5CCS2FC2: Foundations of Computing II

Introduction to Graph Algorithms

Week 4

Dr Christopher Hampson

Department of Informatics

King's College London

Topological Sorting

Directed Acyclic Graphs (DAGs)

- A graph $G = (V, E)$ is said to be a **directed acyclic graph** if
  - it is **irreflexive**, and
  - does not contain any **cycles** of length ≥ 2

> if there is a path $u \rightsquigarrow v$ then there is no path $v \rightsquigarrow u$

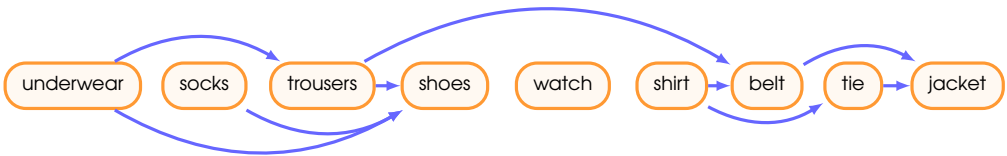
Example: Software Dependency Graphs

`http://cxf.apache.org/docs/cxf-dependency-graphs.html`

- **Topological Sort**
  - A **topological sorting** of a Directed Acyclic Graph $G = (V, E)$ is an ordered list of vertices $L = \langle v_1, v_2, \ldots, v_n \rangle$ such that

> If $v_i \rightsquigarrow v_j$ then $i < j$ for all $i, j \leq n$

(*i.e.* all the arrows point 'downstream' from v_1 to v_n)

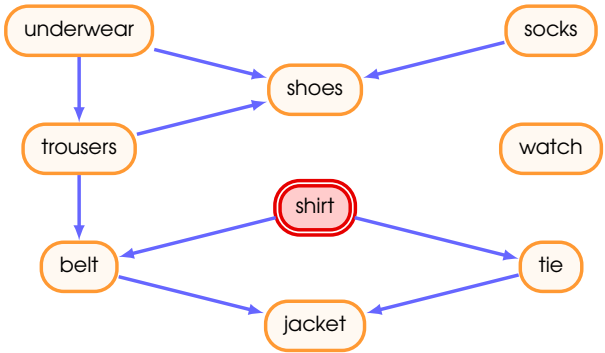


Topological Sorting

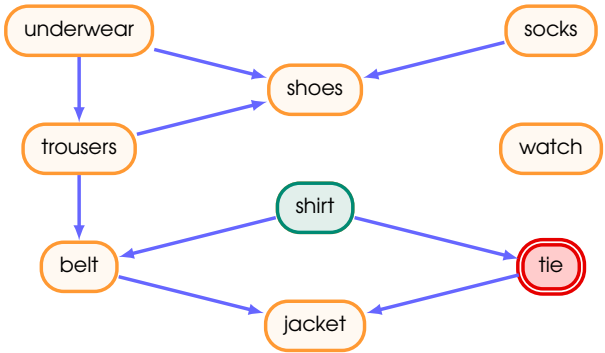
TOPOLOGICAL-SORT (G):

1. **While** verticies remain unsorted:
    2. Select any unsorted vertex and add to a stack
    3. **While** stack is not-empty:
        4. Inspect the top element u on the stack.
        5. **If** u has no unsorted neighbours:
            6. Pop u from the stack and add to list L.
        7. **Else:**
            8. Add any unsorted neighbours of u to the stack.
9. **Return** sorted list L

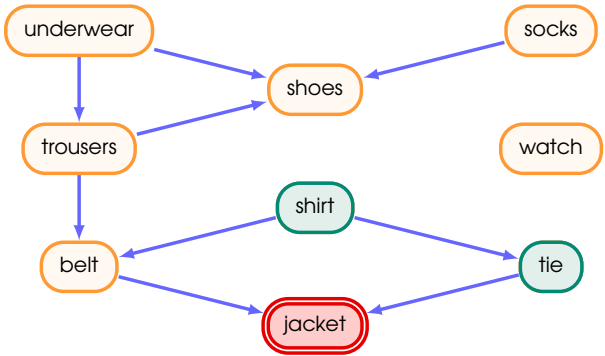
Topological Sorting



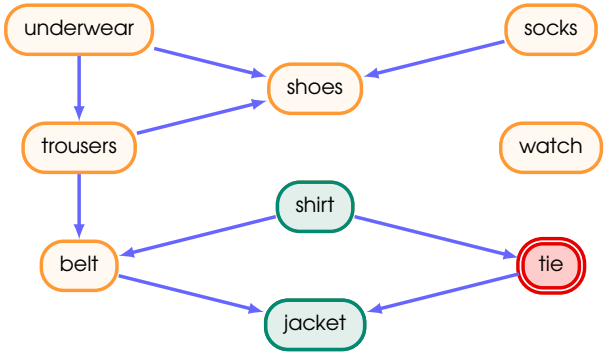
Topological Sorting



Topological Sorting

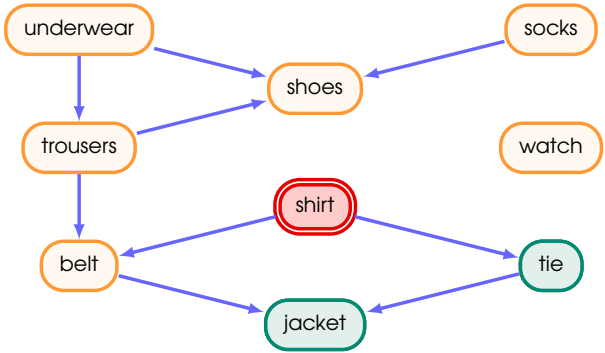


Topological Sorting



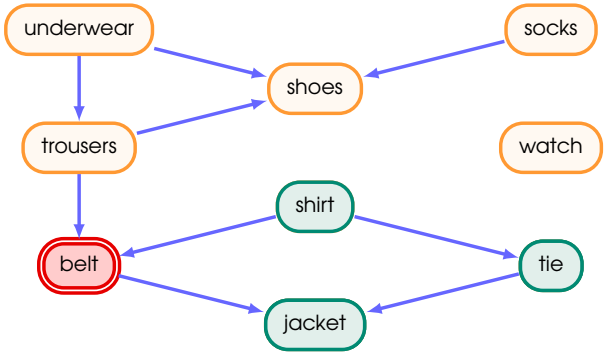
jacket

Topological Sorting



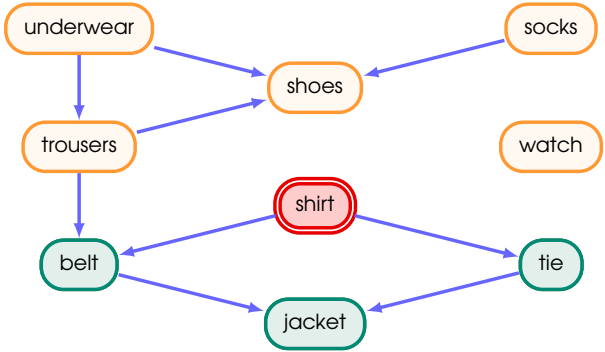
tie jacket

Topological Sorting



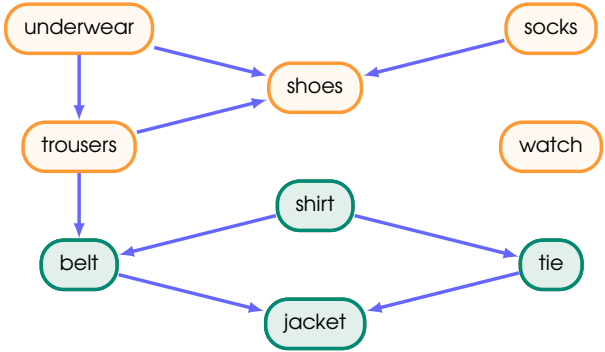
tie jacket

Topological Sorting



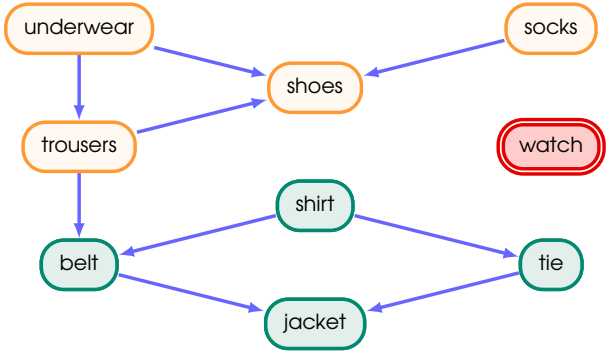
belt tie jacket

Topological Sorting



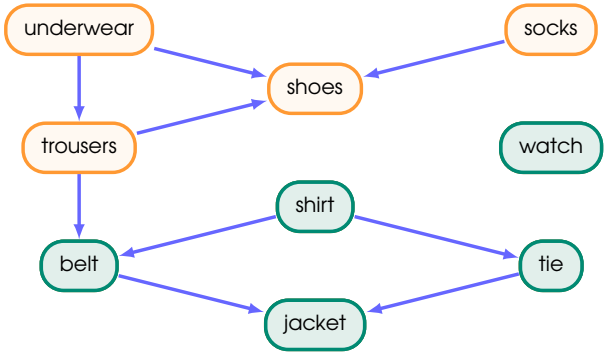
shirt belt tie jacket

Topological Sorting



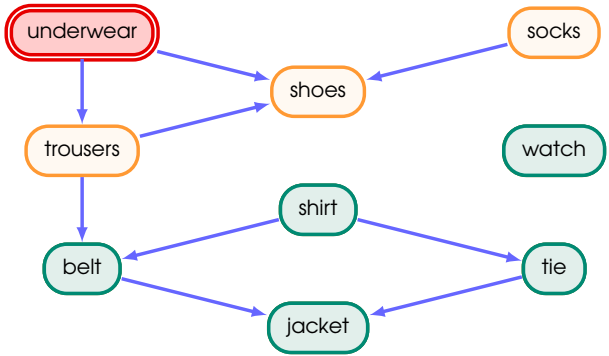
shirt belt tie jacket

Topological Sorting



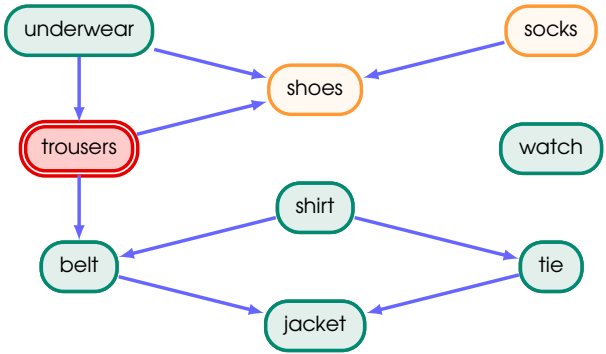
watch shirt belt tie jacket

Topological Sorting



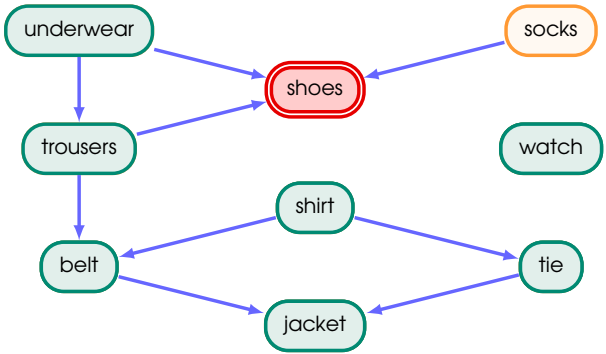
watch shirt belt tie jacket

Topological Sorting



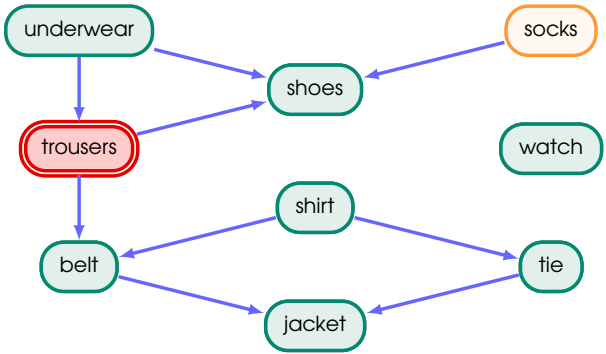
watch shirt belt tie jacket

Topological Sorting



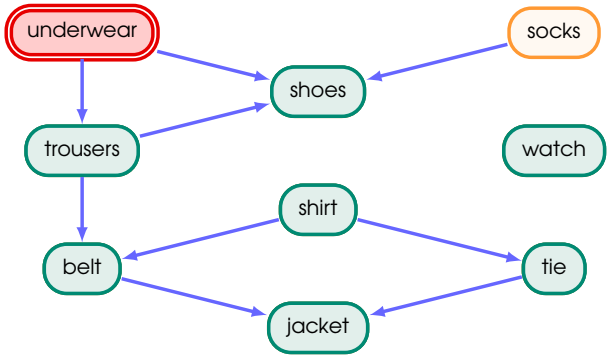
watch shirt belt tie jacket

Topological Sorting



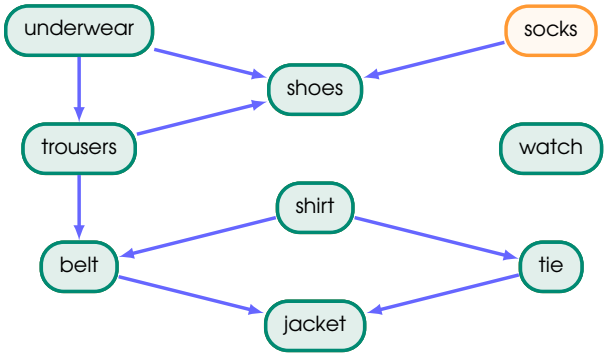
shoes watch shirt belt tie jacket

Topological Sorting



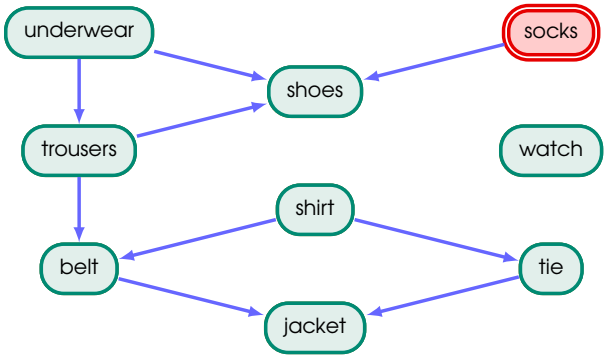
trousers shoes watch shirt belt tie jacket

Topological Sorting



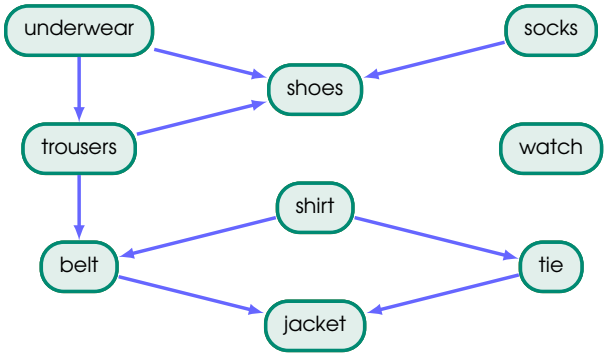
underwear trousers shoes watch shirt belt tie jacket

Topological Sorting



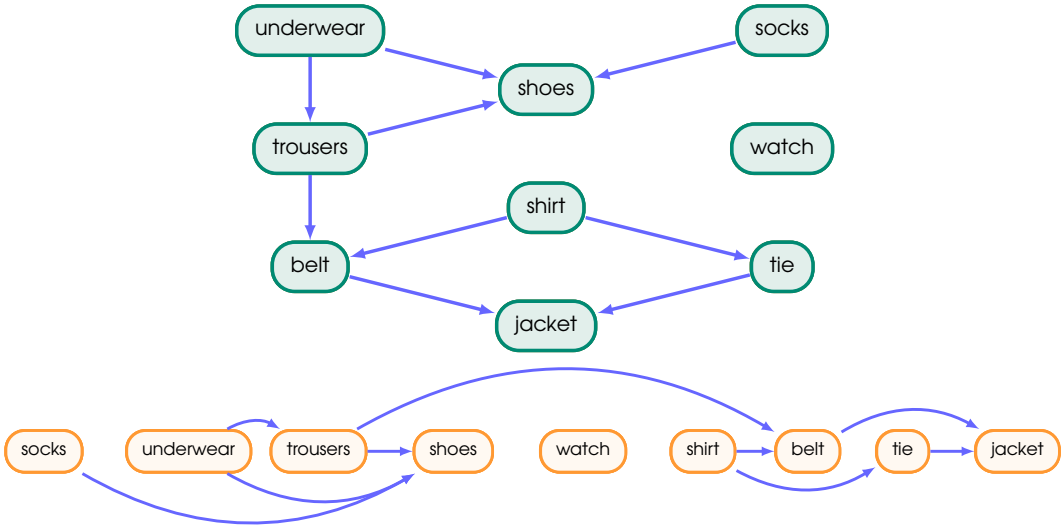
underwear trousers shoes watch shirt belt tie jacket

Topological Sorting



socks underwear trousers shoes watch shirt belt tie jacket

Topological Sorting



Strongly Connected Components

Some Definitions

- **Strong Connected Component (SCC)**
  - We can define an **binary relation** $\sim \subseteq V \times V$ on the set of vertices such that

$$u \sim v \iff u = v \text{ or there is a path } u \rightsquigarrow v \text{ and a path } v \rightsquigarrow u$$

  - A **strongly connected component** is a *maximal* subset $C \subseteq V$, such that

$$u \sim v \quad \text{for all } u, v \in C$$

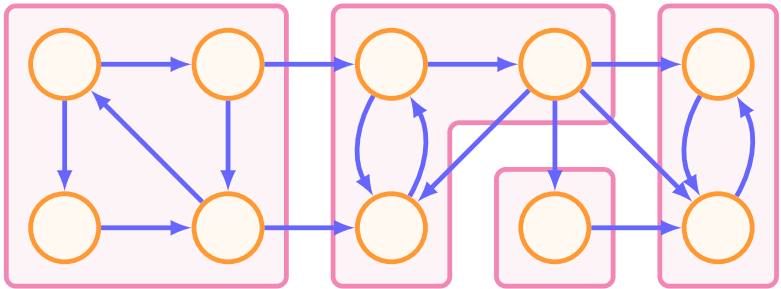
$$[u]_{\equiv} = \{v \in V : v \sim u\}$$

- **Component Graph**
  - The **component graph** of G is a new graph $G^{SCC} = (V^{SCC}, E^{SCC})$, where

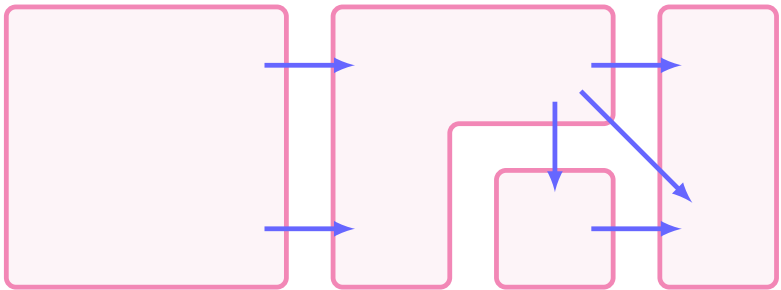
$$V^{SCC} = \{ \text{ all strongly connected components of } G \}$$

$$(A, B) \in E^{SCC} \iff (a, b) \in E \quad \text{for some } a \in A \text{ and } b \in B, \text{ where } A \neq B$$

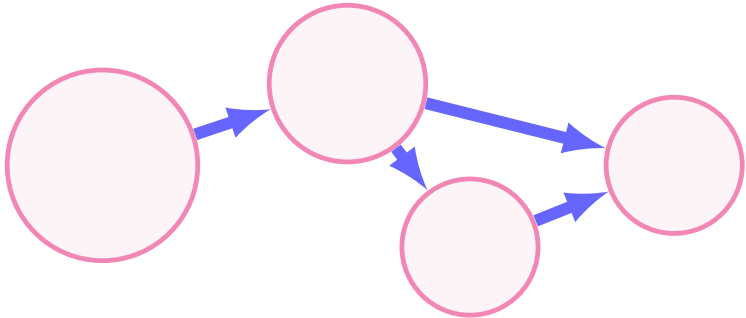
Some Definitions



Some Definitions



Some Definitions

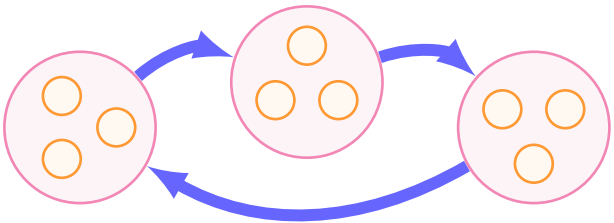


Strongly Connected Components

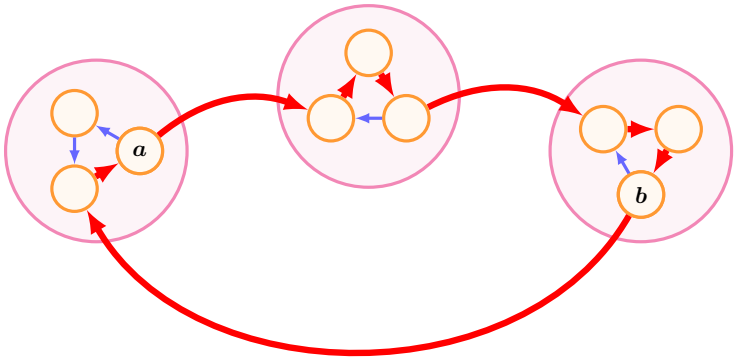
Theorem The *component graph* $G^{\mathbf{SCC}}$ is a directed acylic graph, for any directed graph G.

Proof:

Step 1) Suppose, for **contradiction**, that there is a cycle $\langle C_1, C_2, \ldots, C_n, C_1 \rangle$:

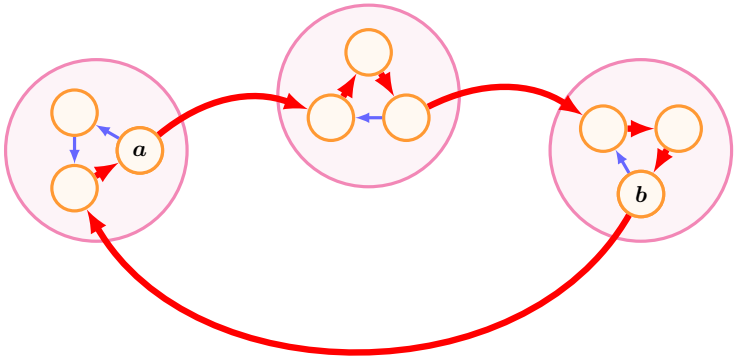


Step 2) Choose any $a \in C_i$ and $b \in C_i$ for some $C_i \neq C_j$.



Strongly Connected Components

Step 3) It follows that there must be a path $a \rightsquigarrow b$ and $b \rightsquigarrow a$



Strongly Connected Components

Step 4) Hence, by **contradiction**, $G^{\mathbf{SCC}}$ cannot contain any cycles, as required.

Q.E.D.

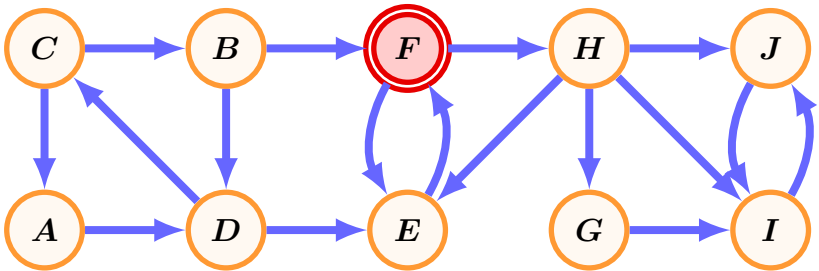
Finding Strongly Connected Components

Strongly Connected Components

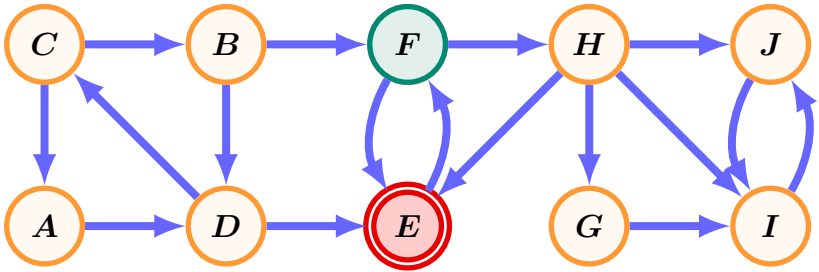
SCC (G) :

1. Call TOPOLOGICAL-SORT (G) to generate ordered list L
2. **While** L is not-empty:
    3. Inspect the first element u in the list.
    4. Perform DFS on the *transpose graph* G^T from u
    5. Add all visited verticies to a new component S
    6. Remove these verticies from the list L.
7. **Return** all components identified

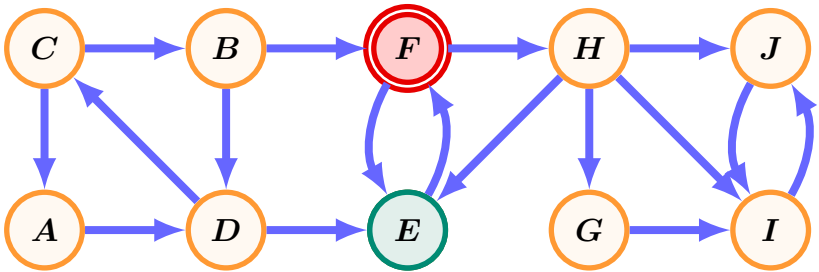
Strongly Connected Components



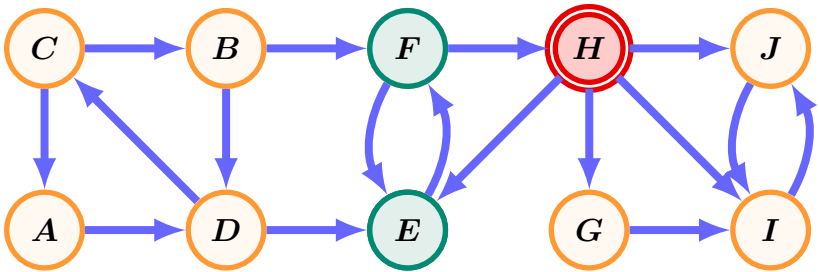
Strongly Connected Components



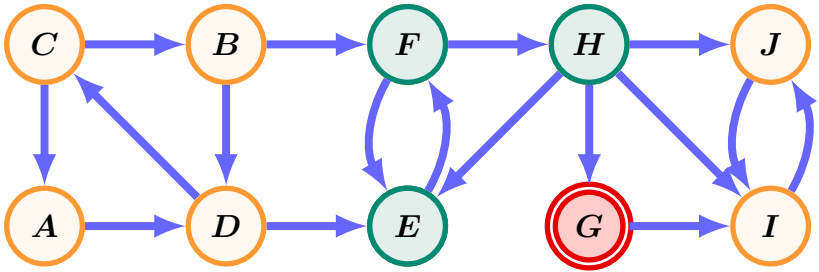
Strongly Connected Components



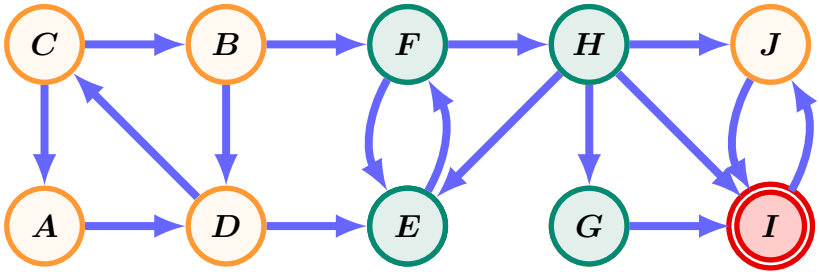
Strongly Connected Components



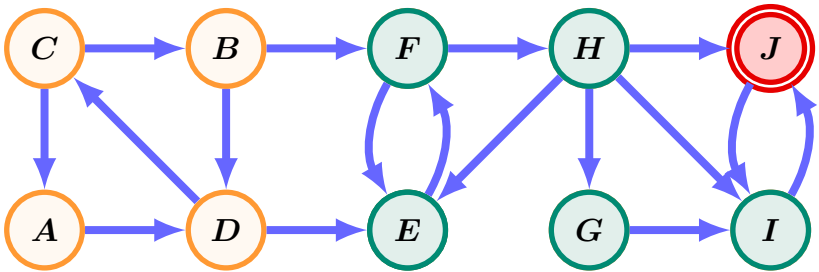
Strongly Connected Components



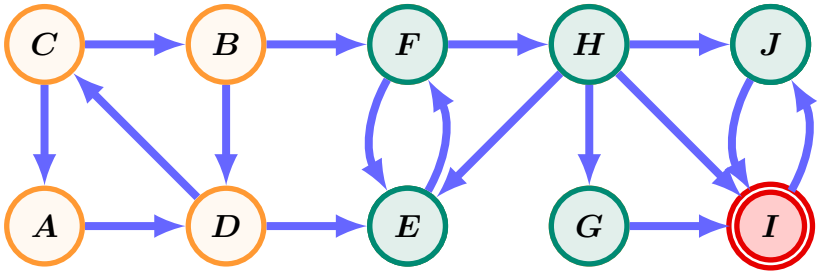
Strongly Connected Components



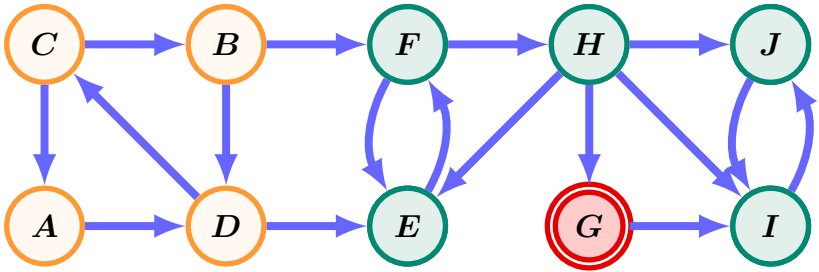
Strongly Connected Components



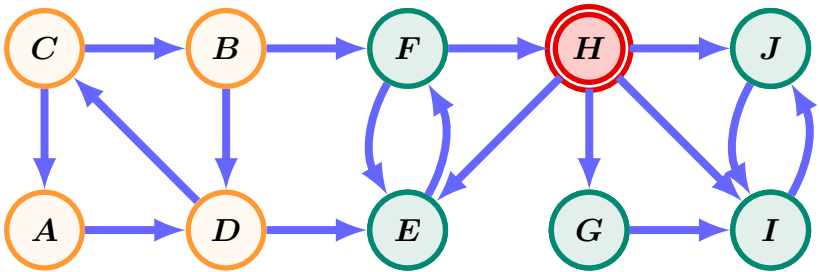
Strongly Connected Components



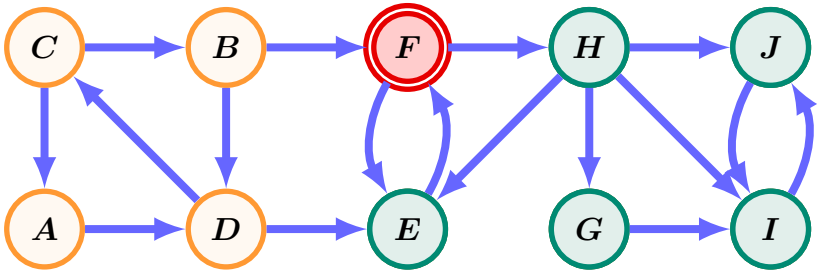
Strongly Connected Components



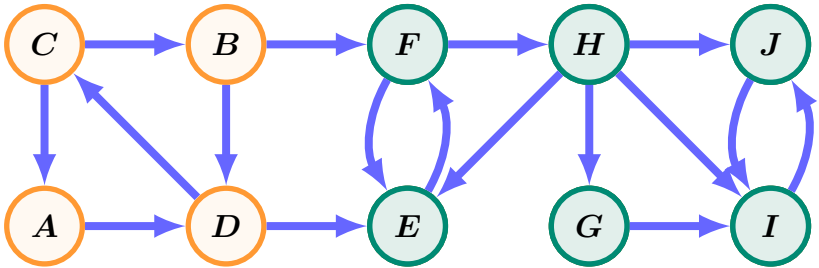
Strongly Connected Components



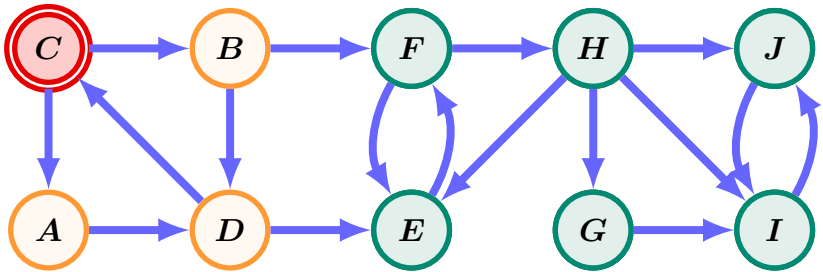
Strongly Connected Components



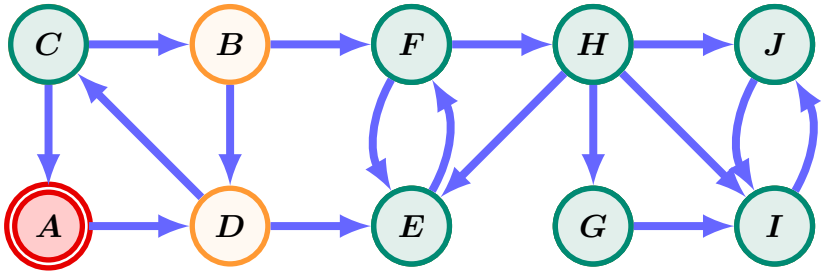
Strongly Connected Components



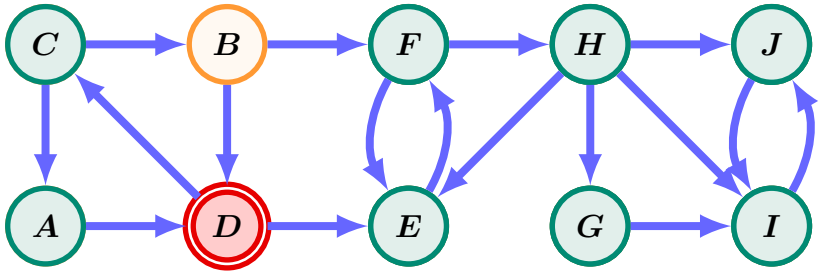
Strongly Connected Components



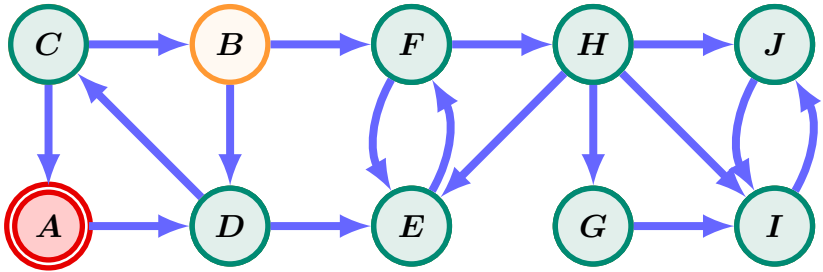
Strongly Connected Components



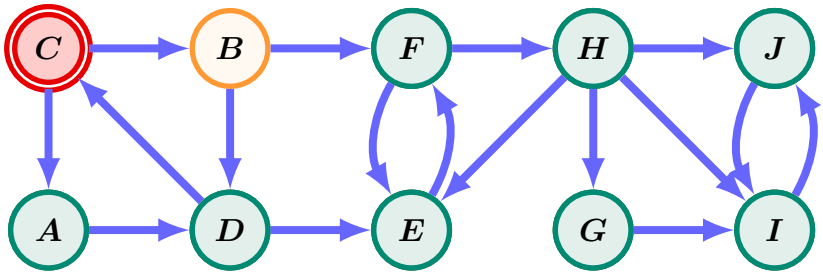
Strongly Connected Components



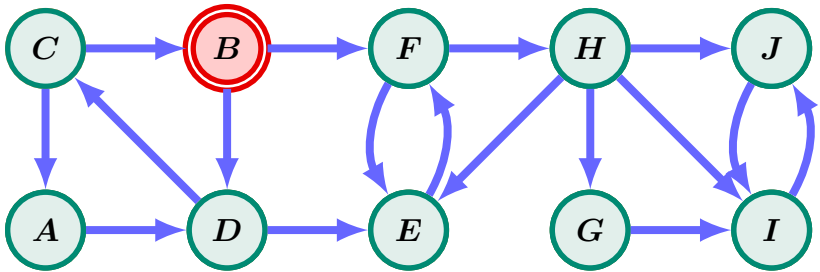
Strongly Connected Components



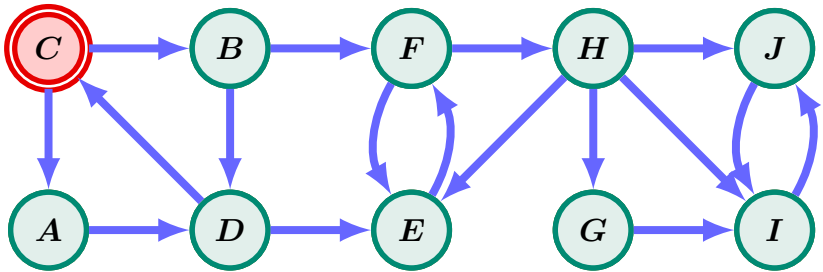
Strongly Connected Components



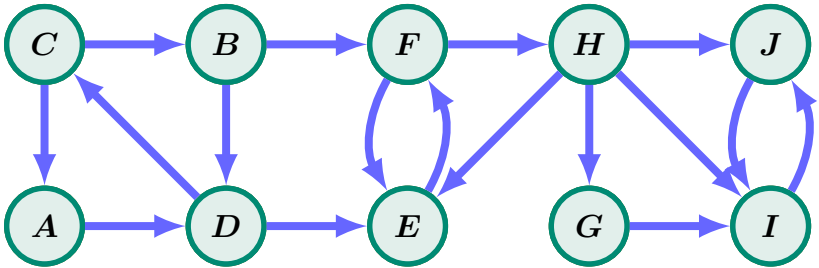
Strongly Connected Components



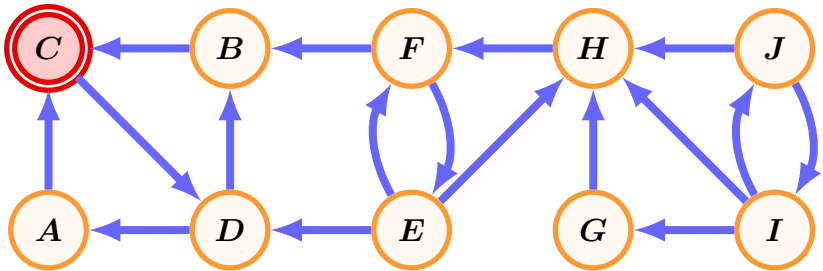
Strongly Connected Components



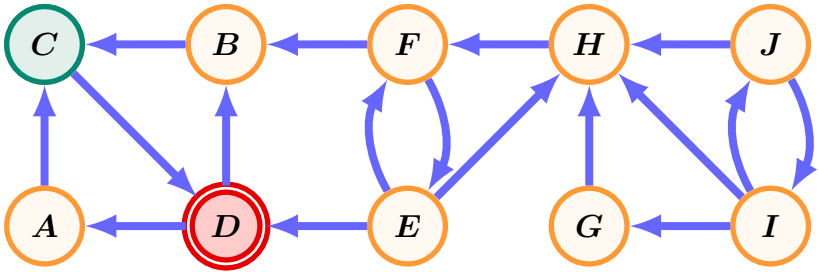
Strongly Connected Components



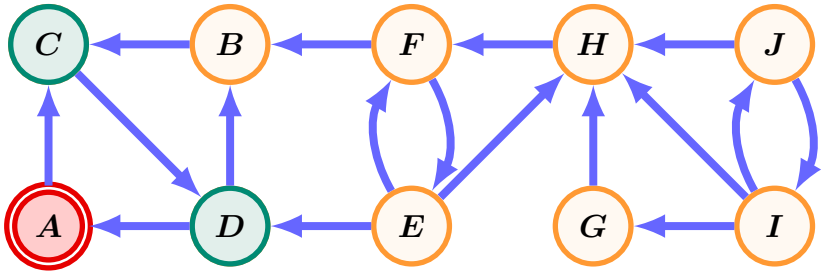
Strongly Connected Components



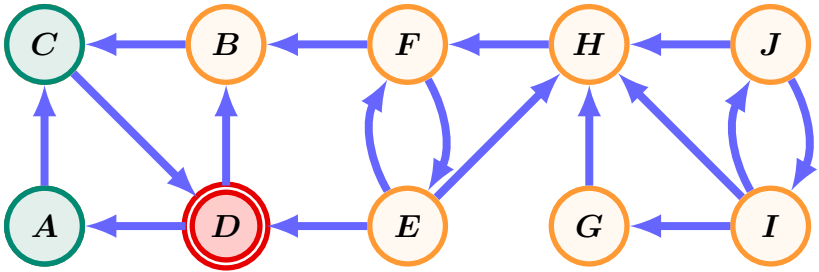
Strongly Connected Components



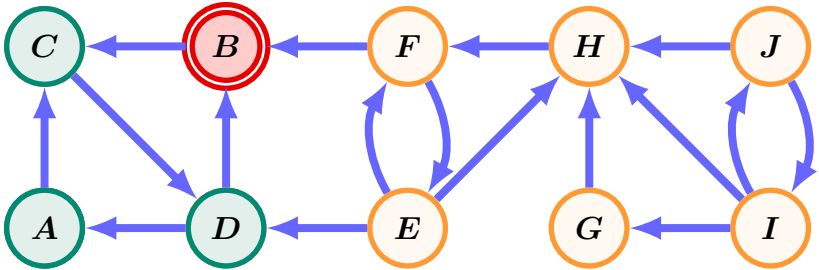
Strongly Connected Components



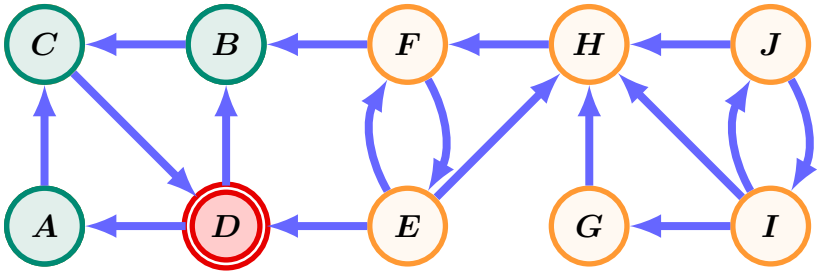
Strongly Connected Components



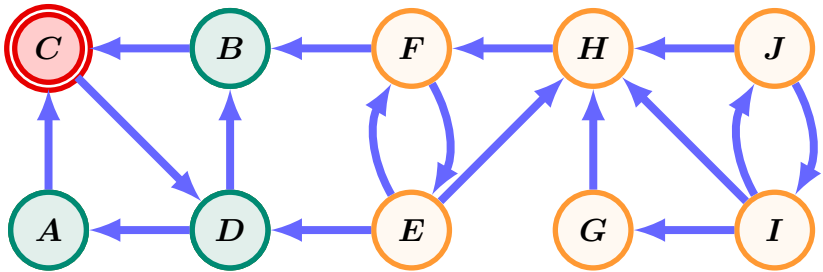
Strongly Connected Components



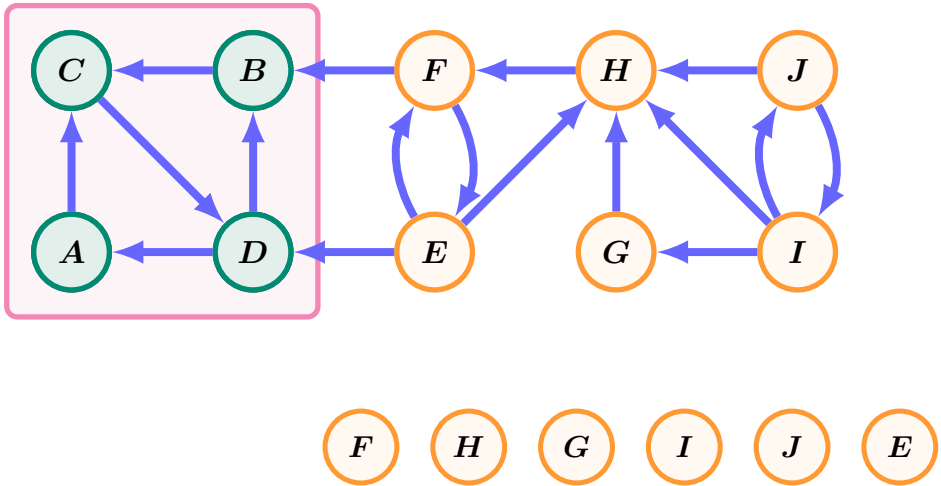
Strongly Connected Components



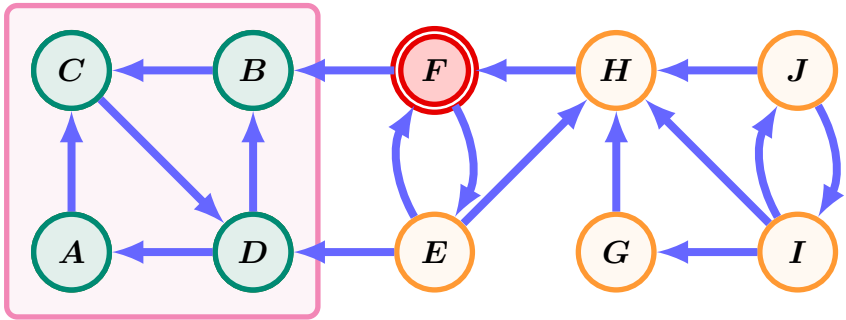
Strongly Connected Components



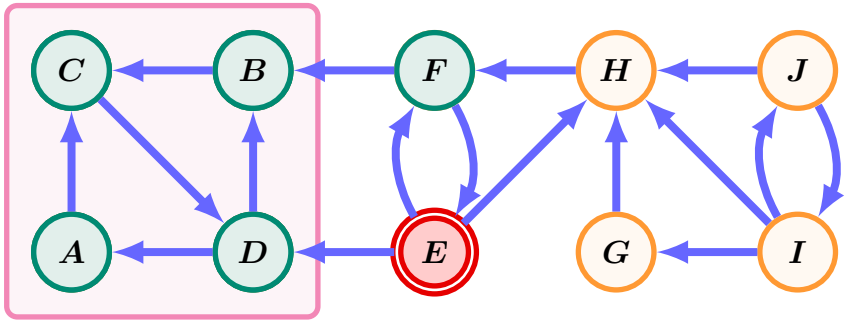
Strongly Connected Components



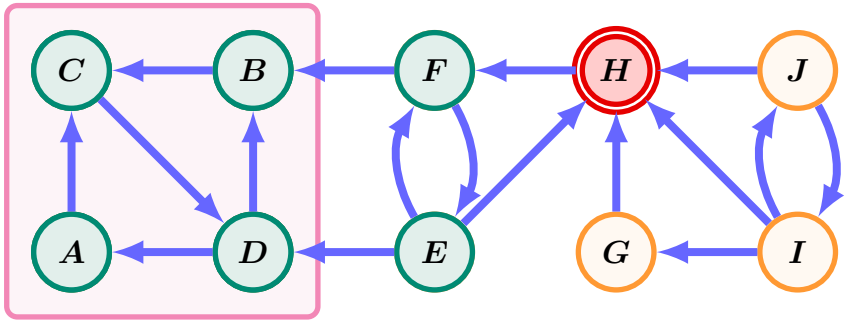
Strongly Connected Components



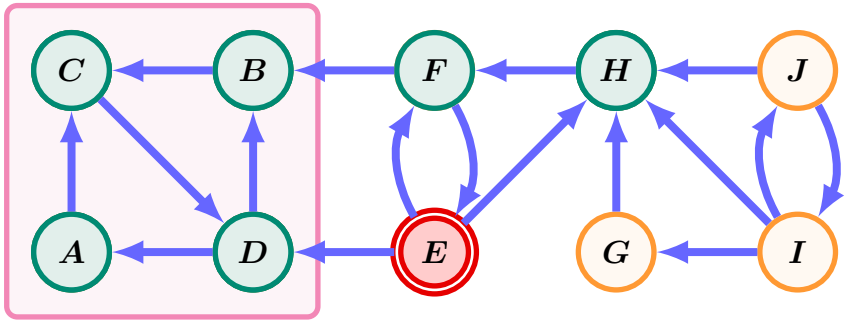
Strongly Connected Components



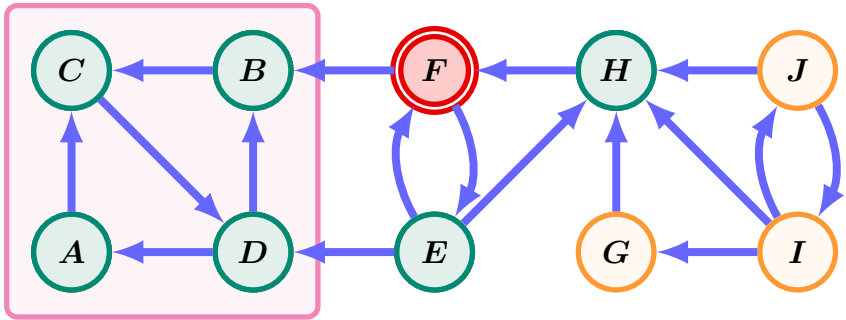
Strongly Connected Components



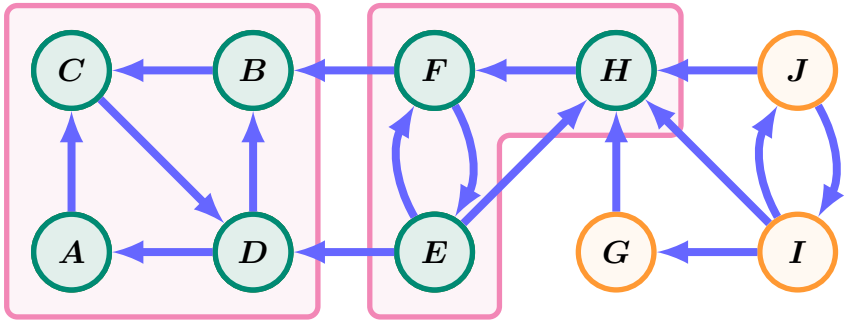
Strongly Connected Components



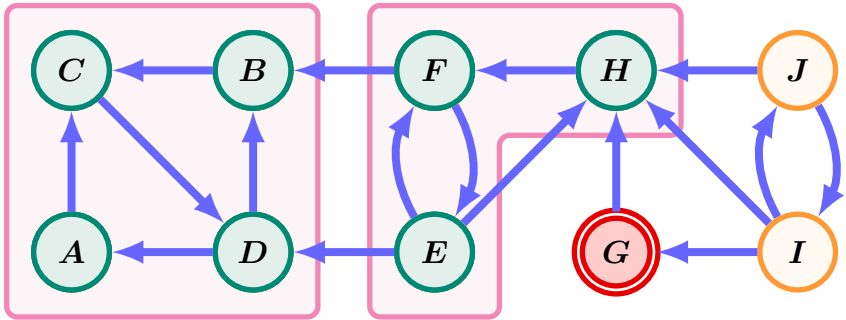
Strongly Connected Components



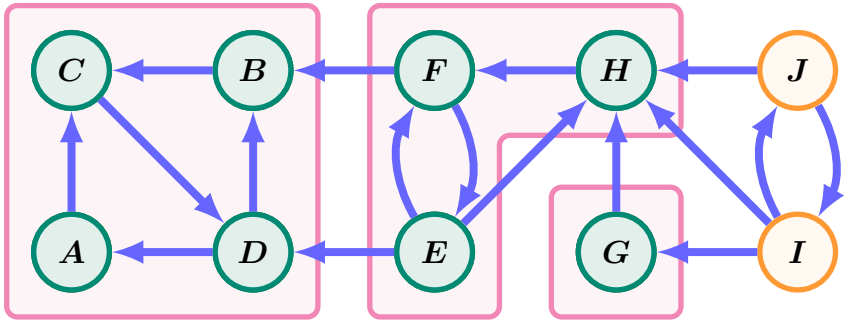
Strongly Connected Components



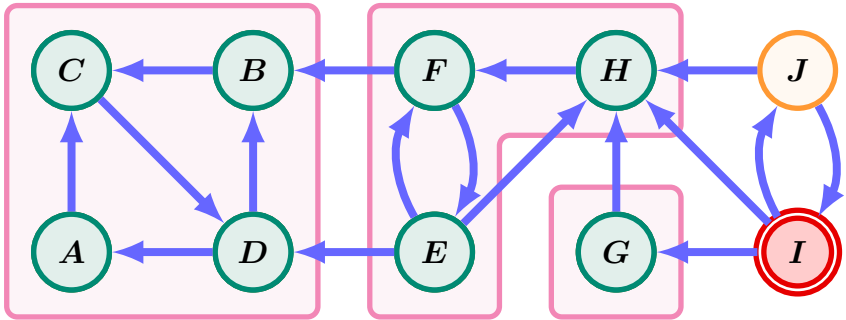
Strongly Connected Components



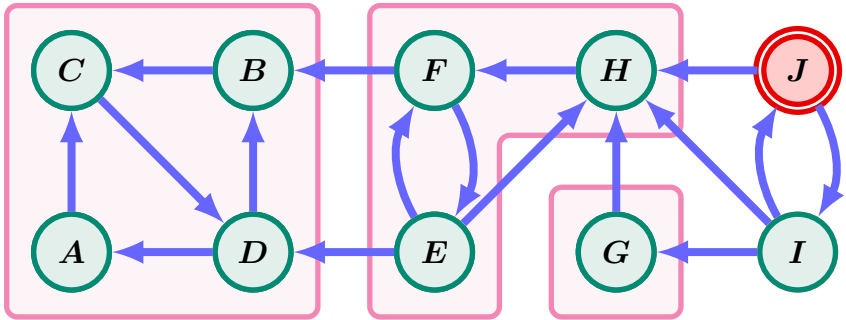
Strongly Connected Components



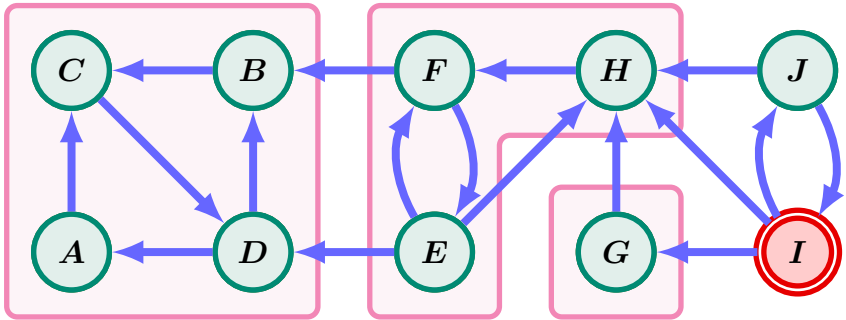
Strongly Connected Components



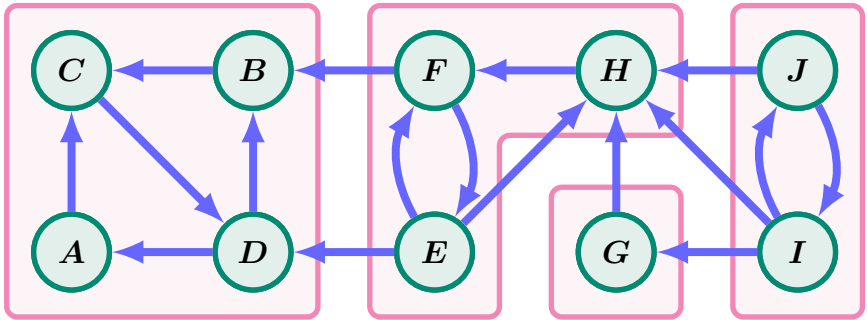
Strongly Connected Components



Strongly Connected Components



Strongly Connected Components



End of Slides!