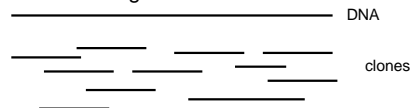


CSE 589
Applied Algorithms
Spring 1999

Contiguous Ordering - PQ Trees
Course Summary

DNA Sequence Reconstruction

- DNA can only be sequenced in relatively small pieces, up to about 1,000 nucleotides.
- By chemistry a much longer DNA sequence can be broken up into overlapping sequences called clones. Clones are 10's of thousands of nucleotides long.

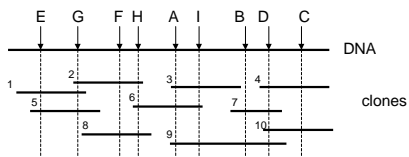


CSE 589 - Lecture 19 - Spring 1999

2

Tagging the Clones

- By chemistry the clones can be tagged by identifying a region of the DNA uniquely.



- Each clone is then tagged correspondingly.

CSE 589 - Lecture 19 - Spring 1999

3

Problem to Solve

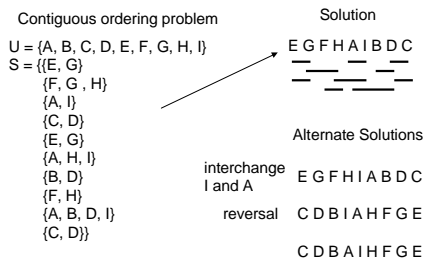
- Given a set of tagged clones, find a consistent ordering of the tags that determines a possible ordering of the DNA molecule.

clone	tag									
1.	{E, G}	<table border="0"> <tr> <td>output</td> <td>E G F H A I B D C</td> </tr> <tr> <td>1</td> <td><u>1</u> <u>2</u> <u>3</u> <u>4</u></td> </tr> <tr> <td>2</td> <td><u>2</u> <u>6</u> <u>7</u> <u>10</u></td> </tr> <tr> <td>3</td> <td><u>5</u> <u>8</u> <u>9</u> <u>10</u></td> </tr> </table>	output	E G F H A I B D C	1	<u>1</u> <u>2</u> <u>3</u> <u>4</u>	2	<u>2</u> <u>6</u> <u>7</u> <u>10</u>	3	<u>5</u> <u>8</u> <u>9</u> <u>10</u>
output	E G F H A I B D C									
1	<u>1</u> <u>2</u> <u>3</u> <u>4</u>									
2	<u>2</u> <u>6</u> <u>7</u> <u>10</u>									
3	<u>5</u> <u>8</u> <u>9</u> <u>10</u>									
input	2. {F, G, H}									
	3. {A, I}									
	4. {C, D}									
	5. {E, G}									
	6. {A, H, I}									
	7. {B, D}									
	8. {F, H}									
	9. {A, B, D, I}									
	10. {C, D}									

CSE 589 - Lecture 19 - Spring 1999

4

Contiguous Ordering Solutions



CSE 589 - Lecture 19 - Spring 1999

5

Linear Time Algorithm

- Booth and Lueker, 1976, designed an algorithm that runs in time $O(n+m+s)$.
 - n is the size of the universe, m is the number of sets, and s is the sum of the sizes of the sets.
- It requires a novel data structure called the PQ tree that represents a set of orderings.
- PQ trees can also be used to test whether an undirected graph is planar.

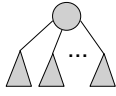
CSE 589 - Lecture 19 - Spring 1999

6

PQ Trees

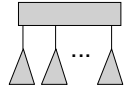
- PQ trees are built from three types of nodes

P node




Children can be reordered.

Q node



Children can be reversed.

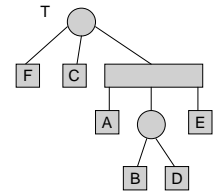
leaf



Each leaf has a unique label.

CSE 589 - Lecture 19 - Spring 1999 7

Example PQ-Tree



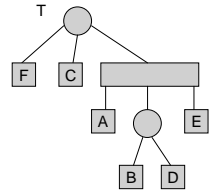
The frontier of T defines the ordering $F(T) = FCABDE$, just read the leaves left to right.

T' is equivalent to T if T can be transformed into T' by reordering the children of P nodes and reversing the children of Q nodes.

CSE 589 - Lecture 19 - Spring 1999 8

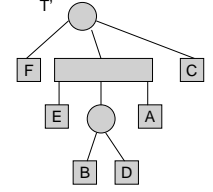
Equivalent PQ Trees

T



FCABDE

T'

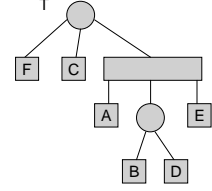


FEBDAC

CSE 589 - Lecture 19 - Spring 1999 9

Orderings Defined by a PQ Tree

- Given a PQ tree T the orderings defined by T is
 - $PQ(T) = \{F(T') : T' \text{ is equivalent to } T\}$



There are $6 \times 2 \times 2 = 24$ distinct orderings in $PQ(T)$.

Generally, if a PQ tree T has q Q nodes and p P nodes with number of children c_1, c_2, \dots, c_p , then the number of orderings in $PQ(T)$ is $2^q c_1! c_2! \dots c_p!$.

$n! = 1 \times 2 \times \dots \times n$

CSE 589 - Lecture 19 - Spring 1999 10

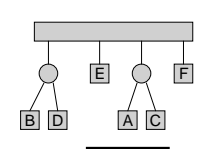
PQ Tree Solution for the Contiguous Ordering Problem

- Input: A universe U and a set $S = \{S_1, S_2, \dots, S_m\}$ of subsets of U .
- Output: A PQ tree T with leaves U with the property that $PQ(T)$ is the set of all orderings of U where each set in S is contiguous in the ordering.

CSE 589 - Lecture 19 - Spring 1999 11

Example Solution

$U = \{A, B, C, D, E, F\}$
 $S = \{\{A, C, E\}, \{A, C, F\}, \{B, D, E\}\}$

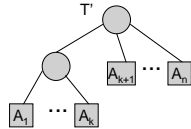


There are 8 orderings that are possible in keeping each of these sets contiguous.

CSE 589 - Lecture 19 - Spring 1999 12

PQ Tree Restriction

- Let $U = \{A_1, A_2, \dots, A_n\}$, $S = \{A_1, A_2, \dots, A_k\}$, and T a PQ tree.
- We will define a function **Restrict** with the following properties:
 - $\text{Restrict}(T, S)$ is a PQ tree.
 - $\text{PQ}(\text{Restrict}(T, S)) = \text{PQ}(T) \cap \text{PQ}(S)$ where



CSE 589 - Lecture 19 - Spring 1999

13

High Level PQ tree Algorithm

- Input is $U = \{A_1, A_2, \dots, A_n\}$, and subsets S_1, S_2, \dots, S_m of U .
- Initialization:
 - $T = P$ node with children A_1, A_2, \dots, A_n
- Calculate m restrictions:
 - for $j = 1$ to m do
 - $T := \text{Restrict}(T, S_j)$
- At the end of iteration k :
 - $\text{PQ}(T)$ = the set of ordering of U where each set S_1, S_2, \dots, S_k are contiguous.

CSE 589 - Lecture 19 - Spring 1999

14

Marking Nodes

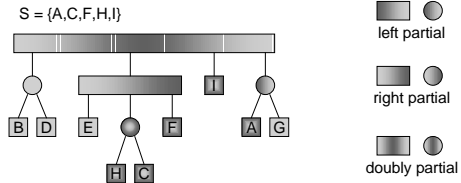
- Given a set S and PQ tree T we can mark nodes either full or partial.
 - A leaf is full if it is a member of S .
 - A node is full if all its children are full.
 - A node is partial if either it has both full and non-full children or it has a partial child.
 - A node is doubly partial if it has two partial children.

CSE 589 - Lecture 19 - Spring 1999

15

Marks of Nodes

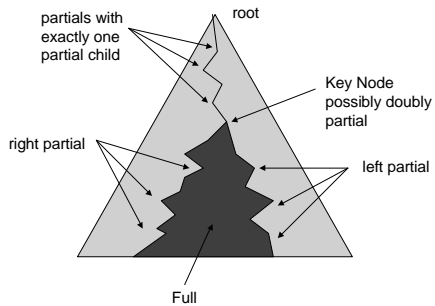
Mark the leaves in S full.
Bottom up mark the nodes full or partial.
The members of S will become contiguous.



CSE 589 - Lecture 19 - Spring 1999

16

Structure of the Marked PQ Tree



CSE 589 - Lecture 19 - Spring 1999

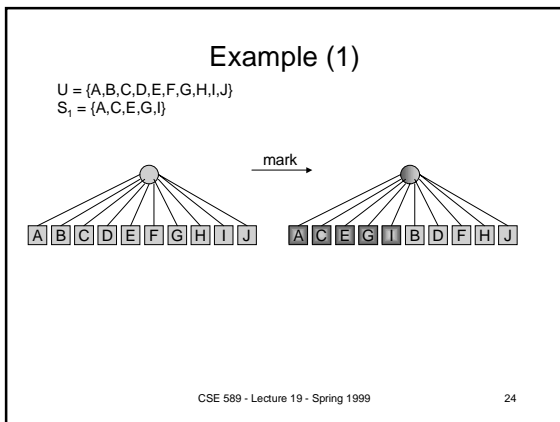
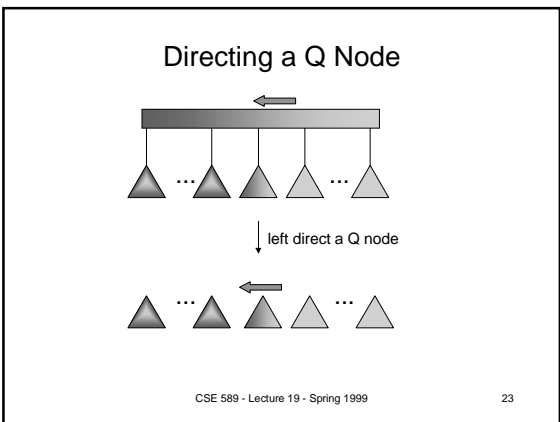
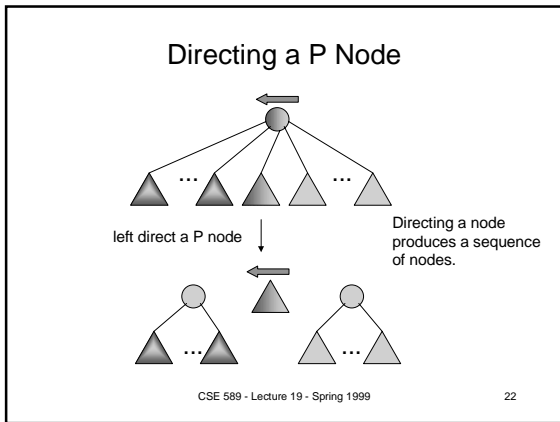
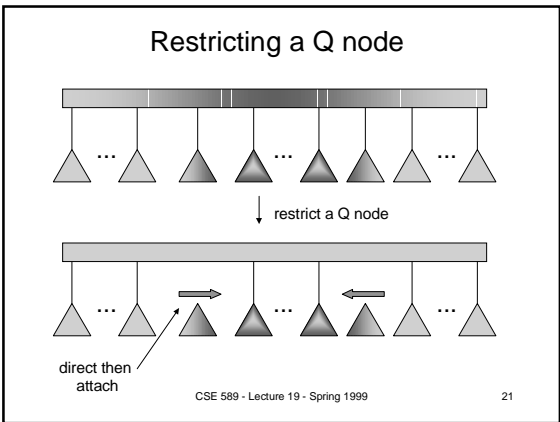
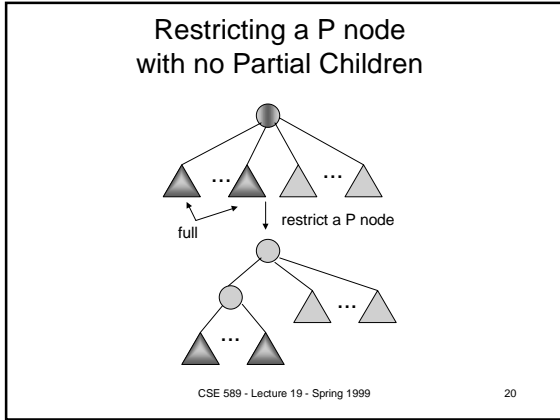
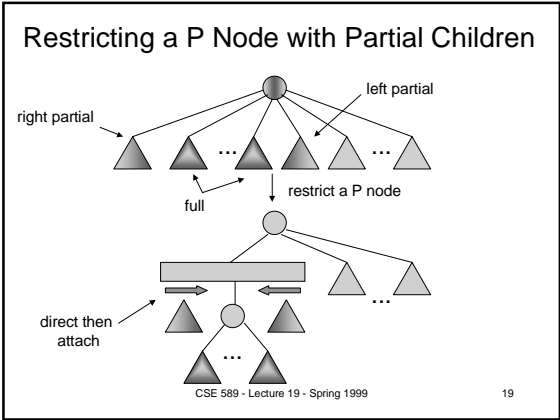
17

Restrict(T, S)

- Mark the full and partial nodes from the bottom up.
 - In the process the marked leaves become contiguous.
- Locate the key node.
 - Deepest node with the property that all its proper ancestors have exactly one partial child.
- Restrict the key node.
 - In the process of restricting the key node we will have to recursively direct partial nodes.
 - Directing a node returns a sequence of nodes.

CSE 589 - Lecture 19 - Spring 1999

18



Example (2)

$U = \{A, B, C, D, E, F, G, H, I, J\}$
 $S_1 = \{A, C, E, G, I\}$

restrict P node

special case because no partial child.

CSE 589 - Lecture 19 - Spring 1999 25

Example (3)

$U = \{A, B, C, D, E, F, G, H, I, J\}$
 $S_2 = \{C, D, F, G, I, J\}$

mark

CSE 589 - Lecture 19 - Spring 1999 26

Example (4)

$U = \{A, B, C, D, E, F, G, H, I, J\}$
 $S_2 = \{C, D, F, G, I, J\}$

restrict P node

CSE 589 - Lecture 19 - Spring 1999 27

Example (5)

$U = \{A, B, C, D, E, F, G, H, I, J\}$
 $S_2 = \{C, D, F, G, I, J\}$

direct P node

CSE 589 - Lecture 19 - Spring 1999 28

Example (6)

$U = \{A, B, C, D, E, F, G, H, I, J\}$
 $S_2 = \{C, D, F, G, I, J\}$

attach

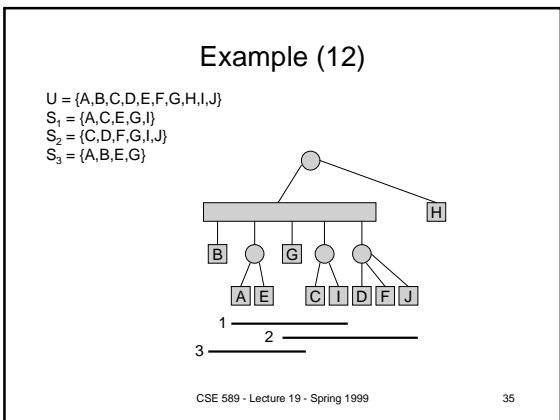
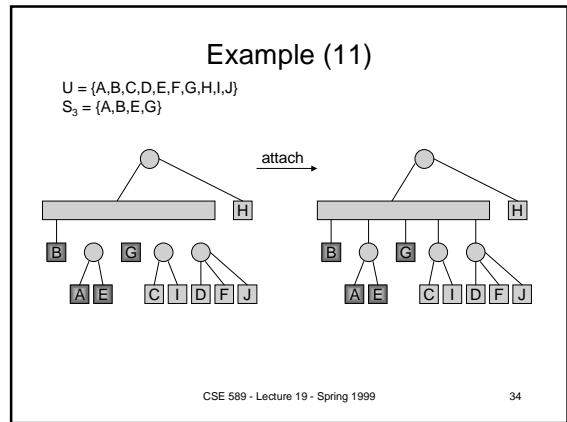
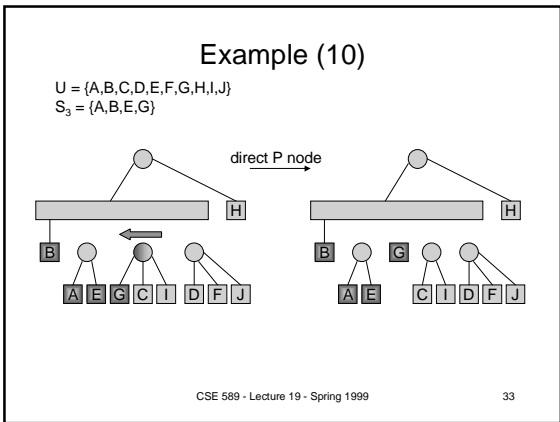
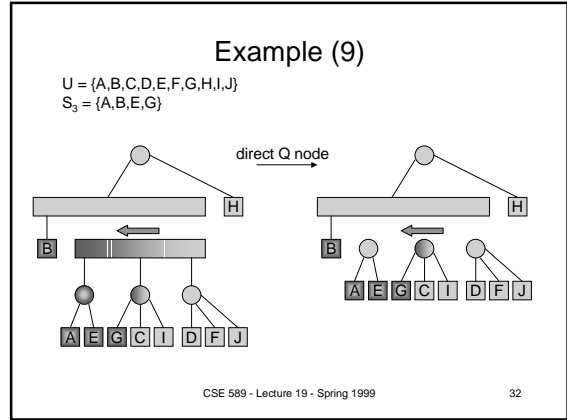
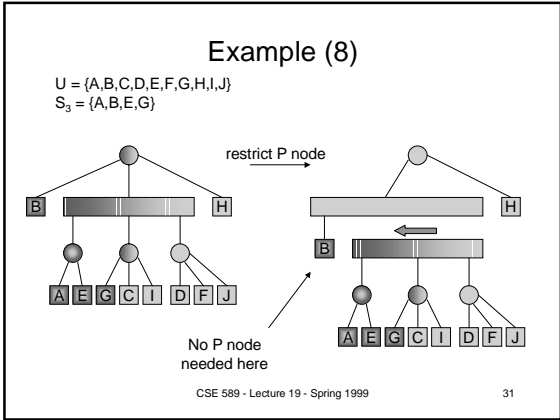
CSE 589 - Lecture 19 - Spring 1999 29

Example (7)

$U = \{A, B, C, D, E, F, G, H, I, J\}$
 $S_3 = \{A, B, E, G\}$

mark

CSE 589 - Lecture 19 - Spring 1999 30



Linear Number of Nodes Processed

- Let n be the size of the universe, m the number of sets, and s the sum of the sizes of the sets.
 - Number of full nodes processed $\leq 2s$.
 - Number of key nodes processed $= m$.
 - Number of partial nodes with partial children processed below the key node $\leq m + n$.
 - Number of partial nodes with no partial children with not partial children $\leq 2m$.
 - Number of partial nodes processed above the key node $\leq m + n$.

CSE 589 - Lecture 19 - Spring 1999 36

Number of Processed Nodes Amortized

partials with exactly one partial child $\leq m+n$

Key Nodes m

partials with partial children $\leq m+n$

Full $\leq 2s$

partials with no partial children $\leq 2m$

n size of universe
 m number of sets
 s sum of size of sets

CSE 589 - Lecture 19 - Spring 1999 37

Partials with Partial Children Below the Key Node

- Amortized complexity argument.
- Consider the quantities:
 - q = number of Q nodes,
 - cp = number of children of P nodes.
 - We examine the quantity $x = q + cp$
 - x is initially n and never negative.
 - Each restrict of a key node increases x by at most 1.
 - Each direct of a partial node with a partial child decreases x by at least 1.
 - Since there are m restricts of a key node then there are most $n + m$ directs of partials with partial children.

CSE 589 - Lecture 19 - Spring 1999 38

Restricting a P Node with Partial Children

restrict a P node

change in $q + cp$ is at most +1.

CSE 589 - Lecture 19 - Spring 1999 39

Restricting a P node with no Partial Children

restrict a P node

change in $q + cp$ is exactly +1.

CSE 589 - Lecture 19 - Spring 1999 40

Restricting a Q node

restrict a Q node

no change in q, cp

direct then attach

CSE 589 - Lecture 19 - Spring 1999 41

Directing a P Node

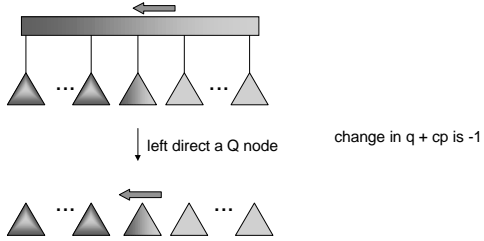
Assume partial child

left direct a P node

change in $q + cp$ is -1

CSE 589 - Lecture 19 - Spring 1999 42

Directing a Q Node



CSE 589 - Lecture 19 - Spring 1999

43

PQ Tree Notes

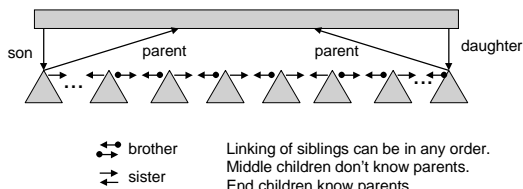
- In algorithmic design only a linear number of nodes are ever processed.
- Designing the data structures to make the linear time processing a reality is very tricky.
- PQ trees solve the idealized DNA ordering problem.
- In reality, because of errors, the DNA ordering problem is NP-hard and other techniques are used.

CSE 589 - Lecture 19 - Spring 1999

44

Example of Data Structure Trick

- Linking the children of a Q node



CSE 589 - Lecture 19 - Spring 1999

45

Applied Algorithms in a Nutshell (1)

- There are genuinely hard problems that require approximate solutions.
 - NP-completeness
 - Branch and Bound for small input size
 - Local search techniques
 - Specialized techniques like GLA.
- Some apparently hard problems are not really so.
 - minimum spanning tree
 - contiguous ordering

CSE 589 - Lecture 19 - Spring 1999

46

Applied Algorithms in a Nutshell (2)

- Cache performance matters.
 - Understanding and controlling cache performance is possible.
- Data compression involves interesting algorithms and theory.
 - Entropy
 - Huffman and arithmetic coding
 - Dictionary coding
 - Sequitur
 - VQ - nearest neighbor search
 - Wavelet compression and SPIHT

CSE 589 - Lecture 19 - Spring 1999

47

Applied Algorithms in a Nutshell (3)

- Computational biology also has interesting algorithms.
 - Approximate matching using dynamic programming.
 - Contiguous ordering using PQ trees.
- Fundamental algorithms should always be available.
 - depth-first search
 - breadth-first search
 - disjoint union/find
 - priority queues (d-heaps)
 - sorting

CSE 589 - Lecture 19 - Spring 1999

48

Applied Algorithms in a Nutshell (4)

- Algorithm evaluation and analysis are critical for understanding correctness and performance.
 - Correctness
 - high level thinking about design.
 - development of good invariants.
 - Analysis of algorithms
 - time and storage analysis.
 - amortized analysis.
 - cache performance analysis.