CSE 312
Winter 2011

More Algorithms:
Dynamic Programming for
Sequence Alignment

Algorithm Design Techniques

Dynamic Programming

- Give a solution to a problem using smaller sub-problems, e.g. a recursive solution
- Useful when the same sub-problems show up again and again in the solution

“Dynamic Programming”

Program –
A plan or procedure for dealing with some matter

– Webster’s New World Dictionary

Pioneered by Richard Bellman in the 1950s.

Dynamic programming = “planning over time”

Secretary of Defense was hostile to mathematical research.

Bellman sought an impressive name to avoid confrontation.

"it's impossible to use dynamic in a pejorative sense"

"something not even a Congressman could object to”

Sequence Similarity: What

G G A C C A

T A C T A A G

T C C A A T

Sequence Similarity: What

G G A C C A

```
T A C T A A G
|  :  |  :  |  |  :
T C C – A A T
```

Sequence Similarity: Why

Most widely used comp. tools in biology

New sequence always compared to sequence data bases

Similar sequences often have similar origin or function

Recognizable similarity after 10^8 –10^9 yr

(not to mention Unix “diff”, module histories in version control systems, a large number of programming assignments with sadly questionable evolutionary history, etc., etc., …)

Bio Example

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

Try it!

pick any protein, e.g. hemoglobin, insulin, exportin,… BLAST to find distant relatives.

Starting with a protein sequence from:

us

I easily found related ones in:

rat
mouse
pig
frog
fly
a worm
fungus
plants
...

Taxonomy Report

```
root ................................    64 hits   16 orgs
. Eukaryota .........................    62 hits   14 orgs
. . Fungi/Metazoa group .............    57 hits   11 orgs
. . . Bilateria .....................    38 hits    7 orgs
. . . . Coelomata ...................    36 hits    6 orgs
. . . . . Tetrapoda .................    26 hits    5 orgs
. . . . . . Eutheria ................    24 hits    4 orgs
. . . . . . . Homo sapiens ..........    20 hits    1 orgs
. . . . . . . Murinae ...............     3 hits    2 orgs
. . . . . . . . Rattus norvegicus ....    2 hits    1 orgs
. . . . . . . . Mus musculus ........     1 hits    1 orgs
. . . . . . . Sus scrofa ............     1 hits    1 orgs
. . . . . . Xenopus laevis ..........     2 hits    1 orgs
. . . . . Drosophila melanogaster ....   10 hits    1 orgs
. . . . Caenorhabditis elegans .......    2 hits    1 orgs
. . . Ascomycota ....................    19 hits    4 orgs
. . . . Schizosaccharomyces pombe ....   10 hits    1 orgs
. . . . Saccharomycetales ...........     9 hits    3 orgs
. . . . . Saccharomyces .............     8 hits    2 orgs
. . . . . . Saccharomyces cerevisiae .    7 hits    1 orgs
. . . . . . Saccharomyces kluyveri ...    1 hits    1 orgs
. . . . . Candida albicans ..........     1 hits    1 orgs
. . Arabidopsis thaliana ............     2 hits    1 orgs
. . Apicomplexa .....................     3 hits    2 orgs
. . . Plasmodium falciparum .........     2 hits    1 orgs
. . . Toxoplasma gondii .............     1 hits    1 orgs
. synthetic construct ...............     1 hits    1 orgs
. lymphocystis disease virus ........     1 hits    1 orgs
```

Terminology

String: ordered list of letters TATAAG

Prefix: consecutive letters from front

empty, T, TA, TAT, ...

Suffix: … from end

empty, G, AG, AAG, ...

Substring: … from ends or middle

empty, TAT, AA, ...

Subsequence: ordered, nonconsecutive

TT, AAA, TAG, ...

Sequence Alignment

a c b c d b
c a d b d

a c – – b c d b
– c a d b – d –

Defn: An *alignment* of strings S, T is a pair of strings S’, T’ (with spaces or ‘-’) s.t.

(1) |S’| = |T’|, and (|S| = “length of S”)

(2) removing all spaces leaves S, T

Alignment Scoring

Mismatch	= -1
Match	= 2

```
a c b c d b
c a d b d d
```

```
a   c   -   -   b   c   d   b
-   c   a   d   b   -   d   d
-1  2  -1  -1   2  -1   2  -1
Value = 3*2 + 5*(-1) = +1
```

The *score* of aligning (characters or spaces) x & y is $\sigma(x,y)$.

Value of an alignment $\sum_{i=1}^{|S'|} \sigma(S'[i], T'[i])$

An *optimal alignment:* one of max value

Optimal Alignment: A Simple Algorithm

(For simplicity, assume |S| = |T| = n)

for all choices of n positions from S&T combined **do**

  align the k chosen letters of S with the k *un*chosen letters of T

  align all other chars to spaces

  compute its value

  retain the max

end

output the retained alignment

S = ab<u>cde</u>
T = <u>vw</u>x<u>y</u>z

```
--abc-de    ab--c-de
vw--xyz-    --vwxyz-
```

(two alignments, but same value)

Analysis

Assume |S| = |T| = n

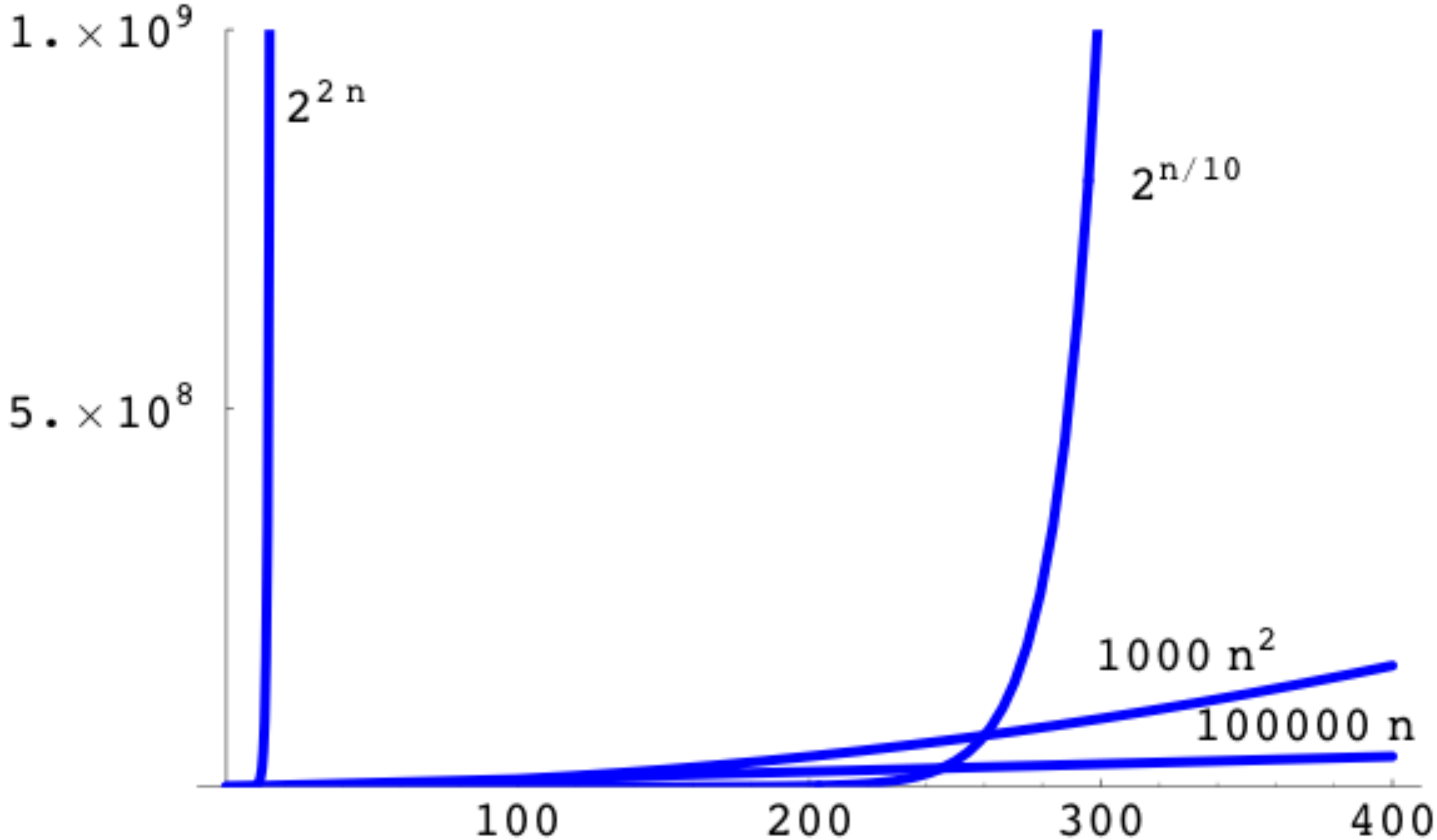
Time to evaluate one alignment: n

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

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

E.g., for n = 20, time is > $2^{40} > 10^{12}$ operations

Polynomial vs Exponential Growth



Dynamic Programming Alg: The Key Idea

Optimal alignment *ends* in 1 of 3 ways:

- last chars of S & T aligned with each other
- last char of S aligned with space in T
- last char of T aligned with space in S
- (never align space with space; $\sigma(-, -) < 0$)

In each case, the *rest* of S & T should be *optimally* aligned to each other
(else you could improve by doing so)

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], …, S[i] with T[1], …, T[j]

for all $0 \le i \le n$, $0 \le j \le m$.

Base Cases

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

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

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

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

General Case

Opt align of S[1], …, S[i] vs T[1], …, T[j]:

$$\begin{bmatrix} \sim\sim\sim\sim & T[j] \\ \sim\sim\sim\sim & S[i] \end{bmatrix}, \quad \begin{bmatrix} \sim\sim\sim\sim & T[j] \\ \sim\sim\sim\sim & - \end{bmatrix}, \text{ or } \begin{bmatrix} \sim\sim\sim\sim & - \\ \sim\sim\sim\sim & S[i] \end{bmatrix}$$

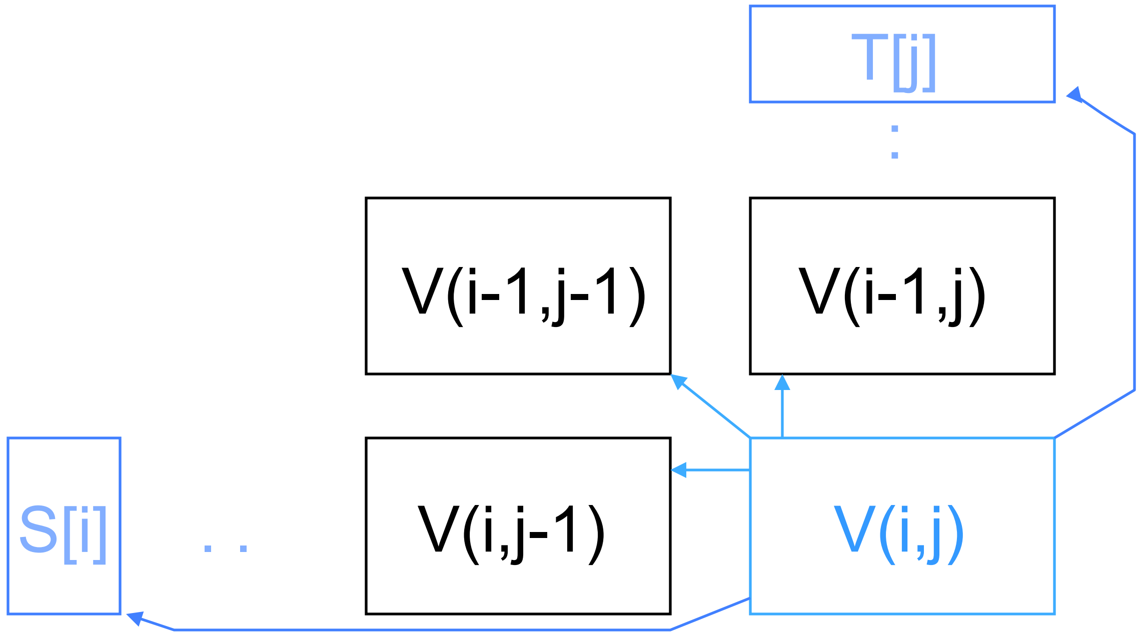
Opt align of $S_1 \ldots S_{i-1}$ & $T_1 \ldots T_{j-1}$

$$V(i,j) = \max \begin{Bmatrix} V(i\text{-}1, j\text{-}1) + \sigma(S[i], T[j]) \\ V(i\text{-}1, j) \quad + \sigma(S[i], \ - \ ) \\ V(i, j\text{-}1) \quad + \sigma(\ - \ , T[j]) \end{Bmatrix}, \quad \begin{matrix} \nwarrow \\ \uparrow \\ \leftarrow \end{matrix}$$

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

Calculating One Entry

$$V(i,j) = \max \begin{cases} V(i\text{-}1,j\text{-}1) + \sigma(S[i],T[j]) \\ V(i\text{-}1,j) \quad + \sigma(S[i],\ \text{-}\ ) \\ V(i,j\text{-}1) \quad + \sigma(\ \text{-}\ ,\ T[j]) \end{cases}$$



Mismatch = -1
Match = 2

Example

j		0	1	2	3	4	5	
i			c	a	d	b	d	←T
0		0	$\boxed{-1}$	-2	-3	-4	-5	
1	a	-1						
2	c	-2						
3	b	-3						
4	c	-4						
5	d	-5						
6	b	-6						

↑
S

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

Mismatch = -1
Match = 2

Example

i	j	0	1	2	3	4	5
			c	a	d	b	d
0		0	-1	-2	-3	-4	-5
1	a	-1					
2	c	-2					
3	b	-3					
4	c	-4					
5	d	-5					
6	b	-6					

←T

↑ S

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

Mismatch = -1
Match = 2

Example

i \ j		0	1	2	3	4	5	
			c	a	d	b	d	←T
0		0	-1	-2	-3	-4	-5	
1	a	-1						
2	c	$\boxed{-2}$						
3	b	-3						
4	c	-4						
5	d	-5						
6	b	-6						
	↑ S							

-	-
a	c

-1

Score(-,c) = -1

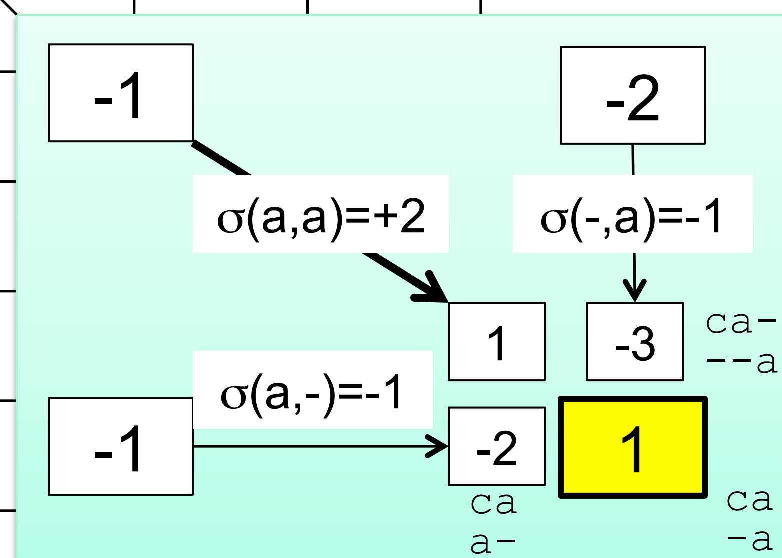
Mismatch = -1
Match = 2

Example

	j	0	1	2	3	4	5
i			c	a	d	b	d
0		0	-1	-2	-3	-4	-5
1	a	-1	-1	1			
2	c	-2					
3	b	-3					
4	c	-4					
5	d	-5					
6	b	-6					

←T

↑
S



Mismatch = -1
Match = 2

Example

j		0	1	2	3	4	5
i			c	a	d	b	d
0		0	-1	-2	-3	-4	-5
1	a	-1	-1	1			
2	c	-2	1				
3	b	-3					
4	c	-4					
5	d	-5					
6	b	-6					$\boxed{}$

← T

↑ S

Time = O(mn)

Mismatch = -1
Match = 2

Example

	j	0	1	2	3	4	5	
i			c	a	d	b	d	←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	b	-3	0	0	-1	2	1	
4	c	-4	-1	-1	-1	1	1	
5	d	-5	-2	-2	1	0	3	
6	b	-6	-3	-3	0	3	$\boxed{2}$	
	↑S							

Finding Alignments: Trace Back

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

i \ j		0	1	2	3	4	5
			c	a	d	b	d ←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	b	-3	0	0	-1	2	1
4	c	-4	-1	-1	-1	1	1
5	d	-5	-2	-2	1	0	3
6	b	-6	-3	-3	0	3	2

↑
S

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.

Significance of Alignments

Is “42” a good score?

Compared to what?

Hypothesis Testing again!

Compared to a specific “null model”, such as “random sequences,” is the data more likely under null or alt hypothesis?

Significance of Alignment

Usual context is “I just searched for S in a big database & my best match, T, scored 42”

Idea 1:

- Align & score S against many *random* sequences
- The fraction of them having better score than alignment of S to T, is the (empirical) probability of a chance alignment as good as observed S:T alignment (e.g., if 1 in 100 are better, then “p ≈ .01”)

Idea 2:

- Just as sums of r.v.’s converge to normal (CLT), *max* of a bunch of r.v.’s also converges to a limit distribution (called EVD), so use tails of that distribution to estimate prob of a chance match

Summary

Sequence similarity has important applications in biology and elsewhere

Surprisingly simple scoring often works well in practice: score positions separately & add

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

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))