

Large-Scale Global Alignments

Multiple Alignments

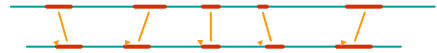


Lecture 10, Thursday May 1, 2003

Rapid global alignment

Genomic regions of interest contain ordered islands of similarity

- E.g. genes
- 1. Find local alignments
- 2. Chain an optimal subset of them

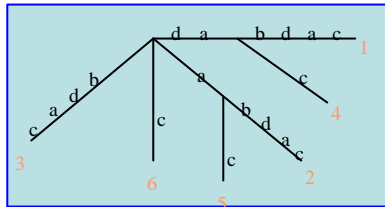


Lecture 10, Thursday May 1, 2003

Suffix Trees

- Suffix trees are a method to find all maximal matches between two strings (and much more)

Example:
x = dabdac



Lecture 10, Thursday May 1, 2003

Application: Find all Matches Between x and y

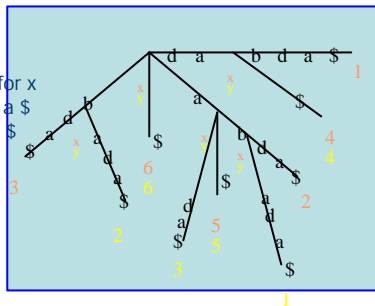
1. Build suffix tree for x, mark nodes with x
 2. Insert y in suffix tree, mark all nodes y "passes from" with y
- The path label of every node marked both 0 and 1, is a common substring

Lecture 10, Thursday May 1, 2003

Example of Suffix Tree Construction for x, y

x = d a b d a \$
y = a b a d a \$

1. Construct tree for x
2. Insert a b a d a \$
3. Insert b a d a \$
4. Insert a d a \$
5. Insert d a \$
6. Insert a \$
6. Insert \$



Lecture 10, Thursday May 1, 2003

Application: Online Search of Strings on a Database

Say a database $D = \{s_1, s_2, \dots, s_n\}$
(eg. proteins)

Question: given new string x, find all matches of x to database

1. Build suffix tree for $\{s_1, \dots, s_n\}$
2. All new queries x take $O(|x|)$ time
(somewhat like BLAST)

Lecture 10, Thursday May 1, 2003

Application: Common Substrings of k Strings

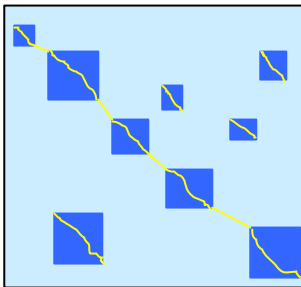
- Say we want to find the longest common substring of s_1, s_2, \dots, s_n
- 1. Build suffix tree for s_1, \dots, s_n
- 2. All nodes labeled $\{s_{i1}, \dots, s_{ik}\}$ represent a match between s_{i1}, \dots, s_{ik}

Lecture 10, Thursday May 1, 2003

Methods to CHAIN Local Alignments

Sparse Dynamic Programming
 $O(N \log N)$

The Problem: Find a Chain of Local Alignments



$(x, y) \rightarrow (x', y')$

requires

$x < x'$
 $y < y'$

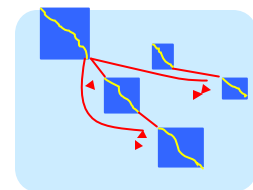
Each local alignment has a weight

FIND the chain with highest total weight

Lecture 10, Thursday May 1, 2003

Quadratic Time Solution

- Build Directed Acyclic Graph (DAG):
 - Nodes: local alignments $(x_a, x_b) \rightarrow (y_a, y_b)$ Directed Edges: any two local alignments that can be chained
- Each local alignment is a node v_i with alignment score s_i



Lecture 10, Thursday May 1, 2003

Quadratic Time Solution (cont'd)

Dynamic programming:

Initialization:

Find each node v_a s.t. there is no edge (u, v_a)
Set score of $V(a)$ to be s_a

Iteration:

For each v_i , optimal path ending in v_i has total score:

$$V(i) = \max (\text{weight}(v_j, v_i) + V(j))$$

Termination:

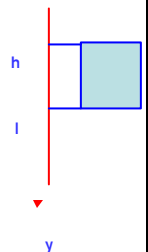
Optimal global chain:
 $j = \text{argmax} (V(j))$; trace chain from v_j

Worst case time: quadratic

Lecture 10, Thursday May 1, 2003

Faster Solution: Sparse DP

- $1, \dots, N$: rectangles
- (h_j, l_j) : y-coordinates of rectangle j
- $w(j)$: weight of rectangle j
- $V(j)$: optimal score of chain ending in j
- L : list of triplets $(l_j, V(j), j)$
 L is sorted by l_j

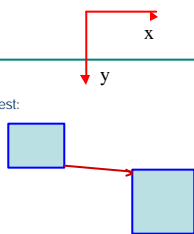


Lecture 10, Thursday May 1, 2003

Sparse DP

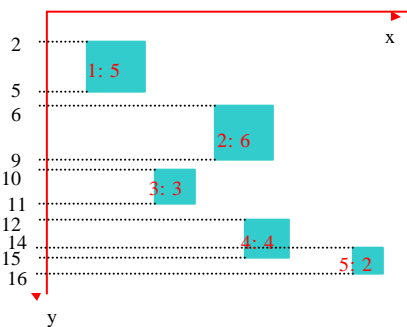
Go through rectangle x coordinates, from lowest to highest:

1. When on the leftmost end of i:
 - a. j: rectangle in L, with largest $l_j < h_i$
 - b. $V(i) = w(i) + V(j)$
2. When on the rightmost end of i:
 - a. j: rectangle in L, with largest $l_j \leq l_i$
 - b. If $V(i) > V(j)$:
 - i. INSERT $(l_i, V(i), i)$ in L
 - ii. REMOVE all $(k, V(k), k)$ with $V(k) \leq V(i)$ & $k \geq l_i$



Lecture 10, Thursday May 1, 2003

Example



Lecture 10, Thursday May 1, 2003

Time Analysis

1. Sorting the x-coords takes $O(N \log N)$
2. Going through x-coords: N steps
3. Each of N steps requires $O(\log N)$ time:
 - Searching L takes $\log N$
 - Inserting to L takes $\log N$
 - All deletions are consecutive, so $\log N$

Lecture 10, Thursday May 1, 2003

Putting it All Together: Fast Global Alignment Algorithms

1. FIND local alignments
2. CHAIN local alignments

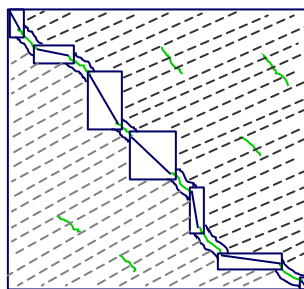
	FIND	CHAIN
GLASS:	k-mers	hierarchical DP
MumMer:	Suffix Tree	Sparse DP
Avid:	Suffix Tree	hierarchical DP
LAGAN	CHAOS	Sparse DP

Lecture 10, Thursday May 1, 2003

LAGAN: Pairwise Alignment

1. FIND local alignments
2. CHAIN local alignments
3. DP restricted around chain

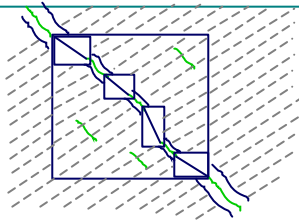
LAGAN



1. Find local alignments
2. Chain - $O(N \log N)$ L.I.S.
3. Restricted DP

Lecture 10, Thursday May 1, 2003

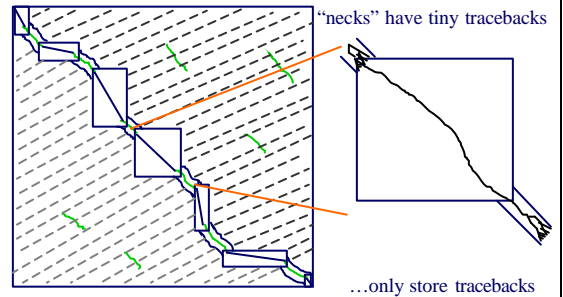
LAGAN: recursive call



- What if a box is too large?
 - Recursive application of LAGAN, more sensitive word search

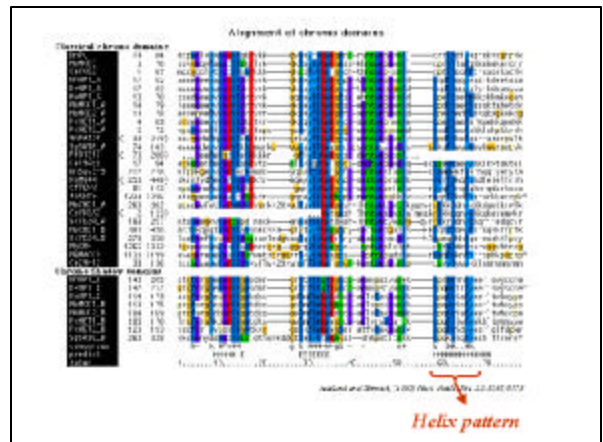
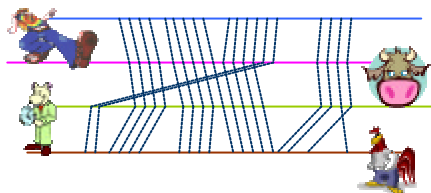
Lecture 10, Thursday May 1, 2003

3. LAGAN: memory



Lecture 10, Thursday May 1, 2003

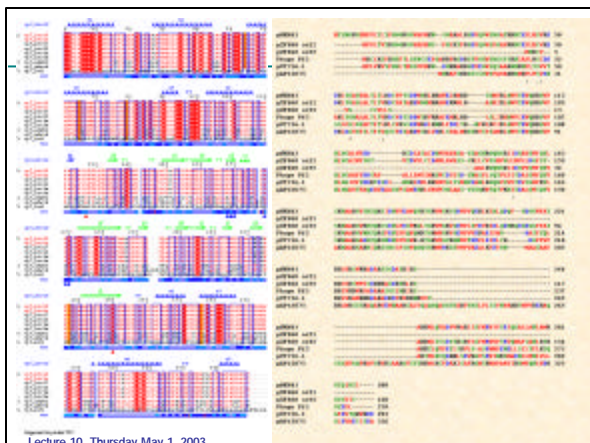
Multiple Sequence Alignments



Overview

- Definition
- Scoring Schemes
- Algorithms

Lecture 10, Thursday May 1, 2003



Lecture 10, Thursday May 1, 2003

Multiple Sequence Alignments

Definition and Motivation

Definition

- Given N sequences x^1, x^2, \dots, x^N :
 - Insert gaps (–) in each sequence x^i , such that
 - All sequences have the same length L
 - Score of the global map is maximum
- Pairwise alignment: a hypothesis on the evolutionary relationship between the letters of two sequences
- Same for a multiple alignment!

Lecture 10, Thursday May 1, 2003

Motivation

- A faint similarity between two sequences becomes very significant if present in many
- Protein domains
- Motifs responsible for gene regulation

Lecture 10, Thursday May 1, 2003

Another way to view multiple alignments

- Pairwise alignment:
Good alignment \triangleright Evolutionary relationship
 - Multiple alignment:
Something in common \triangleright Sequence in common
- *inverse* to pairwise alignment

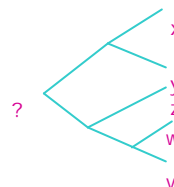
Lecture 10, Thursday May 1, 2003

Multiple Sequence Alignments

Scoring Function

Scoring Function

- Ideally:
 - Find alignment that maximizes probability that sequences evolved from common ancestor



Lecture 10, Thursday May 1, 2003

Scoring Function (cont'd)

- Unfortunately: too many parameters
- Compromises:
 - Ignore phylogenetic tree
 - Statistically independent columns:

$$S(m) = G(m) + \sum_i S(m_i)$$

m: alignment matrix
G: function penalizing gaps

Lecture 10, Thursday May 1, 2003

Scoring Function: Sum Of Pairs

Definition: Induced pairwise alignment

A pairwise alignment induced by the multiple alignment

Example:

x: AC-GCGG-C
y: AC-GC-GAG
z: GCGGC-GAG

Induces:

x: ACGCGG-C; x: AC-GCGG-C; y: AC-GCGAG
y: ACGC-GAC; z: GCGGC-GAG; z: GCGGCGAG

Lecture 10, Thursday May 1, 2003

Sum Of Pairs (cont'd)

- The *sum-of-pairs* score of an alignment is the sum of the scores of all induced pairwise alignments

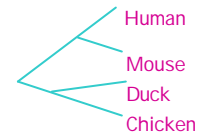
$$S(m) = \sum_{k < l} s(m^k, m^l)$$

$s(m^k, m^l)$: score of induced alignment (k,l)

Lecture 10, Thursday May 1, 2003

Sum Of Pairs (cont'd)

- Heuristic way to incorporate evolution tree:



- Weighted SOP:

$$S(m) = \sum_{k < l} w_{kl} s(m^k, m^l)$$

w_{kl} : weight decreasing with distance

Lecture 10, Thursday May 1, 2003

Consensus

-AGGCTATCACCTGACCTCCAGGCCGA--TGCCC---
TAG-CTATCAC--GACCGC--GGTCGATTGCCCCGAC
CAG-CTATCAC--GACCGC----TCGATTGCTCGAC

CAG-CTATCAC--GACCGC--GGTCGATTGCCCCGAC

- Find optimal consensus string m' to maximize

$$S(m) = \sum_i s(m', m_i)$$

$s(m', m)$: score of pairwise alignment (k,l)

Lecture 10, Thursday May 1, 2003

Multiple Sequence Alignments

Algorithms

Multidimensional Dynamic Programming

Generalization of Needleman-Wunsh:

$$S(m) = \sum_i S(m_i)$$

(sum of column scores)

$$F(i_1, i_2, \dots, i_N) = \max_{\text{(all neighbors of cube)}} (F(\text{nbr}) + S(\text{nbr}))$$

Lecture 10, Thursday May 1, 2003

Multidimensional Dynamic Programming

- Example: in 3D (three sequences):



- 7 neighbors/cell

$$F(i, j, k) = \max \{ \begin{aligned} &F(i-1, j-1, k-1) + S(x_i, x_j, x_k), \\ &F(i-1, j-1, k) + S(x_i, x_j, -), \\ &F(i-1, j, k-1) + S(x_i, -, x_k), \\ &F(i-1, j, k) + S(x_i, -, -), \\ &F(i, j-1, k-1) + S(-, x_j, x_k), \\ &F(i, j-1, k) + S(-, x_j, -), \\ &F(i, j, k-1) + S(-, -, x_k) \end{aligned} \}$$

Lecture 10, Thursday May 1, 2003

Multidimensional Dynamic Programming

Running Time:

1. Size of matrix: L^N ;

Where L = length of each sequence
N = number of sequences

2. Neighbors/cell: $2^N - 1$

Therefore..... $O(2^N L^N)$ 😞

Lecture 10, Thursday May 1, 2003