Algorithms

Lecture Topic: Topological Sort

Anxiao (Andrew) Jiang

Roadmap of this lecture:

1. Application of DFS: Topological Sort.

    1.1 Define "Topological Sort".

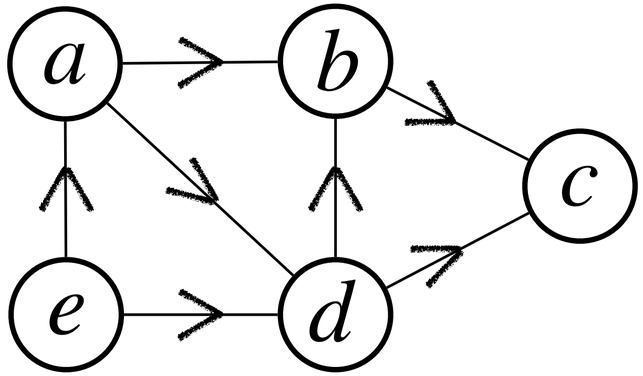
    1.2 Algorithm for topological sort.

Topological Sort

Input: A directed acyclic graph G=(V,E).

Output: Sort vertices of G linearly such that all the edges are from left to right.

Example:

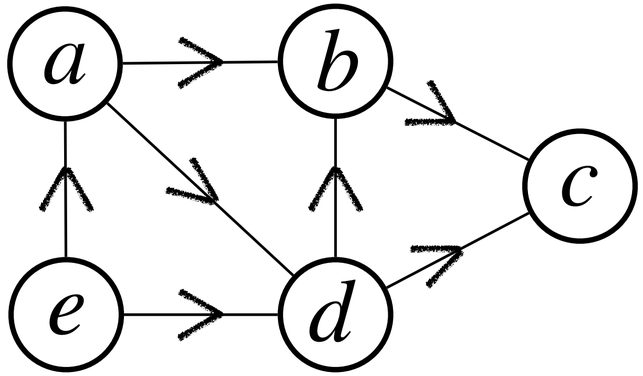


Topological Sort

Input: A directed acyclic graph G=(V,E).

Output: Sort vertices of G linearly such that all the edges are from left to right.

Example:



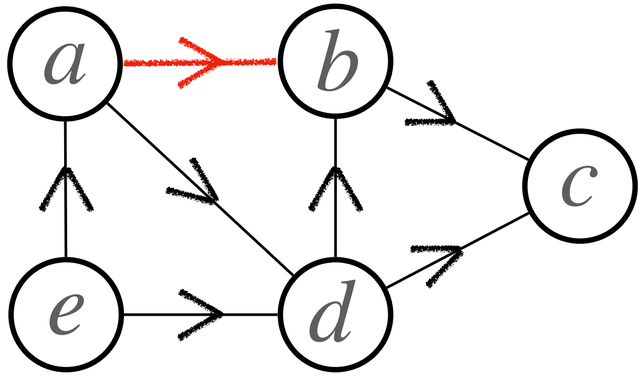
Topological Sort:

Topological Sort

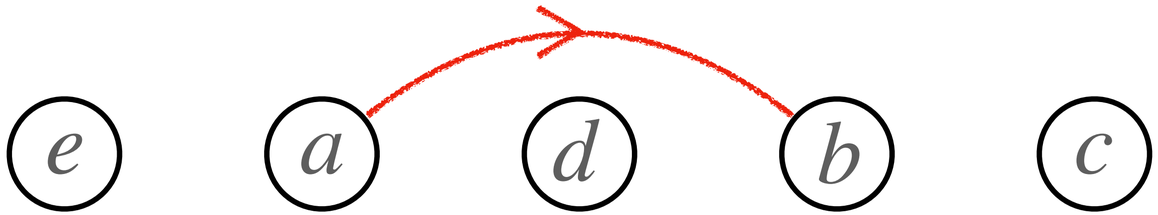
Input: A directed acyclic graph G=(V,E).

Output: Sort vertices of G linearly such that all the edges are from left to right.

Example:



Topological Sort:

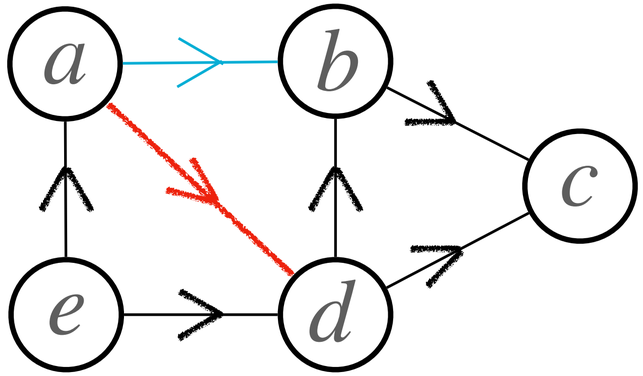


Topological Sort

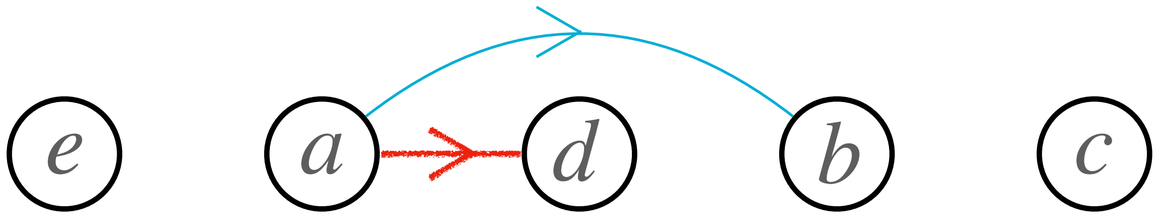
Input: A directed acyclic graph G=(V,E).

Output: Sort vertices of G linearly such that all the edges are from left to right.

Example:



Topological Sort:

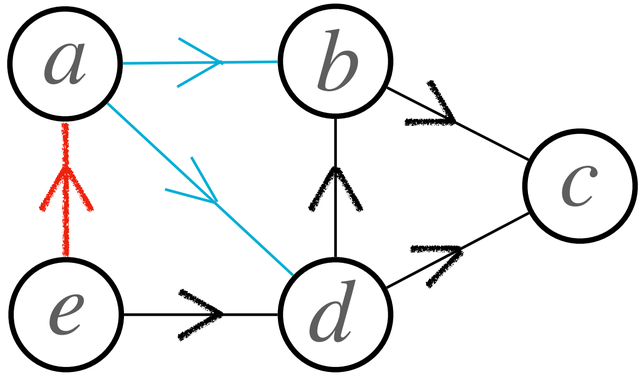


Topological Sort

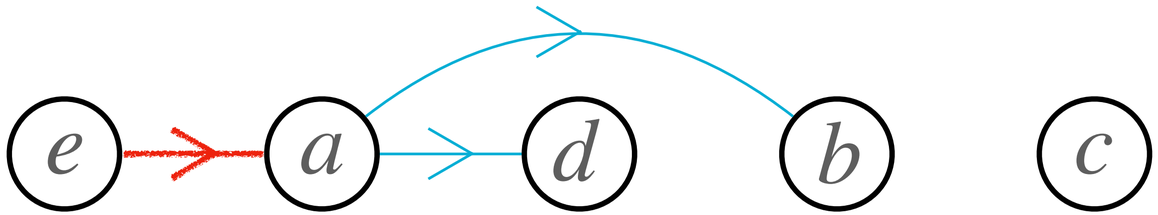
Input: A directed acyclic graph G=(V,E).

Output: Sort vertices of G linearly such that all the edges are from left to right.

Example:



Topological Sort:

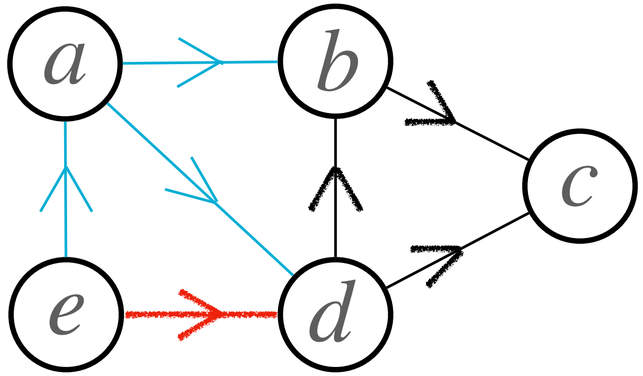


Topological Sort

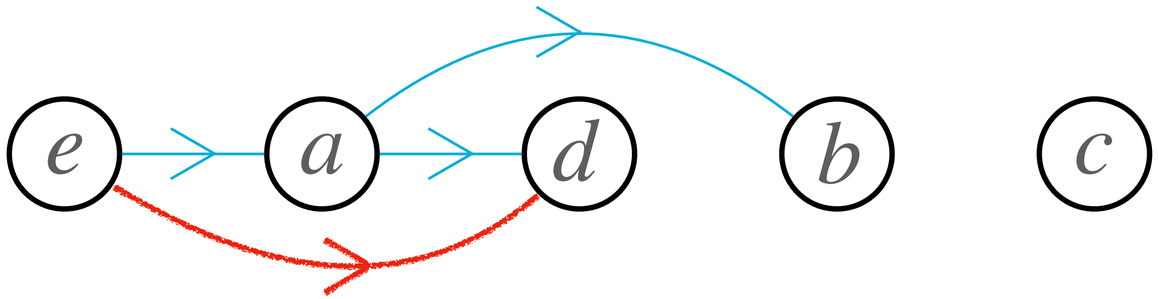
Input: A directed acyclic graph G=(V,E).

Output: Sort vertices of G linearly such that all the edges are from left to right.

Example:



Topological Sort:

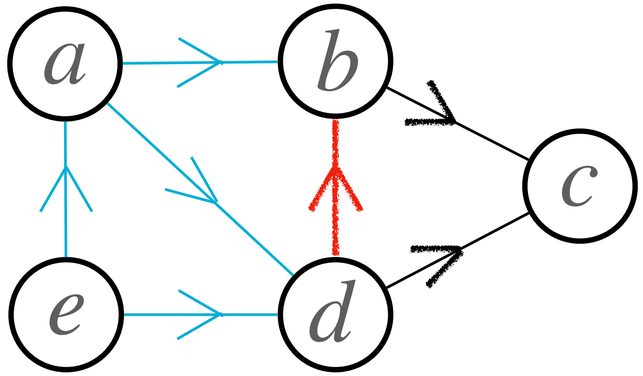


Topological Sort

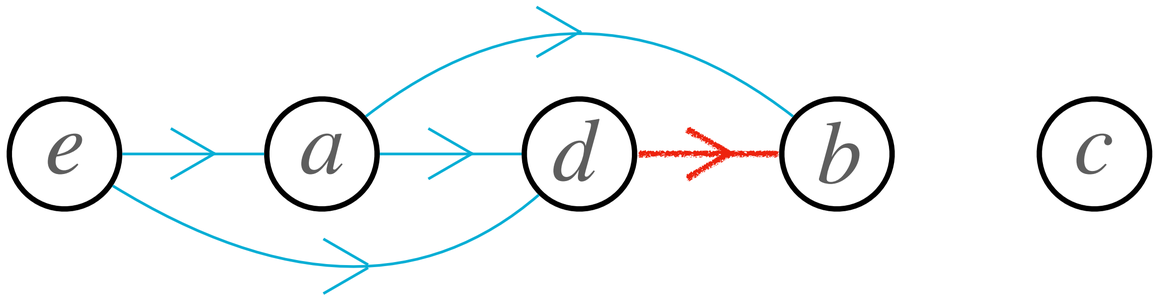
Input: A directed acyclic graph G=(V,E).

Output: Sort vertices of G linearly such that all the edges are from left to right.

Example:



Topological Sort:

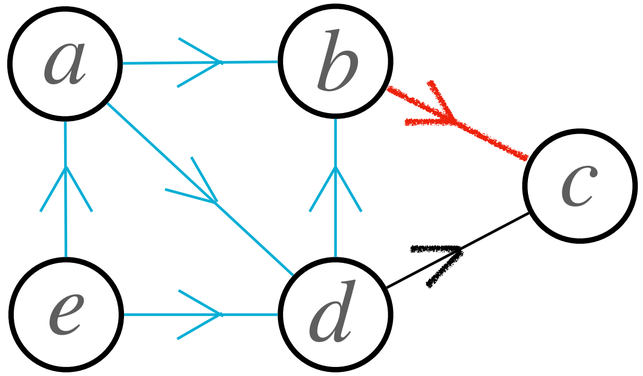


Topological Sort

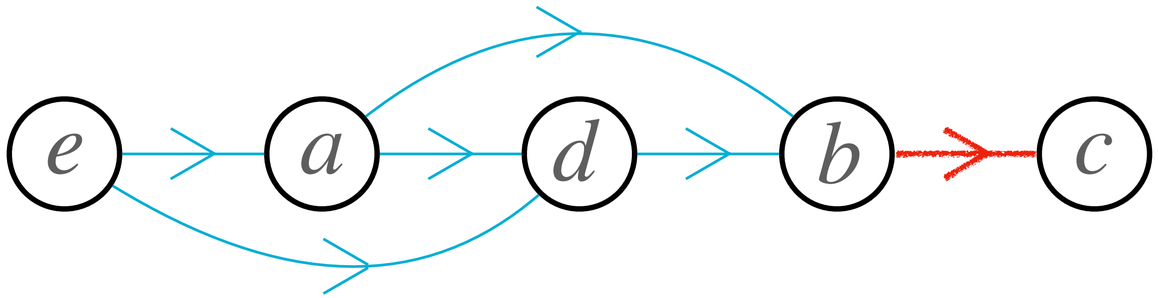
Input: A directed acyclic graph G=(V,E).

Output: Sort vertices of G linearly such that all the edges are from left to right.

Example:



Topological Sort:



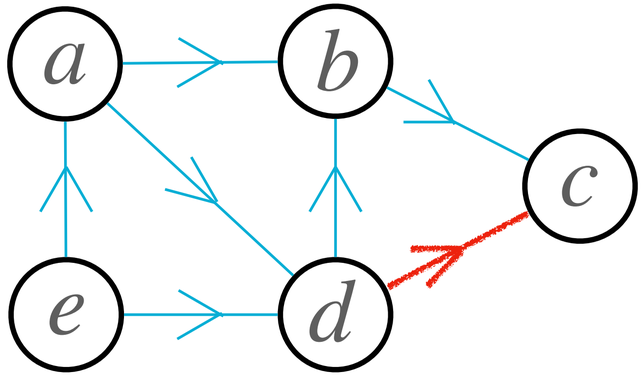
Topological Sort

DAG

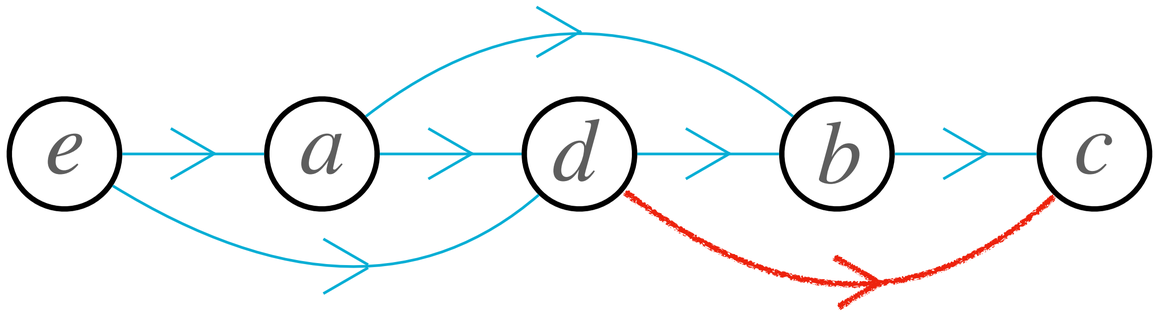
Input: A directed acyclic graph G=(V,E).

Output: Sort vertices of G linearly such that all the edges are from left to right.

Example:



Topological Sort:

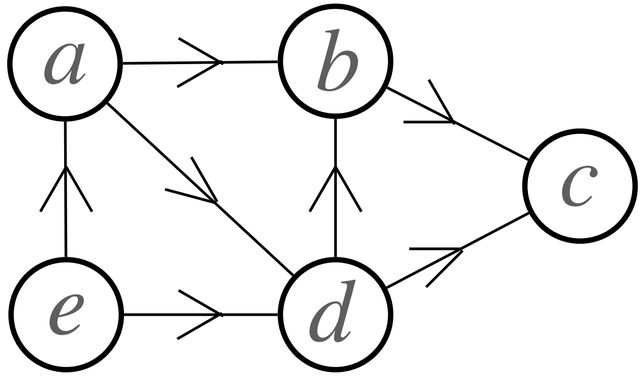


Topological Sort

Input: A directed acyclic graph G=(V,E).

Output: Sort vertices of G linearly such that all the edges are from left to right.

Example:



Topological Sort:

Why topological sort must exist for DAG?

Sink: a node without outgoing edges.

A DAG must have a sink.

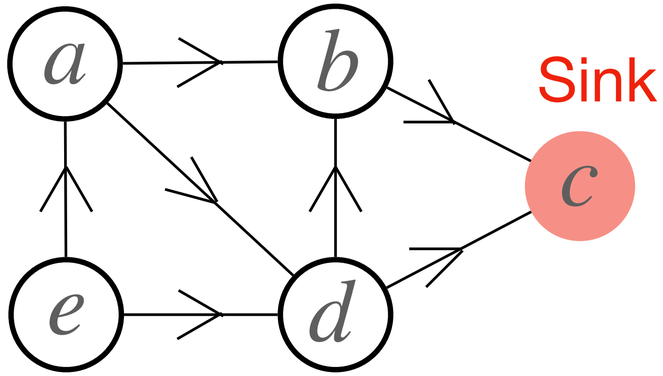
Topological Sort

Input: A directed acyclic graph G=(V,E).

Output: Sort vertices of G linearly such that all the edges are from left to right.

Topological Sort:

Example:



Why topological sort must exist for DAG?

Sink: a node without outgoing edges.

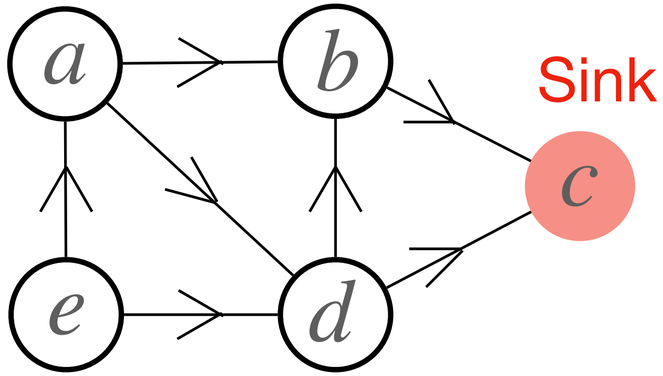
A DAG must have a sink.

Topological Sort

Input: A directed acyclic graph G=(V,E).

Output: Sort vertices of G linearly such that all the edges are from left to right.

Example:



Topological Sort:

Why topological sort must exist for DAG?

Sink: a node without outgoing edges.

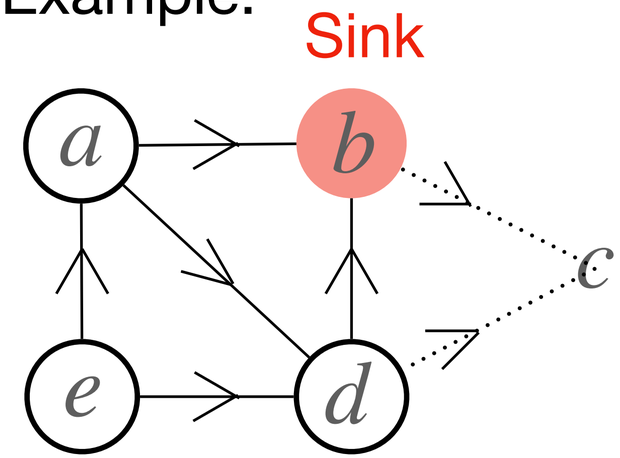
A DAG must have a sink.

Topological Sort

Input: A directed acyclic graph G=(V,E).

Output: Sort vertices of G linearly such that all the edges are from left to right.

Example:



Topological Sort:

Why topological sort must exist for DAG?

Sink: a node without outgoing edges.

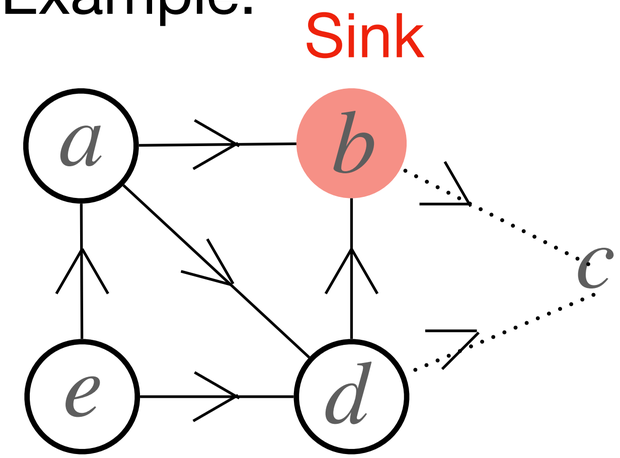
A DAG must have a sink.

Topological Sort

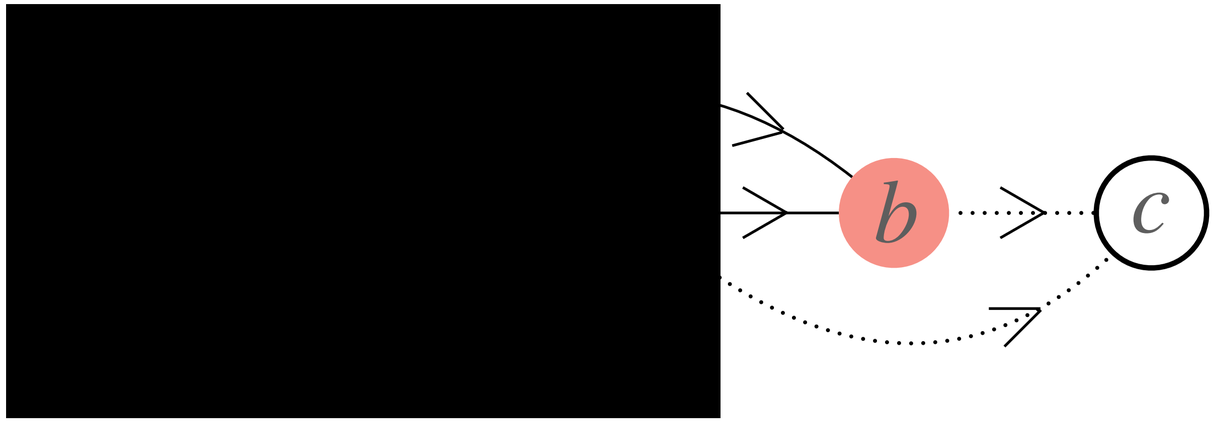
Input: A directed acyclic graph G=(V,E).

Output: Sort vertices of G linearly such that all the edges are from left to right.

Example:



Topological Sort:



Why topological sort must exist for DAG?

Sink: a node without outgoing edges.

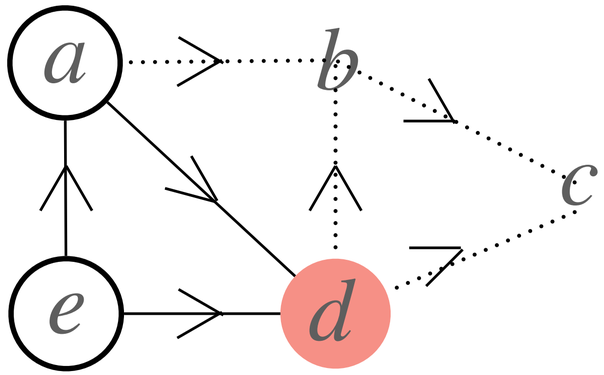
A DAG must have a sink.

Topological Sort

Input: A directed acyclic graph G=(V,E).

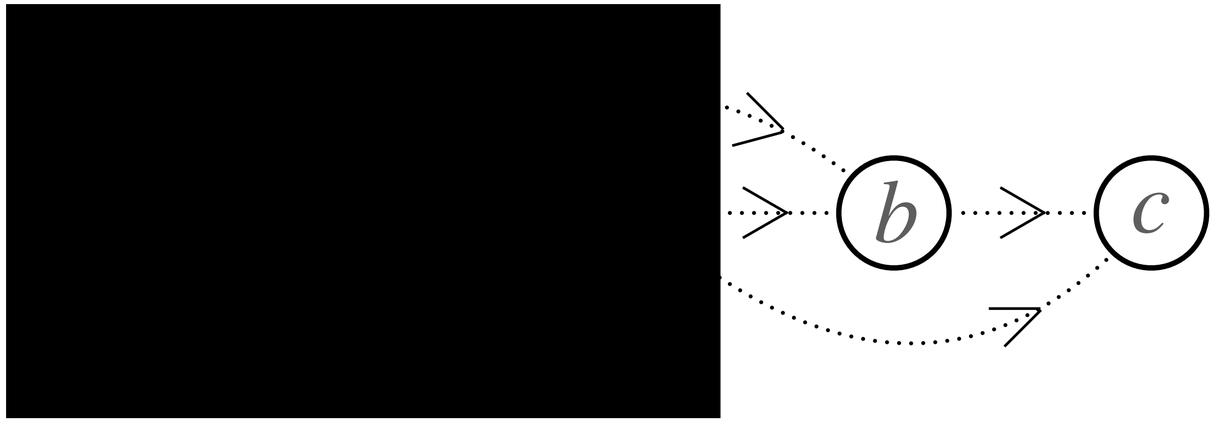
Output: Sort vertices of G linearly such that all the edges are from left to right.

Example:



Sink

Topological Sort:



Why topological sort must exist for DAG?

Sink: a node without outgoing edges.

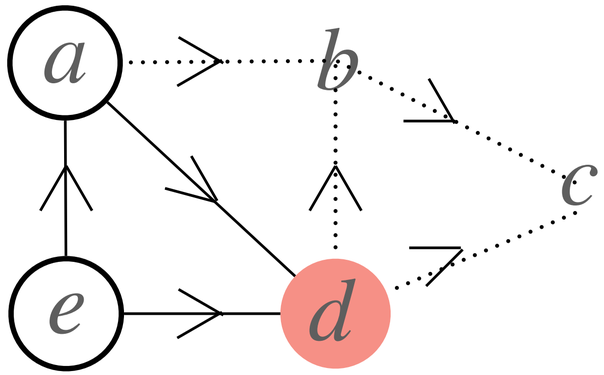
A DAG must have a sink.

Topological Sort

Input: A directed acyclic graph G=(V,E).

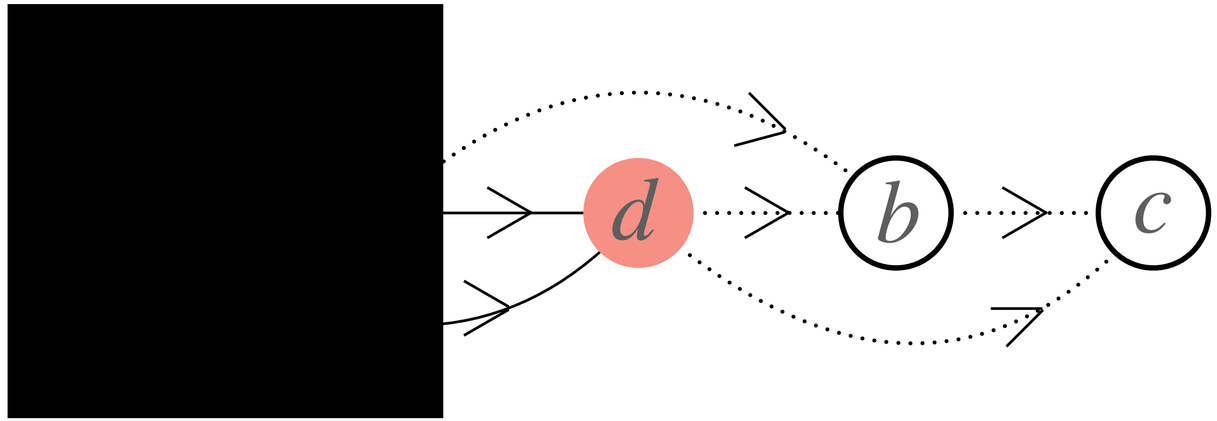
Output: Sort vertices of G linearly such that all the edges are from left to right.

Example:



Sink

Topological Sort:



Why topological sort must exist for DAG?

Sink: a node without outgoing edges.

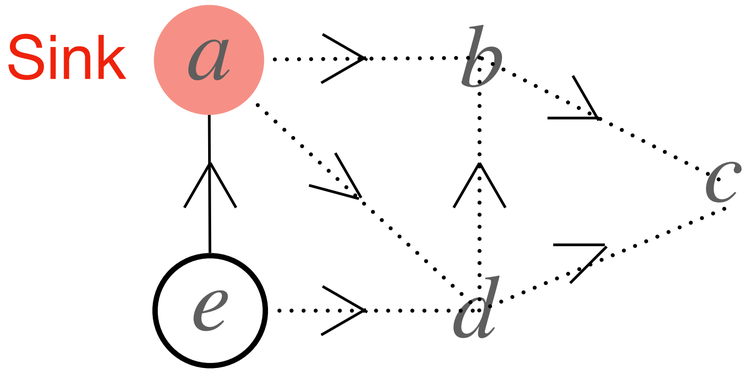
A DAG must have a sink.

Topological Sort

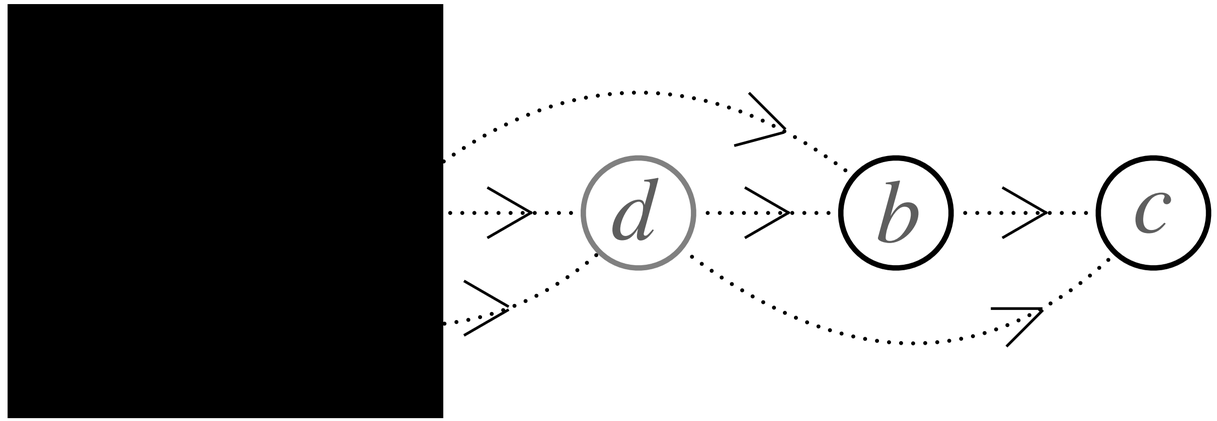
Input: A directed acyclic graph G=(V,E).

Output: Sort vertices of G linearly such that all the edges are from left to right.

Example:



Topological Sort:



Why topological sort must exist for DAG?

Sink: a node without outgoing edges.

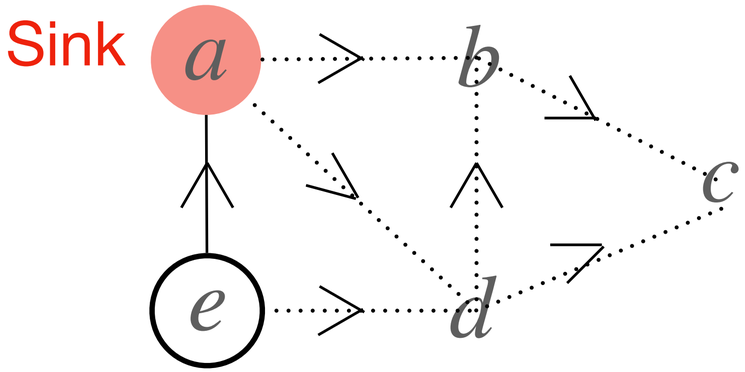
A DAG must have a sink.

Topological Sort

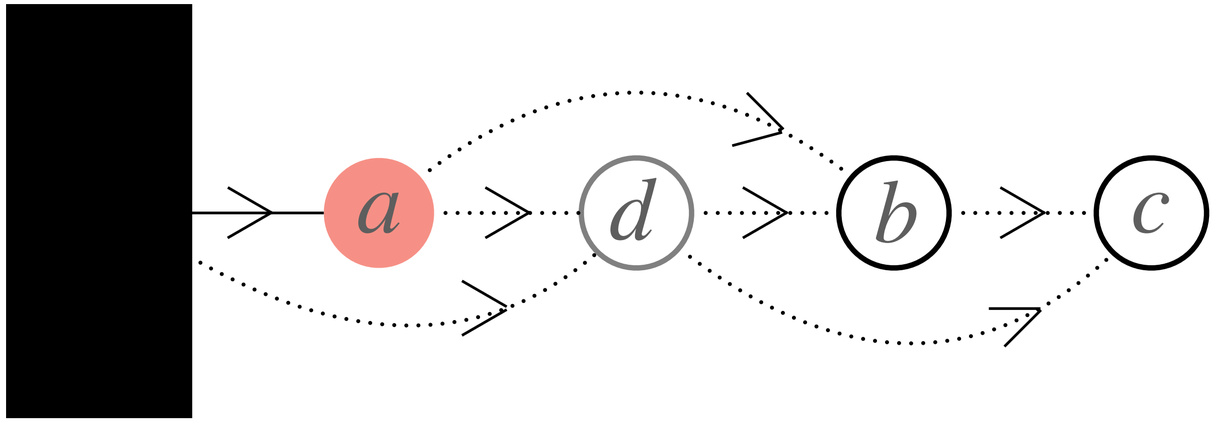
Input: A directed acyclic graph G=(V,E).

Output: Sort vertices of G linearly such that all the edges are from left to right.

Example:



Topological Sort:



Why topological sort must exist for DAG?

Sink: a node without outgoing edges.

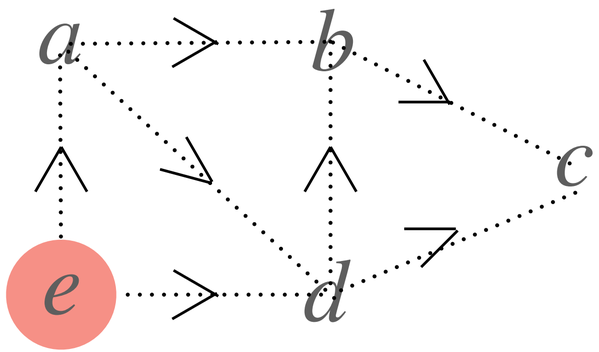
A DAG must have a sink.

Topological Sort

Input: A directed acyclic graph G=(V,E).

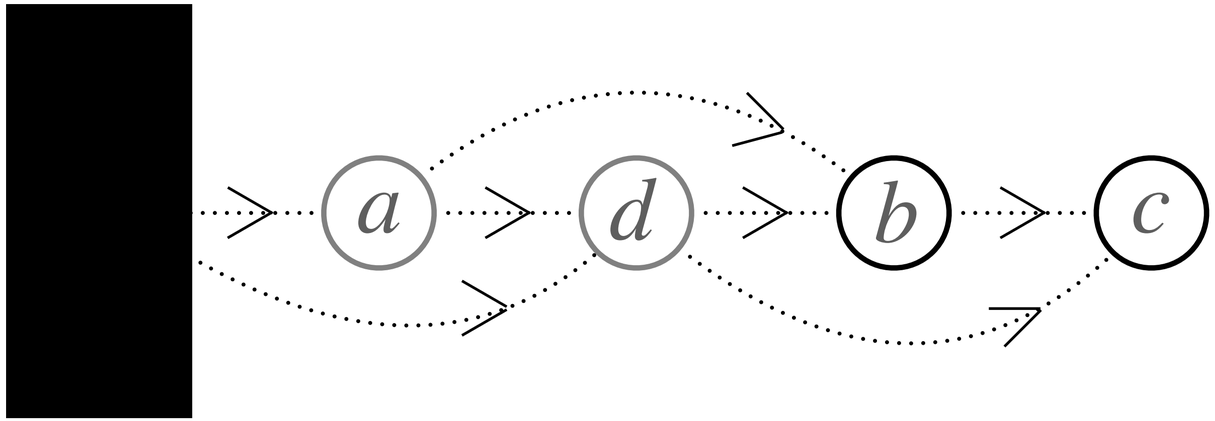
Output: Sort vertices of G linearly such that all the edges are from left to right.

Example:



Sink

Topological Sort:



Why topological sort must exist for DAG?

Sink: a node without outgoing edges.

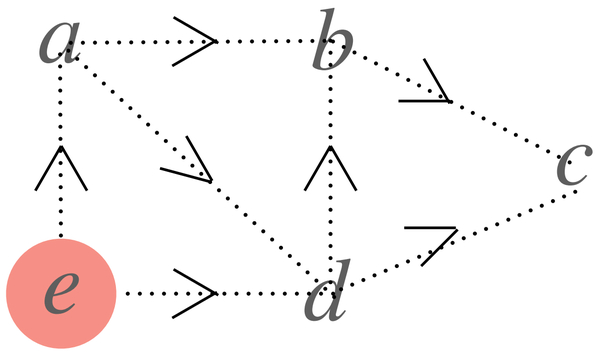
A DAG must have a sink.

Topological Sort

Input: A directed acyclic graph G=(V,E).

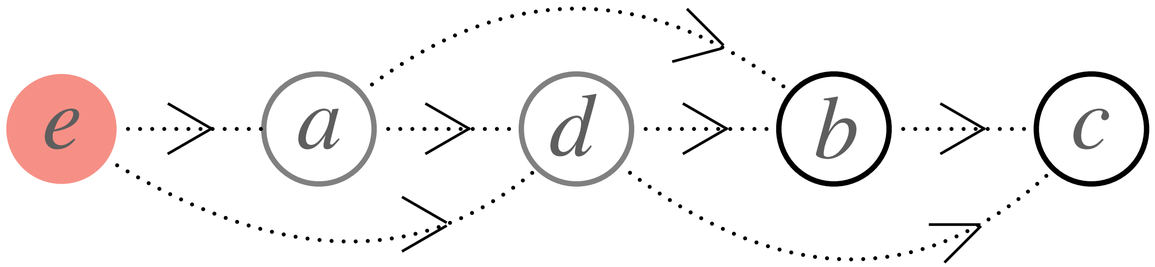
Output: Sort vertices of G linearly such that all the edges are from left to right.

Example:



Sink

Topological Sort:



Why topological sort must exist for DAG?

Sink: a node without outgoing edges.

A DAG must have a sink.

Topological Sort

We actually have an algorithm for “Topological Sort” now!

But is it efficient?

How to get an efficient algorithm of time complexity $O(V + E)$?

Quiz questions:

1. Why does topological sort exist only for acyclic graphs?

2. What are the applications of topological sort?

Roadmap of this lecture:

1. Application of DFS: Topological Sort.

    1.1 Define "Topological Sort".

    1.2 Algorithm for topological sort.

Topological Sort

Theorem: Let G=(V,E) be a directed acyclic graph. When we run DFS on G, for any edge $(u, v) \in E$, we must have $f(u) > f(v)$.

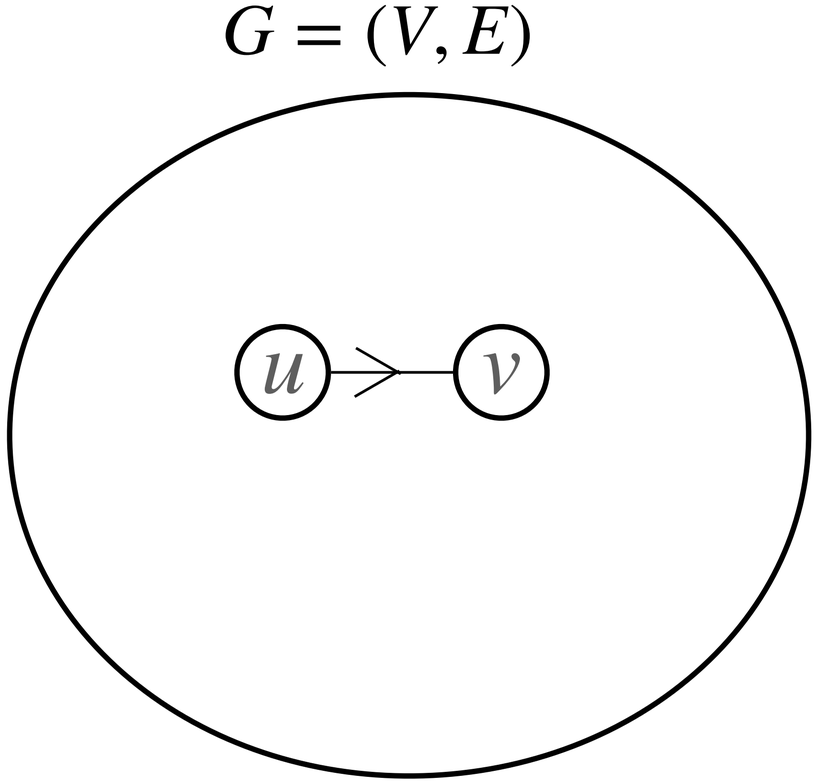
Topological Sort

Theorem: Let G=(V,E) be a directed acyclic graph. When we run DFS on G, for any edge $(u, v) \in E$, we must have $f(u) > f(v)$. Finish time

Topological Sort

Theorem: Let G=(V,E) be a directed acyclic graph. When we run DFS on G, for any edge $(u, v) \in E$, we must have $f(u) > f(v)$. Finish time

Proof:



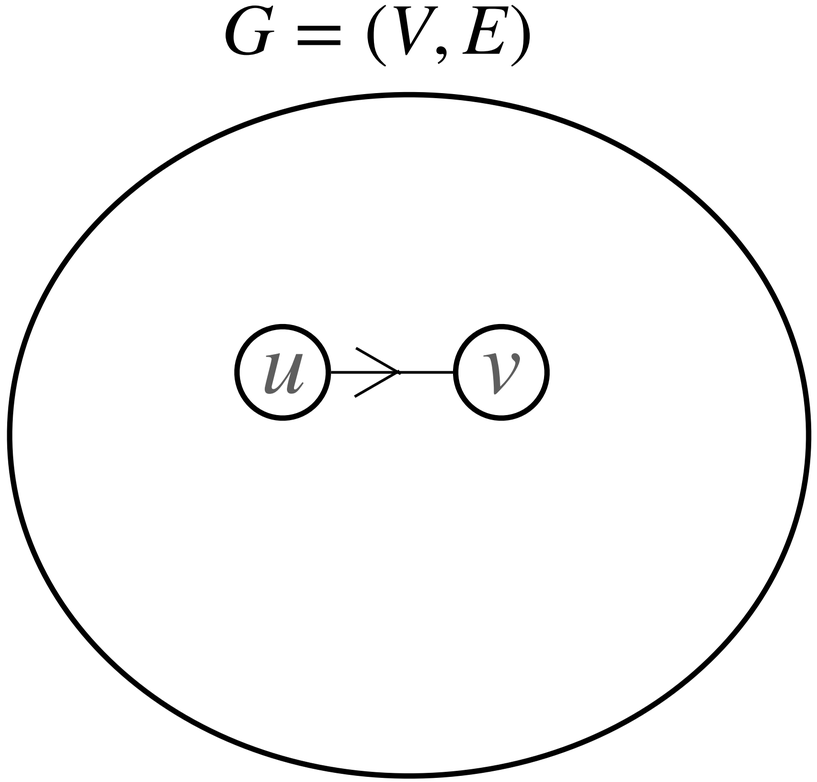
Case 1: $d(u) < d(v)$.

Case 2: $d(u) > d(v)$.

Topological Sort

Theorem: Let G=(V,E) be a directed acyclic graph. When we run DFS on G, for any edge $(u, v) \in E$, we must have $f(u) > f(v)$. Finish time

Proof:



Case 1: $d(u) < d(v)$.

u : ancestor

v : descendant

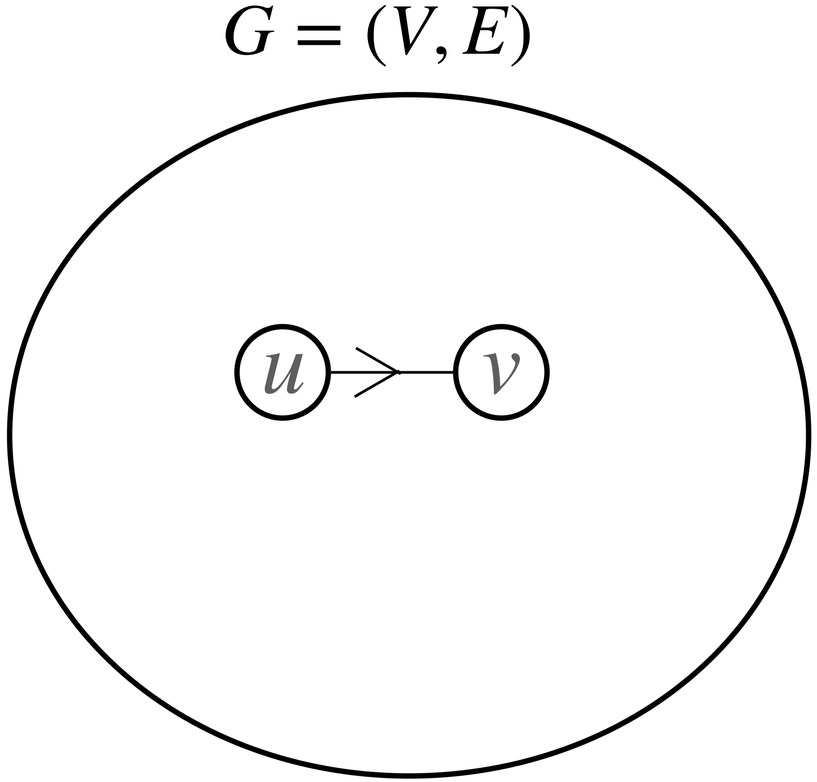
→ $f(u) > f(v)$.

Case 2: $d(u) > d(v)$.

Topological Sort

Theorem: Let G=(V,E) be a directed acyclic graph. When we run DFS on G, for any edge $(u, v) \in E$, we must have $f(u) > f(v)$. Finish time

Proof:



Case 1: $d(u) < d(v)$.

u : ancestor
v : descendant

→ $f(u) > f(v)$.

Case 2: $d(u) > d(v)$.

G is acyclic → v cannot reach u.

v will finish before DFS discovers u.

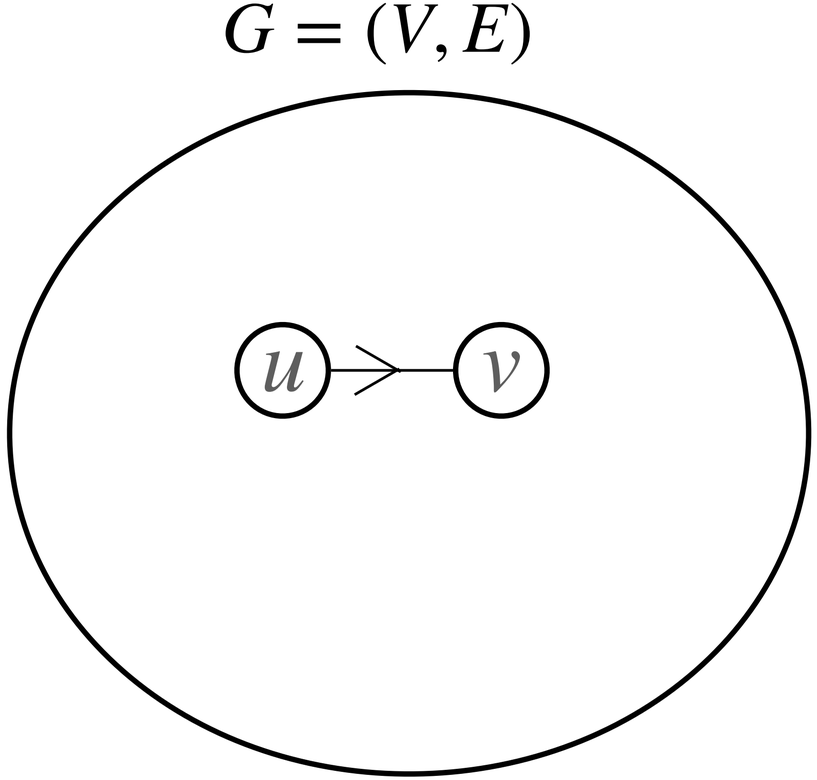
$f(v) < d(u) < f(u)$.

Topological Sort

We can sort edges by their finish time (which is a linear ordering)!

Theorem: Let G=(V,E) be a directed acyclic graph. When we run DFS on G, for any edge $(u, v) \in E$, we must have $f(u) > f(v)$. Finish time

Proof:



Case 1: $d(u) < d(v)$.

u : ancestor
v : descendant

→ $f(u) > f(v)$.

Case 2: $d(u) > d(v)$.

G is acyclic → v cannot reach u.

v will finish before DFS discovers u.

$f(v) < d(u) < f(u)$.

Topological Sort

Idea of Algorithm:

1. Run DFS on graph G=(V,E).

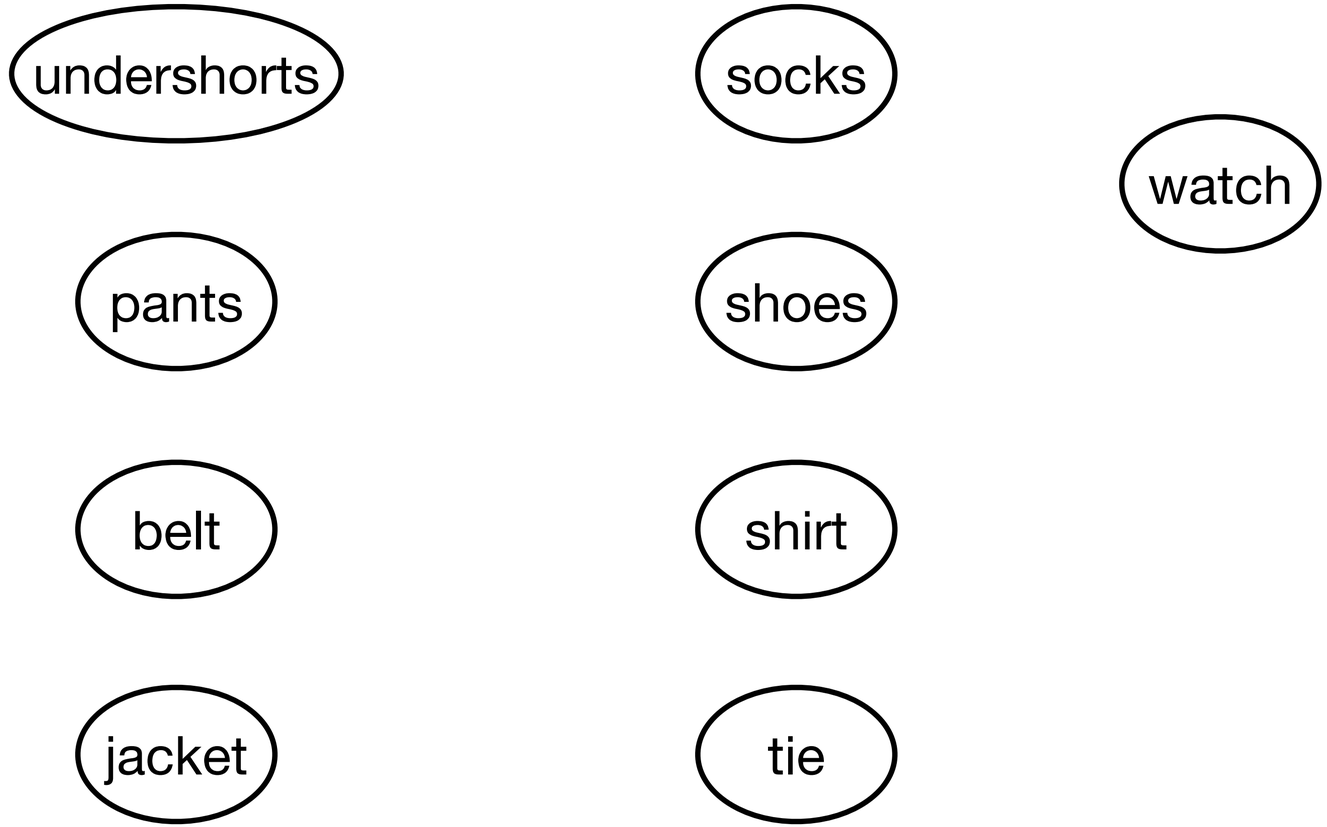
2. Every time a node is finished, put the node to the left end of a linked list.

3. Return the linked list as the topological sort.

Topological Sort

Idea of Algorithm:

1. Run DFS on graph G=(V,E).
2. Every time a node is finished, put the node to the left end of a linked list.
3. Return the linked list as the topological sort.



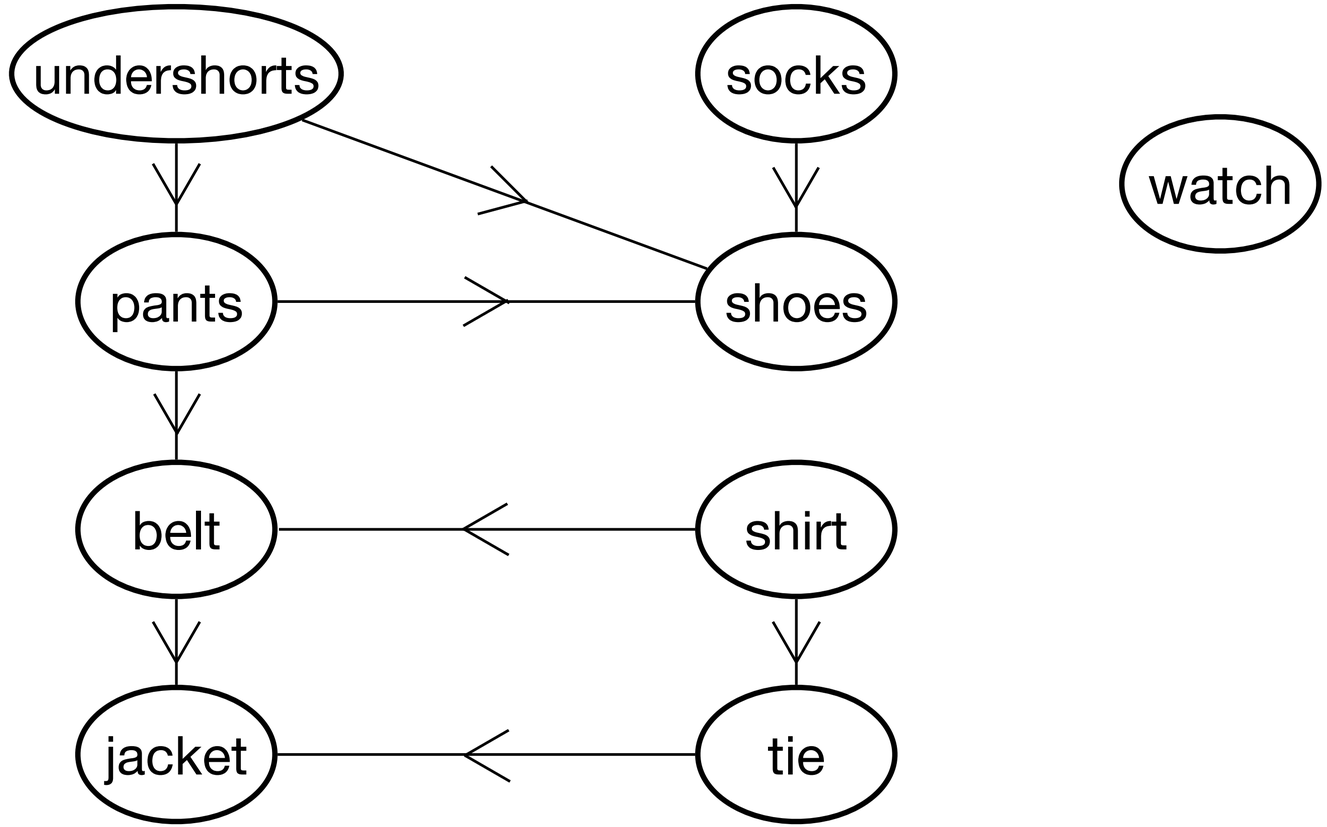
Topological Sort

Idea of Algorithm:

1. Run DFS on graph G=(V,E).

2. Every time a node is finished, put the node to the left end of a linked list.

3. Return the linked list as the topological sort.



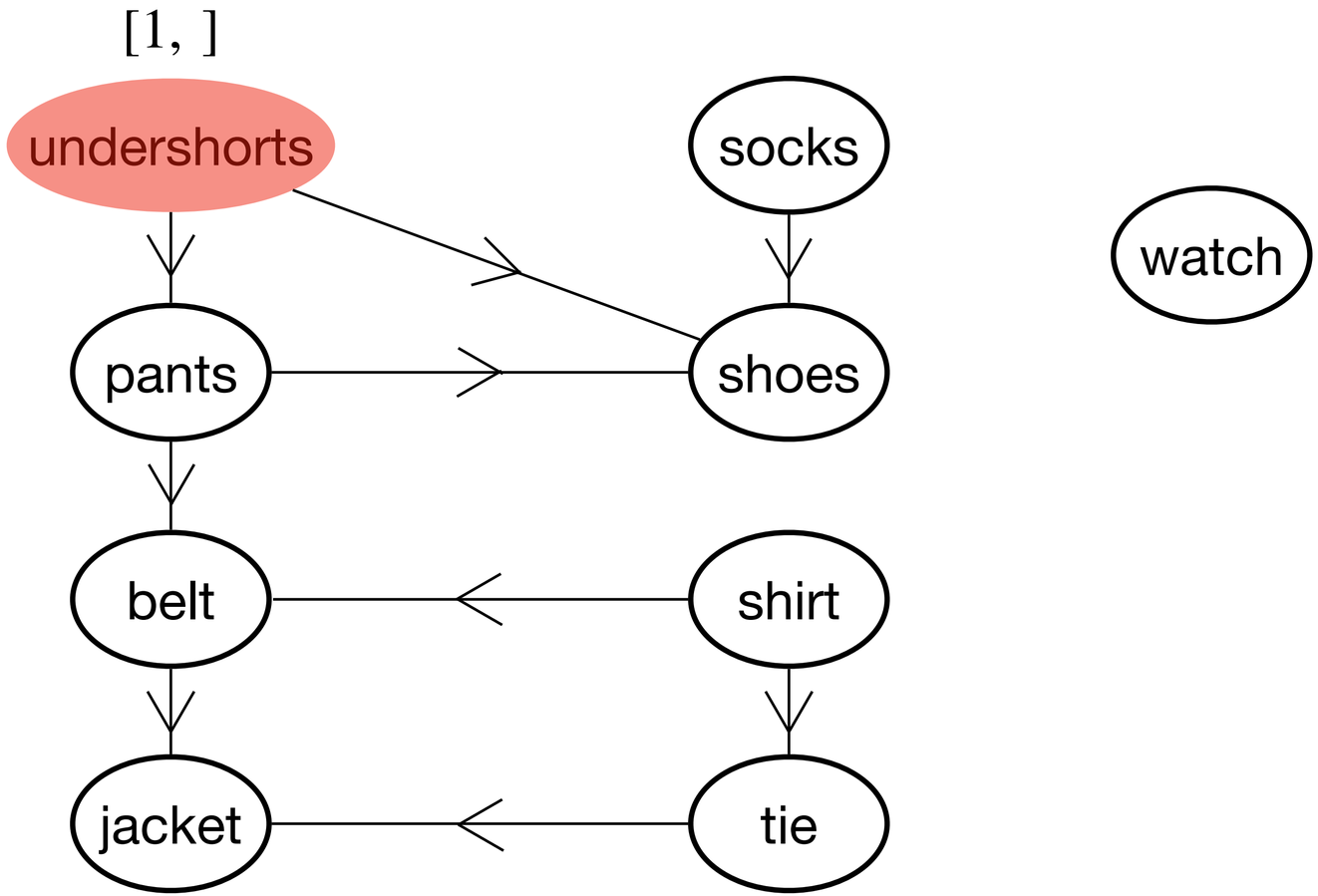
Topological Sort

Idea of Algorithm:

1. Run DFS on graph G=(V,E).

2. Every time a node is finished, put the node to the left end of a linked list.

3. Return the linked list as the topological sort.



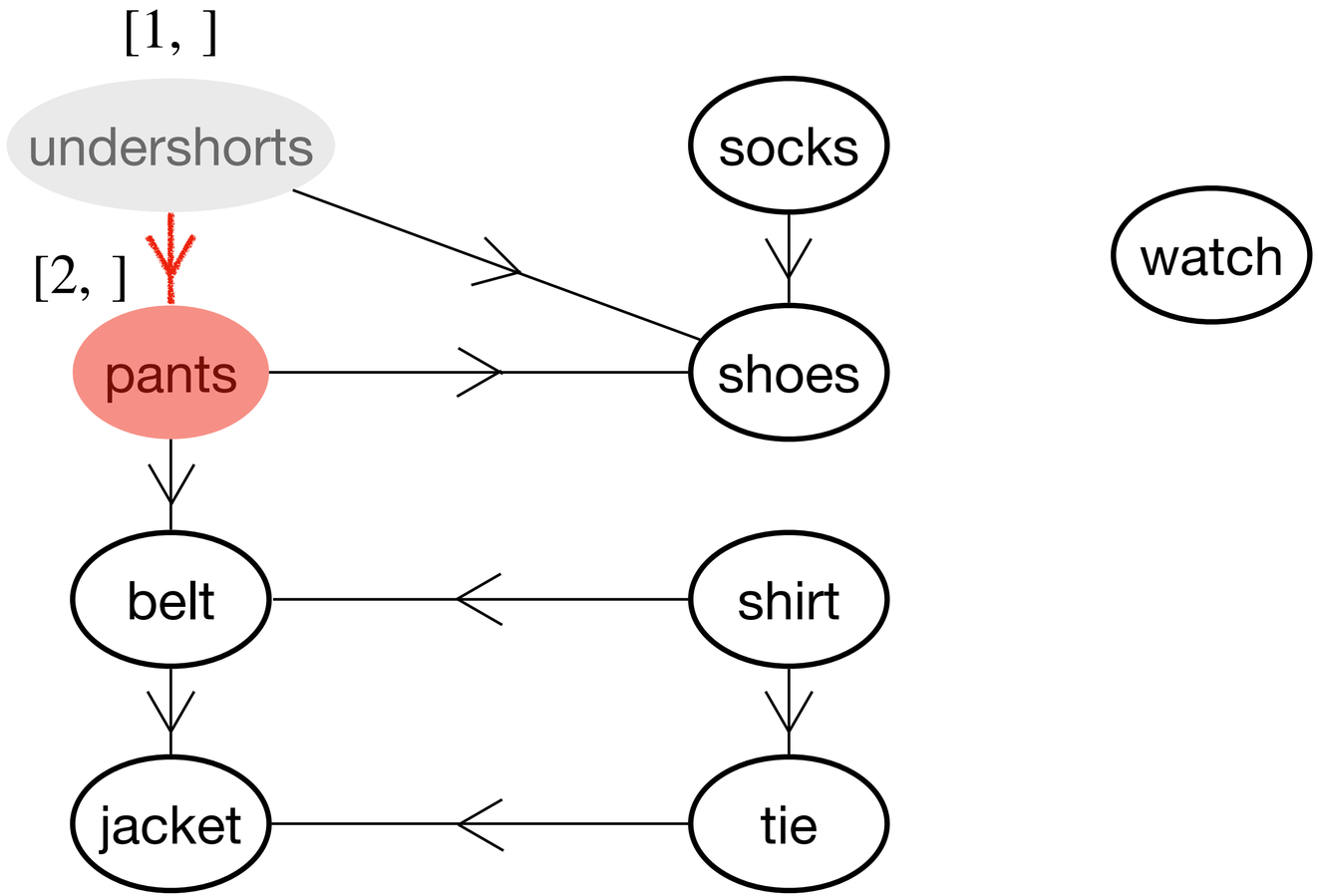
Topological Sort

Idea of Algorithm:

1. Run DFS on graph G=(V,E).

2. Every time a node is finished, put the node to the left end of a linked list.

3. Return the linked list as the topological sort.



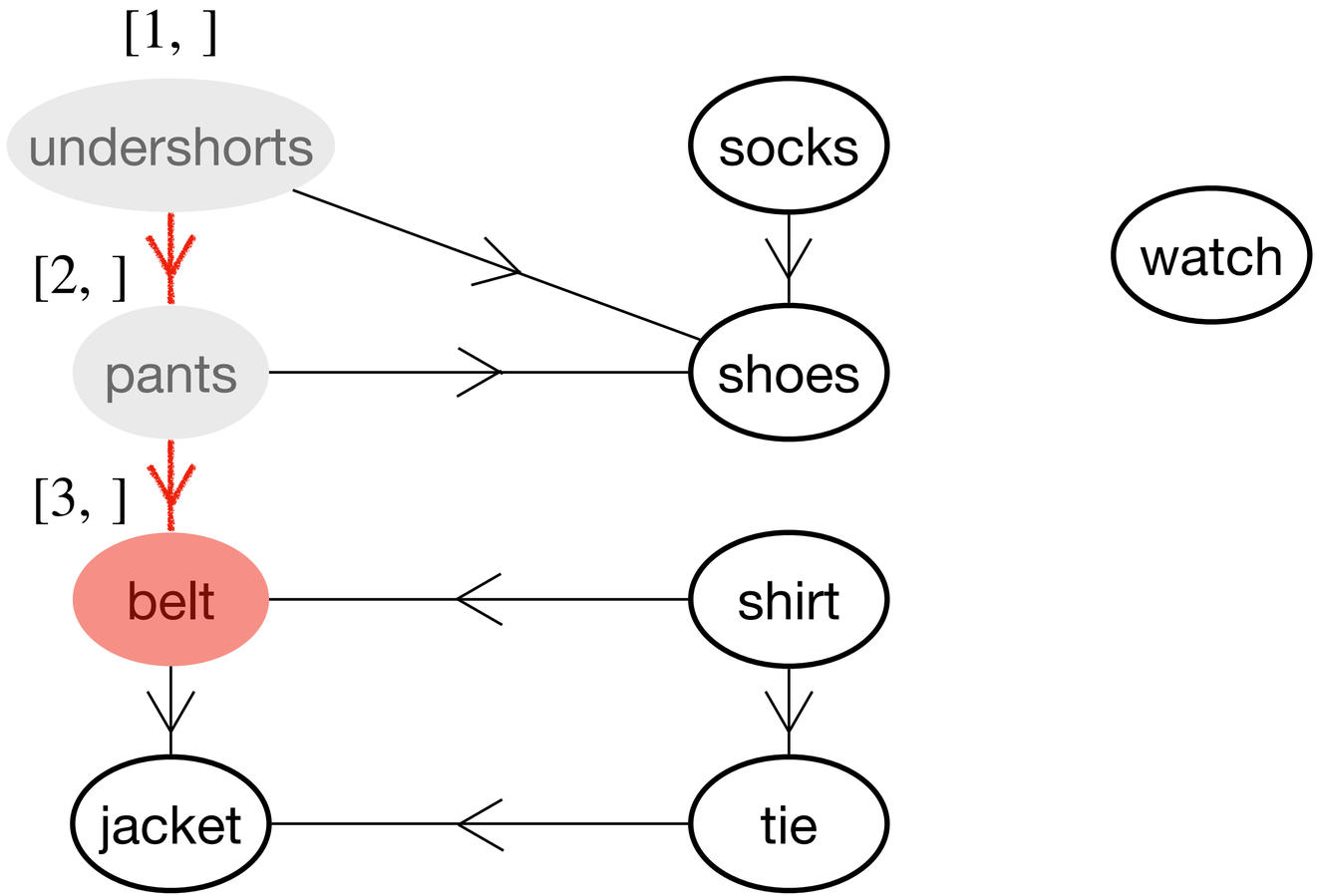
Topological Sort

Idea of Algorithm:

1. Run DFS on graph G=(V,E).

2. Every time a node is finished, put the node to the left end of a linked list.

3. Return the linked list as the topological sort.



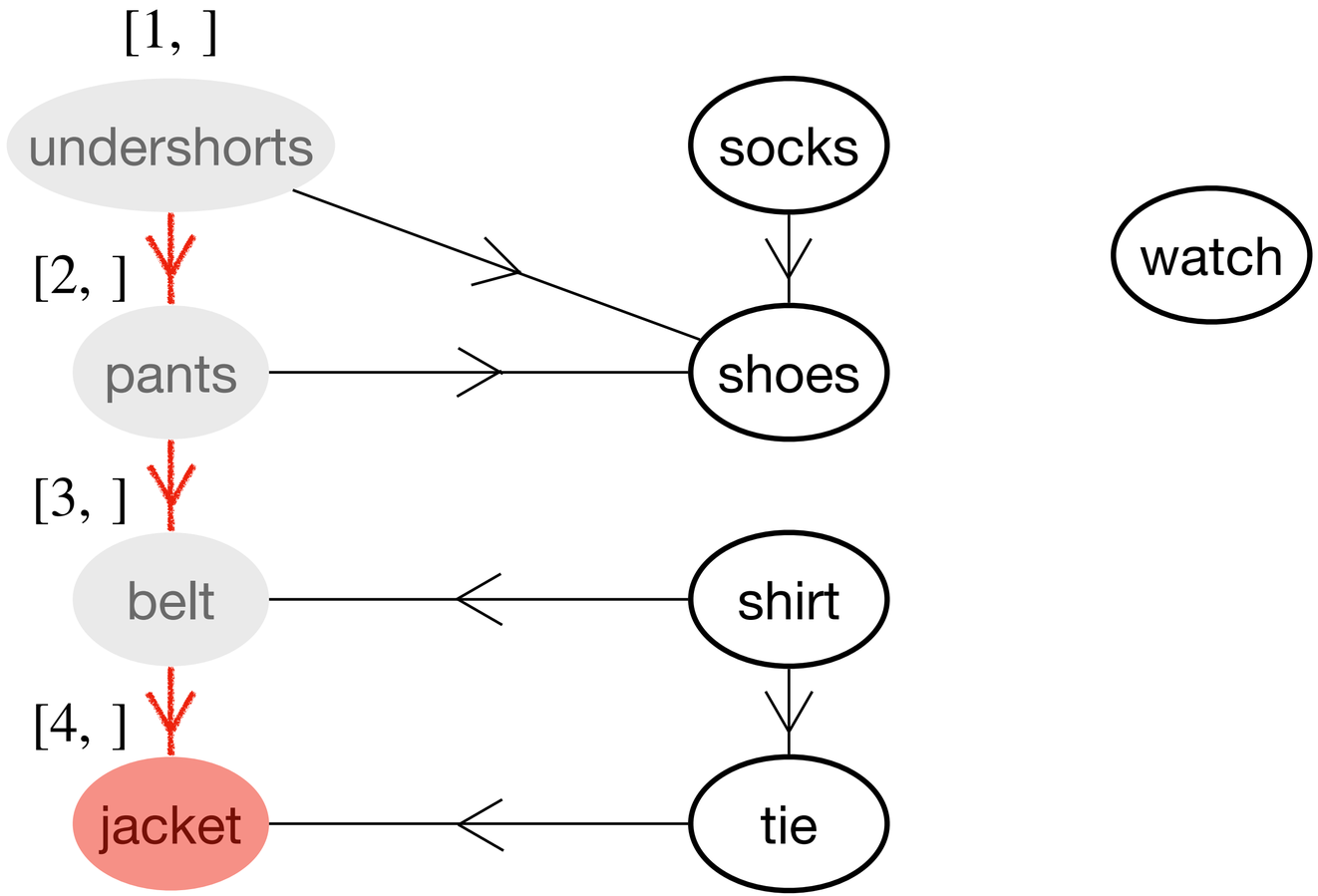
Topological Sort

Idea of Algorithm:

1. Run DFS on graph G=(V,E).

2. Every time a node is finished, put the node to the left end of a linked list.

3. Return the linked list as the topological sort.



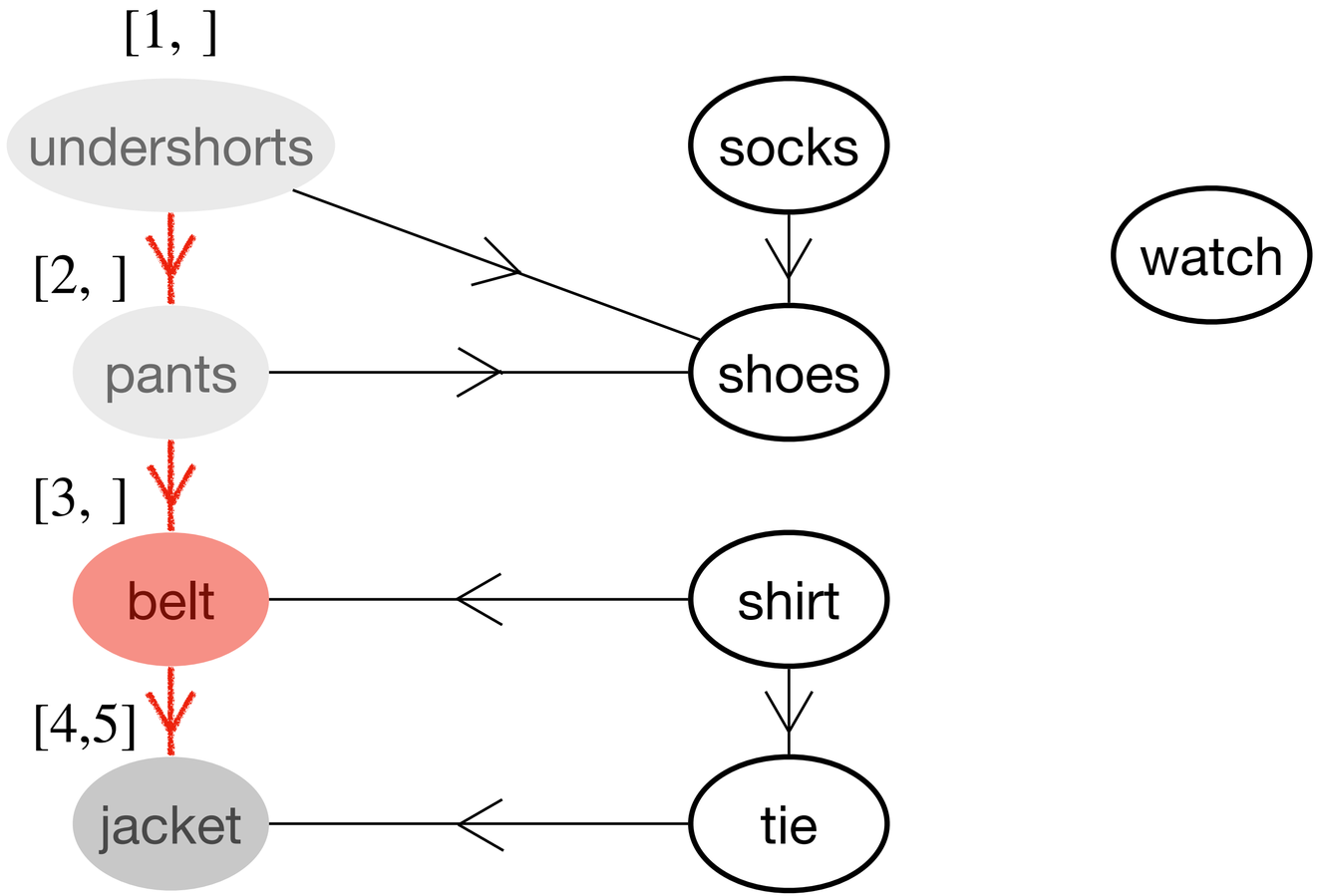
Topological Sort

Idea of Algorithm:

1. Run DFS on graph G=(V,E).

2. Every time a node is finished, put the node to the left end of a linked list.

3. Return the linked list as the topological sort.



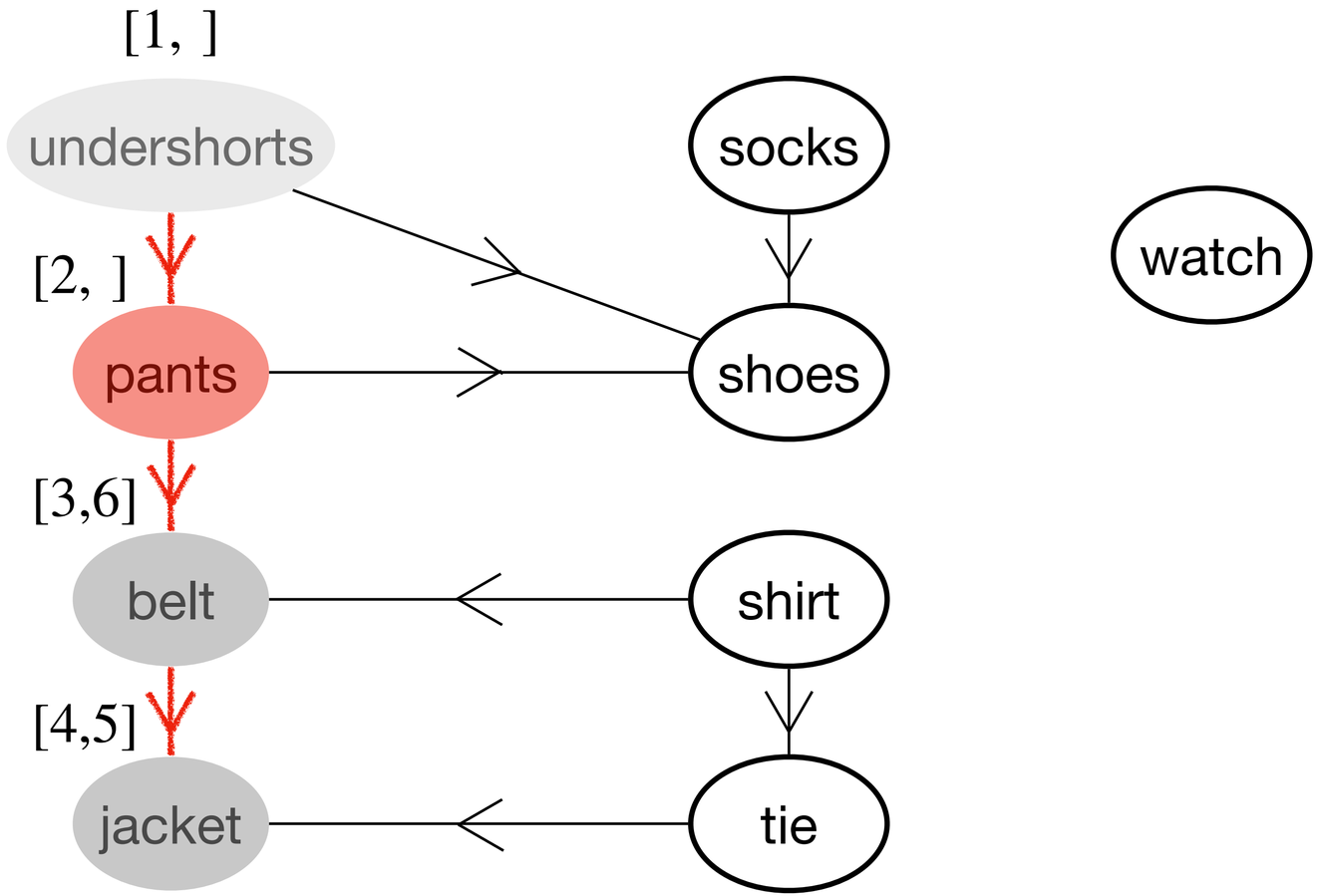
Topological Sort

Idea of Algorithm:

1. Run DFS on graph G=(V,E).

2. Every time a node is finished, put the node to the left end of a linked list.

3. Return the linked list as the topological sort.



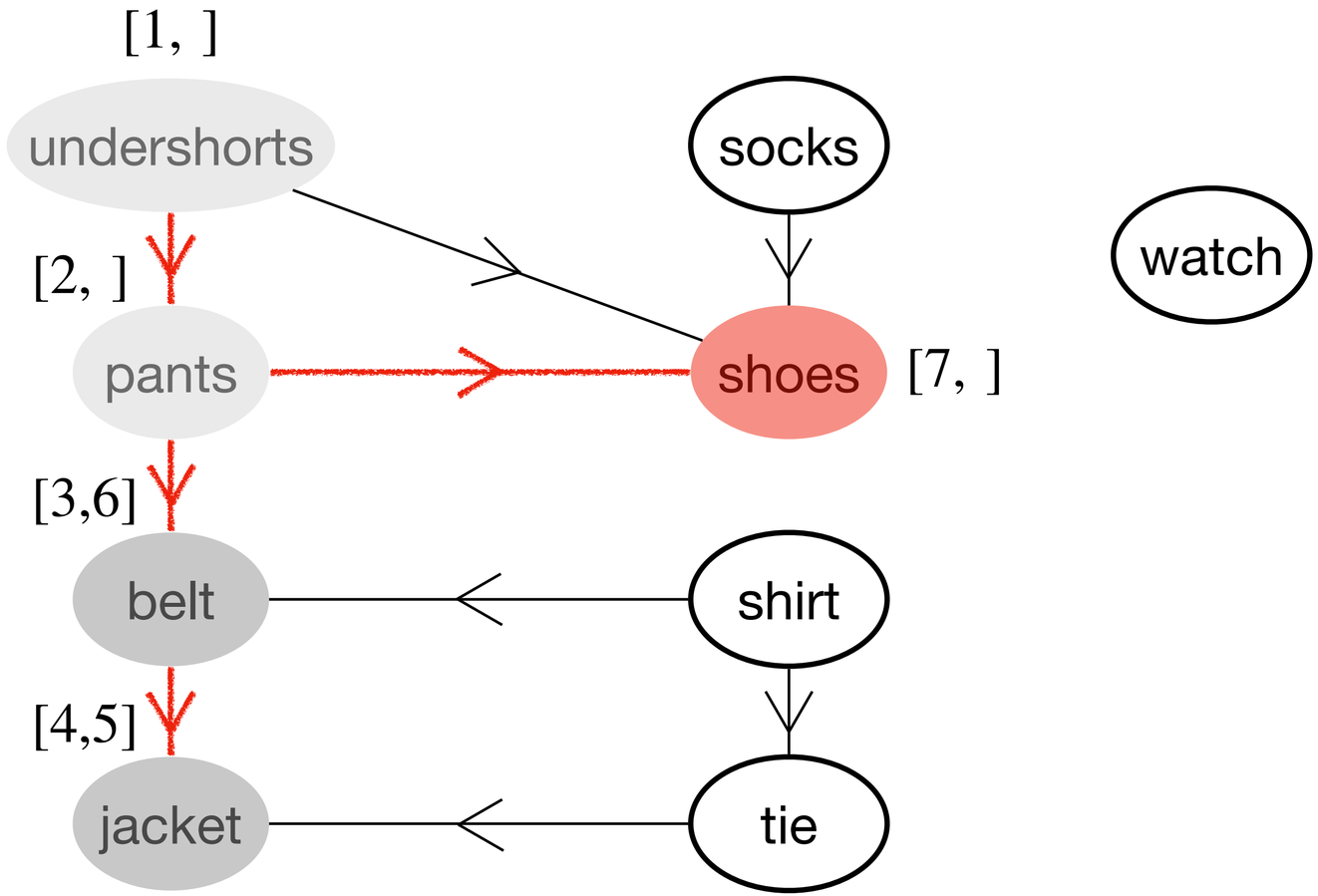
Topological Sort

Idea of Algorithm:

1. Run DFS on graph G=(V,E).

2. Every time a node is finished, put the node to the left end of a linked list.

3. Return the linked list as the topological sort.



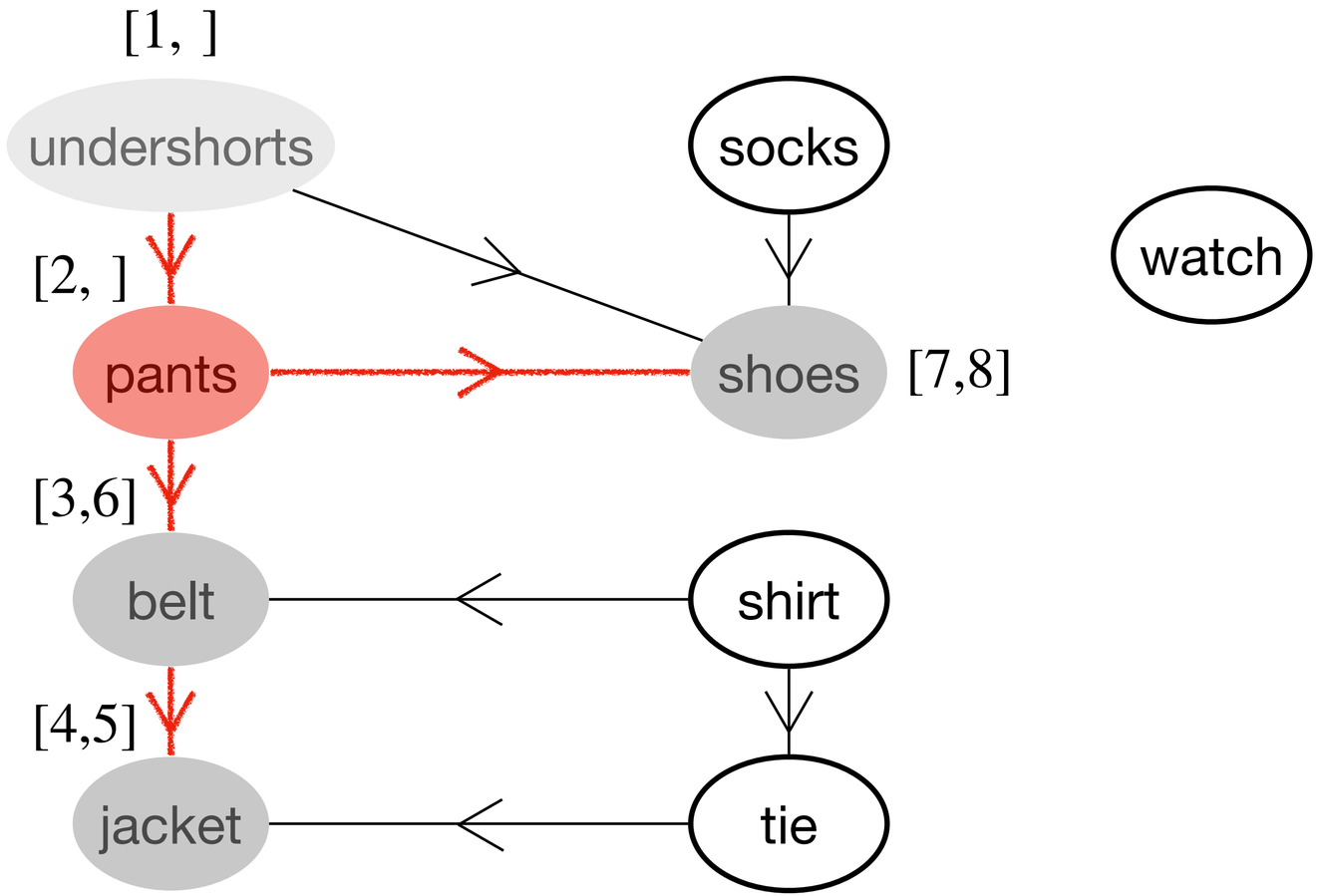
Topological Sort

Idea of Algorithm:

1. Run DFS on graph G=(V,E).

2. Every time a node is finished, put the node to the left end of a linked list.

3. Return the linked list as the topological sort.



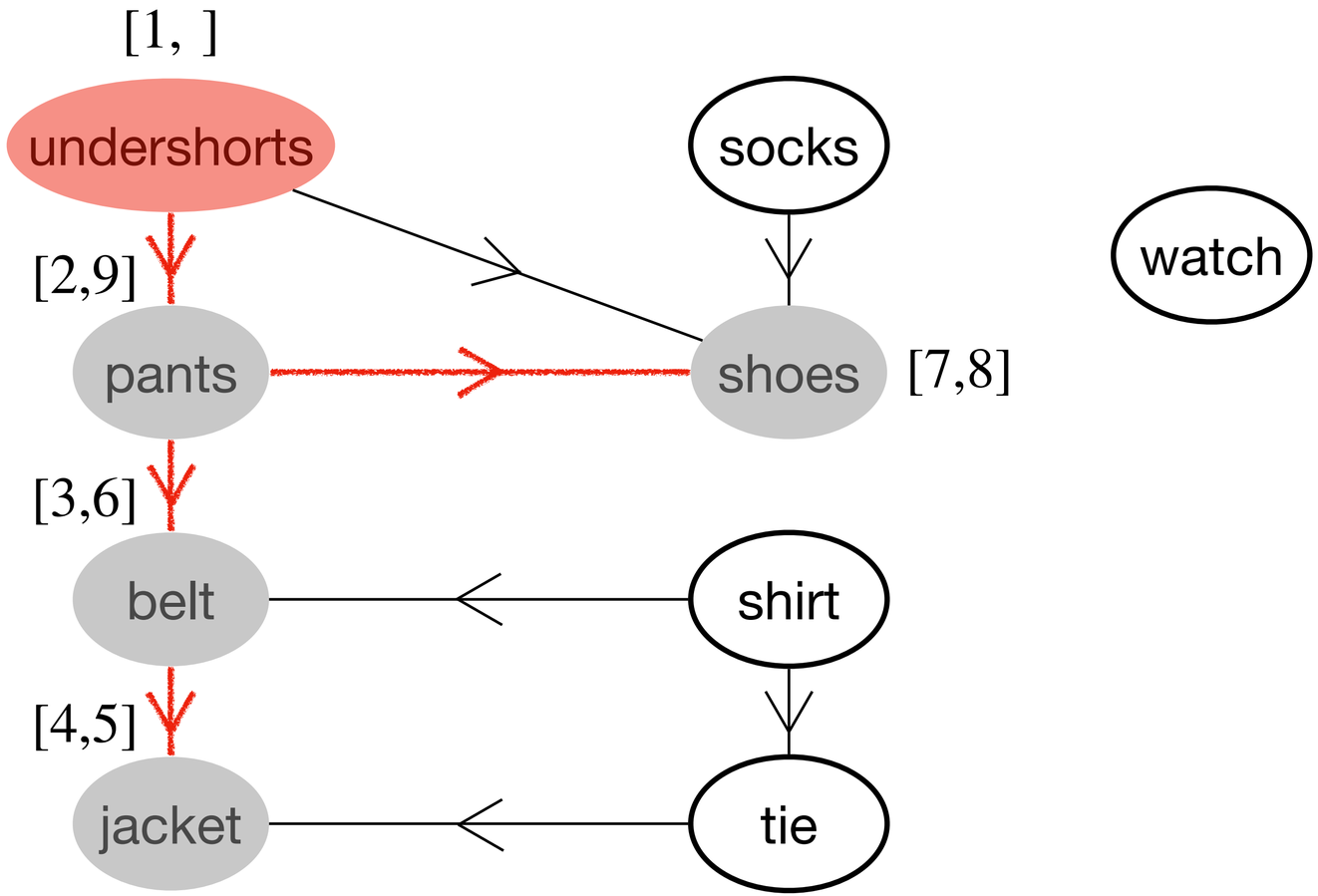
Topological Sort

Idea of Algorithm:

1. Run DFS on graph G=(V,E).

2. Every time a node is finished, put the node to the left end of a linked list.

3. Return the linked list as the topological sort.



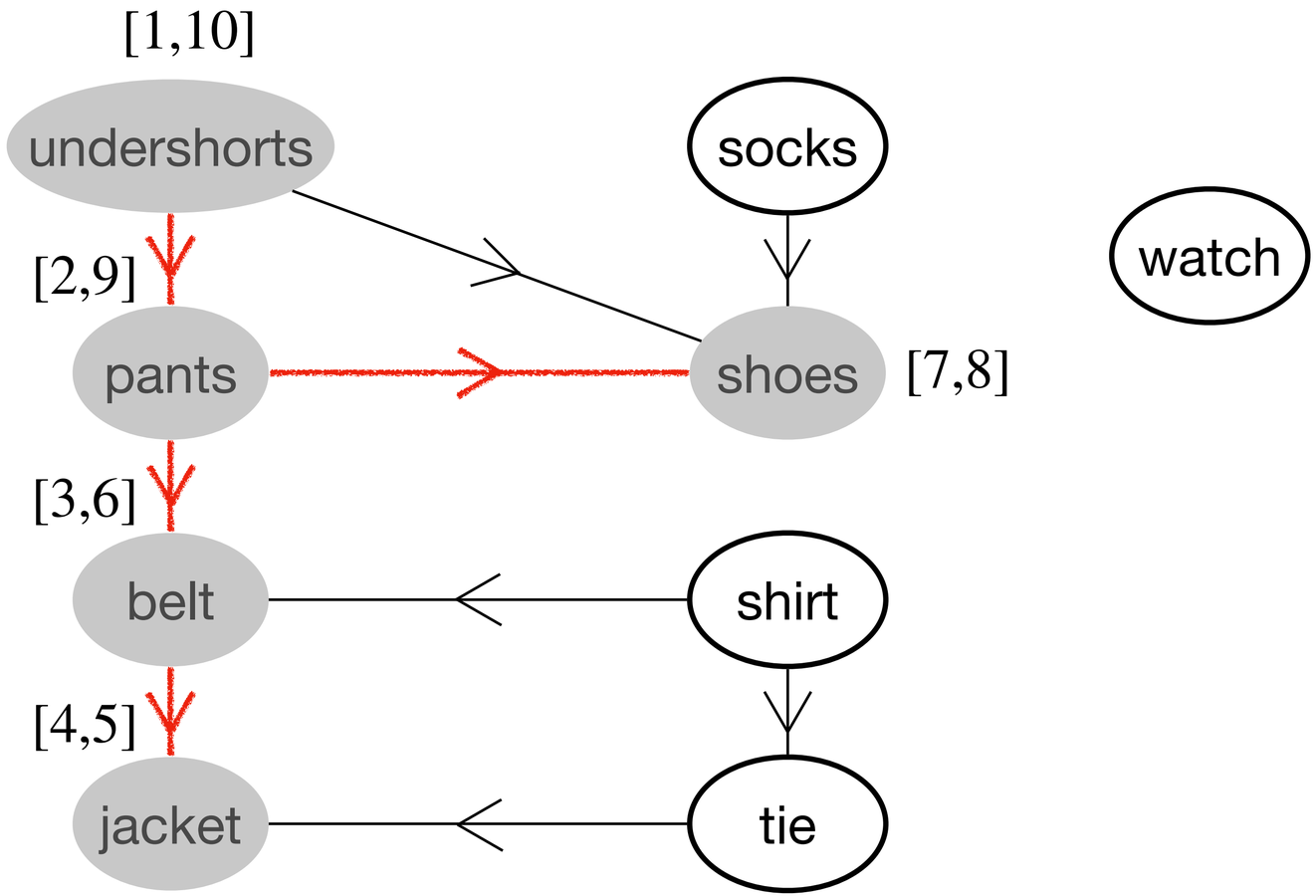
Topological Sort

Idea of Algorithm:

1. Run DFS on graph G=(V,E).

2. Every time a node is finished, put the node to the left end of a linked list.

3. Return the linked list as the topological sort.



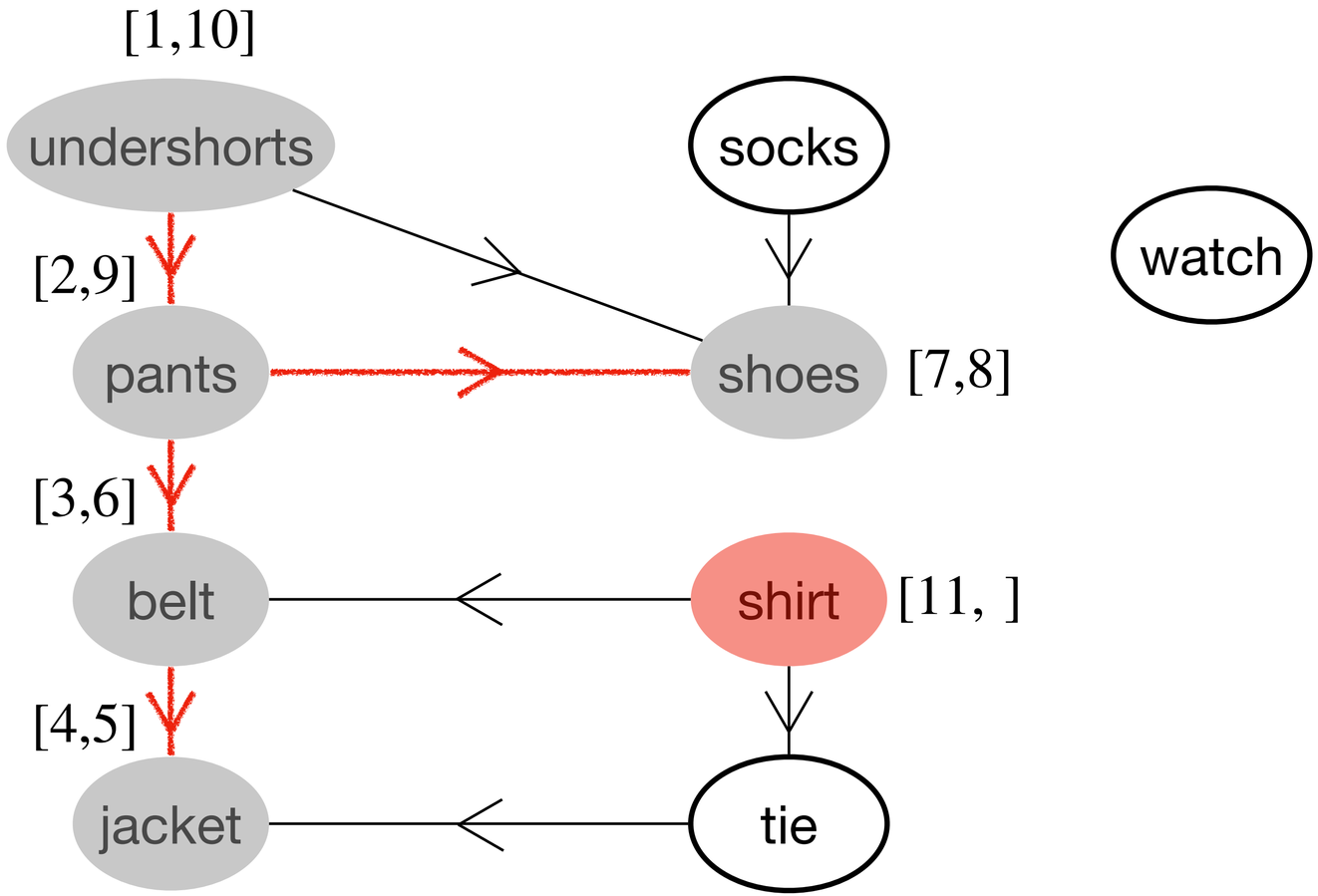
Topological Sort

Idea of Algorithm:

1. Run DFS on graph G=(V,E).

2. Every time a node is finished, put the node to the left end of a linked list.

3. Return the linked list as the topological sort.



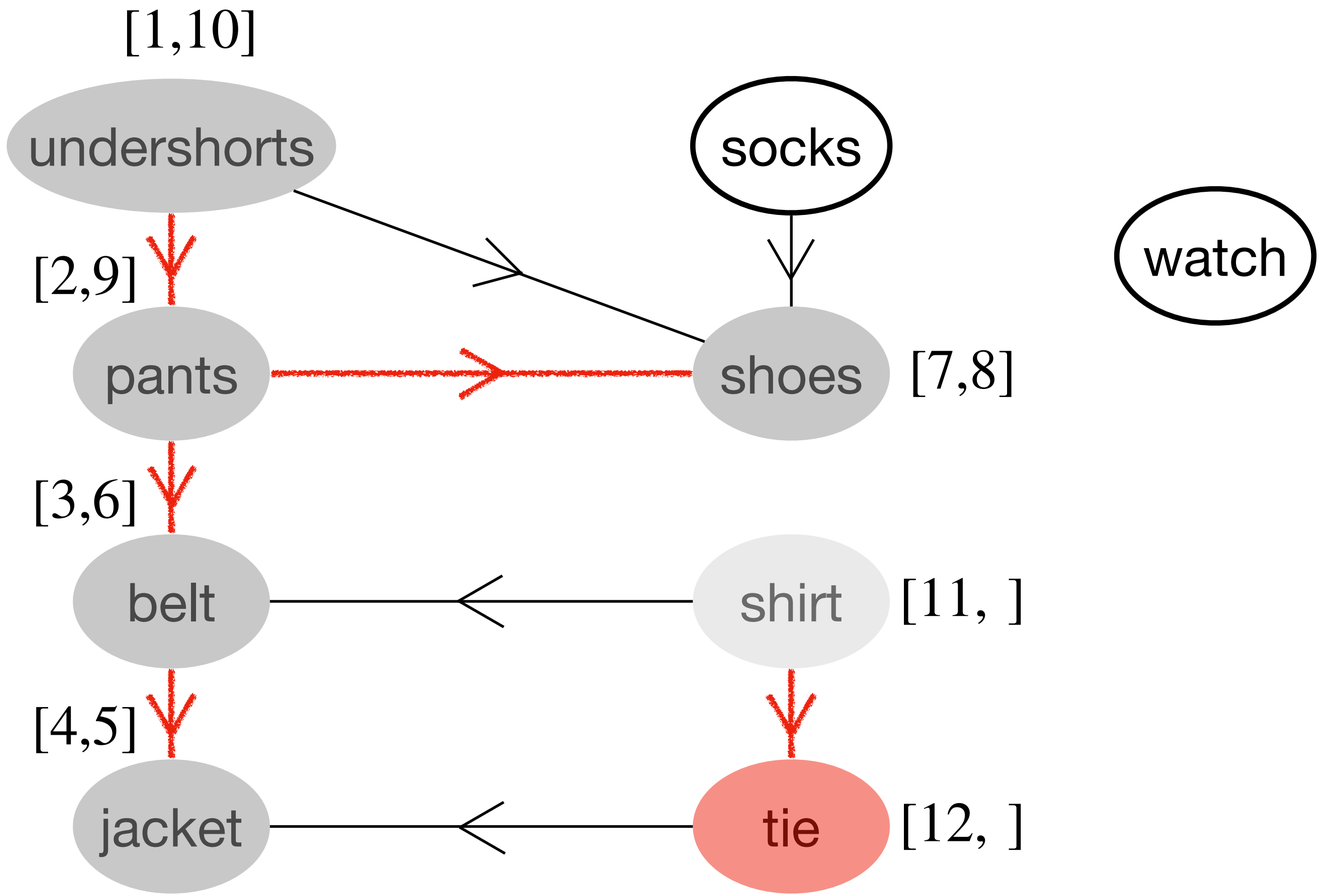
Topological Sort

Idea of Algorithm:

1. Run DFS on graph G=(V,E).

2. Every time a node is finished, put the node to the left end of a linked list.

3. Return the linked list as the topological sort.



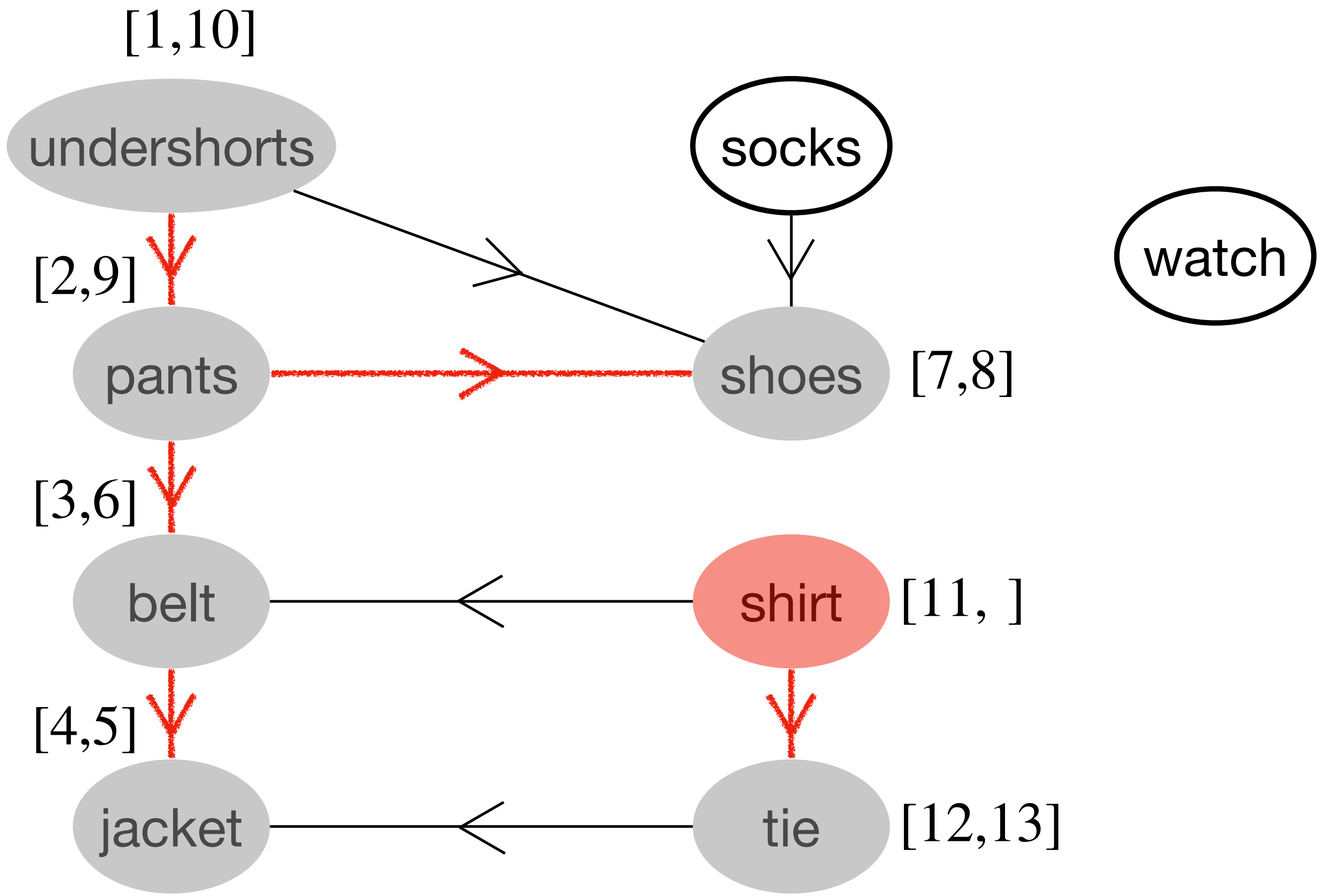
Topological Sort

Idea of Algorithm:

1. Run DFS on graph G=(V,E).

2. Every time a node is finished, put the node to the left end of a linked list.

3. Return the linked list as the topological sort.



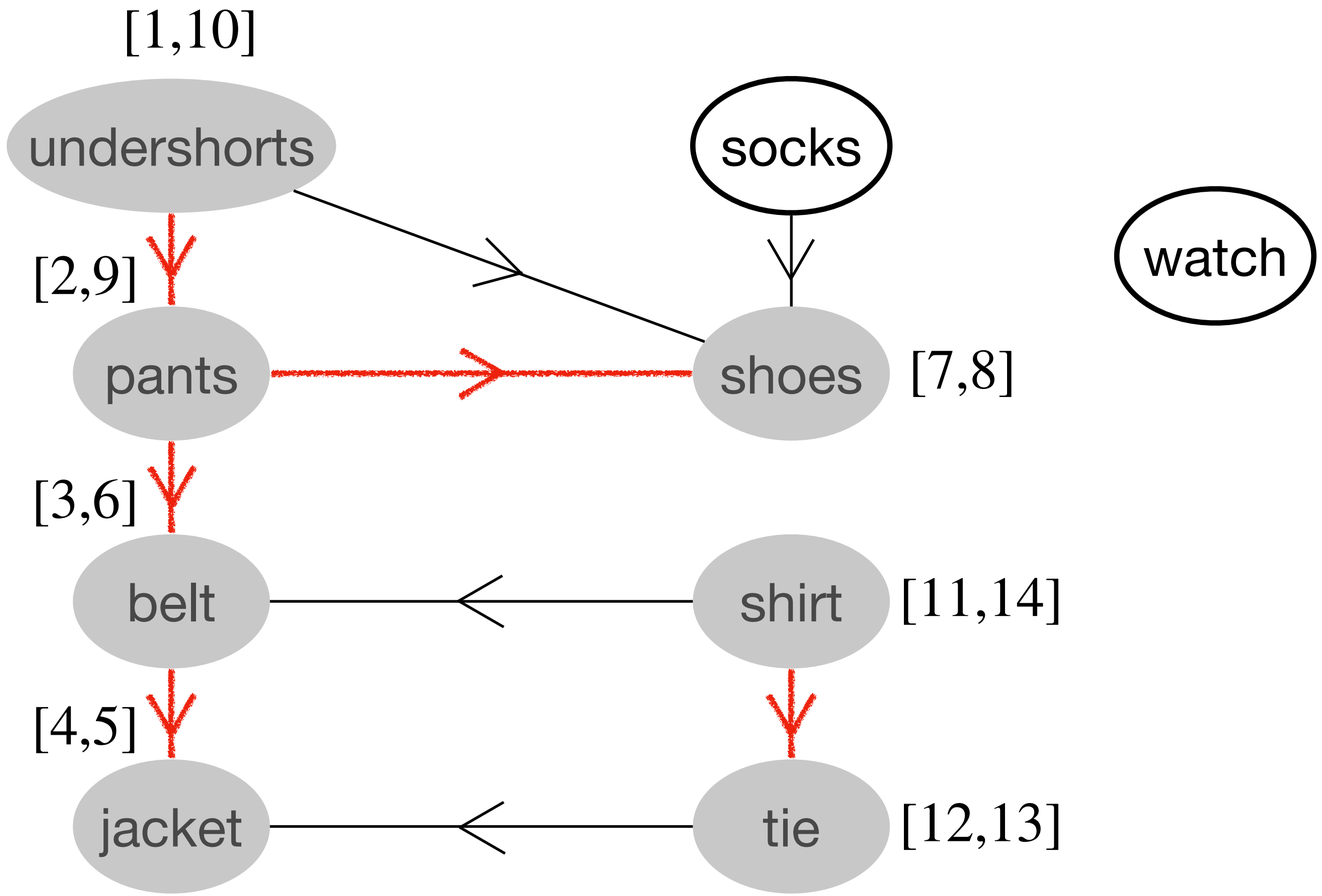
Topological Sort

Idea of Algorithm:

1. Run DFS on graph G=(V,E).

2. Every time a node is finished, put the node to the left end of a linked list.

3. Return the linked list as the topological sort.



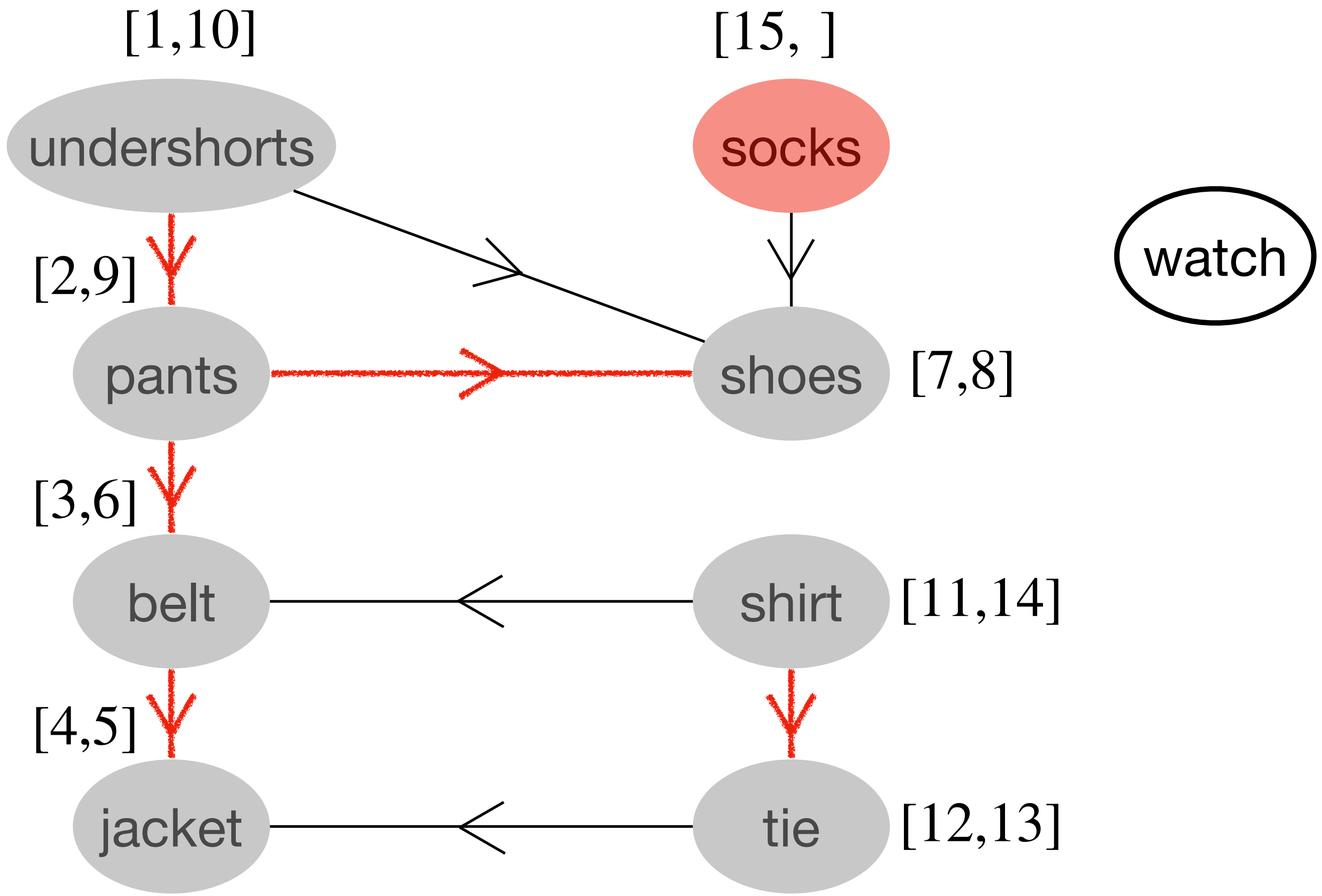
Topological Sort

Idea of Algorithm:

1. Run DFS on graph G=(V,E).

2. Every time a node is finished, put the node to the left end of a linked list.

3. Return the linked list as the topological sort.



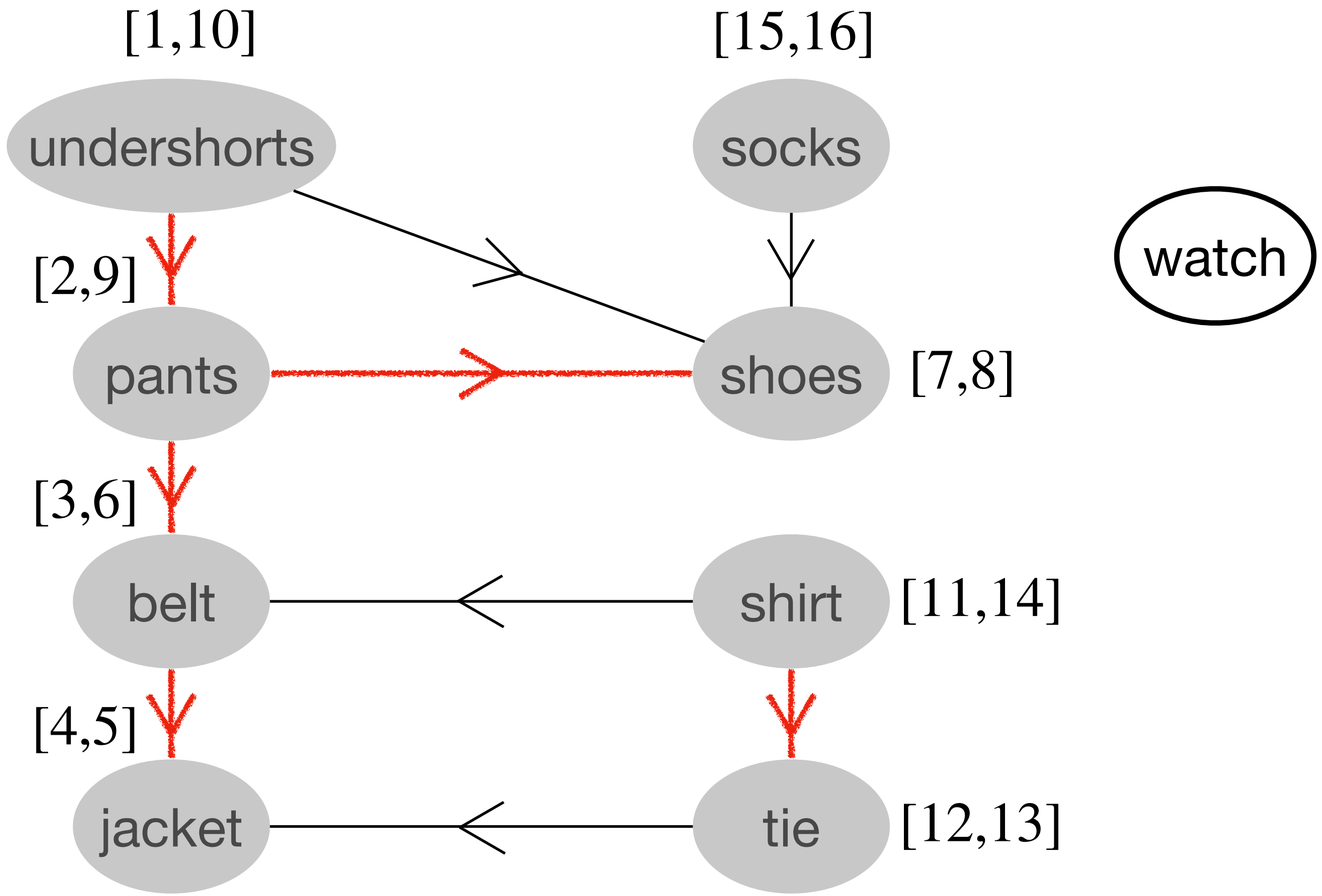
Topological Sort

Idea of Algorithm:

1. Run DFS on graph G=(V,E).

2. Every time a node is finished, put the node to the left end of a linked list.

3. Return the linked list as the topological sort.



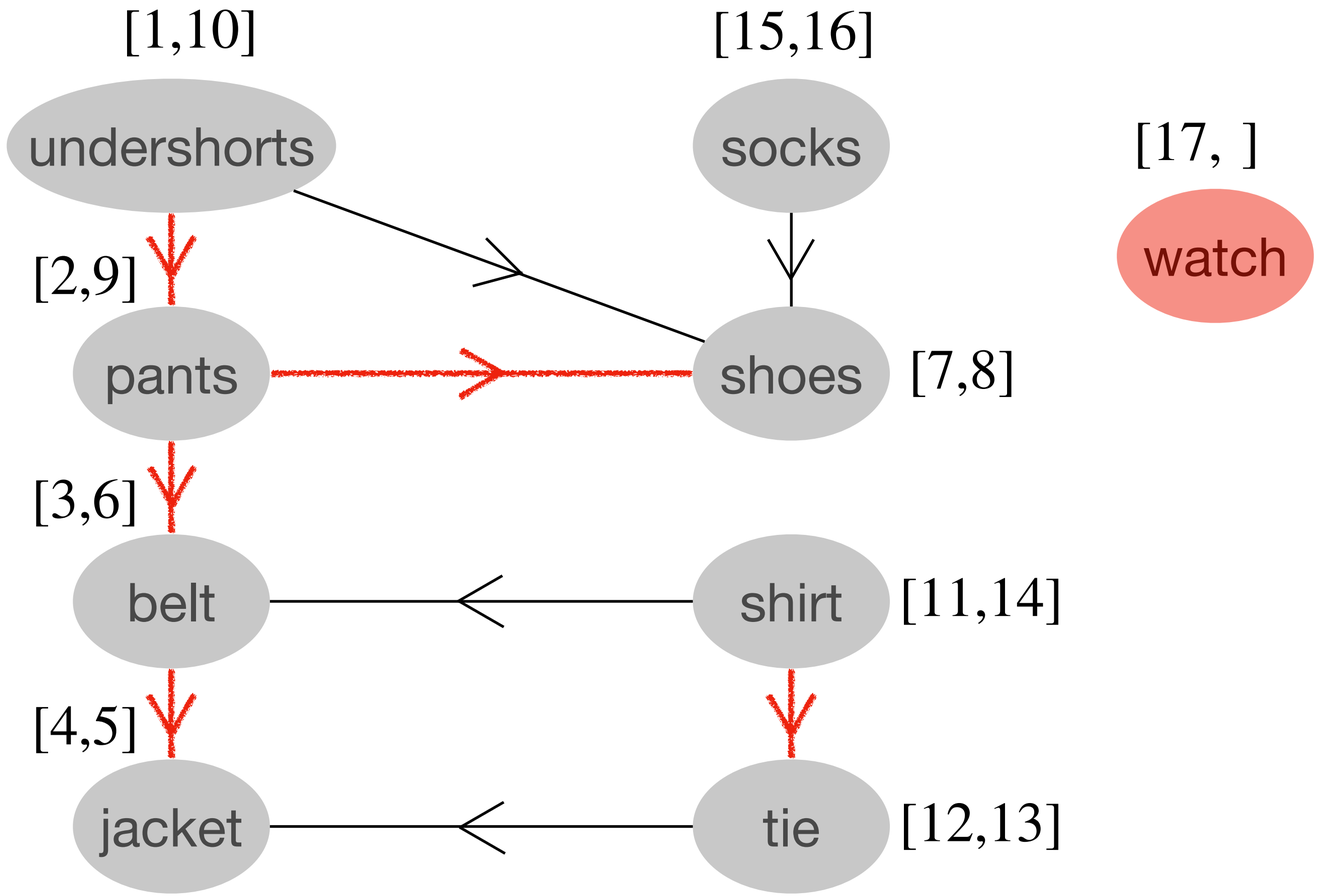
Topological Sort

Idea of Algorithm:

1. Run DFS on graph G=(V,E).

2. Every time a node is finished, put the node to the left end of a linked list.

3. Return the linked list as the topological sort.



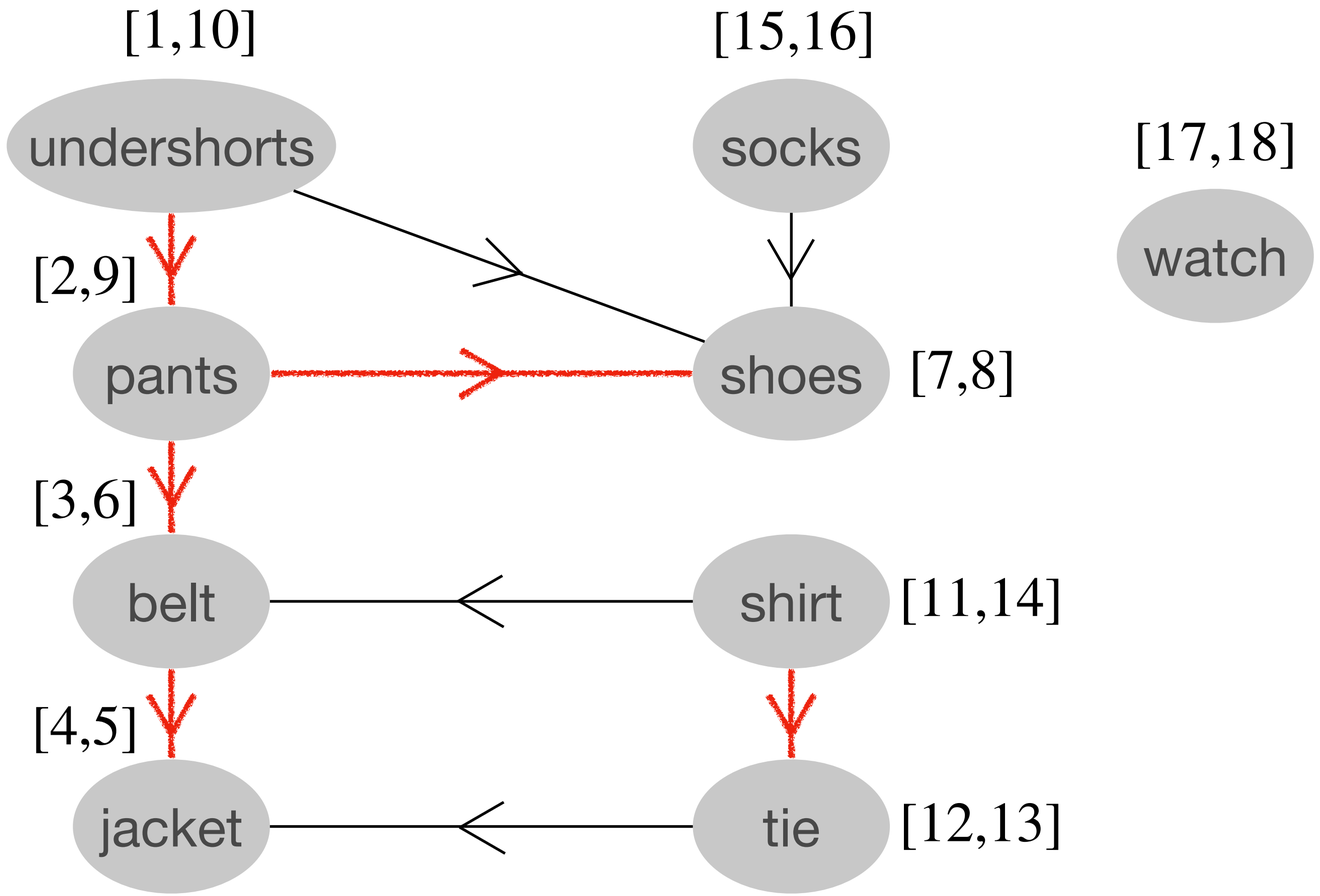
Topological Sort

Idea of Algorithm:

1. Run DFS on graph G=(V,E).

2. Every time a node is finished, put the node to the left end of a linked list.

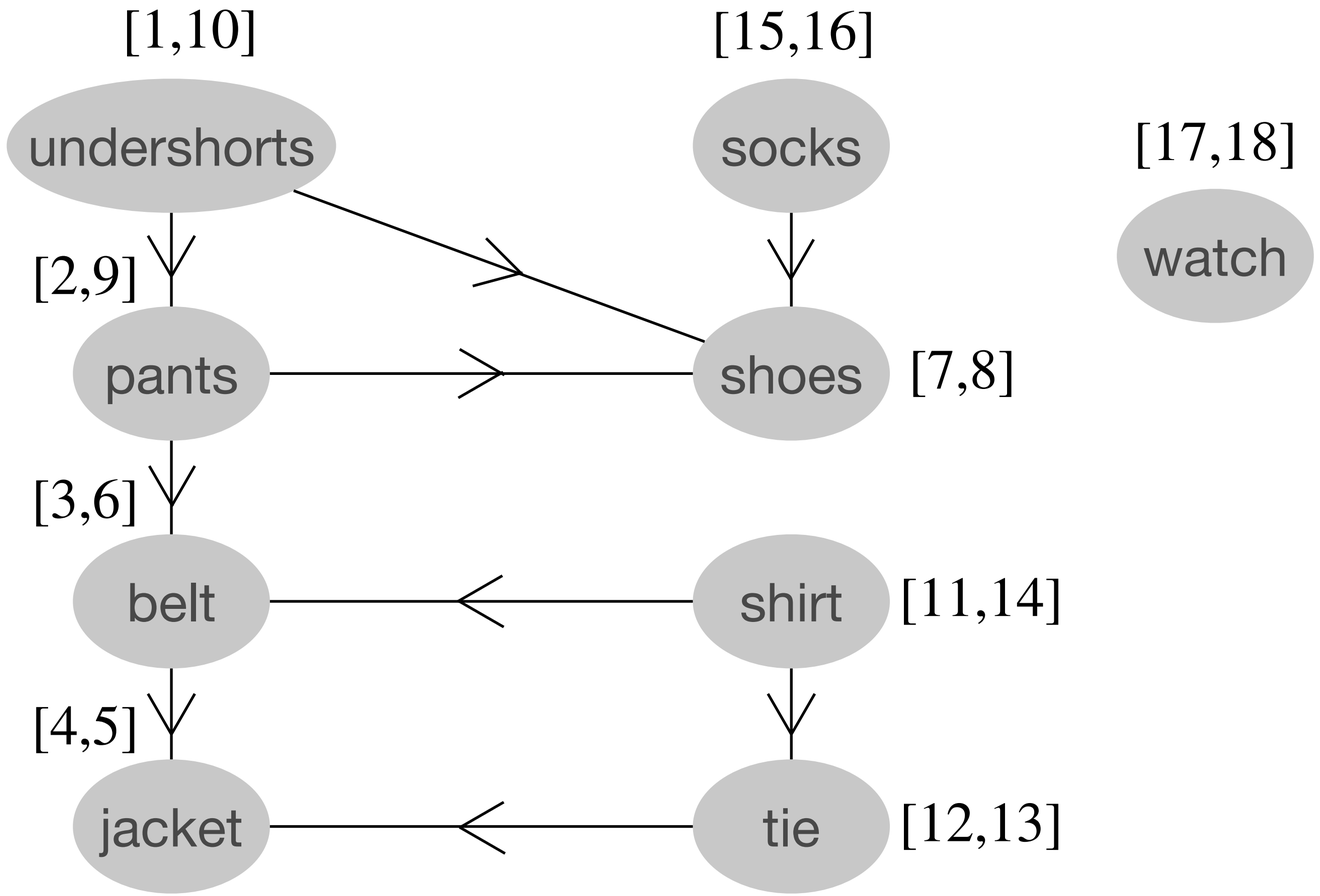
3. Return the linked list as the topological sort.



Topological Sort

Idea of Algorithm:

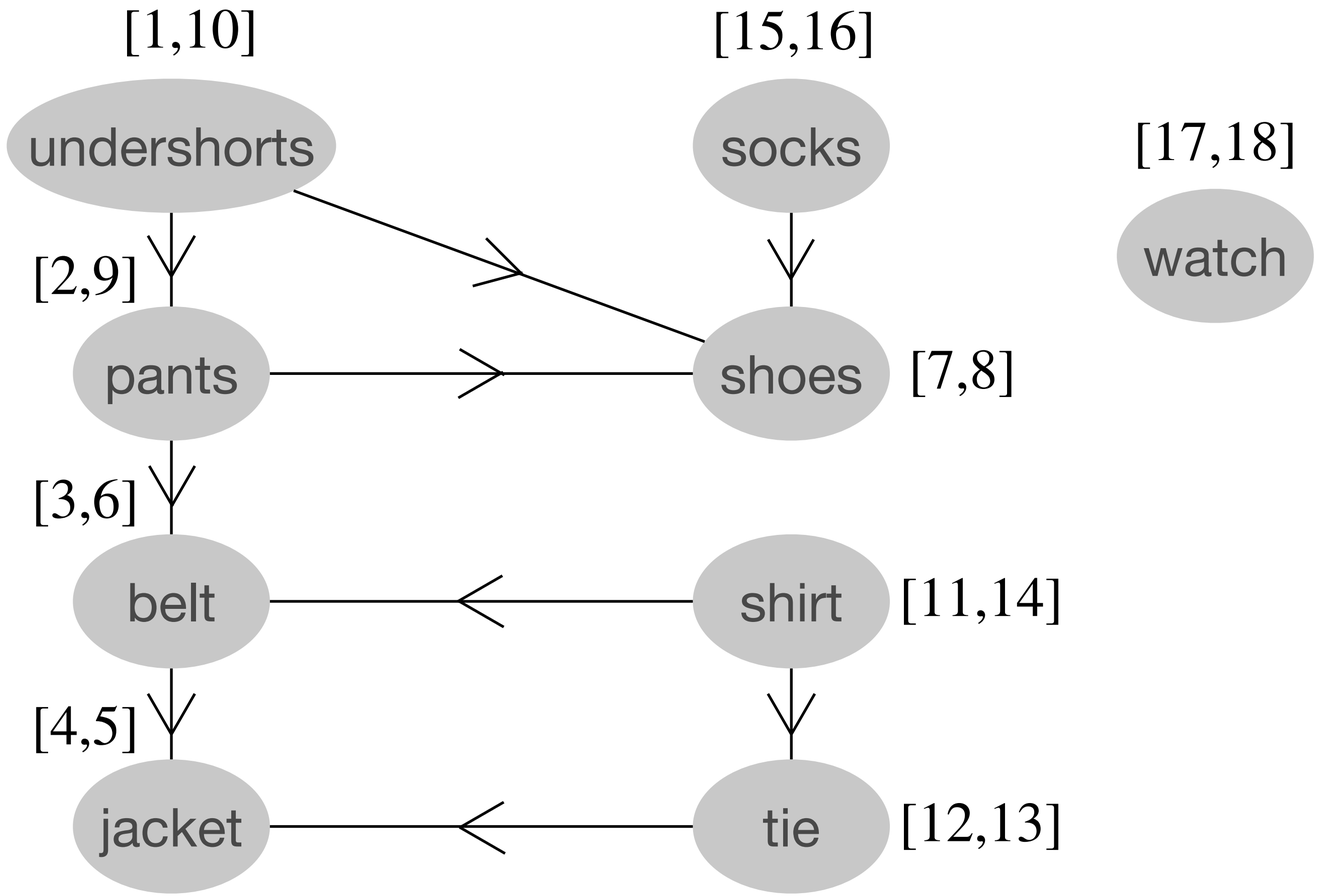
1. Run DFS on graph G=(V,E).
2. Every time a node is finished, put the node to the left end of a linked list.
3. Return the linked list as the topological sort.

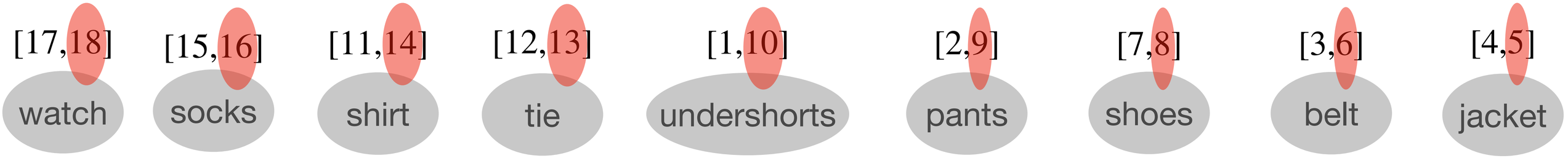


Topological Sort

Idea of Algorithm:

1. Run DFS on graph G=(V,E).
2. Every time a node is finished, put the node to the left end of a linked list.
3. Return the linked list as the topological sort.

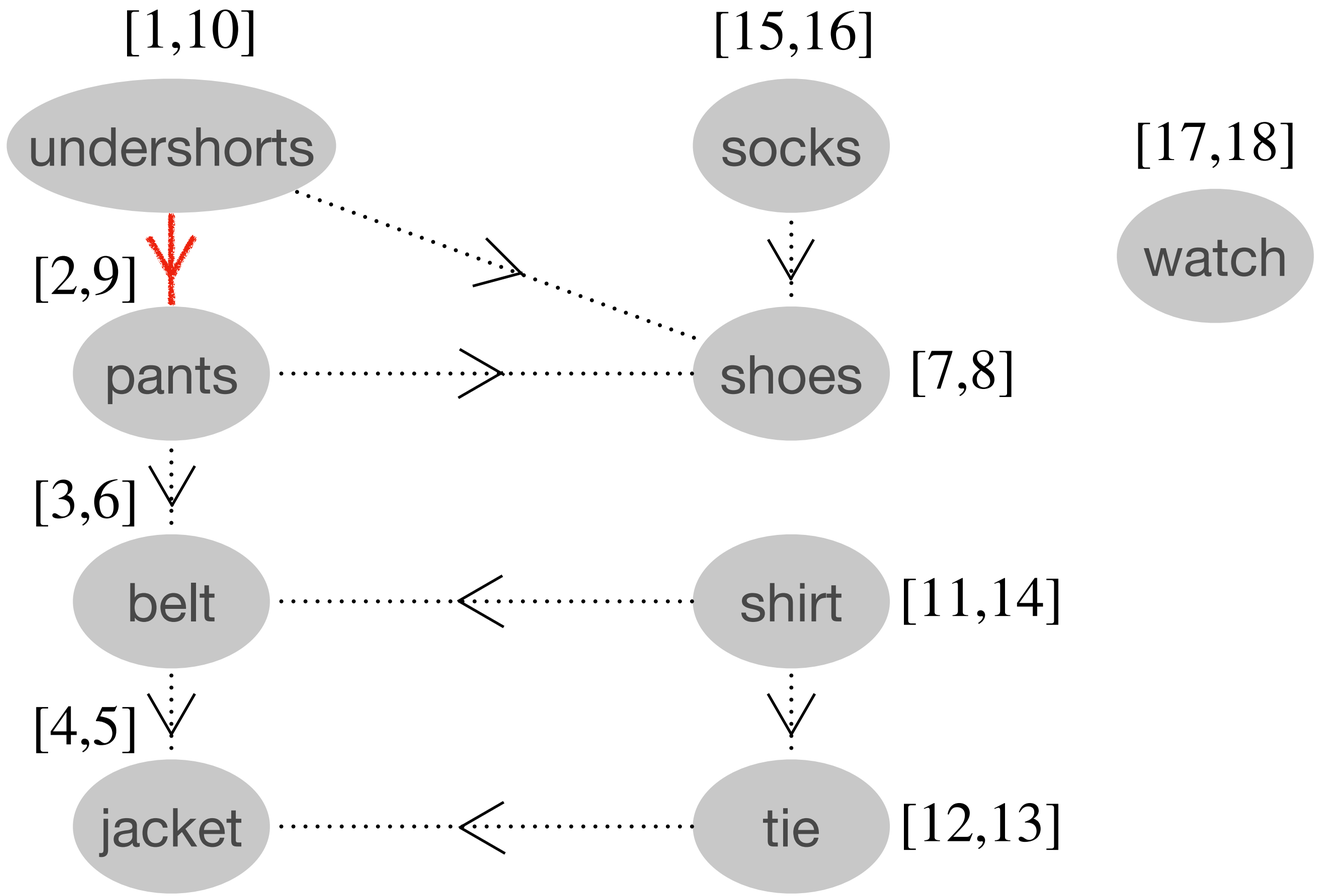




Topological Sort

Idea of Algorithm:

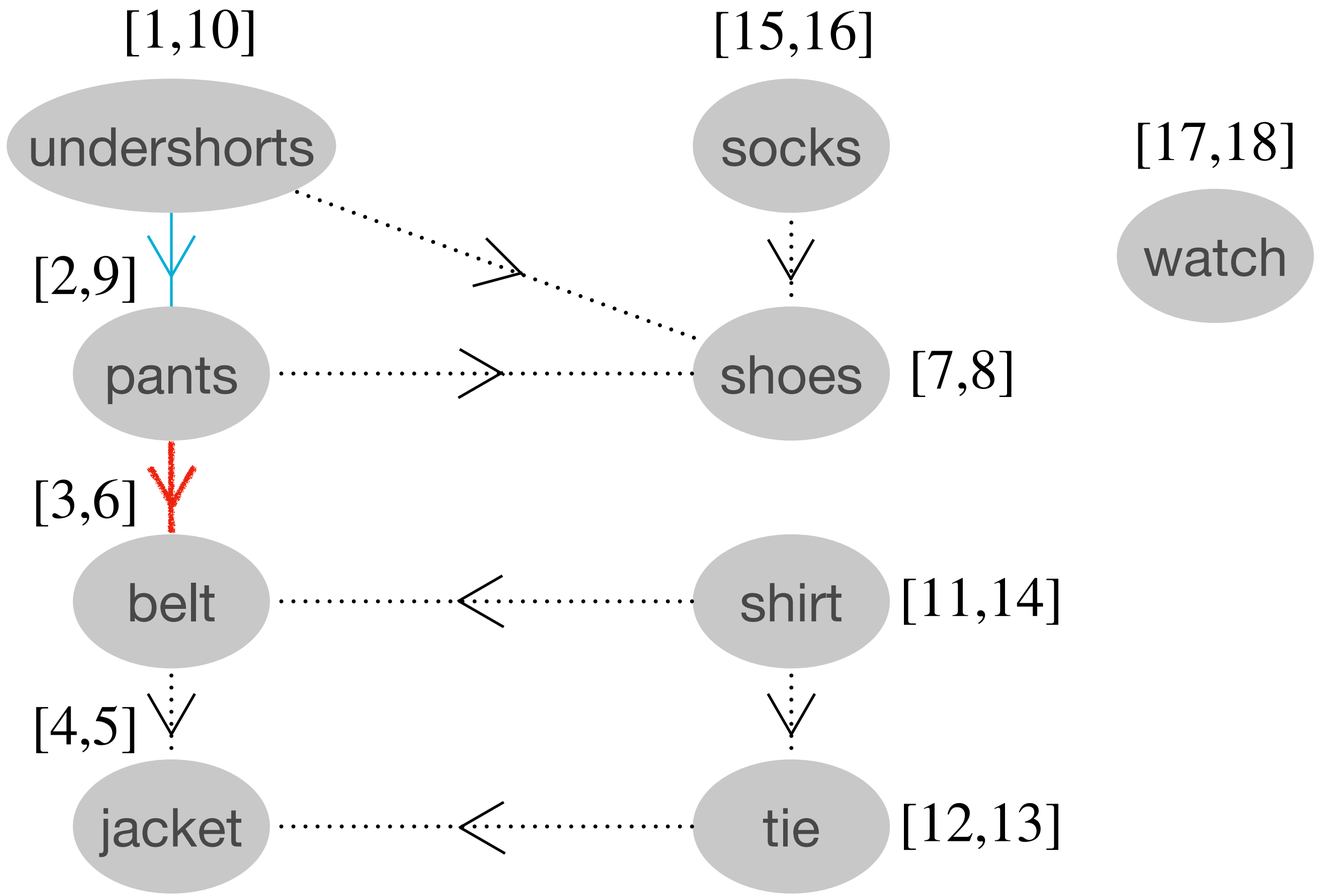
1. Run DFS on graph G=(V,E).
2. Every time a node is finished, put the node to the left end of a linked list.
3. Return the linked list as the topological sort.

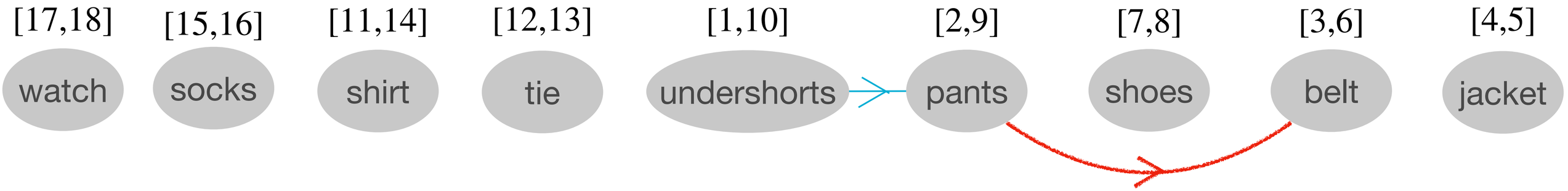


Topological Sort

Idea of Algorithm:

1. Run DFS on graph G=(V,E).
2. Every time a node is finished, put the node to the left end of a linked list.
3. Return the linked list as the topological sort.

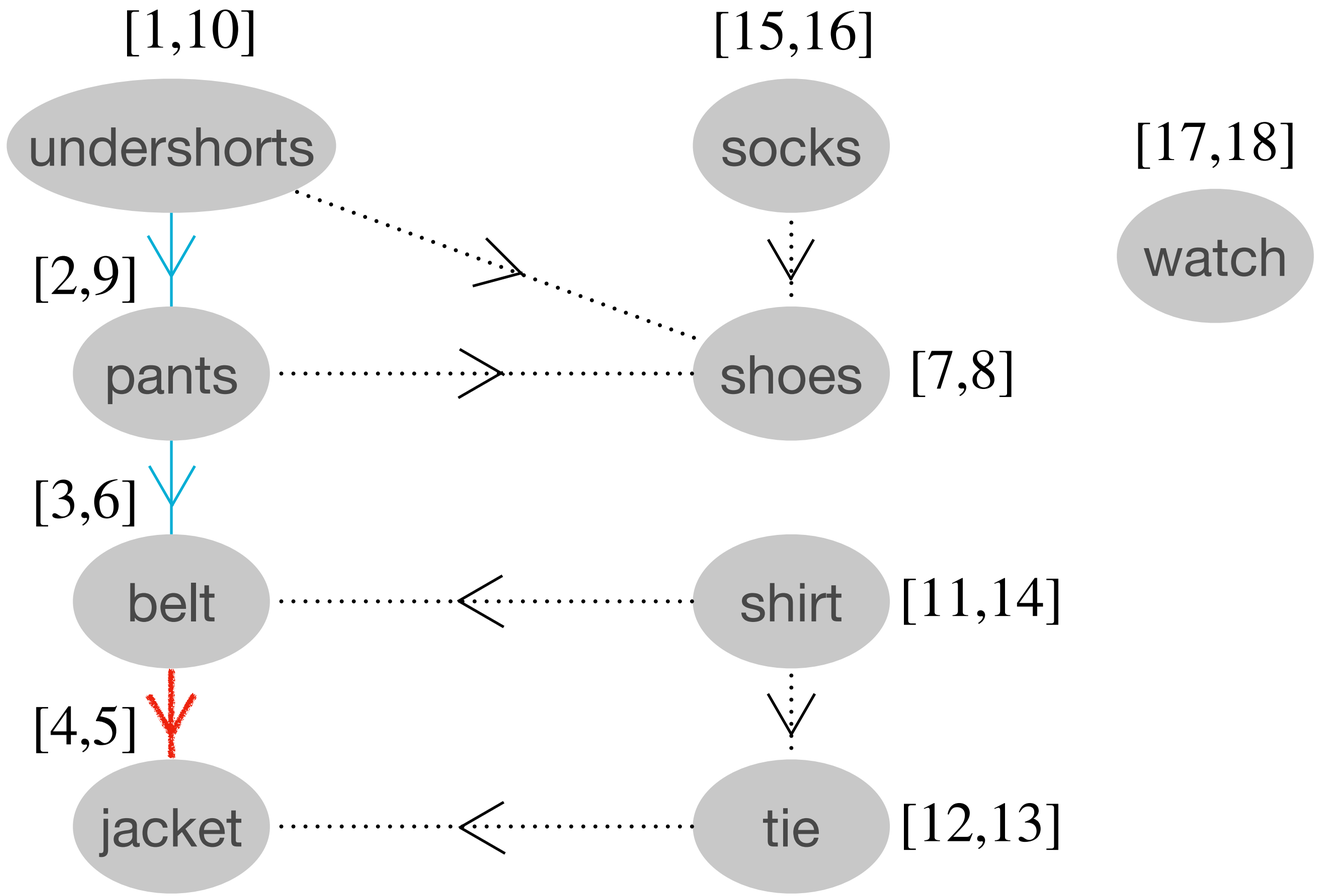


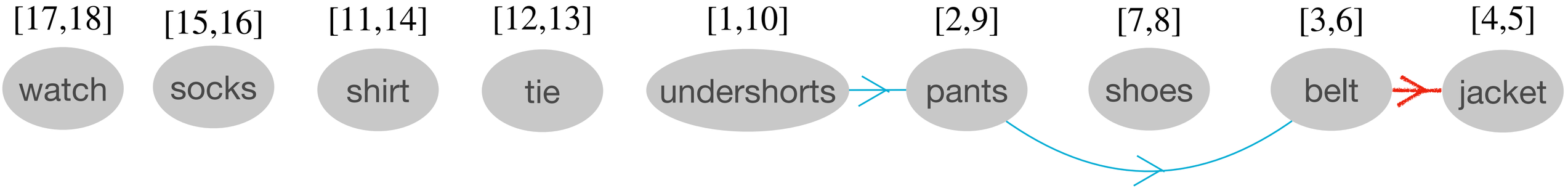


Topological Sort

Idea of Algorithm:

1. Run DFS on graph G=(V,E).
2. Every time a node is finished, put the node to the left end of a linked list.
3. Return the linked list as the topological sort.

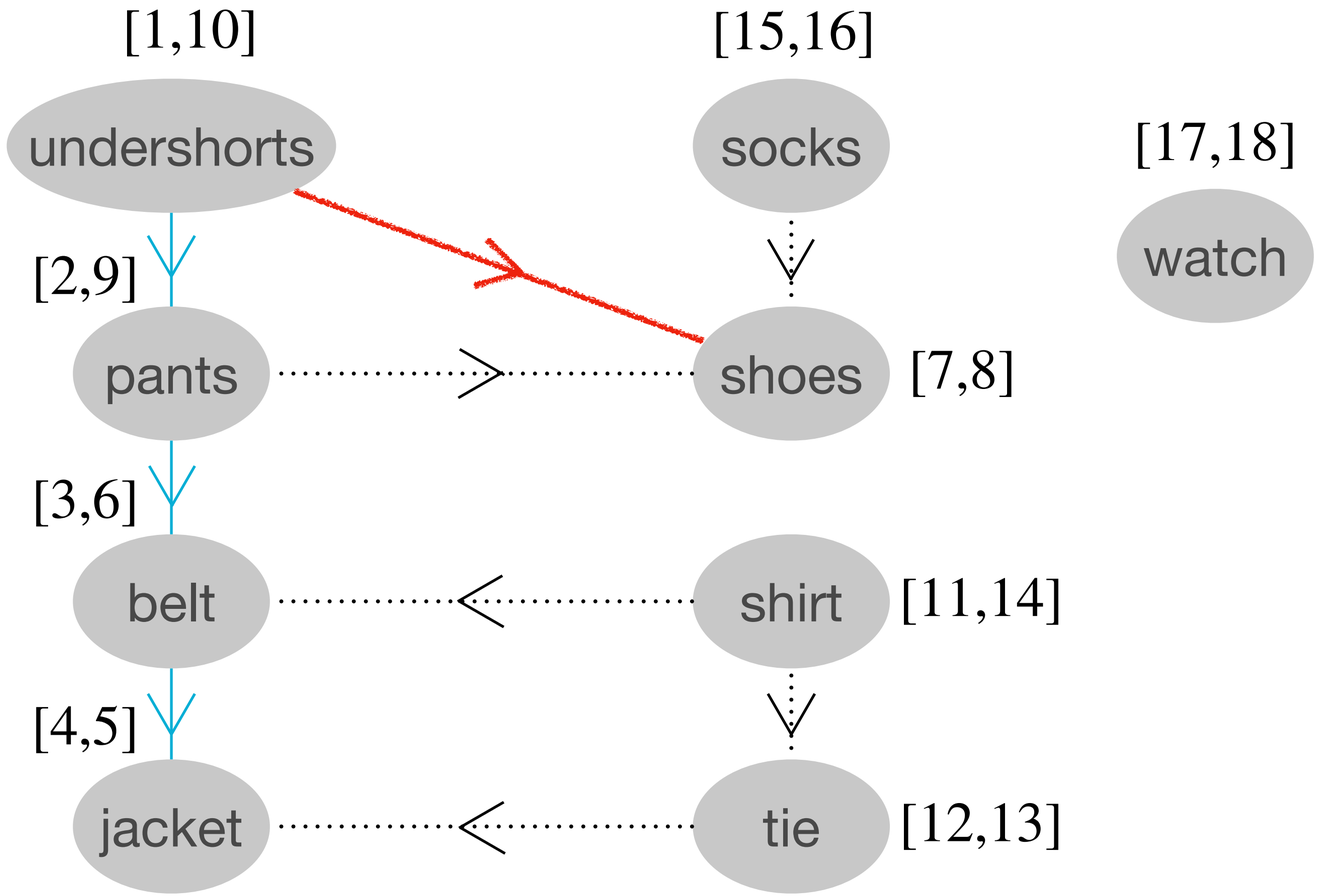


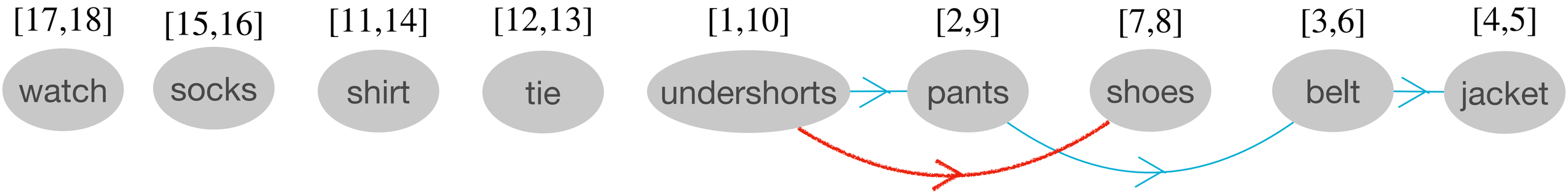


Topological Sort

Idea of Algorithm:

1. Run DFS on graph G=(V,E).
2. Every time a node is finished, put the node to the left end of a linked list.
3. Return the linked list as the topological sort.

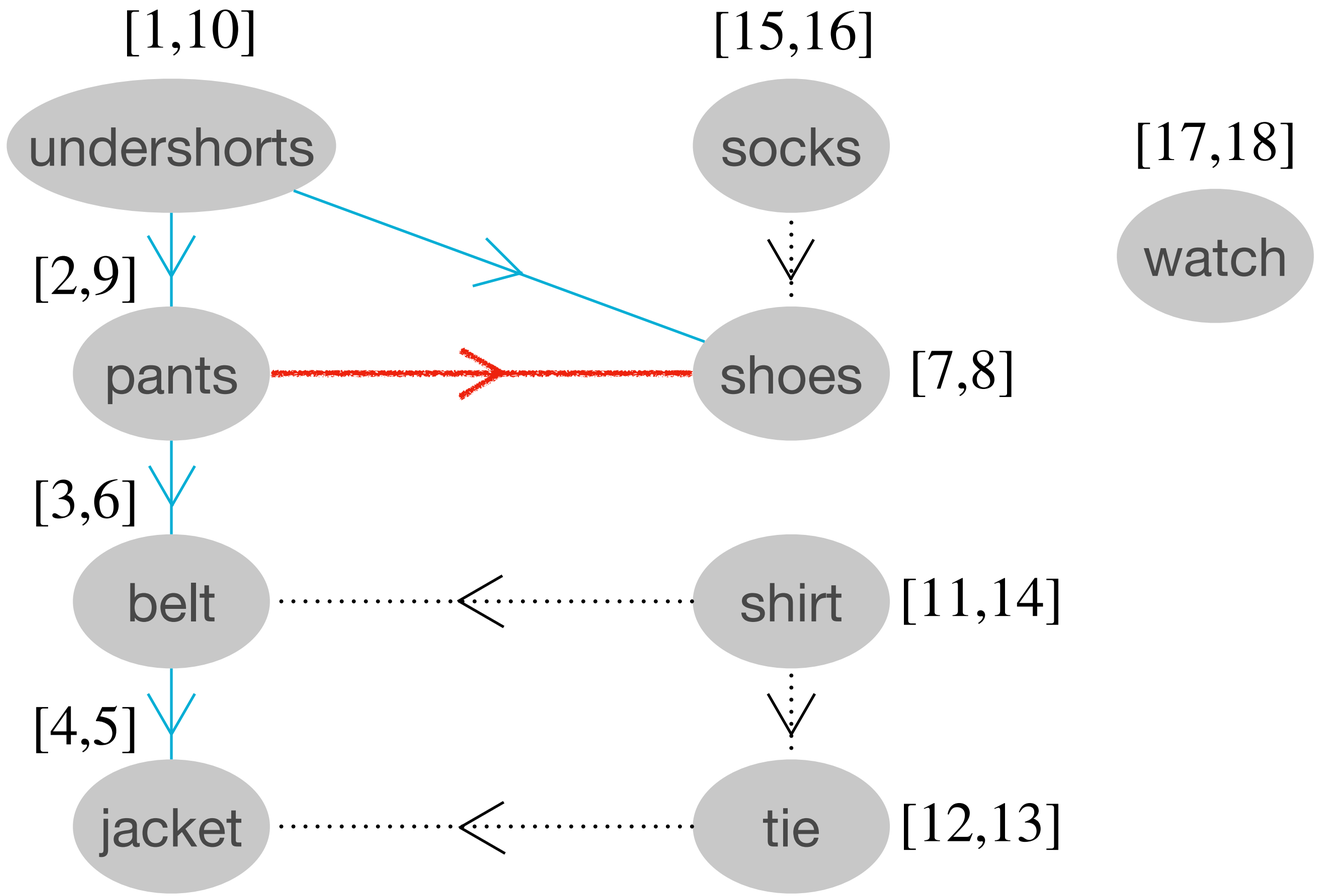


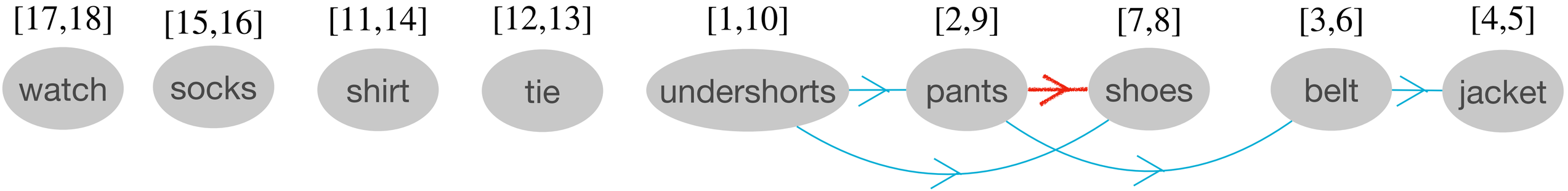


Topological Sort

Idea of Algorithm:

1. Run DFS on graph G=(V,E).
2. Every time a node is finished, put the node to the left end of a linked list.
3. Return the linked list as the topological sort.

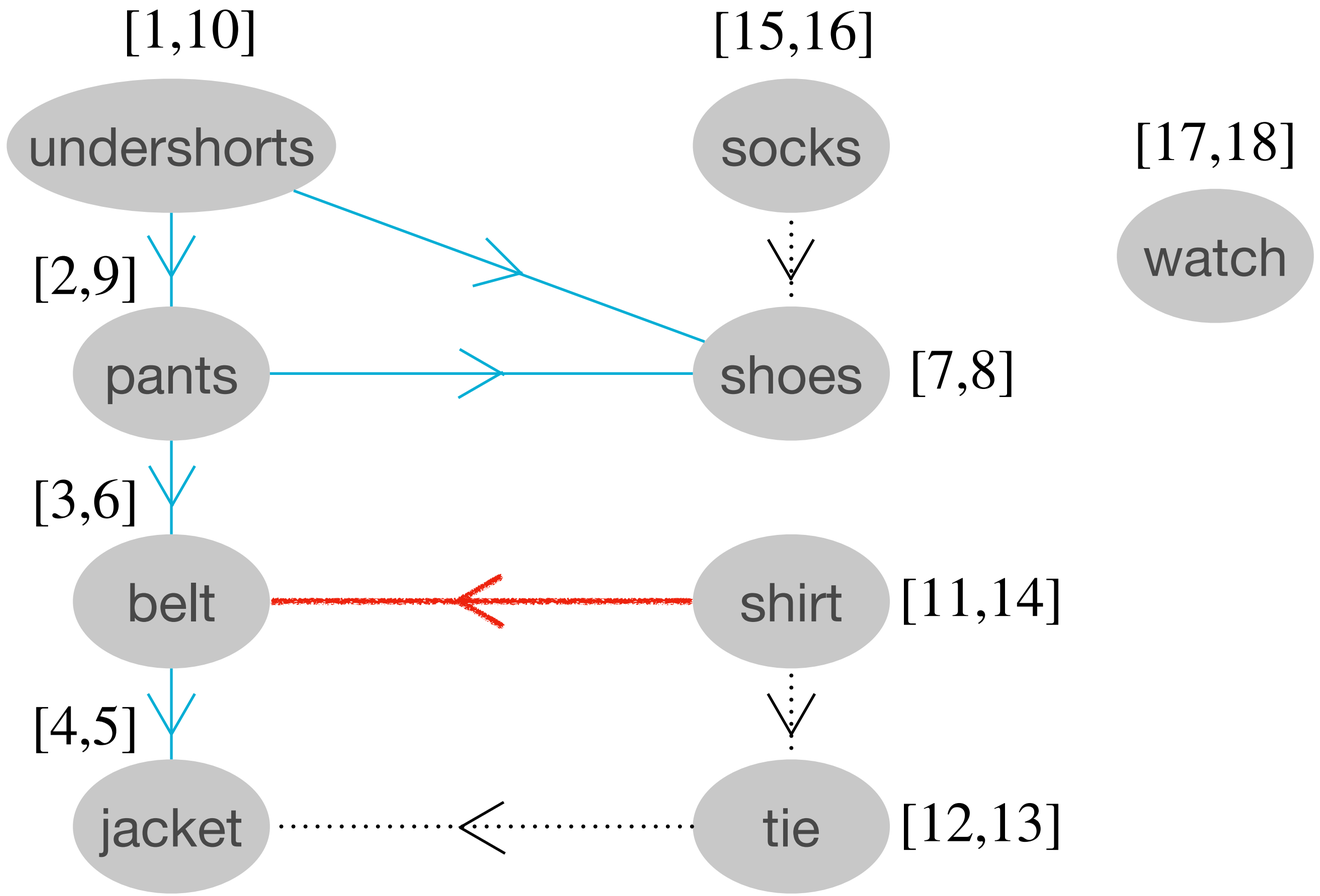


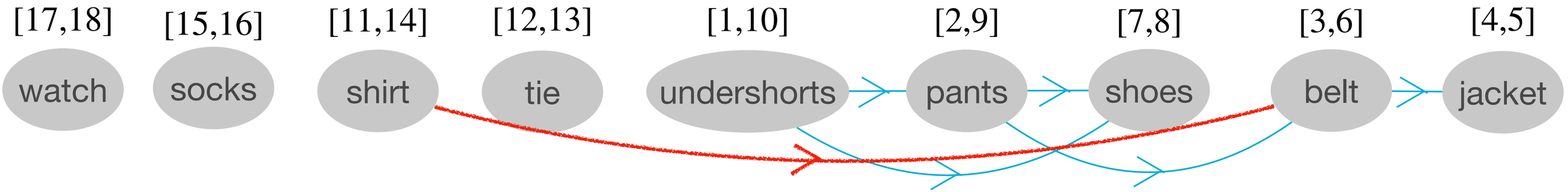


Topological Sort

Idea of Algorithm:

1. Run DFS on graph G=(V,E).
2. Every time a node is finished, put the node to the left end of a linked list.
3. Return the linked list as the topological sort.

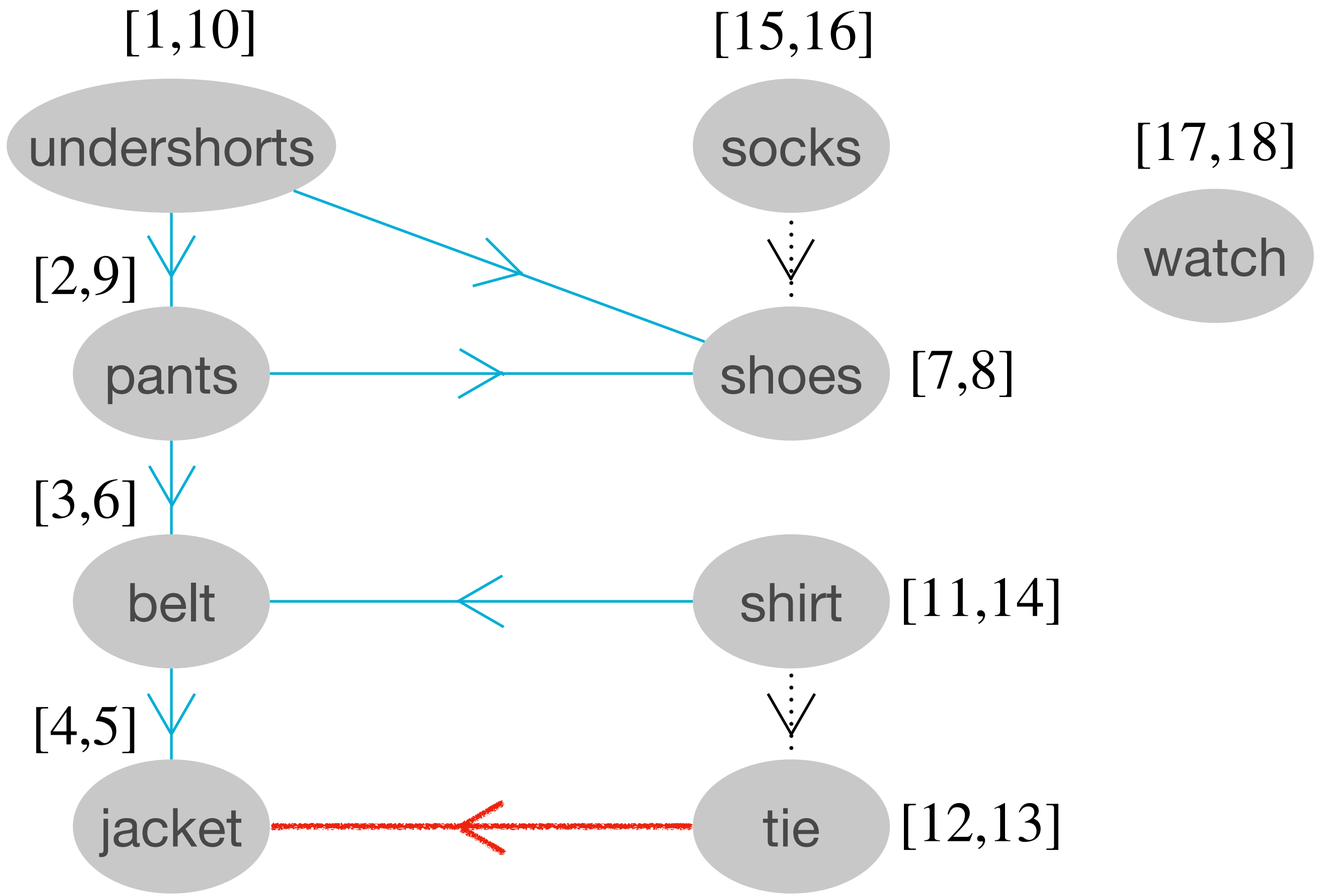


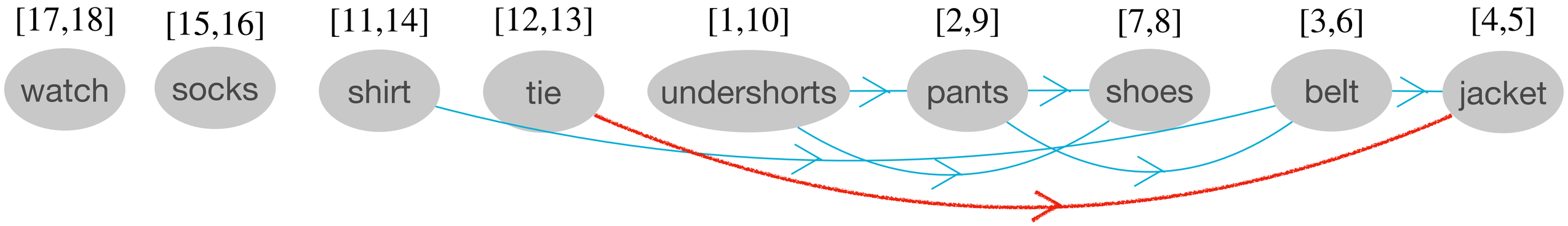


Topological Sort

Idea of Algorithm:

1. Run DFS on graph G=(V,E).
2. Every time a node is finished, put the node to the left end of a linked list.
3. Return the linked list as the topological sort.

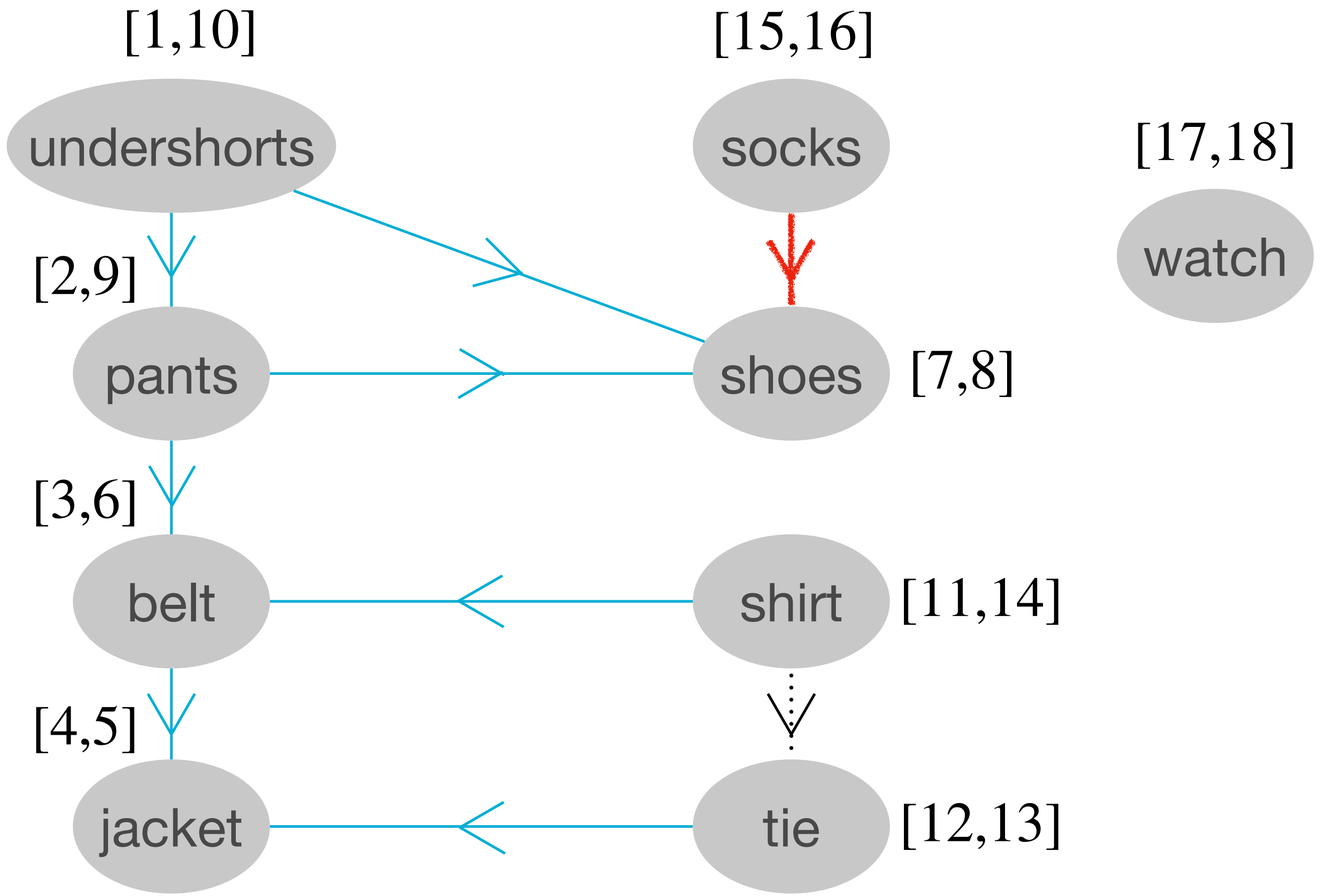


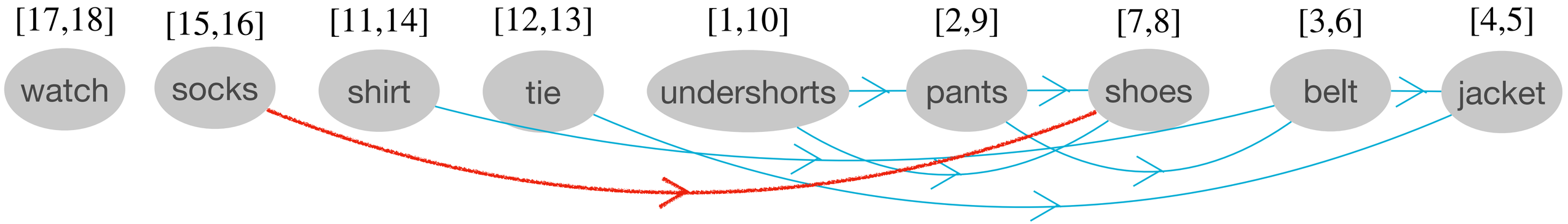


Topological Sort

Idea of Algorithm:

1. Run DFS on graph G=(V,E).
2. Every time a node is finished, put the node to the left end of a linked list.
3. Return the linked list as the topological sort.

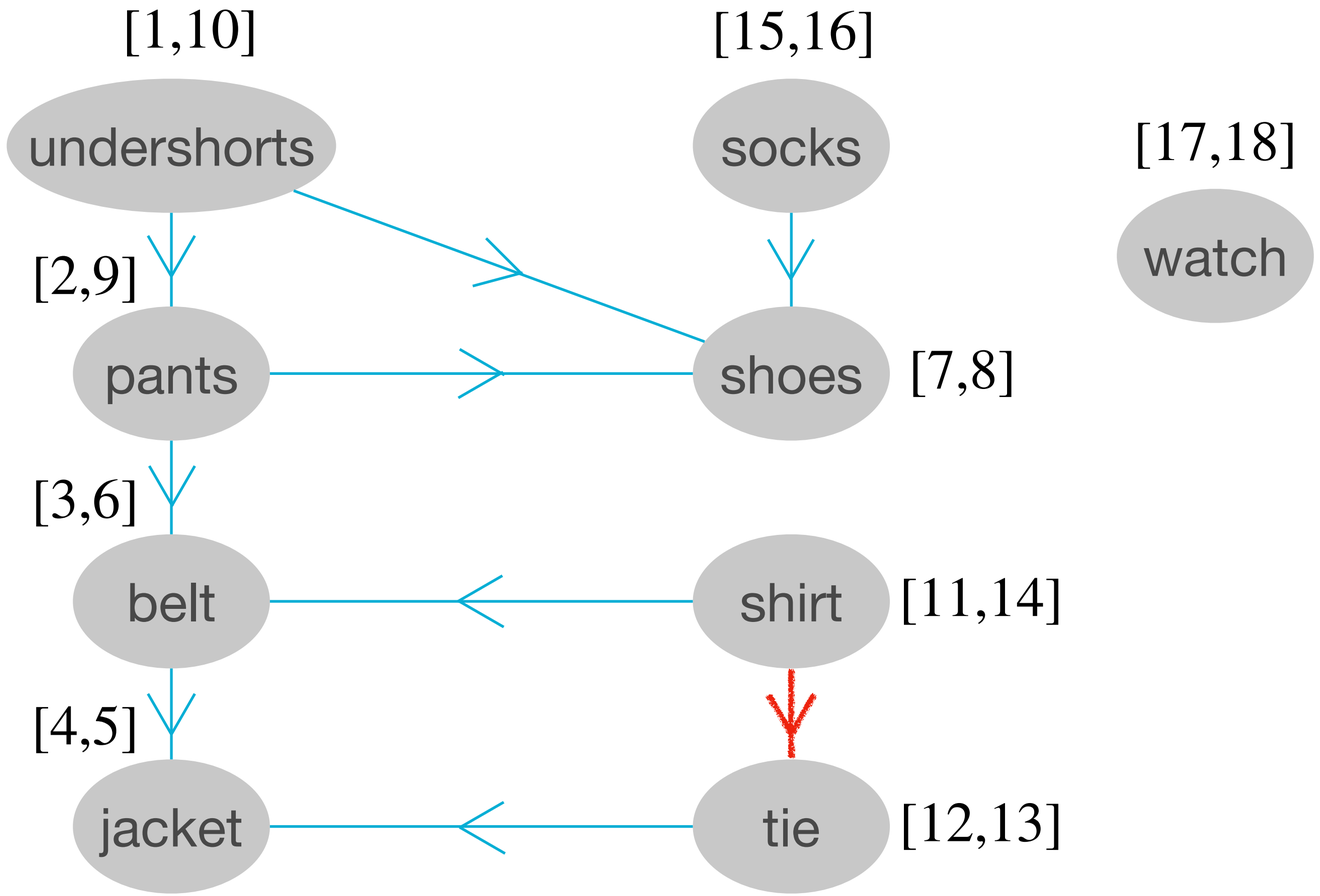


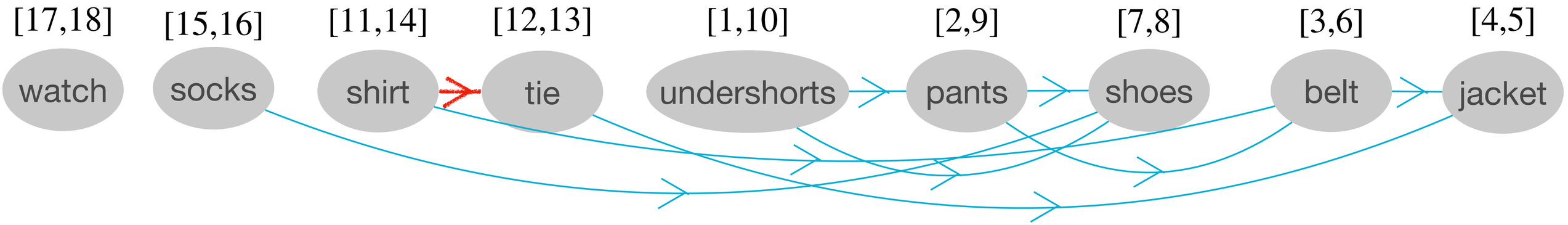


Topological Sort

Idea of Algorithm:

1. Run DFS on graph G=(V,E).
2. Every time a node is finished, put the node to the left end of a linked list.
3. Return the linked list as the topological sort.

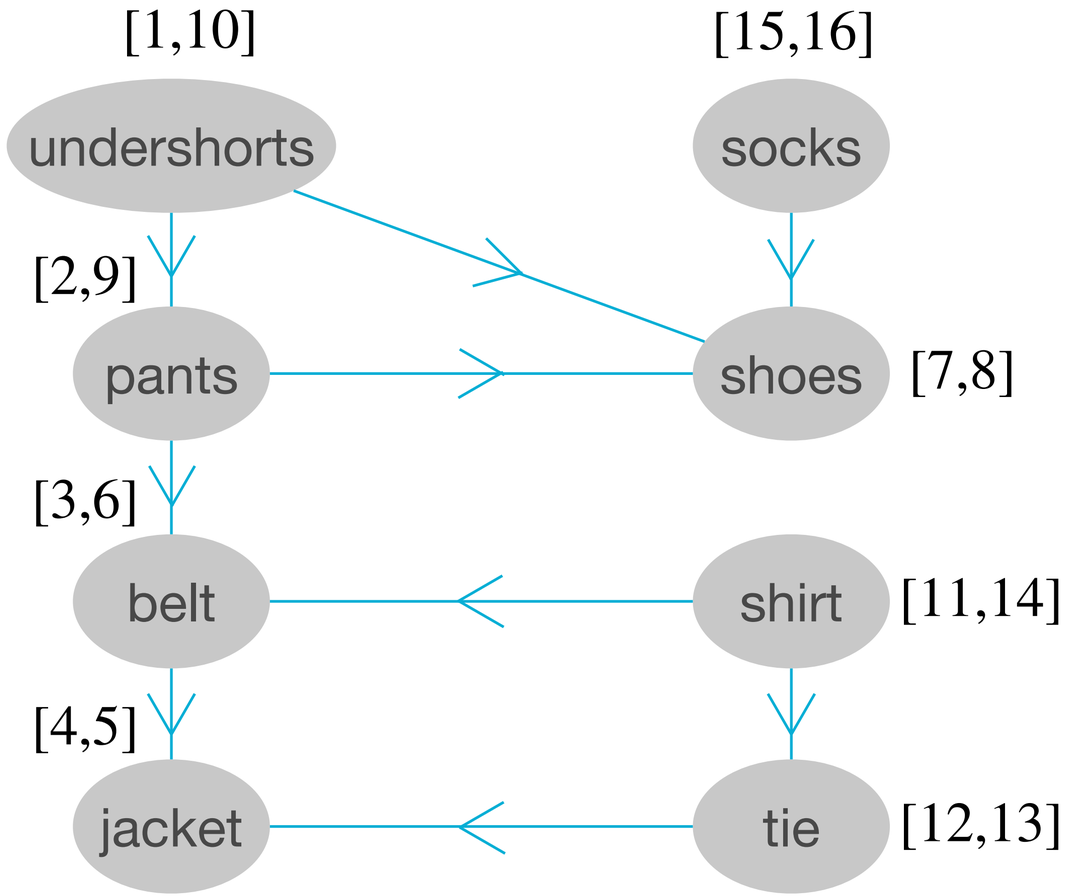




Topological Sort

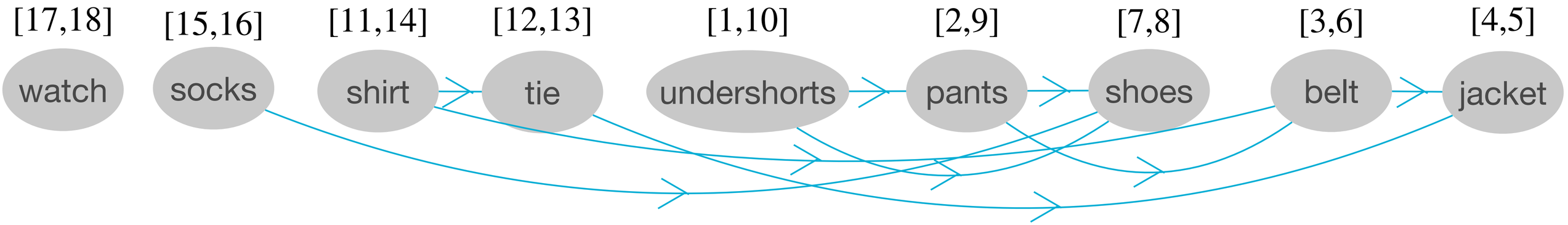
Idea of Algorithm:

1. Run DFS on graph G=(V,E).
2. Every time a node is finished, put the node to the left end of a linked list.
3. Return the linked list as the topological sort.



Time Complexity:

$$O(V+E)$$



Quiz questions:

1. When we run DFS on a DAG, what is the relationship between the finish times of the two endpoints of an edge?

2. How is the above property used by the topological sort algorithm?