A PARALLEL GRAPH ALGORITHM
FOR FINDING CONNECTED COMPONENTS

by

D. S. Hirschberg

October 1975

Technical Report No. 7510
A PARALLEL GRAPH ALGORITHM
FOR FINDING CONNECTED COMPONENTS

D. S. Hirschberg
Rice University

Abstract
A parallel program is presented that determines the connected components of an undirected graph in time \( O(\log n) \)^2 using \( n^2 \) processors. It is assumed that the processors have access to common memory. Simultaneous access to the same location is permitted for fetch, but not store, instructions.

Key Words and Phrases: graph theory, parallel processing, algorithms, connected components

CR Categories: 5.25, 5.32, 6.22
The problem of determining the connected components of a graph using a parallel computer has recently appeared in the literature [1,2]. The result in [1] is based on finding the transitive closure of a matrix in time $O(\log n)$ which can be done using $O(n^2)$ processors. Recently, $O(n^3)$ processors have been shown to be sufficient for the transitive closure problem [3]. We present an algorithm for the connected component problem that requires $O(n^2)$ processors.

**Definition:** A tree-loop is a tree having directed edges going from lower level nodes to upper level nodes (i.e., from leaves towards the root) with one additional edge from the root to one of its descendents. Thus, each node has outdegree one, and there is exactly one loop.

**Lemma:** If $E:V \rightarrow V$ is a function defining directed edges from a finite vertex set into itself, then $E$ is a set of tree-loops.

**Proof** - If we start at any vertex and follow the edges defined by $E$ (each vertex has outdegree exactly one since $E$ is a function), we will eventually loop since the number of vertices is finite.

Any other nodes that can reach any of the nodes in this structure must follow edges already in the structure upon entry. Thus, no second loop can be formed. \[]
We now present algorithm CONNECT that will determine the connected components of an undirected graph.

Input: array $A$ that is dimensioned $n$ by $n$ such that $A(i, j)=1$ if and only if there exists an edge from node $i$ to node $j$.

Output: vector $D$ that has length $n$ such that $D(i)$ is the smallest node $x$ reachable from node $i$ (i.e., is in the same connected component).

**Algorithm CONNECT**

1. Initialize for all $x$:
   
   $D(x) \leftarrow x$
   
   $C(x) \leftarrow \min\{ y \neq x \mid A(x, y)=1 \}$
   
   $\text{FLAG}(x) \leftarrow 1$

2. do steps 2 through 8 for log $n$ iterations

3. For all $x$ s.t. $\text{FLAG}(x)=0$ do $D(x) \leftarrow D(D(x))$

4. For all $x$ do $A[x, D(x)] \leftarrow 1$

5. For all $x, y$ do $A(x, y) \leftarrow A(x, y) \lor A(y, x)$

6. For all $x$ do if $D(x) \neq x$ then $\text{FLAG}(x) \leftarrow 0$

7. For all $x$ s.t. $\text{FLAG}(x)=0$ do
   
   $C(x) \leftarrow \min D(y) \neq D(x)$, if none then $D(x)$

8. For all $x$ s.t. $\text{FLAG}(x)=1$ do
   
   $C(x) \leftarrow \min C(y) \neq D(x)$, if none then $D(x)$
Define node $x$ to be a center in algorithm CONNECT if $\text{FLAG}(x) = 1$.

Define $\text{CEN}$ to be the set of centers of a graph in algorithm CONNECT. Note that, as the algorithm proceeds, $\text{CEN}$ can only lose members.

Initially, all nodes are centers and for each center, $x$, $\text{C}(x)$ points to another center (distinct from $x$) that is reachable from $x$ (if there are any such). Thus, $\text{C}$ defines a set of tree-loops. Also, if $\text{C}(x) = y$ then $x$ is adjacent to $y$.

All centers, $x$, have $\text{D}(x) = x$. $\text{D}(x)$ is the "best" (smallest-valued) node that we have so far been able to ascertain is reachable from $x$.

Lemma: The smallest-valued vertex in a tree-loop defined by $\text{C}$ will be in the loop, and the loop will be of size 2. [4]

Proof - Let $j$ be the vertex with lowest number in the tree-loop. Let $\text{C}(j)$ be $v$. Then $v$ and $j$ are adjacent. Assume $\text{C}(v) = w \neq j$. By definition of $\text{C}$, $\text{C}(v)$ is the smallest-valued node adjacent to $v$. Then $w < j$ which contradicts the assumption that $j$ was minimal. Therefore $\text{C}(v) = j$.

We have exhibited a loop, $(v, j)$, which is the only one in the tree-loop (by definition) and which contains the smallest-valued vertex in the tree-loop and which has size 2. [4]

Lemma: Vertex $x$ in tree-loop $L$, as defined before entry to
statement 3, will have $D(x)=j$, where $j$ is the smallest-valued vertex in $L$, upon exit from statement 3.

**Proof** - Let the vertices in the loop be $j$ (the smallest-valued vertex in the tree-loop $L$) and $v$. Thus, $C(j)=v$ and $C(v)=j$. Also, initially, all vertices $x$ in $L$ have $D(x)=x$. After one iteration, both $D(j)$ and $D(v)$ will have value $j$. Consider node $x$ that is on a path (in $L$) of length $m \leq n$ from the loop (nodes $j$ and $v$). After each iteration, the length of the path will diminish by half (each link will point to 2 nodes ahead instead of one).

Thus, after at most $\log n$ iterations within statement 3, for each node $x$ in $L$, $C(x)=w$ where $w$ is $j$ or $v$ and then $D(x) \leq D[C(x)] \leq \max\{D(j), D(v)\} = j$. \[

Thus, after statement 3, all nodes within a tree-loop, $L$, will point to the smallest-valued vertex, $j$, in $L$.

Statement 4 makes all nodes in $L$ adjacent to $j$ via array $A$.

Statement 5 makes $j$ adjacent to all nodes in $L$ via array $A$.

Statement 6 lets $j$ remain a center while all other nodes in the tree-loop are no longer centers.

We can think of a connected component as consisting of centers (such as $j$) each of which has followers (nodes, such as $x$, that have $D(x)=j$). All followers, $x$ of $j$, have $D(x)=j$ and $A(x,j)=A(j,x)=1$. Thus a connected component, $CC$, is partitioned into clubs, each club consisting of a center and
that center's followers. If CC has more than one club, then each club must be adjacent to another club. That is, within a club there must be at least one node that is adjacent to a member of a different club.

Statement 7 makes all followers x of j that are adjacent to a member of a different club point to that other club (using C). If a follower x of j is not adjacent to a member of a club other than j, then (and only then) does C(x) point to j (which is D(x)).

Finally, statement 8 defines the C function for the next iteration. C(j), j a center, will be the smallest-valued center whose club is adjacent to j's club. Note that if club j is adjacent to club k then club k is adjacent to club j.

The conditions for re-starting at statement 2 are now set. Adjacency now refers to adjacency of clubs.

Statement 2 is used to update the club memberships of non-centers since the club they belonged to may have been absorbed into another club.

We observe that the set of centers is partitioned into tree-loops and that each tree-loop is then reduced to a single center.

Since each tree-loop must contain at least two members, therefore each iteration must reduce the size of CEN by at least a factor of two. Thus, after at most log n iterations, there will be only one center in each connected component.
## Time - Processor Analysis

<table>
<thead>
<tr>
<th>Statement</th>
<th>Time</th>
<th>Processors</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>log n</td>
<td>( n^2 ) [min function requires ( n ) processors and log n time for each node ( x )]</td>
</tr>
<tr>
<td>2-8: multiply time by log n</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>( n )</td>
</tr>
<tr>
<td>3</td>
<td>log n</td>
<td>( n )</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>( n )</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>( n^2 )</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>( n )</td>
</tr>
<tr>
<td>7</td>
<td>log n</td>
<td>( n^2 )</td>
</tr>
<tr>
<td>8</td>
<td>log n</td>
<td>( n^2 )</td>
</tr>
</tbody>
</table>

**Theorem:** Algorithm CONNECT takes time \( O(\log n) \) using \( n \) processors.

**Proof** - see analysis above. \( \Box \)
REFERENCES


4. Ullman J.D. private communication.