Digraph notions

• Vertex y is \textit{reachable} from x if there is a directed path in G from x to y.
  (By convention x is reachable from x by a directed path of length 0.)
• Vertices x and y are \textit{strongly connected} if x is reachable from y and y is reachable from x.
• Vertices x and y are \textit{weakly connected} if they are in the same connected component of the undirected version of G.

Strongly-connected vertices?

This means there are edges in both directions!

Weakly-connected vertices?

Strongly-connected vertices:

Weakly-connected vertices:

Digraph notions cont.

• Strongly connected is an equivalence relation; the equivalence classes are called \textit{strongly connected components}.
• Weakly connected is an equivalence relation; the equivalence classes are called \textit{weakly connected components}.
DFS Applications

- Identify the strongly connected components of a digraph G.
- Identify the weakly connected components of a digraph G. Easy! Do on your own.

DFS(G)

While G has an unvisited vertex x:
   DFS(x)

We'll choose these in alphabetical order

Note: DFS is overloaded!

DFS(x)

Mark x visited
For each edge <x,y>
   If y is unvisited then
      DFS(y)

We'll choose these in alphabetical order

DFS(a)

Visit a
Find <a,d> edge and call DFS(d)

Visit d
All out-edges checked so return

Call Stack:
   DFS(a)
   DFS(d)
   DFS(G)

DFS(d)

Visit d
All out-edges checked so return

Call Stack:
   DFS(d)
   DFS(a)
   DFS(G)
DFS(a)
Alphabetical priority
Visit a
Find <a,d> edge and call
DFS(d)
All out-edges checked so return

Call Stack:
DFS(a)
DFS(G)

DFS(G)
Alphabetical priority
a is unvisited so DFS(a)
b is unvisited so DFS(b)

Call Stack:
DFS(G)

DFS(b)
Alphabetical priority
Visit b
Find edge <b,c> and call
DFS(c)

Call Stack:
DFS(b)
DFS(G)

DFS(c)
Alphabetical priority
Visit c
Find edge <c,a> – no action
Find edge <c,b> – no action
All out-edges checked so return

Call Stack:
DFS(c)
DFS(b)
DFS(G)

DFS(b)
Alphabetical priority
Visit b
Find edge <b,c> and call
DFS(c)
Find edge <b,d> – no action
All out-edges checked so return

Call Stack:
DFS(b)
DFS(G)

DFS(G)
Alphabetical priority
a is unvisited so DFS(a)
b is unvisited so DFS(b)
All nodes checked so return

Call Stack:
DFS(G)
What is the running time of DFS?

- $O(m+n)$
- Every vertex is pushed onto the stack once and popped from the stack once.
- Each out-edge is inspected once.

SCC Problem

- Input: Digraph $G$
- Output: The strongly connected components of $G$.

Naïve Algorithm

For each pair of vertices $x$ and $y$:

- If $x$ is reachable from $y$ and $y$ is reachable from $x$ then $x$ and $y$ are strongly connected.

Run DFS

Naïve Algorithm

For each pair of vertices $x$ and $y$:

- If $x$ is reachable from $y$ and $y$ is reachable from $x$ then $x$ and $y$ are strongly connected.

Mark the vertices unvisited
DFS($y$)
If $x$ is marked visited then $x$ is reachable from $y$ Else $x$ is not reachable from $y$

Naïve algorithm

- Worst case: _______ calls to DFS($x$) so the running time is ___________

All little more sophistication please…

- We can find the strongly connected components of $G$ with two calls to DFS($G$)
- Three ideas
  - DFS Forest
  - Timestamps
  - Reversal of $G$
DFS Forest

- The DFS Forest of $G$ is the subgraph consisting of
  - Every vertex of $G$
  - Each edge traversed in DFS($G$)

![Graph G](image)

DFS tree of $G$: $\rightarrow$

Different selection rules give different results

WARNING

- DFS Forests are sometimes

What can we say?

- If the DFS forest of $G$ ____________ then $x$ and $y$ are strongly connected.
- If $x$ and $y$ are strongly connected then ____________

DFS Forest

- If $x$ and $y$ are strongly connected then they are in the same tree of (every) DFS forest of $G$.

![Strongly connected components of G](image)

DFS Forest

Strongly-connected in $G$ is a refinement of the weakly connected relation of any DFS forest of $G$

![Strongly connected components of G](image)  
DFS forest of $G$
Three ideas

• DFS Forest of G
• Timestamps
• Reversal

DFS(G)
Alphabetical order

Record first-arrival and last-departure times.

<table>
<thead>
<tr>
<th>First-arrival</th>
<th>Last-Departure</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td></td>
</tr>
<tr>
<td>b</td>
<td></td>
</tr>
<tr>
<td>c</td>
<td></td>
</tr>
<tr>
<td>d</td>
<td></td>
</tr>
</tbody>
</table>

DFS(G)
Alphabetical order

<table>
<thead>
<tr>
<th>First-arrival</th>
<th>Last-Departure</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>1</td>
</tr>
<tr>
<td>b</td>
<td>5</td>
</tr>
<tr>
<td>c</td>
<td>6</td>
</tr>
<tr>
<td>d</td>
<td>2</td>
</tr>
</tbody>
</table>

Three ideas

• DFS Forest
• Timestamps
• Reversal of G

$G^R$: Reverse the edges of $G$

$G$

$G^R$

$(G^R)^R$: Reverse the edges of $G^R$

$(G^R)^R = G$
Reachability

\[ X \text{ is reachable from } Y \text{ in } G \iff Y \text{ is reachable from } X \text{ in } G^T \]

SCC

The Strongly Connected Components of \( G \) and \( G^R \) are the same!

SCC Algorithm

- DFS(\( G \)) with timestamp
- DFS(\( G^R \)) using last-departure time decreasing order to produce a DFS forest \( F \) of \( G^R \)
- If \( x \) and \( y \) are weakly in \( F \) if and only if they are strongly connected in \( G \).

DFS(G)

<table>
<thead>
<tr>
<th></th>
<th>First-arrival</th>
<th>Last-Departure</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>b</td>
<td>5</td>
<td>8</td>
</tr>
<tr>
<td>c</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>d</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

DFS(\( G^R \))

Order: \( b,c,a,d \)

DFS Forest

Order: \( b,c,a,d \)
Correctness

Claim: If x and y are in the same strongly connected component of G if and only if x and y are in the same tree of the DFS forest of G.

Claim ⇒
If x and y are in the same strongly connected component of G then they are in the same tree of the DFS forest of G.

Claim ⇒
If x and y are in the same strongly connected component of G then they are in the same tree of the DFS forest of G.

Claim ⇐
If x and y are in the same tree of the DFS forest of G then they are in the same strongly connected component of G.

Claim ⇐
If x and y are in the same tree of the DFS forest of G then they are in the same strongly connected component of G.

1. Show x and r are strongly connected in G.
2. Show y and r are strongly connected in G.

Claim ⇐
x and r are strongly connected in G

1. Since x is reachable from r in G, r is reachable from x in G.
2. Since ___________________, x is reachable from r in G.
Notice

- Last-departure(x) < Last-departure(r).
- If Last-departure(x) < First-arrival(r) then r is not reachable from x in G ⇒ ⇐. So Last-departure(x) > First-arrival(r)
- Hence
  First-arrival(r) < First-arrival(x) < Last-departure(x) < Last-departure(r)

Claim ⇐

x and r are strongly connected in G
1. Since x is reachable from r in G, r is reachable from x in G.
2. Since First-arrival(r) < First-arrival(x) < Last-departure(x) < Last-departure(r), x is reachable from r in G.

Claim ⇐

If x and y are in the same tree of the DFS forest of G^r then they are in the same strongly connected component of G.

Correctness

Claim: If x and y are in the same strongly connected component of G if and only if x and y are in the same tree of the DFS forest of G^r.

Running Time

- We can find the strongly connected components of G in _______________
  - How do we represent G?
  - Do we have to sort the vertices by last-departure time?