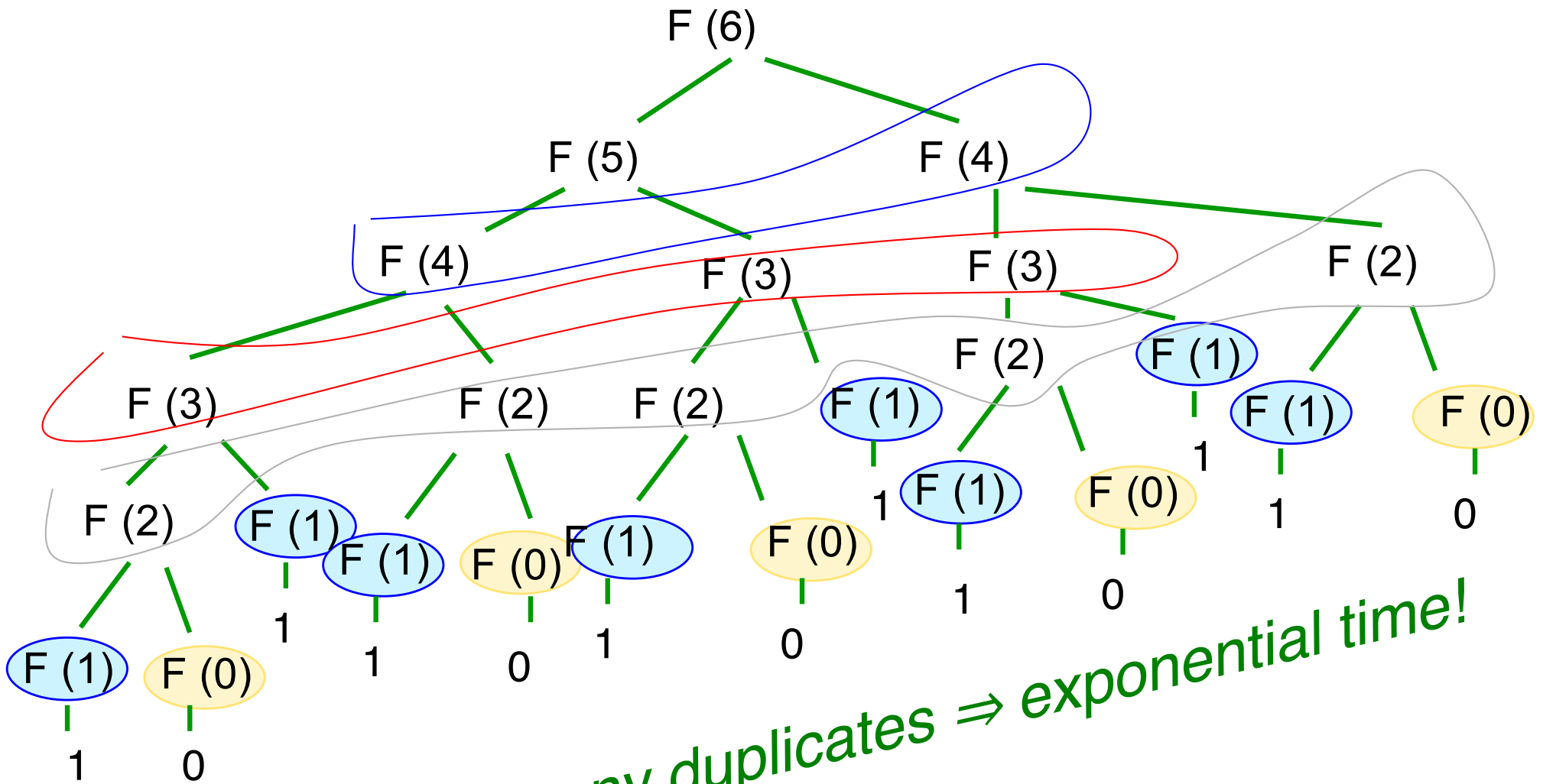
⇒

Time = $\Omega(1.61^n)$

Call tree - start



Full call tree



many duplicates \Rightarrow exponential time!

Fibonacci, II

(dynamic programming)

```
int fibd[n];
fibd[0] = 0;
fibd[1] = 1;
for(i=2; i<=n; i++) {
    fibd[i] = fibd[i-1] + fibd[i-2];
}
return fibd[n];
```

Avoid repeated
subproblems by
tabulating their
solutions

⇒

Time = $O(n)$

(in this case)

When can we use Dynamic Programming?

1. Can we decompose into **subproblems**?

E.g., can we align smaller substrings (say, prefix/suffix in this case), then combine them somehow?

2. Do we have **optimal substructure**?

I.e., is optimal solution to a subproblem *independent of context*? E.g., is appending two optimal alignments also optimal? Perhaps, but some changes at the interface might be needed?

Optimal Substructure (In More Detail)

Optimal alignment *ends* in 1 of 3 ways:

last chars of S & T aligned with each other

last char of S aligned with dash in T

last char of T aligned with dash in S

(assume $\sigma(-, -) < 0$, so never align dash with dash)

*In each case, the **rest** of S & T should be **optimally** aligned to each other*

Optimal Alignment in $O(n^2)$ via “Dynamic Programming”

Input: $S, T, |S| = n, |T| = m$

Output: **value** of optimal alignment

Easier to solve a “harder” problem:

$V(i,j)$ = value of optimal alignment of
 $S[1], \dots, S[i]$ with $T[1], \dots, T[j]$
for **all** $0 \leq i \leq n, 0 \leq j \leq m$.

Base Cases

$V(i,0)$: first i chars of S all match dashes

$$V(i,0) = \sum_{k=1}^i \sigma(S[k], -)$$

$V(0,j)$: first j chars of T all match dashes

$$V(0,j) = \sum_{k=1}^j \sigma(-, T[k])$$

General Case

Opt align of $S[1], \dots, S[i]$ vs $T[1], \dots, T[j]$:

$$\left[\begin{array}{c} \sim\sim\sim\sim S[i] \\ \sim\sim\sim\sim T[j] \end{array} \right], \quad \left[\begin{array}{c} \sim\sim\sim\sim S[i] \\ \sim\sim\sim\sim - \end{array} \right], \quad \text{or} \quad \left[\begin{array}{c} \sim\sim\sim\sim - \\ \sim\sim\sim\sim T[j] \end{array} \right]$$

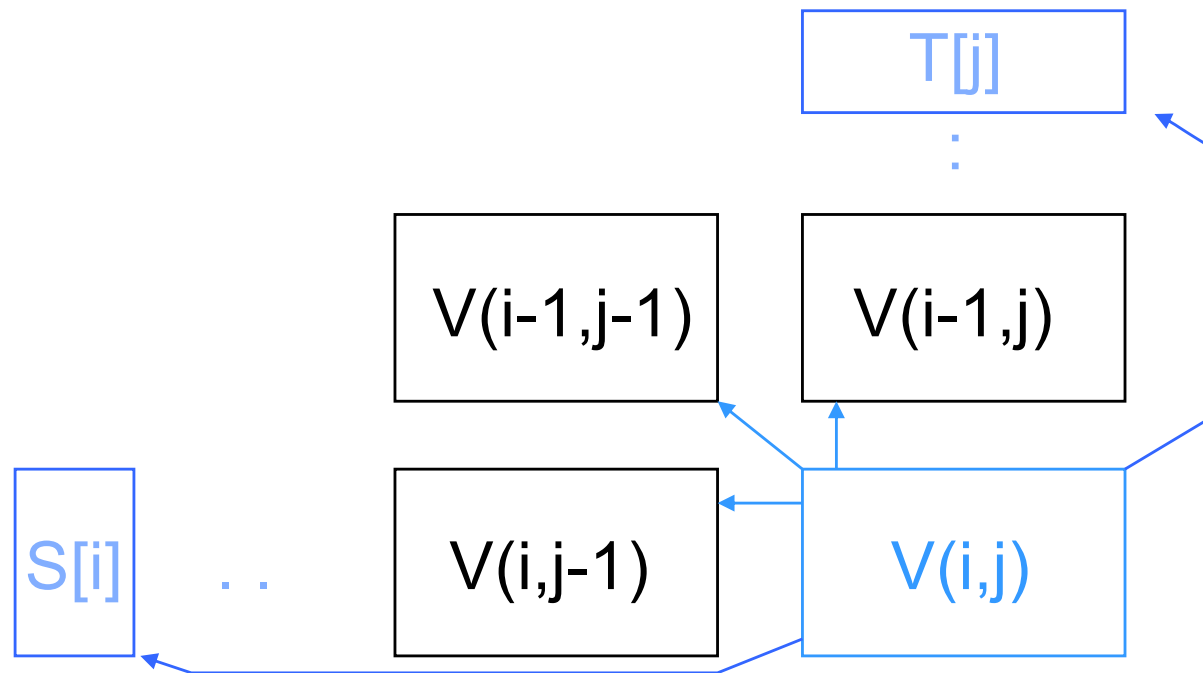
Opt align of
 $S_1 \dots S_{i-1}$ &
 $T_1 \dots T_{j-1}$

$$V(i,j) = \max \left\{ \begin{array}{l} V(i-1,j-1) + \sigma(S[i],T[j]) \\ V(i-1,j) + \sigma(S[i], -) \\ V(i,j-1) + \sigma(-, T[j]) \end{array} \right\},$$

for all $1 \leq i \leq n, 1 \leq j \leq m$.

Calculating One Entry

$$V(i,j) = \max \left\{ \begin{array}{l} V(i-1,j-1) + \sigma(S[i], T[j]) \\ V(i-1,j) + \sigma(S[i], -) \\ V(i,j-1) + \sigma(-, T[j]) \end{array} \right\}$$



Example

Mismatch = -1
Match = 2

	j	0	1	2	3	4	5
i			c	a	t	g	t
0		0	-1	-2	-3	-4	-5
1	a	-1					
2	c	-2					
3	g	-3					
4	c	-4					
5	t	-5					
6	g	-6					

←T

↑S

c
-
Score(c,-) = -1

Mismatch = -1
Match = 2

Example

	j	0	1	2	3	4	5
i			c	a	t	g	t
0		0	-1	-2	-3	-4	-5
1	a	-1					
2	c	-2					
3	g	-3					
4	c	-4					
5	t	-5					
6	g	-6					

←T

↑S

$\begin{matrix} - \\ a \end{matrix}$ Score(-,a) = -1

Example

Mismatch = -1
Match = 2

	j	0	1	2	3	4	5
i			c	a	t	g	t
0		0	-1	-2	-3	-4	-5
1	a	-1					
2	c	-2					
3	g	-3					
4	c	-4					
5	t	-5					
6	g	-6					

←T

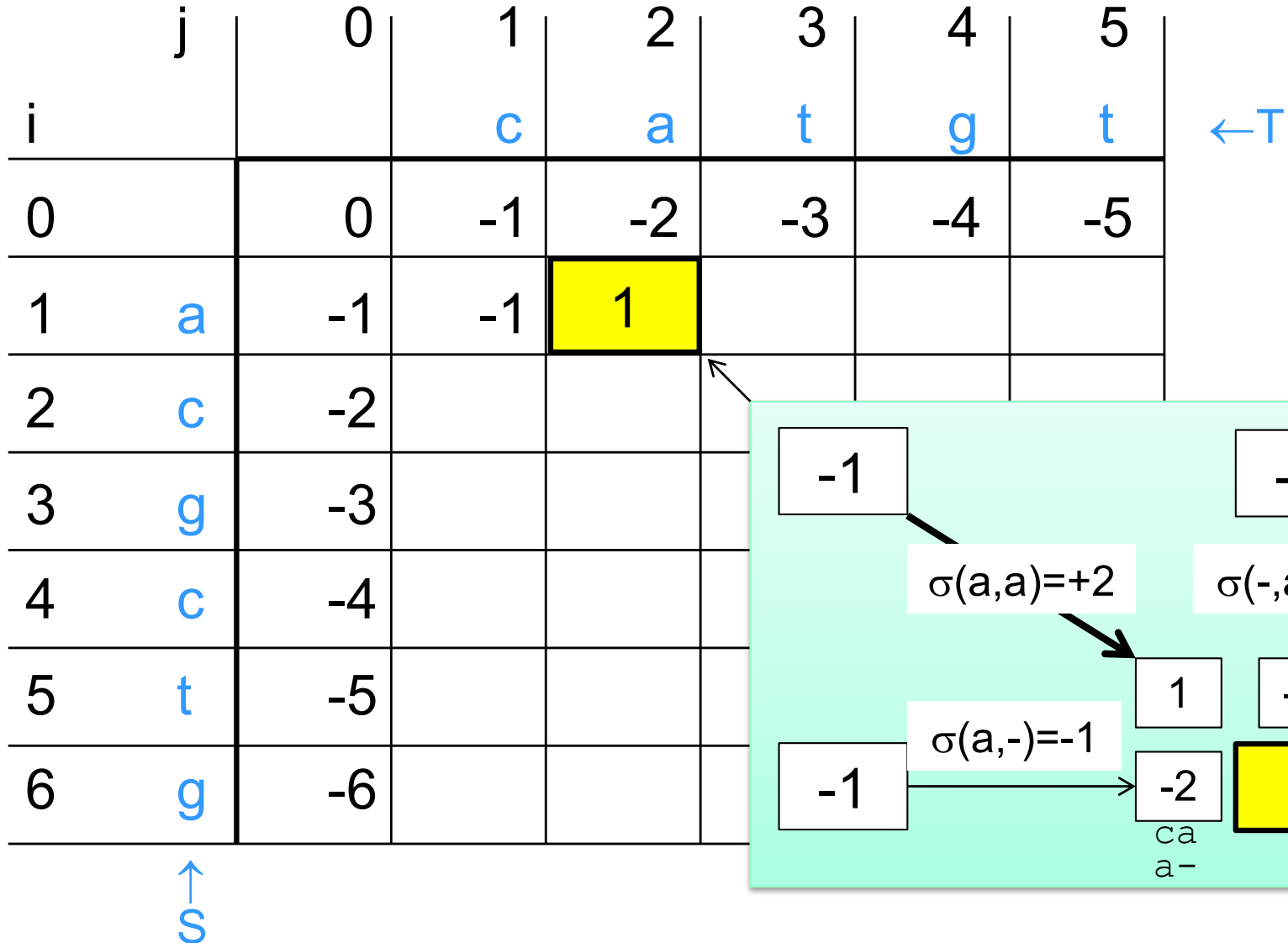
↑S

-	-
a	c
-1	

Score(-,c) = -1

Mismatch = -1
 Match = 2

Example



Example

Mismatch = -1

Match = 2

	j	0	1	2	3	4	5
i			c	a	t	g	t
0		0	-1	-2	-3	-4	-5
1	a	-1	-1	1			
2	c	-2	1				
3	g	-3					
4	c	-4					
5	t	-5					
6	g	-6					

←T

Time =
 $O(mn)$

↑
S

Mismatch = -1
Match = 2

Example

	j	0	1	2	3	4	5
i			c	a	t	g	t
0		0	-1	-2	-3	-4	-5
1	a	-1	-1	1	0	-1	-2
2	c	-2	1	0	0	-1	-2
3	g	-3	0	0	-1	2	1
4	c	-4	-1	-1	-1	1	1
5	t	-5	-2	-2	1	0	3
6	g	-6	-3	-3	0	3	2

←T

↑S

Finding Alignments: Trace Back

Arrows = (ties for) max in $V(i,j)$; 3 LR-to-UL paths = 3 optimal alignments

	j	0	1	2	3	4	5
i			c	a	t	g	t
0		0	-1	-2	-3	-4	-5
1	a	-1	-1	1	0	-1	-2
2	c	-2	1	0	0	-1	-2
3	g	-3	0	0	-1	2	1
4	c	-4	-1	-1	-1	1	1
5	t	-5	-2	-2	1	0	3
6	g	-6	-3	-3	0	3	2

← T

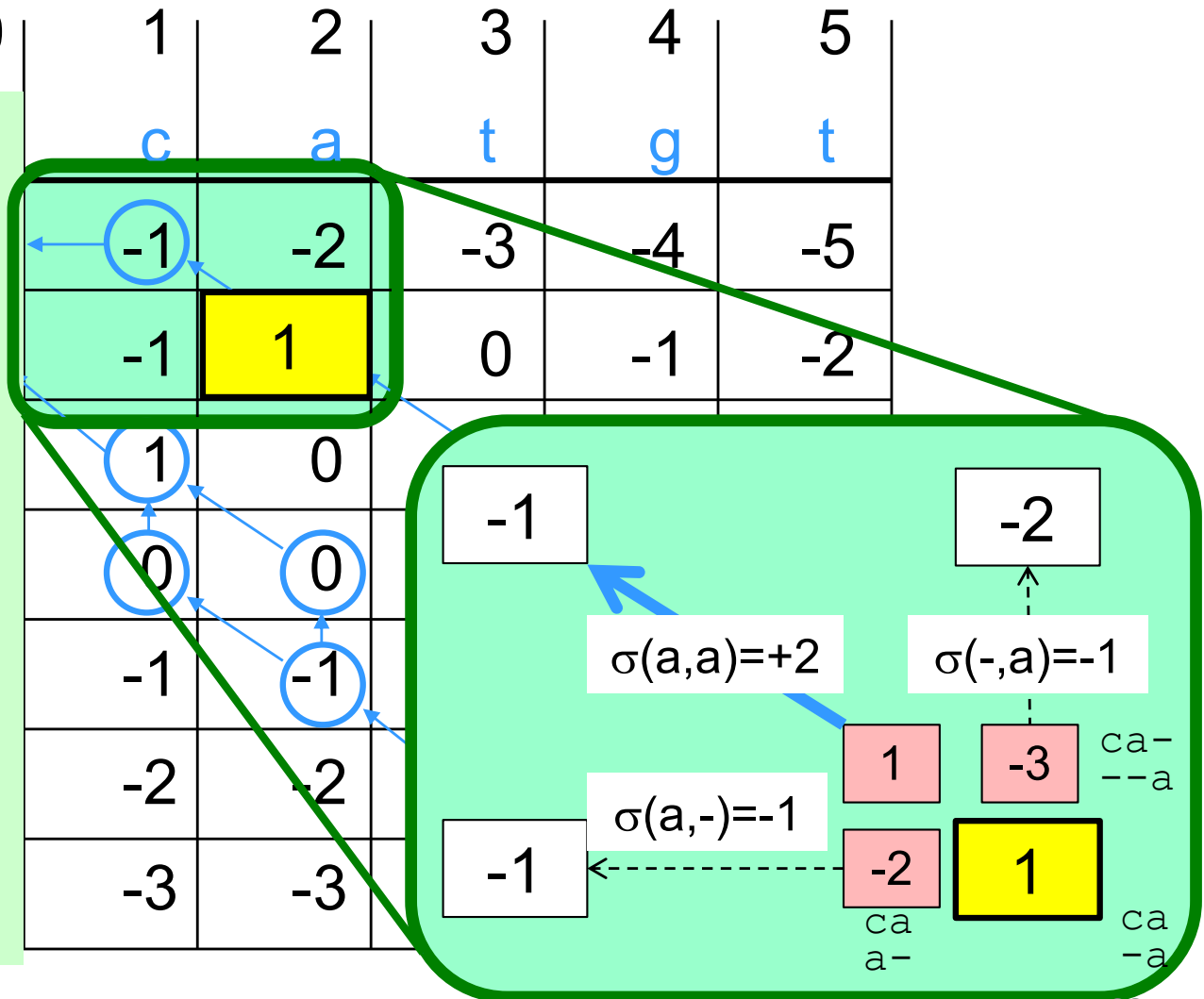
↑ S

Ex: what are the 3 alignments? C.f. slide 12.

Finding Alignments: Trace Back

Arrows = (ties for) max in $V(i,j)$; 3 LR-to-UL paths = 3 optimal alignments

NB: trace back follows max *terms* (pink boxes; $ngbr + \sigma$), not max neighbors (white boxes). E.g., TB from yellow cell is only *diagonal* ($ngbr = -1$, $term = 1$), not to the equally-good horizontal neighbor ($term = -2$)



Complexity Notes

Time = $O(mn)$, (value and alignment)

Space = $O(mn)$

Easy to get **value** in Time = $O(mn)$ and
Space = $O(\min(m,n))$

Possible to get value *and alignment* in
Time = $O(mn)$ and Space = $O(\min(m,n))$,
but tricky (DEKM 2.6)

CSEP 527
Computational Biology
Autumn 2020

Lecture 4
Sequence Alignment, Part II
Local Alignment & Gaps

Variations

Local Alignment

Preceding gives *global* alignment, i.e. full length of both strings;

Might well miss strong similarity of *part* of strings amidst dissimilar flanks

Gap Penalties

10 adjacent dashes cost 10 x one dash?

Many others

Similarly fast DP algs often possible

Local Alignment: Motivations

“Interesting” (evolutionarily conserved, functionally related) segments may be a small part of the whole

- “Active site” of a protein

- Scattered genes or exons amidst “junk”, e.g. retroviral insertions, large deletions

- Don't have whole sequence

Global alignment might miss them if flanking junk outweighs similar regions

Local Alignment

Optimal *local alignment* of strings S & T:
Find substrings A of S and B of T having
max value global alignment

S = abcxdex A = c x d e

T = xxxcde B = c - d e value = 5 (toy σ)

Local Alignment: “Obvious” Algorithm

for all substrings A of S and B of T :
 Align A & B via dynamic programming
 Retain pair with max value
end ;
Output the retained pair

Time: $O(n^2)$ choices for A , $O(m^2)$ for B ,
 $O(nm)$ for DP, so $O(n^3m^3)$ total.

[Best possible? Lots of redundant work...]

Local Alignment in $O(nm)$ via Dynamic Programming

Input: $S, T, |S| = n, |T| = m$

Output: value of optimal **local** alignment

Better to solve a “harder” problem
for all $0 \leq i \leq n, 0 \leq j \leq m$:

$V(i,j) = \mathbf{max}$ value of opt (global)
alignment of a **suffix** of $S[1], \dots, S[i]$
with a **suffix** of $T[1], \dots, T[j]$

Report best i,j

Base Cases

Assume $\sigma(x,-) < 0$, $\sigma(-,x) < 0$

$V(i,0)$: some suffix of first i chars of S ; all match dashes in T ; best suffix is empty

$$V(i,0) = 0$$

$V(0,j)$: similar

$$V(0,j) = 0$$

General Case Recurrences

Opt **suffix** align $S[1], \dots, S[i]$ vs $T[1], \dots, T[j]$:

$$\left[\begin{array}{c} \sim\sim\sim\sim S[i] \\ \sim\sim\sim\sim T[j] \end{array} \right], \left[\begin{array}{c} \sim\sim\sim\sim S[i] \\ \sim\sim\sim\sim - \end{array} \right], \left[\begin{array}{c} \sim\sim\sim\sim - \\ \sim\sim\sim\sim T[j] \end{array} \right], \text{ or } \left[\begin{array}{c} \\ \\ \end{array} \right]$$

Opt align of
suffix of
 $S_1 \dots S_{i-1}$ &
 $T_1 \dots T_{j-1}$

$$V(i,j) = \max \left\{ \begin{array}{l} V(i-1,j-1) + \sigma(S[i], T[j]) \\ V(i-1,j) + \sigma(S[i], -) \\ V(i,j-1) + \sigma(-, T[j]) \\ 0 \end{array} \right\},$$

opt suffix
alignment
has:
2, 1, 1, 0
chars of
S/T

for all $1 \leq i \leq n, 1 \leq j \leq m$.

Scoring Local Alignments

	j	0	1	2	3	4	5	6
i			x	x	x	c	d	e
0		0	0	0	0	0	0	0
1	a	0						
2	b	0						
3	c	0						
4	x	0						
5	d	0						
6	e	0						
7	x	0						

↑
s

← T

Finding Local Alignments

Again, arrows follow max *term* (not max neighbor)

	j	0	1	2	3	4	5	6
i			x	x	x	c	d	e
0		0	0	0	0	0	0	0
1	a	0	0	0	0	0	0	0
2	b	0	0	0	0	0	0	0
3	c	0	0	0	0	2	1	0
4	x	0	2	2	2	1	1	0
5	d	0	1	1	1	1	3	2
6	e	0	0	0	0	0	2	5
7	x	0	2	2	2	1	1	4

← T

One alignment is:

c-de
cxde

What's the other?

↑ S

Notes

Time and Space = $O(mn)$

Space $O(\min(m,n))$ possible with time $O(mn)$, but finding alignment is trickier

Local alignment: “Smith-Waterman”

Global alignment: “Needleman-Wunsch”

Sequence Evolution

“Nothing in Biology Makes Sense Except in the Light of Evolution” – Theodosius Dobzhansky, 1973

Changes happen at random

Deleterious/neutral/advantageous changes
unlikely/possibly/likely spread widely in a population

Changes are less likely to be tolerated in positions
involved in many/close interactions, e.g.

- enzyme binding pocket

- protein/protein interaction surface

- ...

Alignment With Gap Penalties

A Gap: maximal run of dashes in S' or T'

ag--ttc-t 2 gaps in S'

a---ttcgt 1 gap in T'

Motivations, e.g.:

mutation might insert/delete several or even many residues at once

some parts of proteins less critical

matching mRNA (no introns) to genomic DNA (exons and introns)

A Protein Structure: (Dihydrofolate Reductase)



Alignment of 5 Dihydrofolate reductase proteins

```

mouse P00375 ----MVRPLNCIVAVSQNMGIGKNGDLPWPPLRNEFKYFQRM TTTSSVEGKQNLVIMGRK
human P00374 ----MVGSLNCIVAVSQNMGIGKNGDLPWPPLRNEFRYFQRM TTTSSVEGKQNLVIMGKK
chicken P00378 ----VRSLSNIVAVCQNMGIGKDG NLPWPPLRNEYKYFQRM TSTSHVEGKQNAVIMGKK
fly P17719 ----MLR-FNLIVAVCENFGIGIRG DLPWR- IKSELKYFSRTTKRTSDPTKQNAVVMGRK
yeast P07807 MAGGKIPIVGIVACLQPEMGIGFRG GLPWR- LPSEMKYFRQV TSLTKDPNKKNALIMGRK

```

```

      :  . . : . . : : : * * * * * . * * * * : . * : * * : * . : * : * : : * * : *

```

```

P00375 TWFSIPEKNRPLKDRINIVLSRELKEP----PRGAHFLAKSLDDALRLIEQPELASKVDM
P00374 TWFSIPEKNRPLKGRINLVLSRELKEP----PQGAHFLSRSLDDALKLTEQPELANKVDM
P00378 TWFSIPEKNRPLKDRINIVLSRELKEA----PKG AHYLSKSLDDALALLDSP ELKSKVDM
P17719 TYFGVPESKRPLPDRLNIVLSTTLQESDL--PKG-VLLCPNLETAMKILEE---QNEVEN
P07807 TWESIPPKFRPLPNRMNVIISRSFKDDFVHDKERSIVQSNLANAIMNLESN-FKEHLER

```

```

* : . : * . * * * . * : * : : * : : : . . . * * : : . . . : :

```

```

P00375 VWIVGGSSVYQEAMNQP GHLRLFVTRIMQEFESDTFFPEIDL GKYKLLPEYPG-----
P00374 VWIVGGSSVYKEAMNHP GHLKLFVTRIMQDFESDTFFPEIDL EKYKLLPEYPG-----
P00378 VWIVGGTAVYKAAMEKPI NHRLFVTRILHEFESDTFFPEIDYKDFKLLTEYPG-----
P17719 IWIVGGSGVYEEAMASPR CHRLYITKIMQKFD CDTFPPAIP-DSFREVAPDS-----
P07807 IYVIGGGEVYSQIFSI TDHWLITKINPLDKNATPAMDTFLDAKKLEEVFSEQDPAQLKEF

```

```

: : : : * * * * . : . : . . . : : . : . . : . . : .

```

```

P00375 VLSEVQ-----EEKGIKYKFEVYEKKD----
P00374 VLSDVQ-----EEKGIKYKFEVYEKND----
P00378 VPADIQ-----EEDGIQYKFEVYQKSVLAQ
P17719 MPLGVQ-----EENG I KFEYKILEKHS----
P07807 LPPKVELPETDCDQRYSLEEKGYCFEFTLYNRK----

```

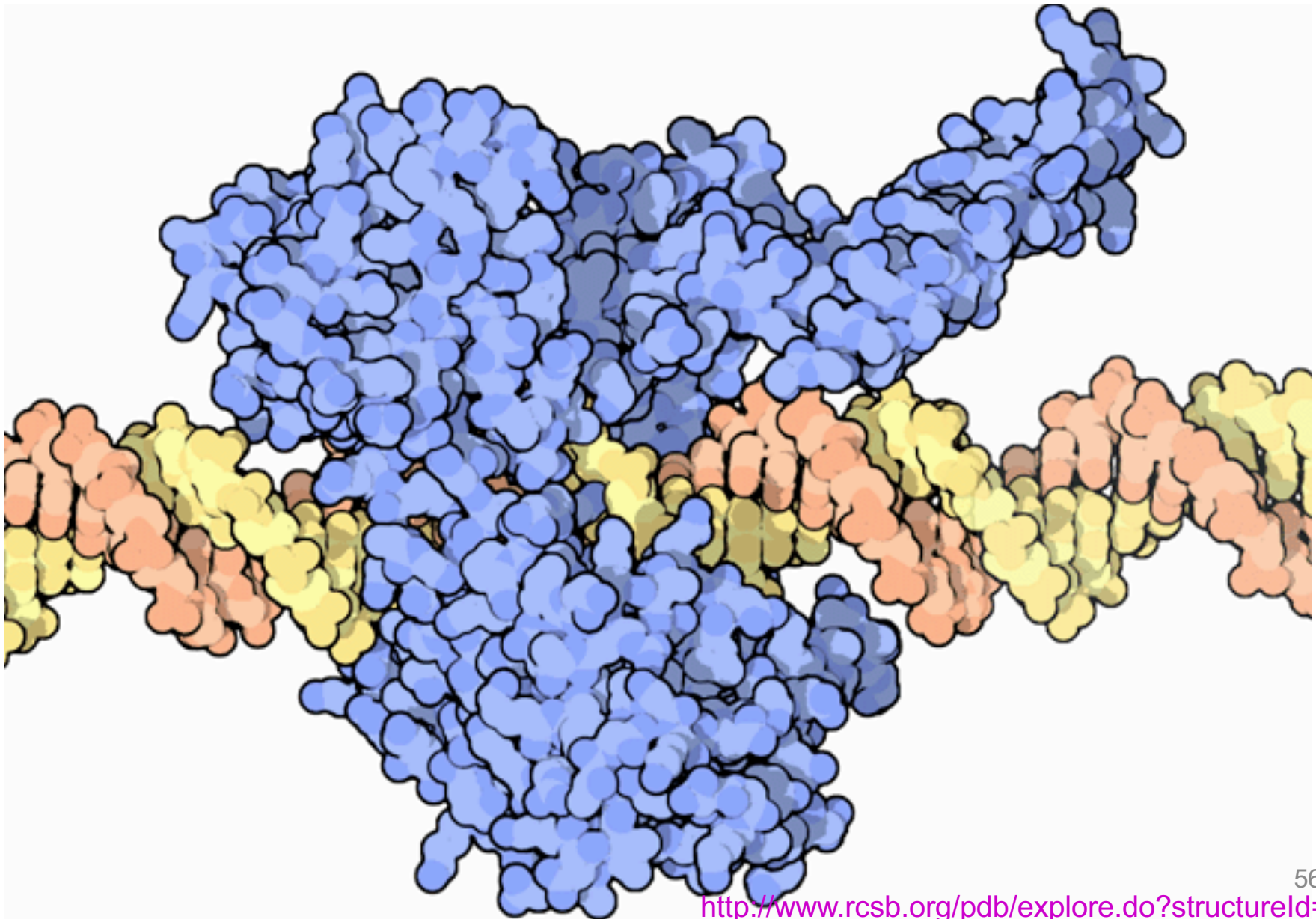
```

: : : * * . * : : : : :

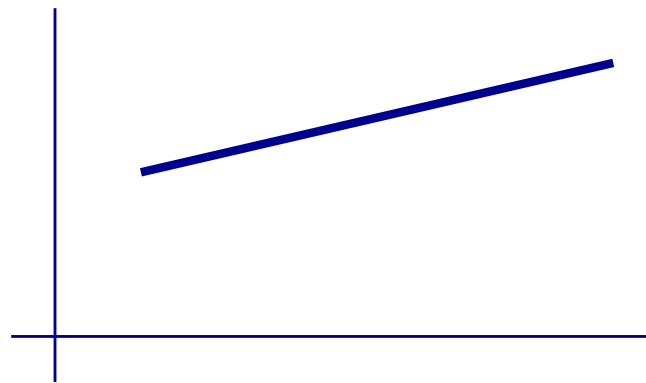
```

CLUSTAL W (1.82) multiple sequence alignment
<http://pir.georgetown.edu/cgi-bin/multaln.pl> 2/11/2013

Topoisomerase I



Affine Gap Penalties



$$\text{Gap penalty} = g + e^*(\text{gaplen}-1), \quad g \geq e \geq 0$$

Note: no longer suffices to know just the *score* of best subproblem(s) – *state* matters: do they end with ‘-’ or not.

Global Alignment with Affine Gap Penalties

$V(i,j)$ = value of opt alignment of $S[1], \dots, S[i]$ with $T[1], \dots, T[j]$

$G(i,j)$ = ..., s.t. last pair matches $S[i]$ & $T[j]$

$F(i,j)$ = ..., s.t. last pair matches $S[i]$ & –

$E(i,j)$ = ..., s.t. last pair matches – & $T[j]$

S	T
x/	x/
–	–
x	x
x	–
–	x

Time: $O(mn)$ [calculate all, $O(1)$ each]

Affine Gap Algorithm

Gap penalty = $g + e^*(\text{gaplen}-1)$, $g \geq e \geq 0$

$V(i,0) = E(i,0) = V(0,i) = F(0,i) = -g - (i-1)*e$

$V(i,j) = \max(G(i,j), F(i,j), E(i,j))$

$G(i,j) = V(i-1,j-1) + \sigma(S[i],T[j])$

$F(i,j) = \max(\boxed{F(i-1,j)-e}, \boxed{V(i-1,j)-g})$

$E(i,j) = \max(\boxed{E(i,j-1)-e}, \boxed{V(i,j-1)-g})$

old gap

new gap


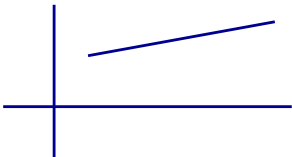
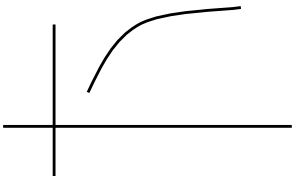
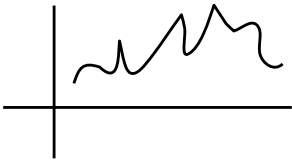
S	T
x/	x/
-	-
x	x
x	-
-	x

Q. Why is the “V” case a “new gap” when V includes E & F?

Other Gap Penalties

Score = $f(\text{gap length})$

Kinds, & best known alignment time

 affine		$O(n^2)$ [really, $O(mn)$]
convex		$O(n^2 \log n)$
general		$O(n^3)$

Summary: Alignment

Functionally similar proteins/DNA often have recognizably similar sequences even after eons of divergent evolution

Ability to find/compare/experiment with “same” sequence in other organisms is a huge win

Surprisingly simple scoring works well in practice: score positions separately & add, usually w/ fancier affine gaps

Simple dynamic programming algorithms can find *optimal* alignments under these assumptions in poly time (product of sequence lengths)

This, and heuristic approximations to it like BLAST, are workhorse tools in molecular biology, and elsewhere.

Summary: Dynamic Programming

Keys to D.P. are to

- a) Identify the subproblems (usually repeated/overlapping)
- b) Solve them in a careful order so all small ones solved before they are needed by the bigger ones, and
- c) Build table with solutions to the smaller ones so bigger ones just need to do table lookups (*no* recursion, despite recursive formulation implicit in (a))
- d) Implicitly, optimal solution to whole problem devolves to optimal solutions to subproblems

A really important algorithm design paradigm

Significance of Alignment Scores

Is “42” a good score?

Compared to what?

Usual approach: compared to a specific “null model”, such as “random sequences”

More on this later; a taste now, for use in next HW

Overall Alignment Significance, I

Empirical p-values (via randomization)

You just searched with x , found “good” score for $x:y$

Generate N random “ y -like” sequences (say $N = 10^3 - 10^6$)

Align x to each & score

If k of them have score than better or equal to that of x to y , then the (empirical) probability of a chance alignment as good as your observed $x:y$ alignment is $(k+1)/(N+1)$

e.g., if 0 of 99 are better, you can say “estimated $p \leq .01$ ”

How to gen “random y -like” seqs? Scores depend on:

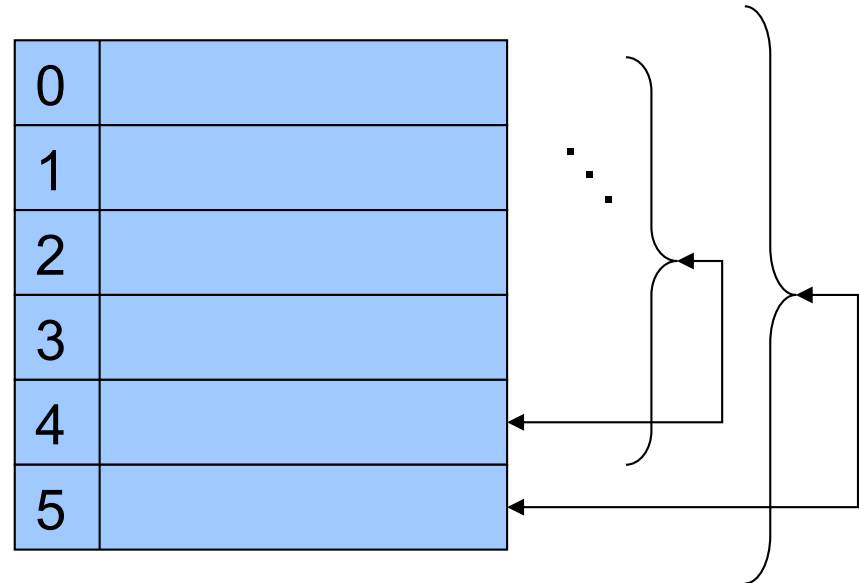
Length, so use same length as y

Sequence composition, so uniform $1/20$ or $1/4$ is a bad idea; even background p_i can be dangerous (if y unusual)

Better idea: *permute* y N times: exactly preserves len & composition

Generating Random Permutations

```
for (i = n-1; i > 0; i--){  
    j = random(0..i);  
    swap X[i] <-> X[j];  
}
```



All $n!$ permutations of the original data equally likely: A specific element will be last with prob $1/n$; given that, another specific element will be next-to-last with prob $1/(n-1)$, ...; overall: $1/(n!)$

C.f. http://en.wikipedia.org/wiki/Fisher–Yates_shuffle and (for subtle way to go wrong) <http://www.codinghorror.com/blog/2007/12/the-danger-of-naivete.html>