Sequencing Alignment I

Lectures 16 – Nov 21, 2011
CSE 527 Computational Biology, Fall 2011

Instructor: Su-In Lee
TA: Christopher Miles

Monday & Wednesday 12:00-1:20
Johnson Hall (JHN) 022

Outline: Sequence Alignment

- What
- Why (applications)
    - Comparative genomics
    - DNA sequencing
- A simple algorithm
- Complexity analysis
- A better algorithm:
    - “Dynamic programming”

Sequence Alignment: What

- Definition
  - An arrangement of two or several biological sequences (e.g. protein or DNA sequences) highlighting their similarity
  - The sequences are padded with gaps (usually denoted by dashes) so that columns contain identical or similar characters from the sequences involved
- Example – pairwise alignment

```
T A C T A A G


T C C A A T
```

Sequence Alignment: What

- Definition
  - An arrangement of two or several biological sequences (e.g. protein or DNA sequences) highlighting their similarity
  - The sequences are padded with gaps (usually denoted by dashes) so that columns contain identical or similar characters from the sequences involved
- Example – pairwise alignment

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

Sequence Alignment: Why

- The most basic sequence analysis task
  - First aligning the sequences (or parts of them) and
  - Then deciding whether that alignment is more likely to have occurred because the sequences are related, or just by chance
- Similar sequences often have similar origin or function
- New sequence always compared to existing sequences (e.g. using BLAST)

Sequence Alignment

- Example: gene HBB
  - Product: hemoglobin
  - Sickle-cell anaemia causing gene
  - Protein sequence (146 aa)

```
MVHLTPEEKS AVTALWGKVN VDEVGGEALG RLLVVYPWTQ RFFESFGDLS TPDAVMGNPK
VKAHGKKVLG AFSDGLAHLD NLKGTFATLS ELHCDKLHVD PENFRLLGNV LVCVLAHHFG
KEFTPPVQAA YQKVVAGVAN ALAHKYH
```

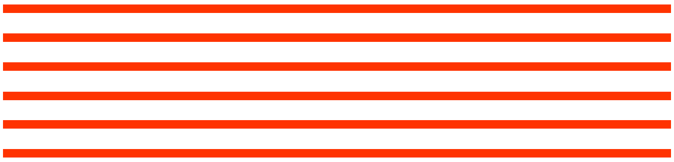
- BLAST (Basic Local Alignment Search Tool)
  - The most popular alignment tool
  - Try it! Pick any protein, e.g. hemoglobin, insulin, exportin,… BLAST to find distant relatives.
  - http://www.ncbi.nlm.nih.gov/blast/

Sequence Alignment: Why

- The most basic sequence analysis task
  - First aligning the sequences (or parts of them) and
  - Then deciding whether that alignment is more likely to have occurred because the sequences are related, or just by chance
- Similar sequences often have similar origin or function
- New sequence always compared to existing sequences (e.g. using BLAST)
- Alignment algorithm is also used when sequencing DNA (reading DNA sequence)

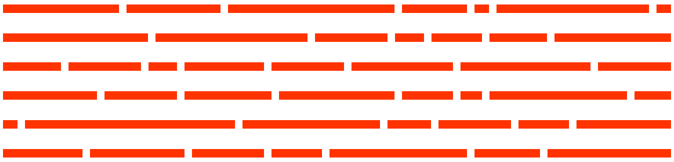
How Do We Read DNA?

- We replicate it

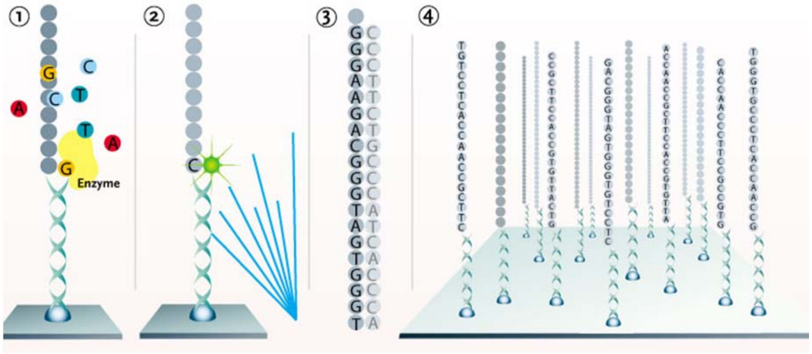


How Do We Read DNA?

- We replicate it
- We shred it



Reading Short DNA



- Use replication machinery with colored bases
- Take pictures of massively parallel reaction
- 10 million reads of 30 per day & $1000

Example

The genome is:

```
TTATGGTCGGTGAGTGTGACTGGTGTTGTCTAA
```

The reads are:

```
GGTCGGTGAG
TGAGTGTGAC
TGGTGTTGTC
TGACTGGTTT
AATGGTCGGT
GAGTGTGACT
AAAAAAAAAA
```

Example

The genome is:

```
TTATGGTCGGTGAGTGTGACTGGTGTTGTCTAA
    ||||||||||
    GGTCGGTGAG
```

The reads are:

```
GGTCGGTGAG
TGAGTGTGAC
TGGTGTTGTC
TGACTGGTTT
AATGGTCGGT
GAGTGTGACT
AAAAAAAAAA
```

Example

The genome is:

```
TTATGGTCGGTGAGTGTGACTGGTGTTGTCTAA
          ||||||||||
          TGAGTGTGAC
```

The reads are:

```
GGTCGGTGAG
TGAGTGTGAC
TGGTGTTGTC
TGACTGGTTT
AATGGTCGGT
GAGTGTGACT
AAAAAAAAAA
```

Example

The genome is:

```
TTATGGTCGGTGAGTGTGACTGGTGTTGTCTAA
                    ||||||||||
                    TGGTGTTGTC
```

The reads are:

```
GGTCGGTGAG
TGAGTGTGAC
TGGTGTTGTC
TGACTGGTTT
AATGGTCGGT
GAGTGTGACT
AAAAAAAAAA
```

Example

The genome is:

```
TTATGGTCGGTGAGTGTGACTGGTGTTGTCTAA
                |||||||| |
                TGACTGGTTT
```

The reads are:

GGTCGGTGAG
TGAGTGTGAC
TGGTGTTGTC
TGACTGGTTT
AATGGTCGGT
GAGTGTGACT
AAAAAAAAAA

Example

The genome is:

```
TTATGGTCGGTGAGTGTGACTGGTGTTGTCTAA
  |||||||||
 AATGGTCGGT
```

The reads are:

GGTCGGTGAG
TGAGTGTGAC
TGGTGTTGTC
TGACTGGTTT
AATGGTCGGT
GAGTGTGACT
AAAAAAAAAA

Example

The genome is:

```
TTATGGTCGGTGAGTGTGACTGGTGTTGTCTAA
    GGTCGGTGAG
          TGAGTGTGAC
                    TGGTGTTGTC
                TGACTGGTTT
 AATGGTCGGT
           GAGTGTGACT
```

Assembled genome sequence is:

```
 AATGGTCGGTGAGTGTGACTGGTTTTGTC
```

Computational Problem

- **Goal:** Align reads to genome
- **Input:**
  Many reads: l-long strings $S_1,\ldots,S_m$
  Approximate reference genome: string R
- **Output:**
  $x_1,\ldots,x_m$ along R where reads match, resp.
- **Complications:**
  - Errors
  - Differences

Sequence Alignment

- What
- Why (applications)
  - Comparative genomics
  - DNA sequencing
- A simple algorithm
- Complexity analysis
- A better algorithm:
  - “Dynamic programming”

Sequence Alignment: Key Issues

- What sorts of alignment should be considered
- The scoring system used to rank alignments
- The algorithm used to find optimal (or good) scoring alignments
- The statistical methods used to evaluate the significance of an alignment score

Terminology (CS, not necessarily Bio)

- *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

Definition: An *alignment* of strings S, T is a pair of strings S', T' (with spaces) s.t.

(1) |S'| = |T'|, and (|S| = "length of S")

(2) removing all spaces leaves S, T

Alignment Scoring

Mismatch = -1
Match = 2

S	a c b c d b	**S'**	a	c	-	-	b	c	d	b
T	c a d b d	**T'**	-	c	a	d	b	-	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 all subseqs A of S, B of T s.t. |A| = |B| do
    align A[i] with B[i], 1 ≤ i ≤ |A|
    align all other chars to spaces
    compute its value
    retain the max
end
output the retained alignment
```

Example

S = abcd → A = cd
T = wxyz → B = xz

```
-abc-d  a-bc-d
w--xyz  -w-xyz
```

Outline: Sequence Alignment

- What
- Why (applications)
  - Comparative genomics
  - DNA sequencing
- A simple algorithm
- Complexity analysis
- A better algorithm:
  - "Dynamic programming"

Complexity Analysis

n=4
k=1

abcd---
w---xyz

- Assume |S| = |T| = n
- Cost of evaluating one alignment: ≥ 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 *un*picked chars of T

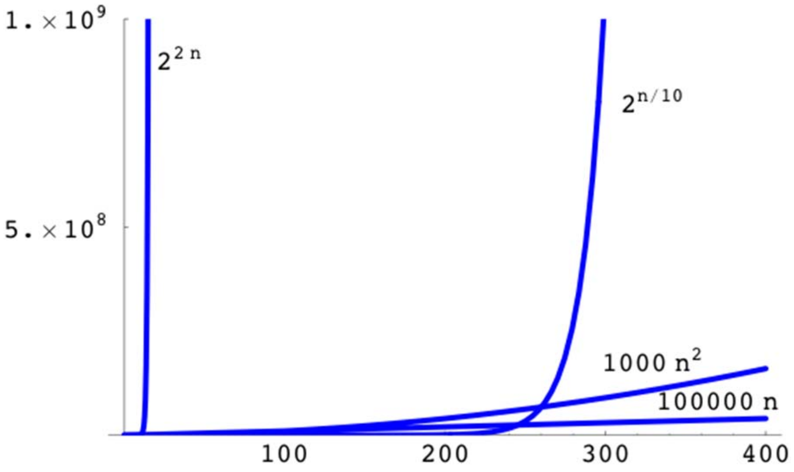
k

S = abcd
T = wxyz

$\underbrace{}_{k}\underbrace{}_{n-k}$

- 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



Outline: Sequence Alignment

- What
- Why (applications)
  - Comparative genomics
  - DNA sequencing
- A simple algorithm
- Complexity analysis
- A better algorithm:
  - “Dynamic programming”

Alignment Scoring

Mismatch = -1
Match = 2

S a c b c d b
T c a d b d

S’ a c – – b c d b
T’ – c a d b – 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 σ(x,y): e.g. σ(a,-)=-1, σ(c,c)=2.
- *Value* of an alignment $\sum_{i=1}^{|S'|} \sigma(S'[i], T'[i])$
- An *optimal alignment:* one of max value
- A simple algorithm: complexity $>2^{2n}$

Needleman-Wunsch Algorithm

- Align by “Dynamic programming”
- Key idea: Build up an optimal alignment using previous solutions for optimal alignments of smaller subsequences.
- Optimal alignment between S & T *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; σ(–, –) < 0)
  - In each case, the *rest* of S & T should be *optimally* aligned to each other

$$\begin{bmatrix} \boxed{\begin{matrix}\sim\sim\sim\sim \\ \sim\sim\sim\sim\end{matrix}} & \begin{matrix} S[n] \\ T[m]\end{matrix}\end{bmatrix}, \begin{bmatrix} \boxed{\begin{matrix}\sim\sim\sim\sim \\ \sim\sim\sim\sim\end{matrix}} & \begin{matrix} S[n] \\ -\end{matrix}\end{bmatrix}, \text{or} \begin{bmatrix} \boxed{\begin{matrix}\sim\sim\sim\sim \\ \sim\sim\sim\sim\end{matrix}} & \begin{matrix} - \\ T[m]\end{matrix}\end{bmatrix}$$

Opt align of $S_1 \ldots S_{n-1}$ & $T_1 \ldots T_{m-1}$

Opt align of $S_1 \ldots S_{n-1}$ & $T_1 \ldots T_m$

Opt align of $S_1 \ldots S_n$ & $T_1 \ldots T_{m-1}$

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

General Case

Optimal align of S[1], ..., S[i] vs T[1], ..., T[j]:

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

Opt align of $S_1 \ldots S_{i-1}$ & $T_1 \ldots T_{j-1}$
Value = V(i-1, j-1)

Opt align of $S_1 \ldots S_{i-1}$ & $T_1 \ldots T_j$
Value = V(i-1, j)

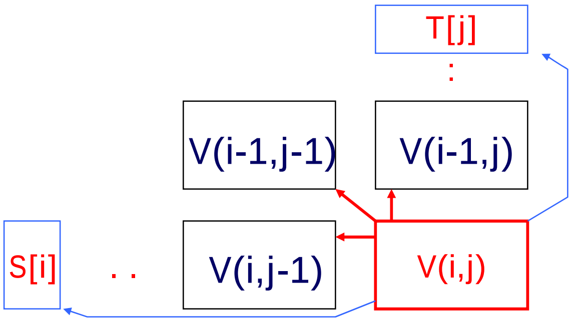
Opt align of $S_1 \ldots S_i$ & $T_1 \ldots T_{j-1}$
Value = V(i, j-1)

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

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

Calculating One Entry

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



T[j]

:

V(i-1,j-1)

V(i-1,j)

S[i]

. .

V(i,j-1)

V(i,j)

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

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

Example

Mismatch = -1
Match = 2

- V(i,j) = value of optimal alignment of S[1], ..., S[i] with T[1], ..., T[j]

	j	0	1	2	3	4	5
i			**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

c
-
Score(c,-) = -1

Example

Mismatch = -1
Match = 2

	j	0	1	2	3	4	5
i			**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

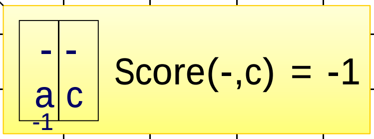
↑
S

-
a
Score(-,a) = -1

Example

Mismatch = -1
Match = 2

	j	0	1	2	3	4	5	
i			**c**	**a**	**d**	**b**	**d**	**←T**
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						
	↑ **S**							



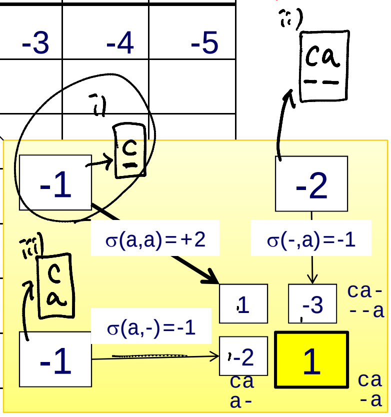
Score(-,c) = -1

Example

Mismatch = -1
Match = 2

T : ca
S : a

	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				
2	**c**	-2						
3	**b**	-3						
4	**c**	-4						
5	**d**	-5						
6	**b**	-6						
	↑ **S**							



Example

Mismatch = -1
Match = 2

	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				
2	**c**	-2	1					
3	**b**	-3						
4	**c**	-4						
5	**d**	-5						
6	**b**	-6						
	↑ S							

Example

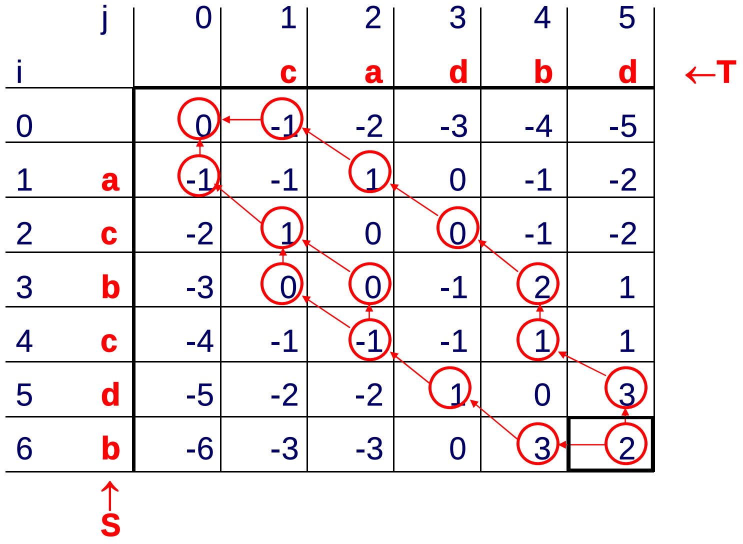
Mismatch = -1
Match = 2

	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							

Time = O(mn)

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**	**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?*

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

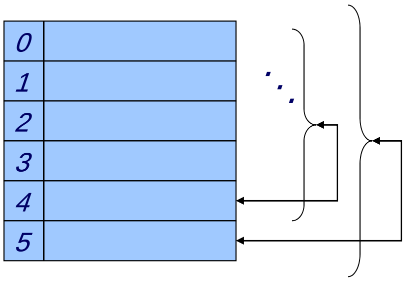
Overall Alignment Significance, II
Empirical (via randomization)

- Generate N random sequences (say N = 10^3 - 10^6)
- Align x to each & score
- If k of them have better score than alignment of x to y, then the (empirical) probability of a chance alignment as good as observed x:y alignment is (k+1)/(N+1)
  - e.g., if 0 of 99 are better, you can say “estimated p < .01”

- How to generate “random” sequences?
  - Scores are often sensitive to sequence composition
  - So uniform 1/20 or 1/4 is a bad idea
  - Even background p_i can be dangerous
  - Better idea: *permute* y N times

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: Why? A specific element will be last with prob 1/n; given that, a specific other element will be next-to-last with prob 1/(n-1), ...; overall: 1/(n!)