Graphs

- Definition
  - G = (V, E)
- Representation
  - Edge set
  - Adjacency matrix
  - Adjacency lists
- Graph API
  - create
  - addEdge(v,w)
  - Adj(v)
- Performance
  - Described in |E| and |V|

DFS and BFS

DFS (to visit a vertex v)

Mark v as visited.

Recursively visit all unmarked vertices w adjacent to v.

BFS (from source vertex s)

Put s onto a FIFO queue, and mark s as visited.

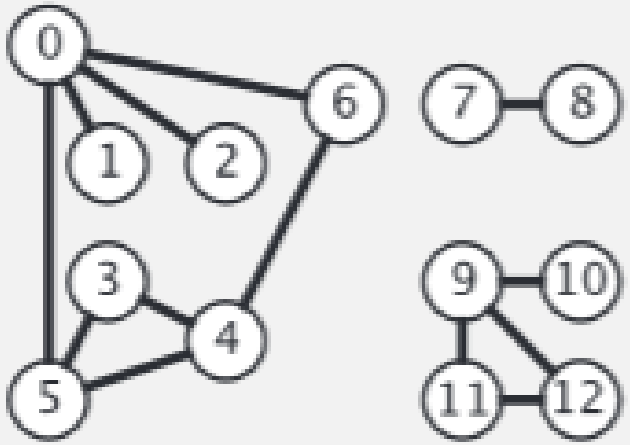
Repeat until the queue is empty:

- remove the least recently added vertex v
- add each of v's unvisited neighbors to the queue, and mark them as visited.

Connected components

Initialize all vertices v as unmarked.

For each unmarked vertex v, run DFS to identify all vertices discovered as part of the same component.



Challenge 1

- Determine if a graph is bipartite
- G = (U,V,E)
  - U and V are two independent sets whose union is the vertex set
  - Each edge in E connects a vertex in U to a vertex in V

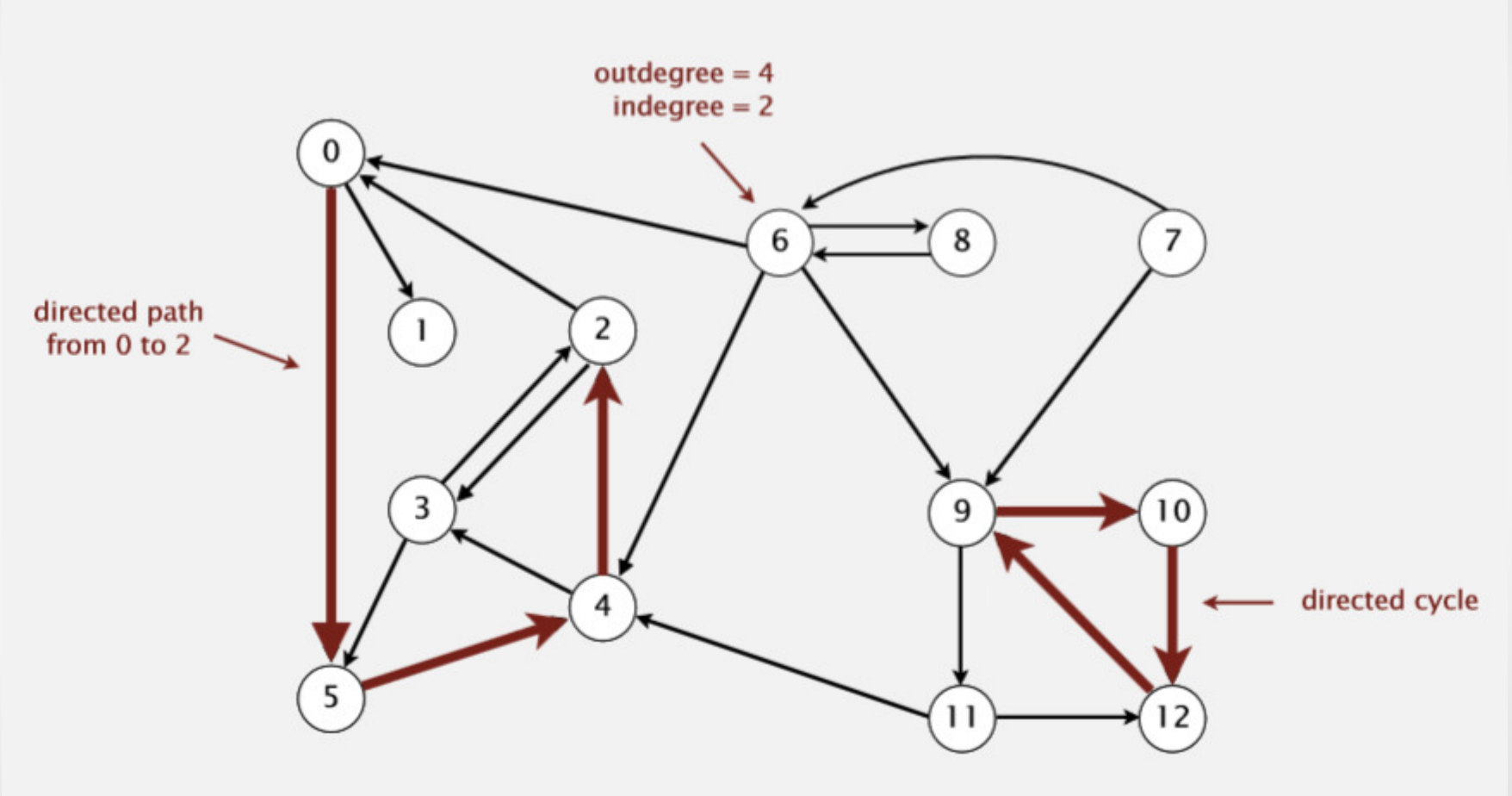
Challenges

- Find a cycle in a graph (if any)
- Find a cycle in a graph that uses every edge exactly once (Euler path)
- Find a cycle that uses every vertex exactly once (Hamilton cycle)
- Are two graphs isomorphic?
- Layout a graph on a plane w/o crossing edges (Tarjan, 1970)

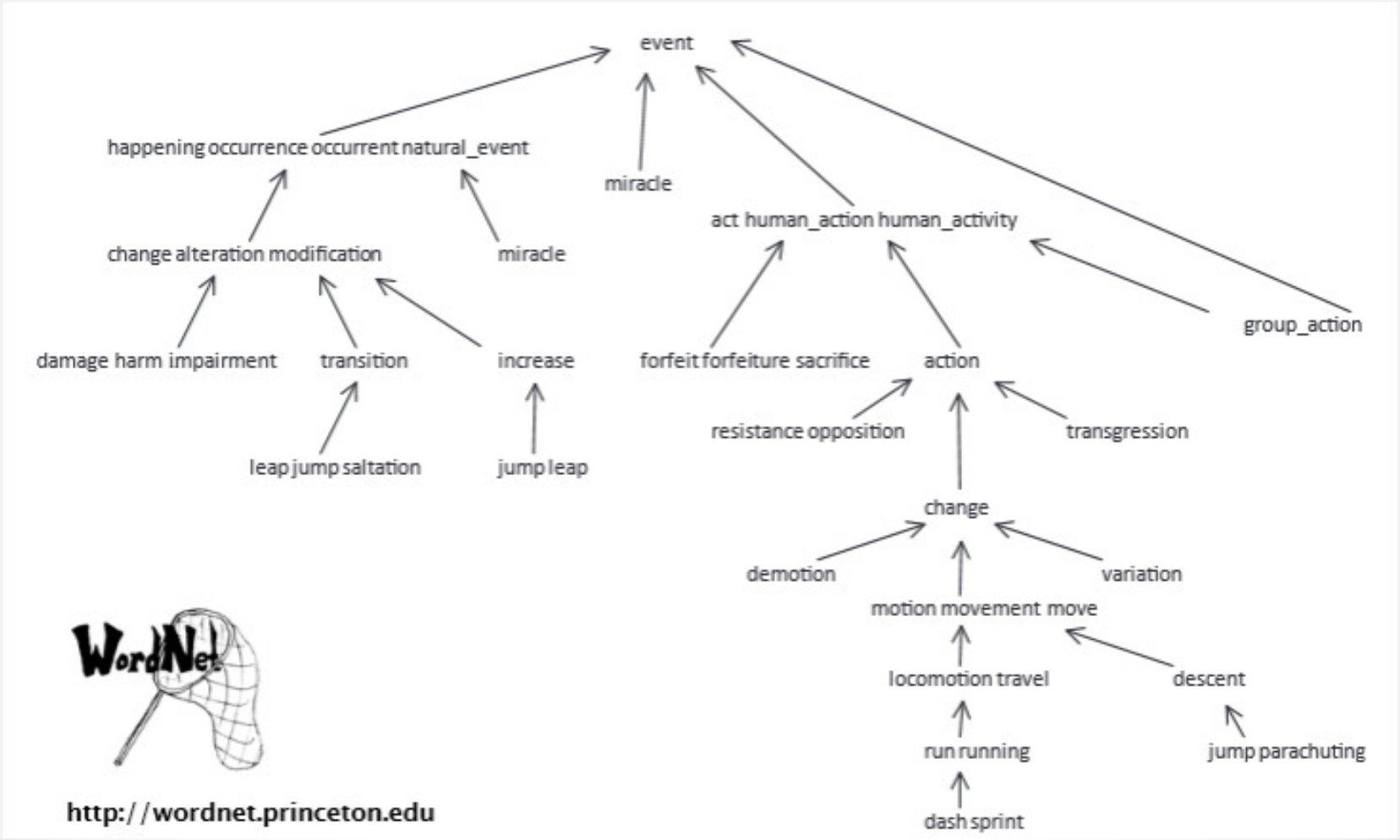
What we know

problem	BFS	DFS	time
path between s and t	✔	✔	$E+V$
shortest path between s and t	✔		$E+V$
connected components	✔	✔	$E+V$
biconnected components		✔	$E+V$
cycle	✔	✔	$E+V$
Euler cycle		✔	$E+V$
Hamilton cycle			$2^{1.657\,V}$
bipartiteness	✔	✔	$E+V$
planarity		✔	$E+V$
graph isomorphism			$2^{c\sqrt{V \log V}}$

Directed Graphs



Vertex = synset; edge = hypernym relationship.

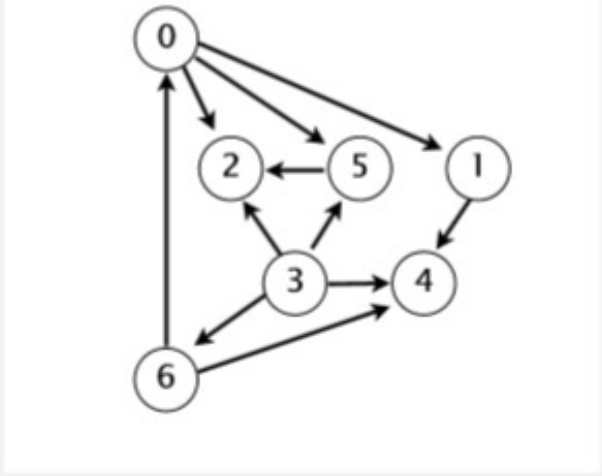


http://wordnet.princeton.edu

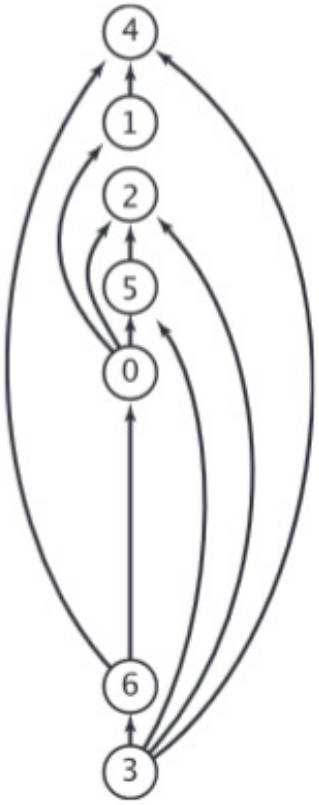
representation

representation	space	insert edge from v to w	edge from v to w?	iterate over vertices pointing from v?
list of edges	E	1	E	E
adjacency matrix	V^2	$1^\dagger$	1	V
adjacency lists	$E + V$	1	$outdegree(v)$	$outdegree(v)$

Topological sort



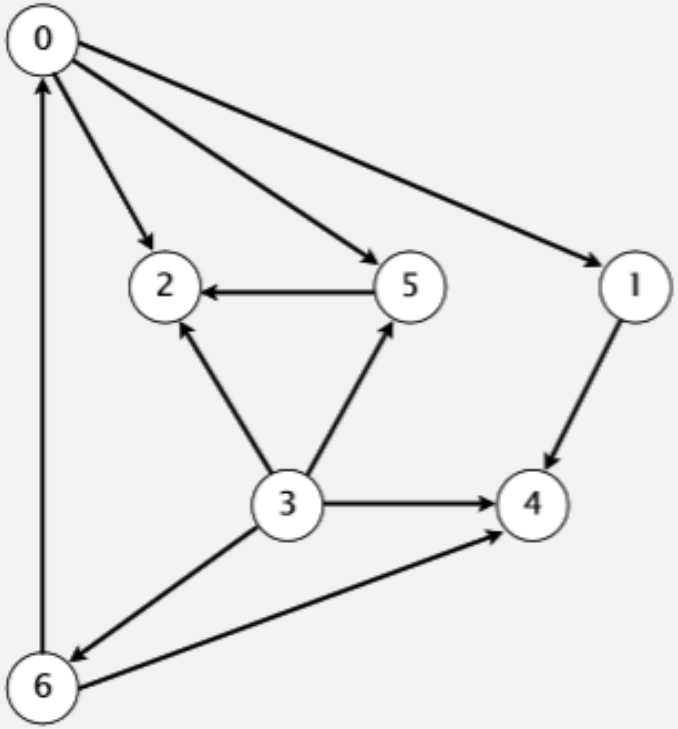
precedence constraint graph



feasible schedule

Find a topological sort

- Run depth-first search.
- Return vertices in reverse postorder.



postorder

4 1 2 5 0 6 3

topological order

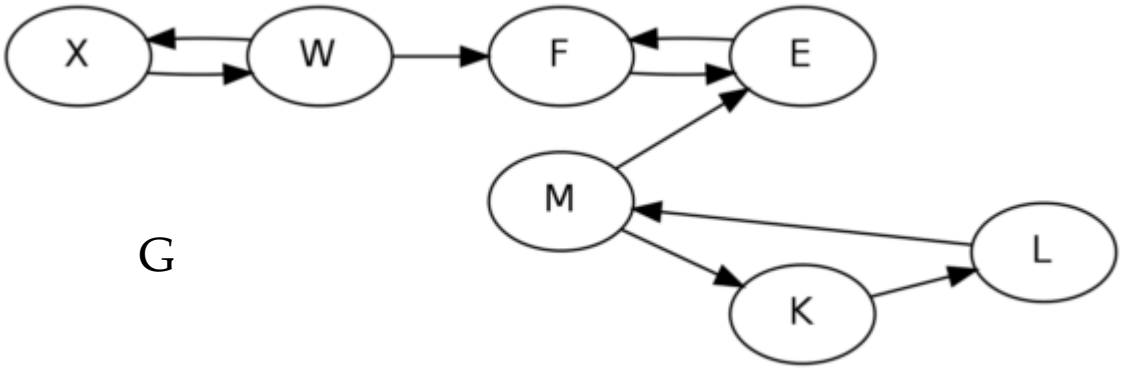
3 6 0 5 2 1 4

DFS orderings

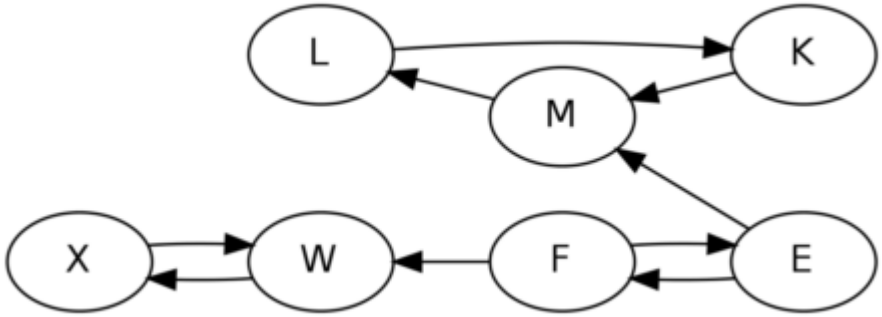
Orderings.

- Preorder: order in which `dfs()` is called.
- Postorder: order in which `dfs()` returns.
- Reverse postorder: reverse order in which `dfs()` returns.

What is the relation between these two graphs?



G



G^R