Overall Strategy

Initially: Each vertex is distinct component candidate

Basic Step: Merge component-candidates if they are connected by edge repeatedly

Finally: Left with connected components

End Result: CRCW PRAM (Arbitrary write succeeds) with $2m + n$ processors can find connected components in time $O(\log n)$

Speedup: $\frac{m+n}{\log n}$

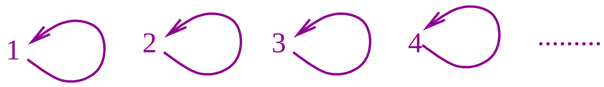
Efficiency: $O\left(\frac{1}{\log n}\right)$

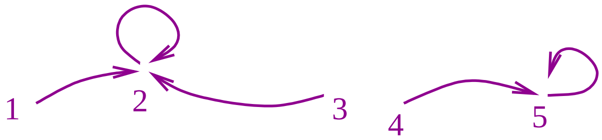
Implementation Ideas

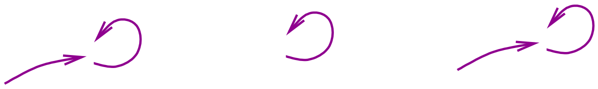
Invariant: $D(i) = i$ For some vertex in each component

Initialization: $D(i) = i$ for all i

Progress: More and more vertices in a component get identical $D(i)$.

Initial Picture: 

Final Picture: 

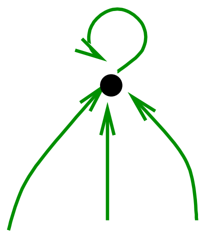
Intermediate: 

Definitions

Rooted Directed Tree: Tree in which

1. Edges are directed from child to parent
2. Root may have a self loop

Rooted Directed Forest: Collection of Rooted directed trees



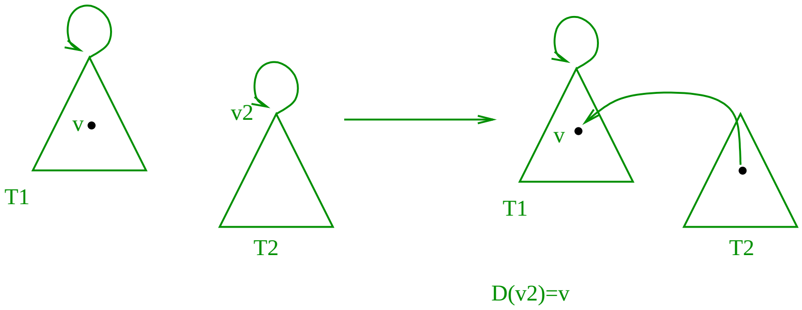
Rooted Star: (Rooted directed tree of height−1)

Algorithm Invariant: D forms a rooted directed forest throughout the execution

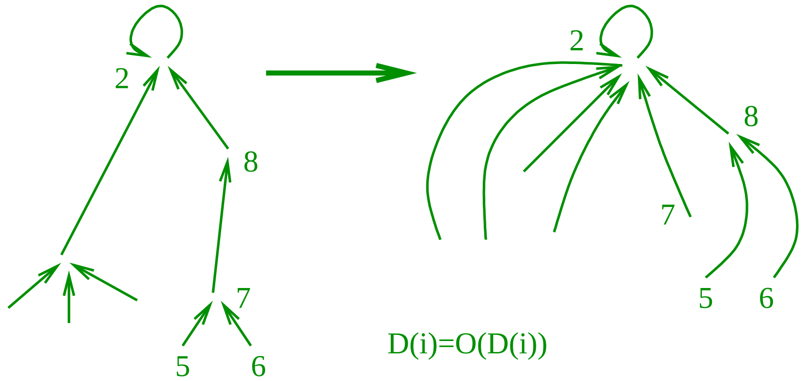
Algorithm Goal: For any component D forms a rooted star

Basic Steps

Grafting (Merge Trees):



Pointer Jumping (Reduce Tree height):



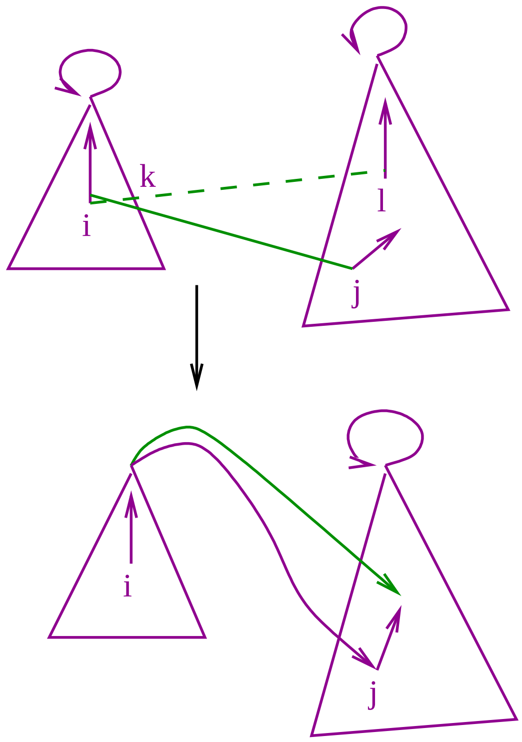
How can we do parallel grafting?

Algorithm (Attempt)

Main Idea: Graft large vertices onto small vertices $\Rightarrow$ Invariant: Root $<$ Child $<$ Grand child $\ldots \Rightarrow$ No cycle during grafting

Algorithm: Do several times:

1. (Ordered Grafting) For all edges (i, j) **if** $(D(i) = D(D(i))$ **and** $D(j) < D(i))$



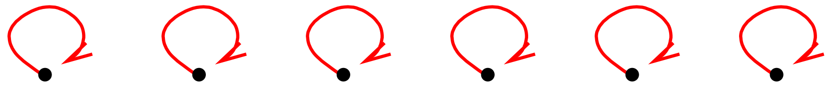
   then $D(D(i)) = D(j)$
2. (Pointer Jumping)**for** all vertices i $D(i) = D(D(i))$

Correct, but slow

Example

Input Graph:

D initially:



Ordered Grafting:
$\forall (i, j)$:
if $(D(i) = D(D(i))$ **and** $D(j) < D(i))$ **then** $D(D(i)) = D(j)$

Star grafting:
$\forall (i, j)$:
if $i \in \text{Star}$ **and** $D(j) \neq D(i)$ **then** $D(D(i)) = D(j)$

Shortcutting:
$\forall i$:
$D(i) = D(D(i))$

Algorithm

1. For all vertices i: $D(i) = i$
2. Do $k \log N$ times:

   Ordered Grafting: For all edges (i, j)
   if $D(i) = D(D(i))$ and $D(j) < D(i)$
   then $D(D(i)) = D(j)$

   Star Grafting: For all edges (i, j):
   if $i \in$ a star inactive in step 1
   and $D(i) \neq D(j)$
   then $D(D(i)) = D(j)$

   Pointer Doubling: For all vertices i:
   $D(i) = D(D(i))$

INVARIANTS: Before every ordered grafting step and on exit from the algorithm:

1. i and $D(i)$ are in same connected component.
2. D forms a pseudo forest.
3. Unless i is a leaf, $D(i) \leq i$

Lemma

D forms a pseudo forest throughout execution.

Proof: By induction over the number of iterations. Assume the Lemma is true at the beginning of an iteration.

In ordered grafting, we only graft a larger numbered root onto a smaller numbered root. Hence cycles cannot form.

Suppose a cycle forms in star grafting. Then it must only involve vertices which were roots of stars inactive during ordered grafting.

Thus there must be at least two stars which were inactive during ordered grafting, and connected by an edge.

But this is impossible, since the larger numbered root would have grafted onto the smaller, or some other in ordered grafting.

Lemma

At the beginning/end of each iteration, unless i is a leaf, $D(i) \leq i$.

Proof: By induction over the number of iterations. Suppose the lemma holds at the beginning of some iteration.

Ordered grafting will graft larger numbered roots onto smaller numbered vertices, hence the Lemma will continue to hold after the step.

In star grafting, some smaller vertex may get grafted onto a larger vertex. This will violate the lemma.

However, this will get cleaned up in the subsequent short cutting step, in which the internal node at which the lemma does not hold will turn into a leaf.

Lemma

If a tree does not change during an entire iteration then it spans a complete component.

Proof:

T does not participate in pointer doubling $\Rightarrow$ Star

Inactive in star grafting $\Rightarrow$

Forall edges (i, j), $i \in T$: $D(i) = D(j)$.

$\Rightarrow$ Complete component.

Time Analysis

$h_i(T)$: Vertex height of tree T after iteration i.

K : some connected component

$h_i(K)$: Sum of heights of trees comprising K after iteration i.

Lemma: $h_i(K) \leq \left(\frac{3}{4}\right)^i |K|$

Proof:

1. Grafting does not increase $h_i(K)$: In general it will decrease, except for the case when a single vertex gets grafted onto another single vertex. Here the combined height will neither decrease nor increase.

2. Pointer Doubling gives:

$$h_{i+1}(T) \leq \left(\frac{3}{4}\right) h_i(T)$$

for every non star tree. In general a path of edge length L will shrink to have edge length $\left\lceil \frac{L}{2} \right\rceil$. Thus a path of vertex length L' will shrink to have vertex length $\left\lceil \frac{L'-1}{2} \right\rceil + 1$. This has minimum value $\frac{3}{4}$ at $L' = 4$.

Corollary

All connected components become stars in $\log_{4/3} N$ steps.

Exercises

1. Show that if the star grafting step is not used, then the algorithm can take time linear in n, on a CRCW PRAM in which writes are resolved arbitrarily. So you may need to say as a part of your counterexample what resolution is needed to make the example work badly.

2. Suppose the star grafting step is not used. How long will the algorithm take on a CRCW PRAM in which the minimum value gets written? What if the value that gets written is chosen uniformly randomly from the values that different processors are attempting to write?