Graphs, Search, Components

Undirected Graph

Representations:

Adjacency Matrix

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>b</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>c</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>d</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Adjacency Lists

- a: b, d
- b: a, c
- c: b
- d: a
Di(srected) Graph

vertex
arc (or edge)
\( h(e) \) head
\( t(e) \) tail

acyclic digraph

topological order: \( i \to j \iff h(i) < h(j) \)
Search

Explore vertices systematically by traversing edges

Mark vertices when visited

Traverse an untraversed edge from a visited vertex or else
Start a new search from an unvisited vertex

Breadth-first search: queue of visited vertices

Depth-first search: stack of visited vertices

\[ \text{dfs}(v): \{ \text{previsit}(v); \text{scan}(v); \text{postvisit}(v) \} \]

\[ \text{scan}(v): \text{for } e \in \text{arcs out}(v) \Rightarrow \text{traverse}(e) \]

\[ \text{traverse}(e): \{ \text{advance}(e); \\ \text{if not previsited}(\text{t}(e)) \Rightarrow \text{dfs}(\text{t}(e)); \\ \text{retreat}(e) \} \]

\[ \text{explore}(G): \text{for } v \in G \Rightarrow \text{if not previsited}(v) \Rightarrow \text{dfs}(v) \]
Strong Components by DFS (Gabow)

Maintain stack of tentative components

Add each new vertex to stack as a singleton component

When advancing along an edge, if it leads from a component to a lower component on stack, combine all components down to the lower component

When postvisiting the last vertex in a component, output the component

Needs disjoint set union to maintain components:

\[ O((m+n)\alpha(n)) \]

Components output in reverse topological order
Linear-Time Version

Maintain stack of vertices not in permanent components in preorder

Observe: tentative components are intervals on this stack

Maintain separate stack if (indices of) bottom vertices in tentative components

vertex number: 0 if not previsited
  stack position if positive
  - component number if negative
Blocks

Number vertices in pre(visit) order

\[ \text{low}(v) = \min \{ \text{pre}(v) \} \forall v \in \text{pre}(w) \exists (x, w) \text{ with } x \text{ a descendant of } v \}\]

\( v \) is a cut vertex iff start vertex with degree \( \geq 2 \)

or \( \exists \) child \( w \) of \( v \) with \( \text{pre}(v) \leq \text{low}(w) \)

Algorithm

Initialize \( \text{low}(v) = \text{pre}(v) \forall v \), stack empty

Advance \((v, w)\) \( \rightarrow \) add \((v, w)\) to stack

Retreat \((v, w)\) : if \((v, w)\) not a tree edge \( \rightarrow \\
\text{low}(v) = \min \{ \text{low}(v), \text{pre}(w) \} \)

else \( \text{low}(v) = \min \{ \text{low}(v), \text{low}(w) \} \)

if \( \text{low}(w) \geq \text{pre}(v) \), pop edges
down to and including \((v, w)\)
from stack to form a block