Finding Strongly Connected Components with DFS

The Sharir-Kosaraju Algorithm
DFS(G)
1. For each vertex $u \in G.V$
2. \hspace{1em} $u.color = \text{WHITE}$
3. \hspace{1em} $u.\pi = \text{NIL}$
4. \hspace{1em} $\text{time} = 0$
5. For each vertex $u \in G.V$
6. \hspace{1em} if $u.color == \text{WHITE}$
7. \hspace{2em} DFS-Visit(G,u)

DFS-Visit(G,u)
1. \hspace{1em} $\text{time} = \text{time} + 1$
2. \hspace{1em} $u.d = \text{time}$
3. \hspace{1em} $u.color = \text{GRAY}$
4. For each vertex $v \in G.\text{Adj}[u]$
5. \hspace{2em} if $v.color == \text{WHITE}$
6. \hspace{3em} $v.\pi = u$
7. \hspace{3em} DFS-Visit(G,v)
8. \hspace{1em} $u.color = \text{BLACK}$
9. \hspace{1em} $\text{time} = \text{time} + 1$
10. \hspace{1em} $u.f = \text{time}$

\textbf{Lem 1:} The graph $G_{\pi} = (V, E_{\pi}), \ E_{\pi} = \{ (v_{\pi}, v) \mid v_{\pi} \neq \text{NIL} \}$ is a forest (DFS-forest).
Each node $v$ is associated with the interval $[v.d, v.f]$
DFS(G)
1. For each vertex \( u \in G.V \)
2. \( u.\text{color} = \text{WHITE} \)
3. \( u.\pi = \text{NIL} \)
4. \( \text{time} = 0 \)
5. For each vertex \( u \in G.V \)
6. if \( u.\text{color} == \text{WHITE} \)
7. \( \text{DFS-Visit}(G,u) \)

DFS-Visit(G,u)
1. \( \text{time} = \text{time} + 1 \)
2. \( u.d = \text{time} \)
3. \( u.\text{color} = \text{GRAY} \)
4. For each vertex \( v \in G.Adj[u] \)
5. if \( v.\text{color} == \text{WHITE} \)
6. \( v.\pi = u \)
7. \( \text{DFS-Visit}(G,v) \)
8. \( u.\text{color} = \text{BLACK} \)
9. \( \text{time} = \text{time} + 1 \)
10. \( u.f = \text{time} \)

**Thm 2:** \([v.d, v.f] \subset [u.d, u.f]\) iff \( v \) is a descendant of \( u \) in the DFS-forest, \([v.d, v.f] \cap [u.d, u.f] = \emptyset\) iff \( u \) and \( v \) are unrelated in the DFS-forest.
DFS(G)
1. For each vertex $u \in G.V$
2. $u.color = \text{WHITE}$
3. $u.\pi = \text{NIL}$
4. time = 0
5. For each vertex $u \in G.V$
6. if $u.color == \text{WHITE}$
7. DFS-Visit(G,u)

DFS-Visit(G,u)
1. time = time + 1
2. u.d = time
3. u.color = GRAY
4. For each vertex $v \in G.Adj[u]$
5. if $v.color == \text{WHITE}$
6. $v.\pi = u$
7. DFS-Visit(G,v)
8. u.color = BLACK
9. time = time + 1
10. u.f = time

Thm 3: $v$ is a descendant of $u$ in the DFS-forest iff when we invoke DFS-Visit(u) there is a white path from $u$ to $v$. 
Edge classification

•树边：(u,v)，DFS-Visit(v)是从DFS-visit(u)调用的
•回边：(u,v) 使得v是u在DFS森林的祖先
•前向边：非树边(u,v) 使得v是u的后裔
•交叉边：(u,v) 使得 v.f < u.d

Observe, among non-tree edges:
1. (Only) backward edges go to a vertex with a later finish time
2. (Only) forward edges go to a vertex with later discovery time
Strongly connected components (SCC)

Def: Equivalence classes of the relation \((u, v)\) iff there is a path from \(u\) to \(v\) and a path from \(v\) to \(u\)
SCC graph $G^{SCC} = (V^{SCC}, E^{SCC})$

Def: $V^{SCC}$ has a vertex $c_i$ per strong component $C_i$.

$E^{SCC}$ has an edge $(c_i, c_j)$ if there is an edge in $G$ from a vertex in $C_i$ to a vertex in $C_j$. 
SCC graph $G^{SCC} = (V^{SCC}, E^{SCC})$

Def: $V^{SCC}$ has a vertex $c$ per strong component $C$.

$E^{SCC}$ has an edge $(c, c')$ if there is an edge in $G$ from a vertex in $C$ to a vertex in $C'$.
SCC graph $G^{SCC} = (V^{SCC}, E^{SCC})$

Def: $V^{SCC}$ has a vertex $c$ per strong component $C$.

$E^{SCC}$ has an edge $(c, c')$ if there is an edge in $G$ from a vertex in $C$ to a vertex in $C'$

obs: $G^{SCC}$ is a DAG

proof: 1/2

proof: 1/2
Computing strongly connected components
Let's make a DFS

Thm 7: If there is an edge from $u \in C$ to $v \in C'$ then $f(C) \geq f(C')$
בשני מקומות כאןצריך להיות\[d(C)\]
ולא\[f(C)\]

\[f(c) = x \cdot d\]

\[f(c) > f(c')\]

**proof**

\[d(c) < d(c')\]

לעוז עוזרא

\[x \cdot d = f(c)\]

כרזים על סיווג

\[C \cup C' \subseteq \mathbb{R} \times \mathbb{R}\]

(خارجית)

\[1 \leq d(c)\]

(.inner)

\[y \cdot f < x \cdot f + y \in C \cup C'\]

Forest:\[f(c) < f(c')\]
\[ d(c') < d(c) \]

\[ \forall c, d = d(c') \]

\[ c' \in \text{Set} \]

\[ f(c) = f(c') \]

\[ \forall c \in \text{Set} \]

\[ f^* \text{ is a bijection} \]
Reverse the graph

Replace \((u, v)\) by \((v, u)\) \(\implies\) you get \(G^T\)
An algorithm to compute SCC’s

**STRONGLY-CONNECTED-COMPONENTS** \((G)\)

1. call **DFS**\((G)\) to compute finishing times \(u.f\) for each vertex \(u\)
2. compute \(G^T\)
3. call **DFS**\((G^T)\), but in the main loop of **DFS**, consider the vertices in order of decreasing \(u.f\) (as computed in line 1)
4. output the vertices of each tree in the depth-first forest formed in line 3 as a separate strongly connected component

**Thm 8:** Each DFS tree of \(DFS(G^T)\) is a strongly connected component
למען \(DFS(C_{\gamma})\) ו- \(DFS(C_{\gamma}^\perp)\) \(C\), \(C_{\gamma}\) \(C_{\gamma}^\perp\)

\(\phi\) \(\theta\)

\(\text{סיווג: }\)
לא צומת 7.
Computing $G^{SCC} = (V^{SCC}, E^{SCC})$

Remove parallel edges using an array..