Directed Acyclic Graph (DAG)

To prepare revealing the connectivity-structure of directed graphs, we first introduce a spsecial class of directed graphs, directed acyclic graphs (DAGs).

Definition 1 (DAG)**.** A directed graph $G = (V, E)$ is *acyclic* if and only if G does not contain cycles.

Let $G = (V, E)$ be a directed graph. If a vertex $v \in V$ does not have any in-edges (i.e., *in-degree* is 0), we call it *source vertex*; if a vertex $v \in V$ does not have any out-edges (i.e., *out-degree* is 0), we call it *sink vertex*.

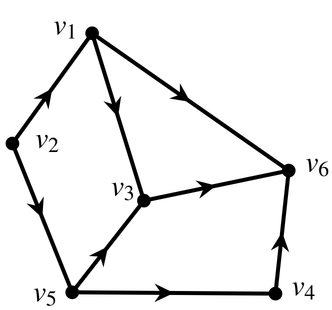
A directed graph may contain multiple source vertices or sink vertices, or may not have any source vertex or sink vertex. (Can you give such examples?)

Claim 1. A DAG $G = (V, E)$ always has source vertex and sink vertex.

Proof. Let's prove it by contradiction. Assume that G does not contain any source. First, G must not contain self-loop as otherwise G won't be a DAG. Let v be any vertex in V. As v is not a source, we know that there exists some vertex u points to v, i.e., $(u, v) \in E$. Now since u is not a source then there must exist another vertex w such that $(w, u) \in E$. Notice that $w \neq v$ as otherwise there will be a cycle: $v = w \to u \to v$. This means that w is a new vertex. Again as w is not a source, there must exist another *new* vertex points to it. This process can be extended infinitely following the fact and assumption that G is a DAG and all vertices not are sources, but this is not possible as the number of vertices is limited. The existence of sink can be proved symmetrically. □

Definition 2 (Linearization / Topological Sorting)**.** Let $G = (V, E)$ be a directed graph. Let X be an ordering of V. If X satisfies: if $(v_i, v_j) \in E$, then v_i is before v_j in X, then we say X is a linearization (or toplogical sorting) of G.

See some examples below.



$(v_2, v_1, v_5, v_3, v_4, v_6)$ linearization? Yes

$(v_2, v_5, v_1, v_4, v_3, v_6)$ linearization? Yes

$(v_2, v_5, v_3, v_4, v_1, v_6)$ linearization? No, see edge (v_1, v_3)

$(v_2, v_1, v_4, v_5, v_3, v_6)$ linearization? No, see edge (v_5, v_4)

Figure 1: Examples of linearization.

If a directed graph G admits a linearization, then we say G can be *linearized*. We now show that linearization is an *equivalent* characterization of DAGs.

Claim 2. A directed graph G can be linearized if and only if G is a DAG.

Proof. Let's first prove that if G can be linearized, then G is a DAG. This is equivalent to proving its contraposition: if G contains a cycle, then G cannot be linearized. Suppose that there exists an cycle $v_{i_1} \to v_{i_2} \to \cdots \to v_{i_k} \to v_{i_1}$ in G. Then the linearization X must satisfy that v_{i_j} is before $v_{i_{j+1}}$ for all $j = 1, 2, \cdots, k-1$, and that v_{i_k} is before v_{i_1}, in X. Clearly, this is not possible.

The other side of the statement, i.e., if G is a DAG, then G can be linearized, can be proved constructively.

We will design an algorithm (see below), that constructs a linearization for any DAG. The idea of the algorithm is to iteratively finds source vertex and removes it and its out-edges.

Algorithm find-linearization ($G = (V, E)$)
    init X as empty list;
    while (G is not empty)
        arbitrarily find a source vertex u of G;
        add u to the end of X;
        update G by removing u and its out-edges;
    end while;
end algorithm;

This algorithm is correct. First, when a vertex u is added to X, it is a source vertex of the current graph, which means that $\{w \mid (w, u) \in E\}$ is either empty or all of them have been added to X. Second, X will include all vertices. This is because, a source always exists in a DAG (as we just proved). The above algorithm is more a framework, as how we update the graph is not given specifically, and which affects the running time.

Above algorithm gives a constructive proof that, a DAG can always be linearized. This completes the proof for the fact that, a directed graph is a DAG if and only if it can be linearized. □

Meta-Graph

For a directed graph $G = (V, E)$, its structure of connectivity can be represented as a new directed graph, called *meta-graph*, denoted as $G_M = (V_M, E_M)$. Each of the vertices of the meta-graph corresponds to a connected component of G, and two vertices $C_i, C_j \in V_M$ are connected by edge $(C_i, C_j) \in E_M$ if and only if there exists edge $(u, v) \in E$ such that $u \in C_i$ and $v \in C_j$. An example of meta-graph is given below.

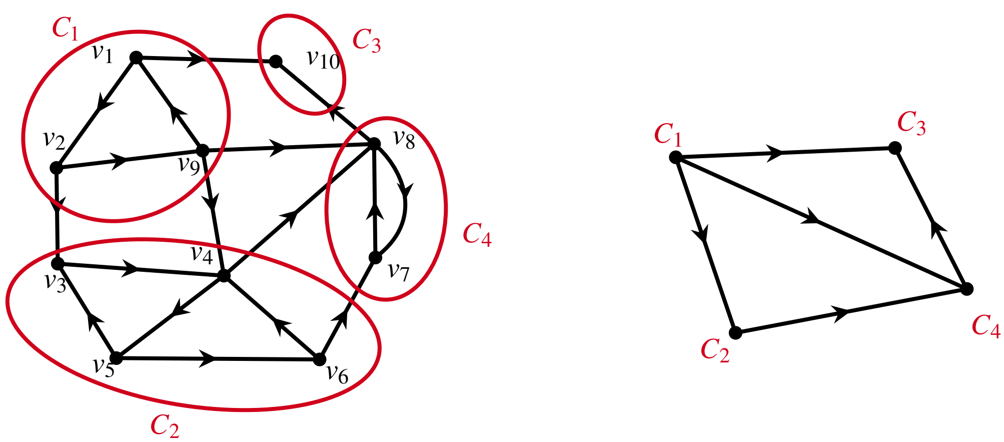


Figure 2: Example of meta-graph.

Meta-graph has an important property: it does not contain cycles, i.e., it is a directed acyclic graph (DAG).

Claim 3. The meta-graph G_M of any directed graph G is a directed acyclic graph.

Proof. Suppose conversely that G_M contains a cycle, $C_1 \to C_2 \to C_k \to C_1$, then the union of the vertices in these connected components form a single connected component, contradicting to the *maximal* property of connected component. $\square$

Reverse Graph

Definition 3. Let $G = (V, E)$ be a directed graph. The *reverse graph* of G, denoted as $G_R = (V, E_R)$, has the same set of vertices and edges with reversed direction, i.e., $(u, v) \in E$ if and only if $(v, u) \in E_R$.

Following properties can be easily proved using above definition.

Property 1. $(G_R)_R = G$.

Property 2. X is a linearization of DAG G if and only if the reverse of X is a linearization of G_R.

Property 3. There is a path from u to v in G if and only if there is a path from v to u in G_R. In other words, u can reach v in G if and only if u is reachable from v in G_R.

Property 4. G and G_R has the same set of connected components.

Property 5. The meta-graph of G_R is the reverse graph of the meta-graph of G. Formally, $(G_R)_M = (G_M)_R$.

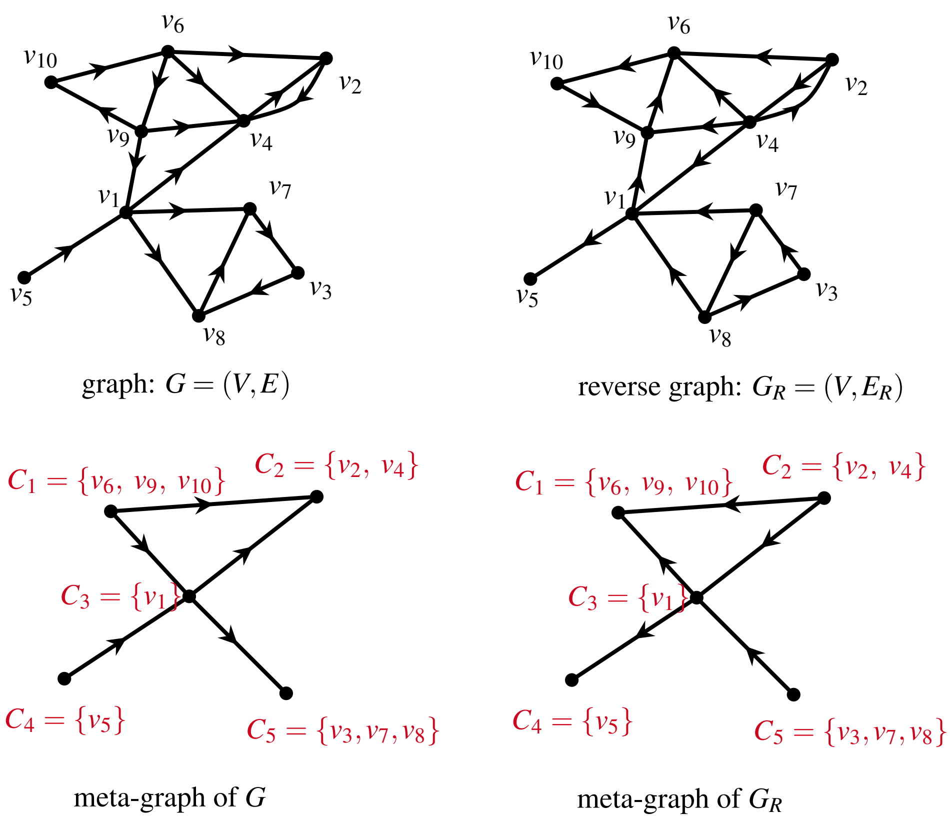


Figure 3: Graph and its reverse graph, and the corresponding meta-graphs.