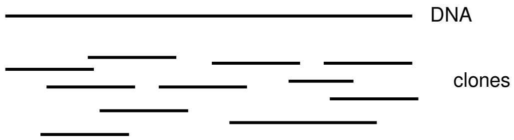
CSEP 521
Applied Algorithms
Autumn 2009

Contiguous Ordering - PQ Trees

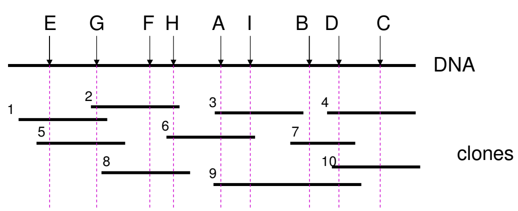
DNA Sequence Reconstruction

- 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.



Tagging the Clones

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



- Each clone is then tagged correspondingly.

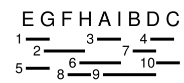
Problem to Solve

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

input

clone	tag
1.	{E, G}
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}

output



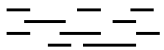
Contiguous Ordering Solutions

Contiguous ordering problem

U = {A, B, C, D, E, F, G, H, I}
S = {{E, G}
{F, G , H}
{A, I}
{C, D}
{E, G}
{A, H, I}
{B, D}
{F, H}
{A, B, D, I}
{C, D}}

Solution

E G F H A I B D C



Alternate Solutions

interchange I and A — E G F H I A B D C

reversal — C D B I A H F G E

C D B A I H F G E

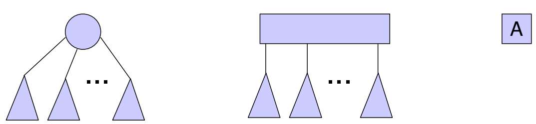
Linear Time Algorithm

- 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.

PQ Trees

- PQ trees are built from three types of nodes

P node Q node leaf

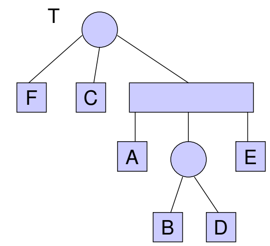


Children can be reordered.

Children can be reversed.

Each leaf has a unique label.

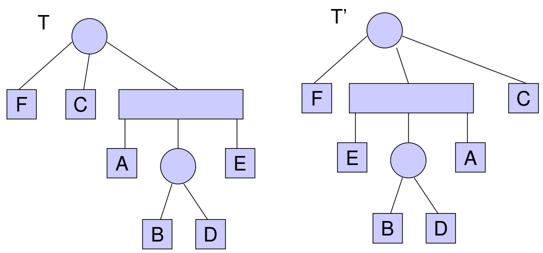
Example PQ-Tree



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.

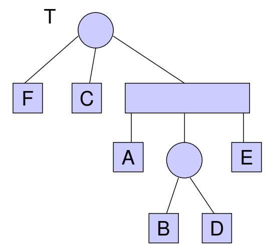
Equivalent PQ Trees



FCABDE FEBDAC

Orderings Defined by a PQ Tree

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



There are 6 x 2 x 2 = 24 distinct orderings in PQ(T).

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

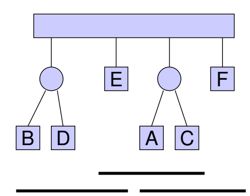
$n! = 1 \times 2 \times \ldots \times n$

PQ Tree Solution for the Contiguous Ordering Problem

- Input: A universe U and a set $S = \{S_1, S_2, \ldots, S_m\}$ 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.

Example Solution

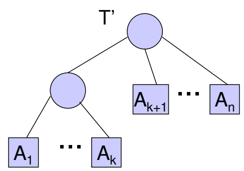
U = {A,B,C,D,E,F}
S = {{A,C,E}, {A,C,F}, {B,D,E}}



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

PQ Tree Restriction

- Let $U = \{A_1,A_2,\ldots,A_n\}$, $S = \{A_1,A_2,\ldots,A_k\}$, and T a PQ tree.
- We will define a function Restrict with the following properties:
  - Restrict(T,S) is a PQ tree.
  - PQ(Restrict(T,S)) = PQ(T) intersect PQ(T') where



High Level PQ tree Algorithm

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

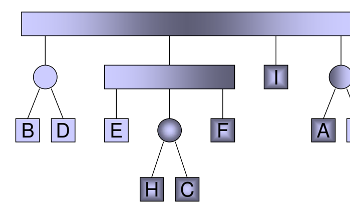
Marking Nodes

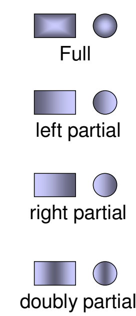
- Given a set S and PQ tree T we can mark nodes either full or partial.
  - A leaf is full if it is a member of S.
  - A node is full if all its children are full.
  - A node is partial if either it has both full and non-full children or it has a partial child.
  - A node is doubly partial if it has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

Marks of Nodes

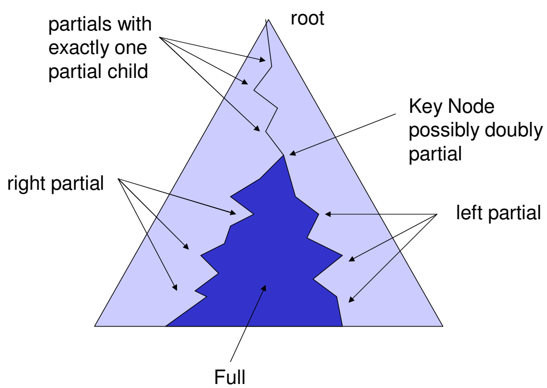
Mark the leaves in S full.
Bottom up mark the nodes full or partial.
The members of S will become contiguous.

S = {A,C,F,H,I}





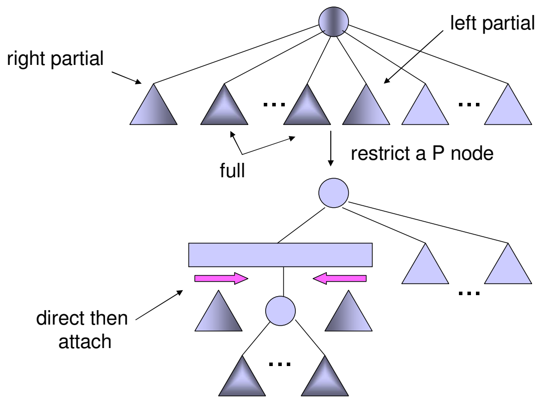
Structure of the Marked PQ Tree



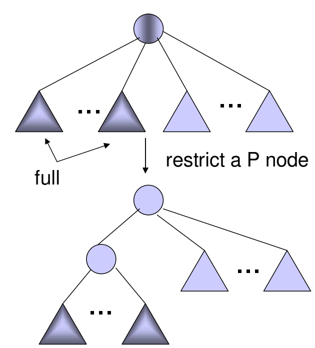
Restrict(T,S)

- 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 the full leaves are descendents of the node.
- 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.

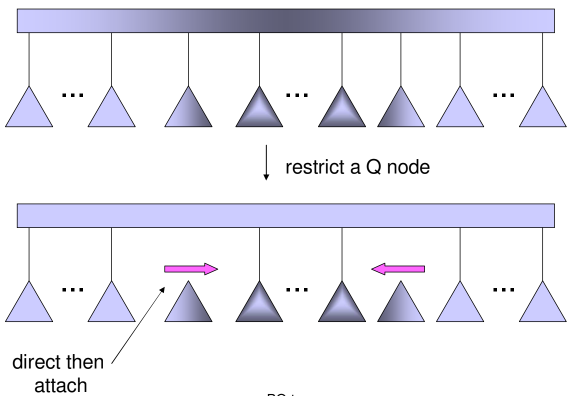
Restricting a P Node with Partial Children



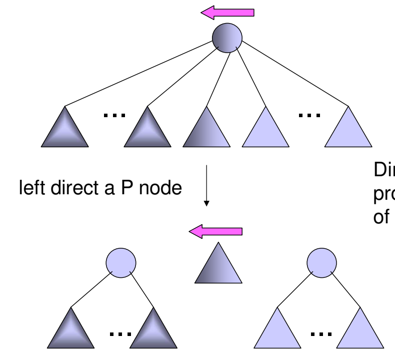
Restricting a P node with no Partial Children



Restricting a Q node

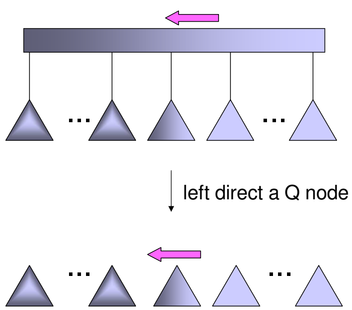


Directing a P Node



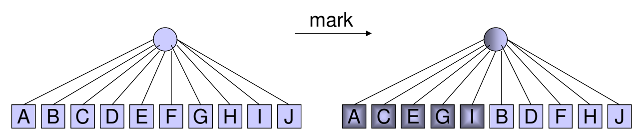
Directing a node produces a sequence of nodes.

Directing a Q Node



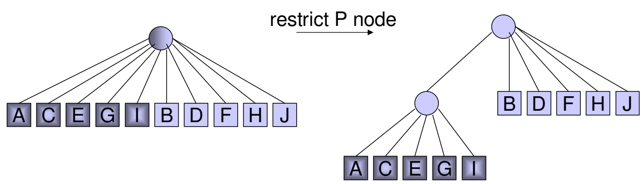
Example (1)

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



Example (2)

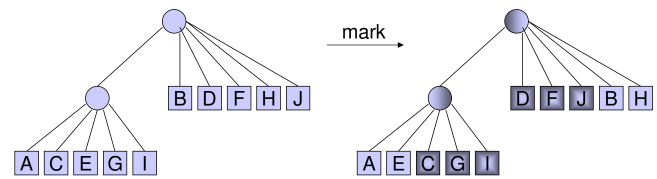
U = {A,B,C,D,E,F,G,H,I,J}
S_1 = {A,C,E,G,I}



special case because
no partial child.

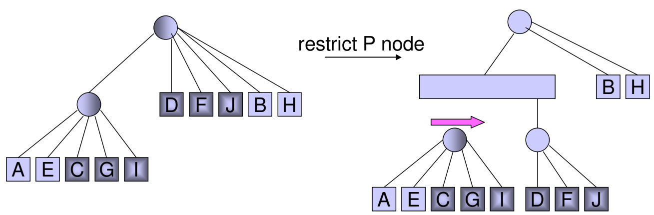
Example (3)

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



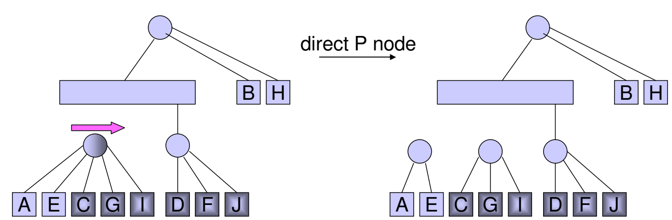
Example (4)

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



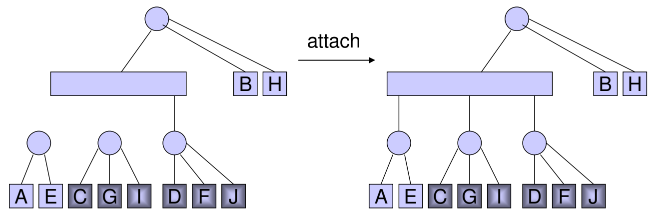
Example (5)

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



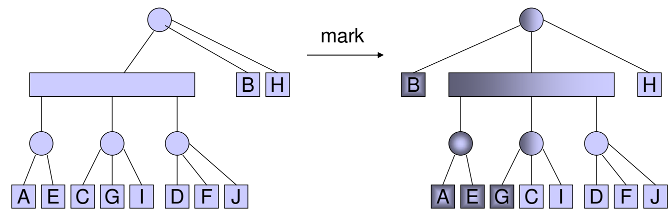
Example (6)

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



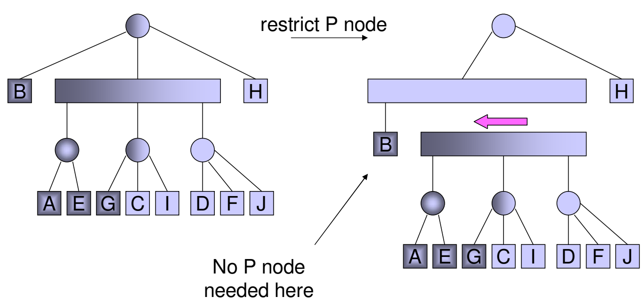
Example (7)

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



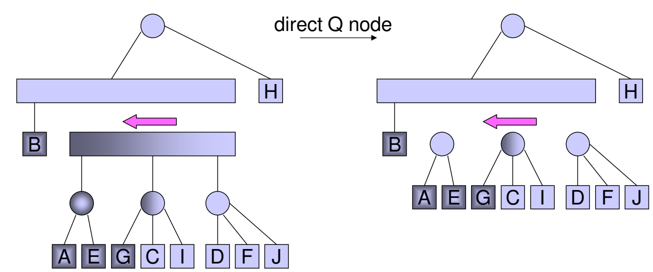
Example (8)

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



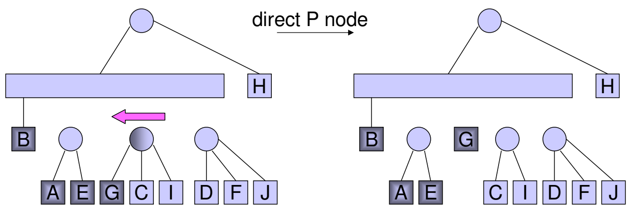
Example (9)

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



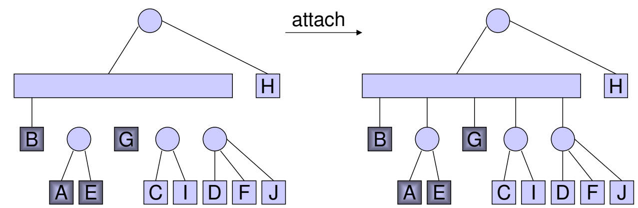
Example (10)

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



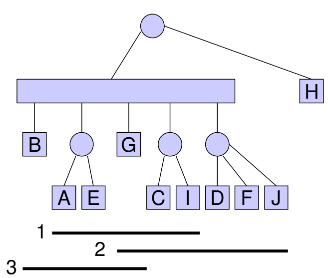
Example (11)

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



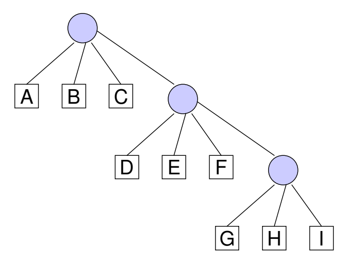
Example (12)

U = {A,B,C,D,E,F,G,H,I,J}
S_1 = {A,C,E,G,I}
S_2 = {C,D,F,G,I,J}
S_3 = {A,B,E,G}



Exercise

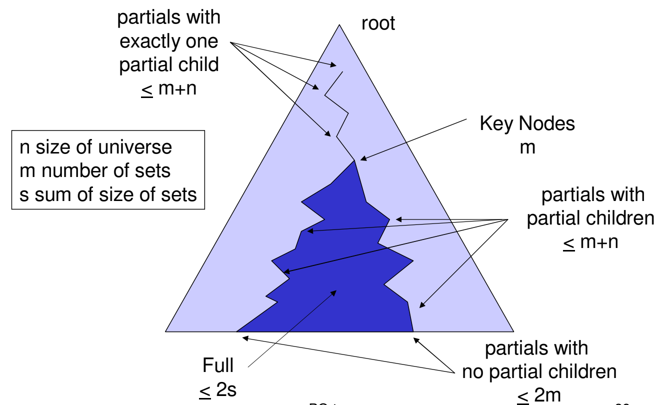
- Restrict with to make {A,B,D,E,G} contiguous



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 $\leq$ 2m.
  – Number of partial nodes processed above the key node $\leq$ m + n.

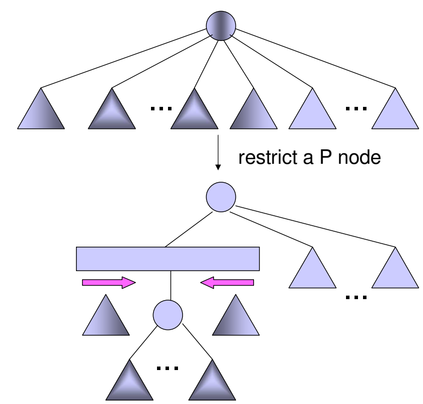
Number of Processed Nodes Amortized



Partials with Partial Children Below the Key Node

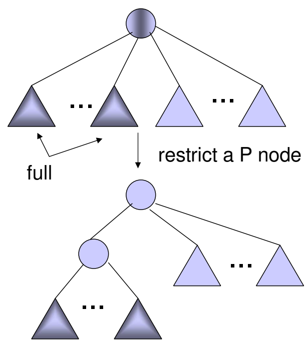
- Amortized complexity argument.
- Consider the quantities:
  – q = number of Q nodes,
  cp = number of children of P nodes.
  – We examine the quantity x = q + cp
  – x is initially n and never negative.
  – Each restrict of a key node increases x by at most 1.
  – Each direct of a partial node with 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.

Restricting a P Node with Partial Children



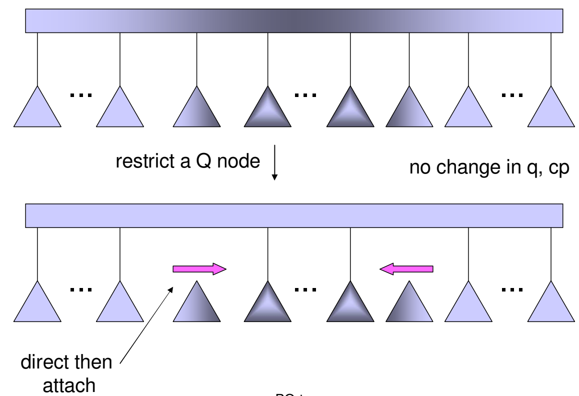
change in q + cp is at most +1.

Restricting a P node with no Partial Children



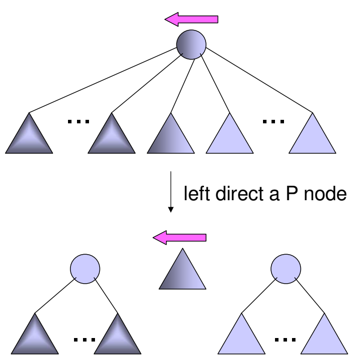
change in q + cp is exactly +1.

Restricting a Q node



no change in q, cp

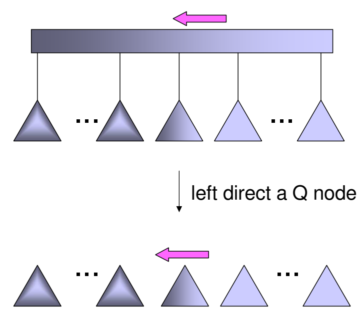
Directing a P Node



Assume partial child

change in q + cp is -1

Directing a Q Node



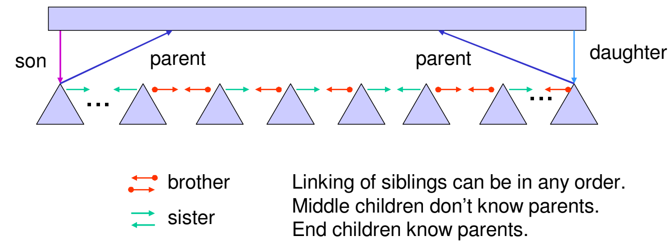
change in q + cp is -1

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.

Example of Data Structure Trick

- Linking the children of a Q node



Linking of siblings can be in any order.
Middle children don't know parents.
End children know parents.