Depth First Search (DFS)

\[ G \text{ is graph} \]

\[ V : \text{set of vertices} = \{ v_1, v_2, \ldots, v_n \} \quad |V| = n \]

\[ E : \text{set of edges} \subseteq V \times V \]

\[ (u, v) \in E \implies \text{there is an edge from } u \text{ to } v \]

\[ |E| = m \]

Undirected

vertex order in \((u, v)\) does not matter

\( (u, v) = (v, u) \)

\( u \) & \( v \) are adjacent

Directed

vertex order matters

\( (u, v) \implies u \) is source and \( v \) is the target/destination

\( (v, u) \implies v \) is source & \( u \) is the target

Neighbors

\( G = (V, E) \), undirected (unweighted)

For any \( u \in V \)

\[ N(u) = \{ v \mid (u, v) \text{ is an edge in } E \} \]

\[ N(u) = \{ x_1, x_2, x_3 \} \]
Degree \( d(u) : |N(u)| = \text{# of neighbors} \)

Directed graphs

\[ \begin{align*}
    x_1 & \xrightarrow{e_1} x_2 \\
    x_2 & \xrightarrow{e_2} x_3 \\
    x_3 & \xrightarrow{e_3} x_4 \\
    x_4 & \xrightarrow{e_4} x_5 \\
    x_5 & \xrightarrow{e_5} x_6
\end{align*} \]

In neighbors: \( u \rightarrow N^-(u) \)

\[ \text{In degree } (u) = \left\{ v \mid (v, u) \in E \right\} \]

Out neighbors: \( \{ v \mid (u, v) \in E \} : N^+(u) \)

Out degree \( (u) = |N^+(u)| \)

Walk: an alternating sequence of vertices and edges

\( (2, 4, 1, 4, 3) \)

Graph:

\[ \begin{align*}
    1 & \xrightarrow{e_1} 2 \\
    2 & \xrightarrow{e_2} 3 \\
    3 & \xrightarrow{e_4} 4 \\
    4 & \xrightarrow{e_5} 5 \\
    e_1 & \xrightarrow{e_2} 2 \\
    e_2 & \xrightarrow{e_3} 3 \\
    e_3 & \xrightarrow{e_4} 4 \\
    e_4 & \xrightarrow{e_5} 5
\end{align*} \]

Explicit:

\( a_1, a_2, a_3, a_4, \ldots \)

Trail: a walk with no repeated edge

\( (2, 4, 1, 4, 3) \) is a walk, but not a trail.

Path: a walk with no repeated vertex

\( (2, 4, 1, 4, 3) \) is also not a path.
Graph Traversal

- Depth First Search (DFS)
- Breadth First Search (BFS)

DFS:
- Connectivity
- Cycles

Undirected graphs

$\text{dfs}(G, u)$:

$\begin{align*}
\rightarrow \text{visited}(u) &= \text{True} \\
\text{previsit}(u) &\triangleright \text{enter } u \\
\text{for } v \in N(u) : \\
\quad \text{if not visited}(v) \\
\quad \quad \text{dfs}(G, v) \\
\text{postvisit}(u) &\triangleright \text{leave } u
\end{align*}$

Connectedness

$\text{dfs}(G)$

$N(x)$ will be looked at in alphabetic order

$\text{dfs-Tree}$

Red edge:
- Tree edges (forward)
- Non-tree edges
- Back edges

DFS also gives us connected components

Component: largest set of vertices such as $C$.
Components
(maximal)
Containment relationship

\[ \text{cc}(G) : \]
\[
\begin{cases}
\text{ccnum} = 0 \\
\text{for } u \in V : \]
\[
\begin{cases}
\text{if not visited}(u) : \\
\text{dfs}(G, u, \text{ccnum}) \\
\text{ccnum} = \text{ccnum} + 1
\end{cases}
\end{cases}
\]

Previsit
\[
\begin{cases}
\text{visited}(u) = \text{True} \\
\text{ccnum}(u) = \text{ccnum} \\
\text{for } v \in N(u) \\
\text{dfs}(G, v, \text{ccnum})
\end{cases}
\]

\[ \Omega(N(u)) \]

\[ O(V) + O(E) \]

\[ O(V+E) \]

previsit & postvisit numbers

\[ \text{previsit}(u) \]

Increment a counter for both events

\[ \text{postvisit}(u) \]

Think of this as an interval
Directed graphs

- Directed acyclic graphs (DAGs)
- Strongly Connected Components

DFS order: in the DFS tree, if u is before v, then \( \text{post}(u) > \text{post}(v) \)

Proof:
A directed graph contains a cycle iff there exists a back edge.
contains a cycle if \( E \) a back edge!

edge from a node to its ancestor \((F, \theta)\)

cross-edge:
edge from a node \( u \) to another node \( v \)

such that \( v \) has already been visited before

\( u \) is seen

\( \text{pre}(u) > \text{post}(v) \) \((H, \delta)\)

Detecting cycle:

Is cyclic \((G)\): directed graph

1) do pre-post numbering in \(O(V+E)\) time

\text{linear}

2) during pre-post numbering,

output False if we encounter a back edge

\( \exists (V, u) \in E \quad \forall \ u \in N(v) \) in dfs

\([\text{pre}(v), \text{post}(v)] \subseteq [\text{pre}(u), \text{post}(u)]\)

In a directed graph

Weakly Connected Components (WCC)

A strongly connected component (SCC)

\( \exists \) a path between all pairs of vertices in the component

we do not have a path between all pairs of vertex
pair between all pairs of vertices

Any directed graph is a DAG over its SCCs.