16.4

DFS in Directed Graphs

DFS
16.4.1
DFS in Directed Graphs: Pre/Post numbering

DFS
DFS in Directed Graphs

**DFS(G)**

Mark all nodes $u$ as unvisited

$T$ is set to $\emptyset$

$\text{time} = 0$

while there is an unvisited node $u$ do

$\text{DFS}(u)$

Output $T$

**DFS(u)**

Mark $u$ as visited

$\text{pre}(u) = \text{++time}$

for each edge $(u, v)$ in $\text{Out}(u)$ do

if $v$ is not visited

add edge $(u, v)$ to $T$

$\text{DFS}(v)$

$\text{post}(u) = \text{++time}$
Example of **DFS** in directed graph
Example of DFS in directed graph
DFS Properties

Generalizing ideas from undirected graphs:

- **DFS** \( G \) takes \( O(m + n) \) time.
- Edges added form a **branching**: a forest of out-trees. Output of \( DFS(G) \) depends on the order in which vertices are considered.
- If \( u \) is the first vertex considered by \( DFS(G) \) then \( DFS(u) \) outputs a directed out-tree \( T \) rooted at \( u \) and a vertex \( v \) is in \( T \) if and only if \( v \in rch(u) \).
- For any two vertices \( x, y \) the intervals \([pre(x), post(x)]\) and \([pre(y), post(y)]\) are either disjoint or one is contained in the other.

**Note:** Not obvious whether \( DFS(G) \) is useful in directed graphs but it is.
DFS Properties

Generalizing ideas from undirected graphs:

1. **DFS**\((G)\) takes \(O(m + n)\) time.

2. Edges added form a **branching**: a forest of out-trees. Output of **DFS**\((G)\) depends on the order in which vertices are considered.

3. If \(u\) is the first vertex considered by **DFS**\((G)\) then **DFS**\((u)\) outputs a directed out-tree \(T\) rooted at \(u\) and a vertex \(v\) is in \(T\) if and only if \(v \in rch(u)\).

4. For any two vertices \(x, y\) the intervals \([pre(x), post(x)]\) and \([pre(y), post(y)]\) are either disjoint or one is contained in the other.

**Note:** Not obvious whether **DFS**\((G)\) is useful in directed graphs but it is.
DFS Properties

Generalizing ideas from undirected graphs:

1. **DFS**\((G)\) takes \(O(m + n)\) time.

2. Edges added form a **branching**: a forest of out-trees. Output of **DFS**\((G)\) depends on the order in which vertices are considered.

3. If \(u\) is the first vertex considered by **DFS**\((G)\) then **DFS**\((u)\) outputs a directed out-tree \(T\) rooted at \(u\) and a vertex \(v\) is in \(T\) if and only if \(v \in rch(u)\)

4. For any two vertices \(x, y\) the intervals \([\text{pre}(x), \text{post}(x)]\) and \([\text{pre}(y), \text{post}(y)]\) are either disjoint or one is contained in the other.

**Note:** Not obvious whether **DFS**\((G)\) is useful in directed graphs but it is.
DFS Properties

Generalizing ideas from undirected graphs:

1. **DFS**(G) takes $O(m + n)$ time.

2. Edges added form a branching: a forest of out-trees. Output of **DFS**(G) depends on the order in which vertices are considered.

3. If $u$ is the first vertex considered by **DFS**(G) then **DFS**(u) outputs a directed out-tree $T$ rooted at $u$ and a vertex $v$ is in $T$ if and only if $v \in rch(u)$

4. For any two vertices $x, y$ the intervals $[pre(x), post(x)]$ and $[pre(y), post(y)]$ are either disjoint or one is contained in the other.

Note: Not obvious whether **DFS**(G) is useful in directed graphs but it is.
DFS Properties

Generalizing ideas from undirected graphs:

1. **DFS**($G$) takes $O(m + n)$ time.

2. Edges added form a branching: a forest of out-trees. Output of **DFS**($G$) depends on the order in which vertices are considered.

3. If $u$ is the first vertex considered by **DFS**($G$) then **DFS**($u$) outputs a directed out-tree $T$ rooted at $u$ and a vertex $v$ is in $T$ if and only if $v \in rch(u)$

4. For any two vertices $x, y$ the intervals $[\text{pre}(x), \text{post}(x)]$ and $[\text{pre}(y), \text{post}(y)]$ are either disjoint or one is contained in the other.

Note: Not obvious whether **DFS**($G$) is useful in directed graphs but it is.
DFS tree and related edges

Edges of $G$ can be classified with respect to the DFS tree $T$ as:

1. **Tree edges** that belong to $T$

2. A **forward edge** is a non-tree edges $(x, y)$ such that $\text{pre}(x) < \text{pre}(y) < \text{post}(y) < \text{post}(x)$.

3. A **backward edge** is a non-tree edge $(y, x)$ such that $\text{pre}(x) < \text{pre}(y) < \text{post}(y) < \text{post}(x)$.

4. A **cross edge** is a non-tree edges $(x, y)$ such that the intervals $[\text{pre}(x), \text{post}(x)]$ and $[\text{pre}(y), \text{post}(y)]$ are disjoint.
Types of Edges
THE END

... 

(for now)