NPTEL MOOC,JAN-FEB 2015
Week 3, Module 5

DESIGN AND ANALYSIS OF ALGORITHMS

Applications of BFS and DFS

MADHAVAN MUKUND, CHENNAI MATHEMATICAL INSTITUTE
http://www.cmi.ac.in/~madhavan

Graphs, formally

G = (V,E)

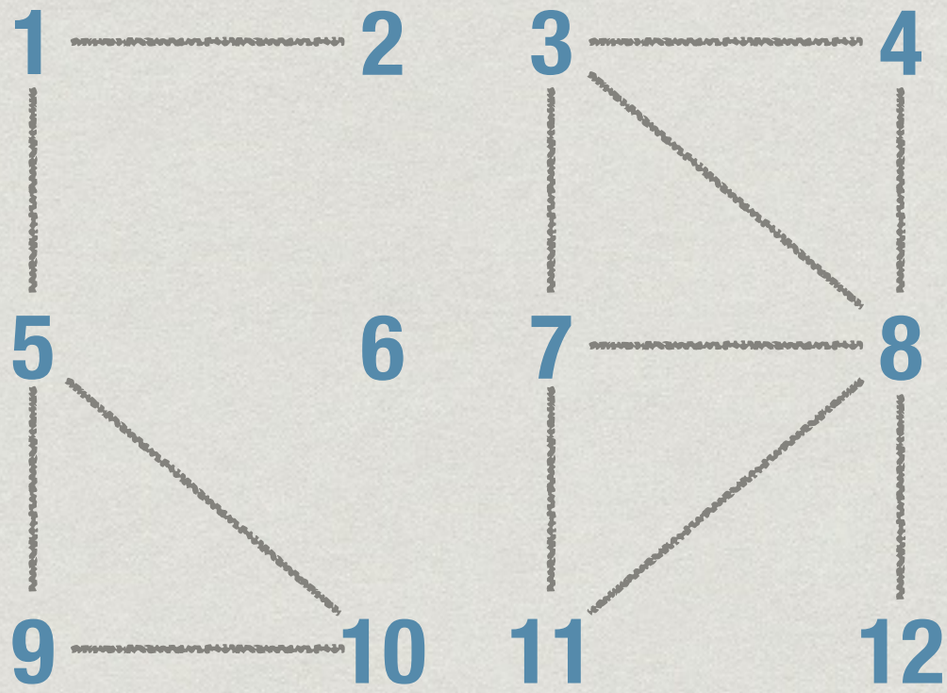
- Set of vertices V
- Set of edges E
  - E is a subset of pairs (v,v'): $E \subseteq V \times V$
  - Undirected graph: (v,v') and (v',v) are the same edge
  - Directed graph:
    - (v,v') is an edge from v to v'
    - Does not guarantee that (v',v) is also an edge

Exploring graph structure

- Breadth first search
  - Level by level exploration
- Depth first search
  - Explore each vertex as soon as it is visited
  - DFS numbering
- What can we find out about a graph using BFS/ DFS?

Connectivity

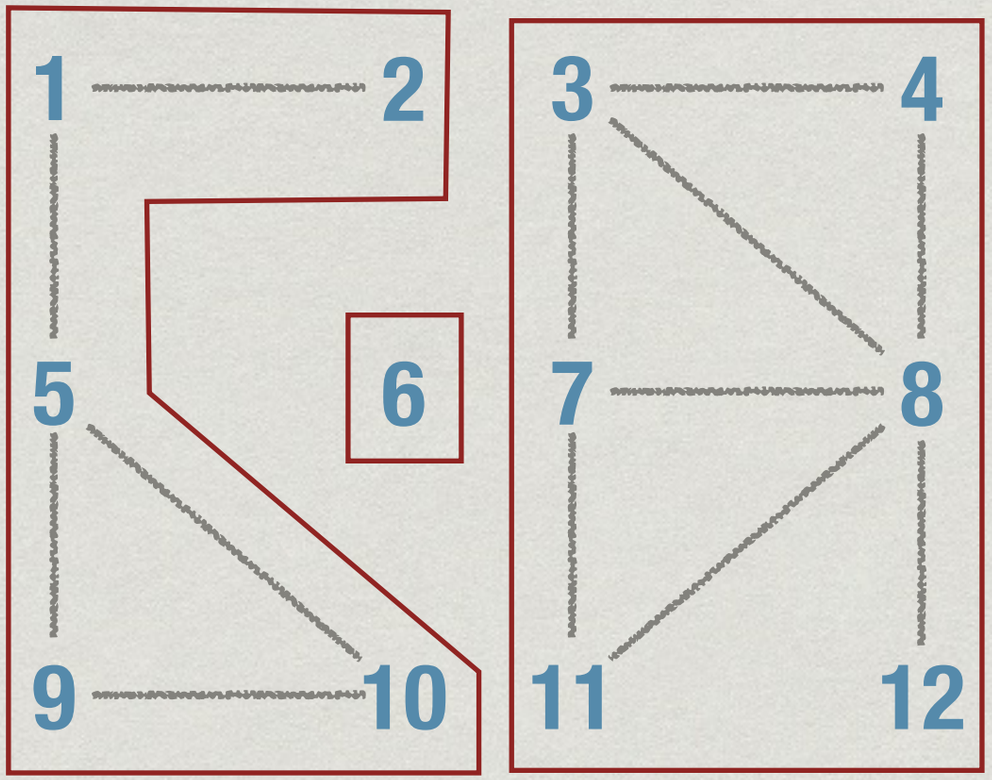
Connected graph



Disconnected graph

Connectivity

Connected graph



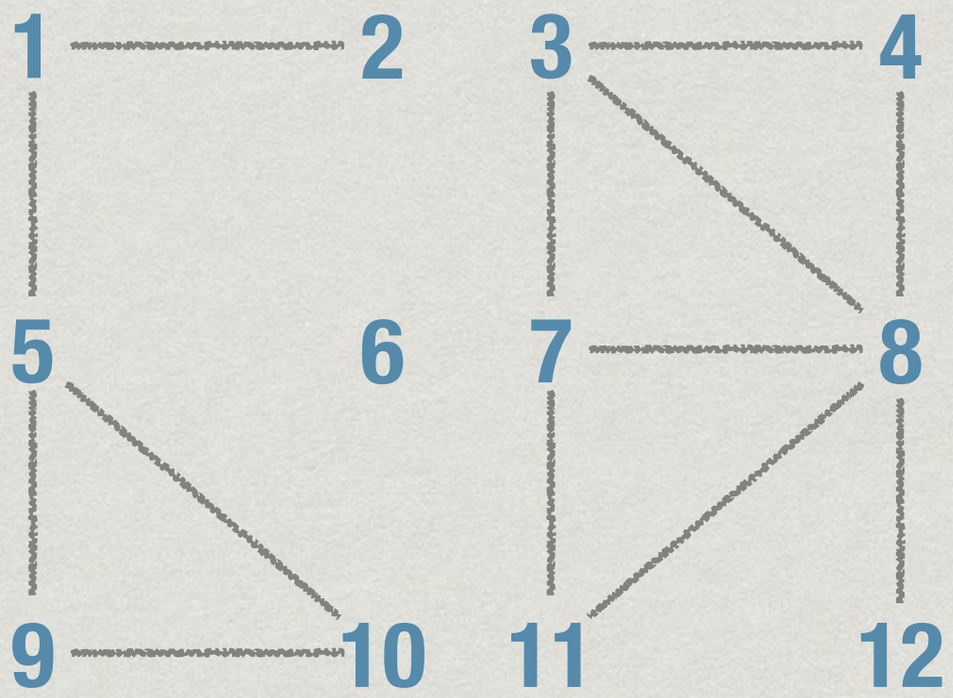
Disconnected graph

Connected components

Identifying connected components

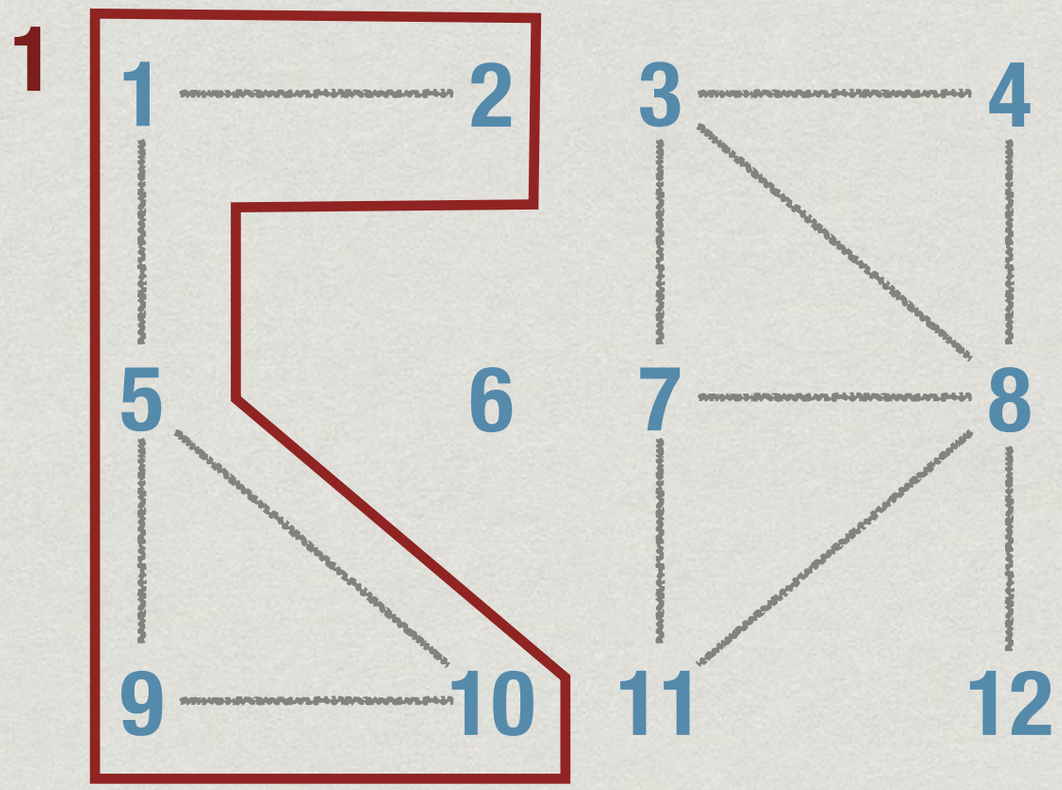
- Vertices {1,2,…,N}
- Start BFS or DFS from 1
  - All nodes marked Visited form a connected component
  - Pick first unvisited node, say j, and run BFS or DFS from j
  - Repeat till all nodes are visited
- Update BFS/DFS to label each visited node with component number

Connected components



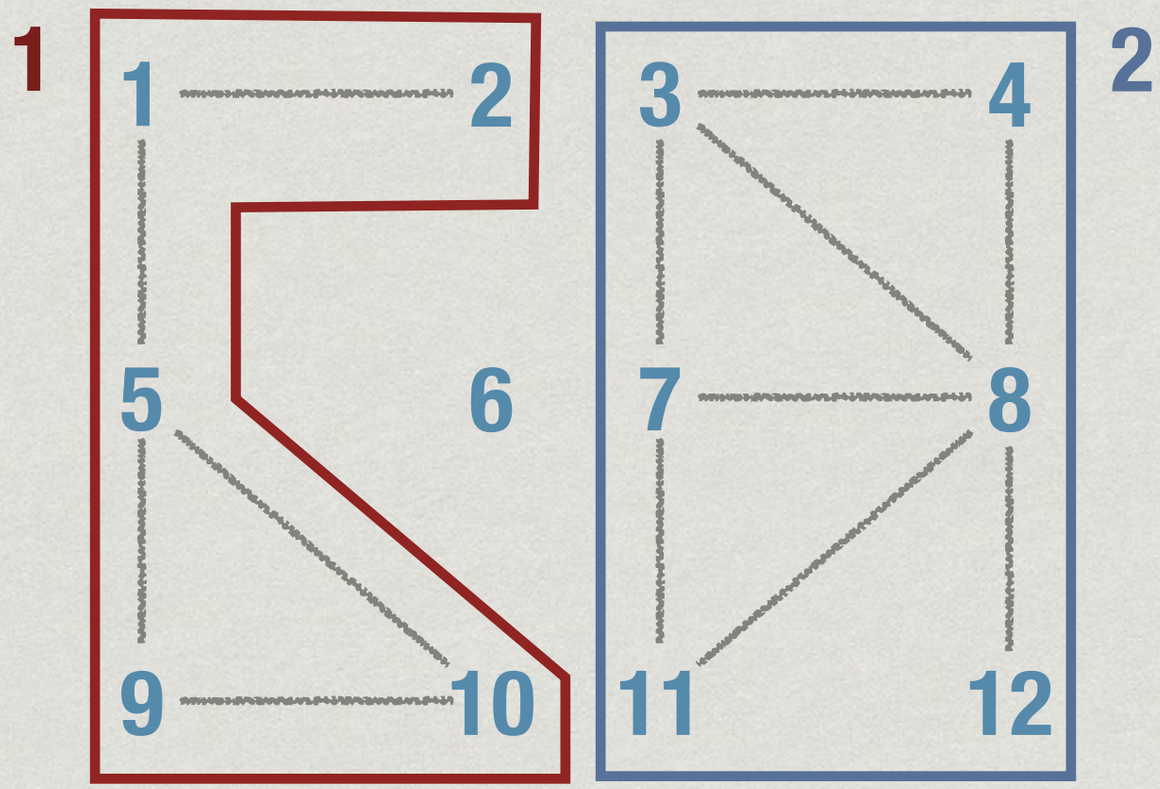
- Add a counter `comp` to number components
- Increment counter each time a fresh BFS/DFS starts
- Label each visited node `j` with `component[j] = comp`

Connected components



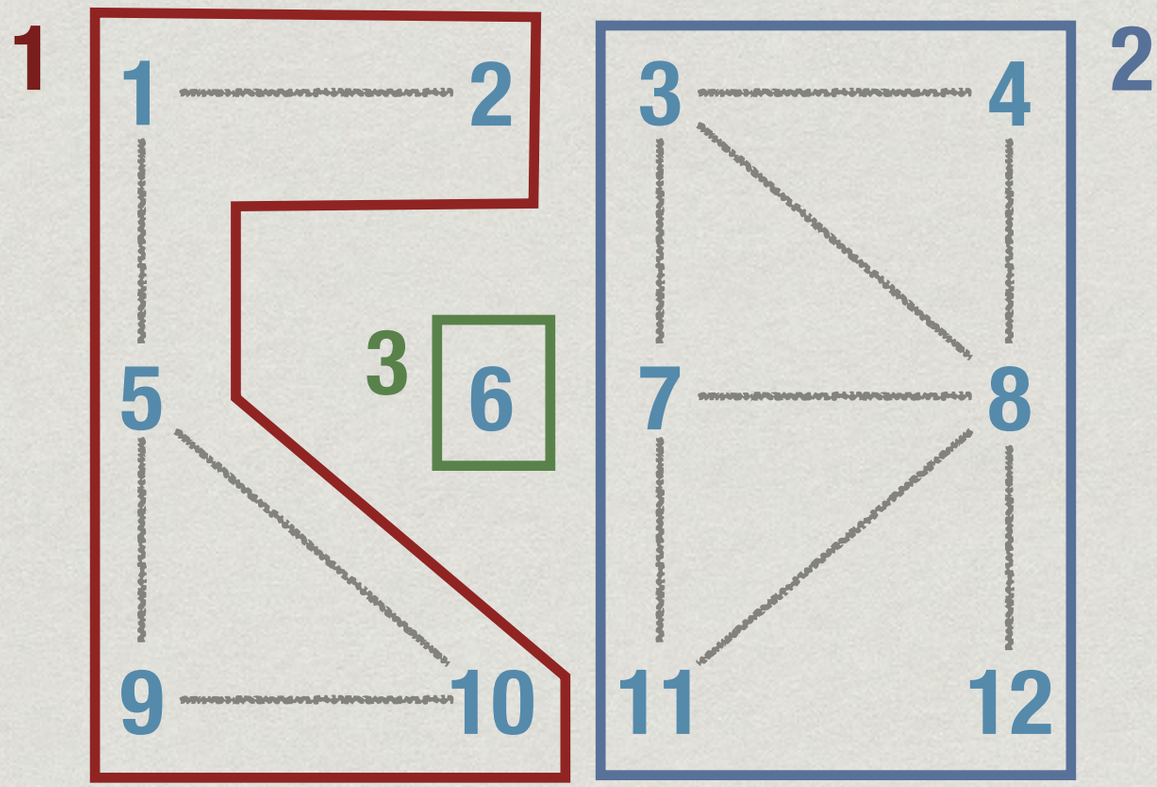
- Add a counter `comp` to number components
- Increment counter each time a fresh BFS/DFS starts
- Label each visited node `j` with `component[j] = comp`

Connected components



- Add a counter `comp` to number components
- Increment counter each time a fresh BFS/DFS starts
- Label each visited node `j` with `component[j] = comp`

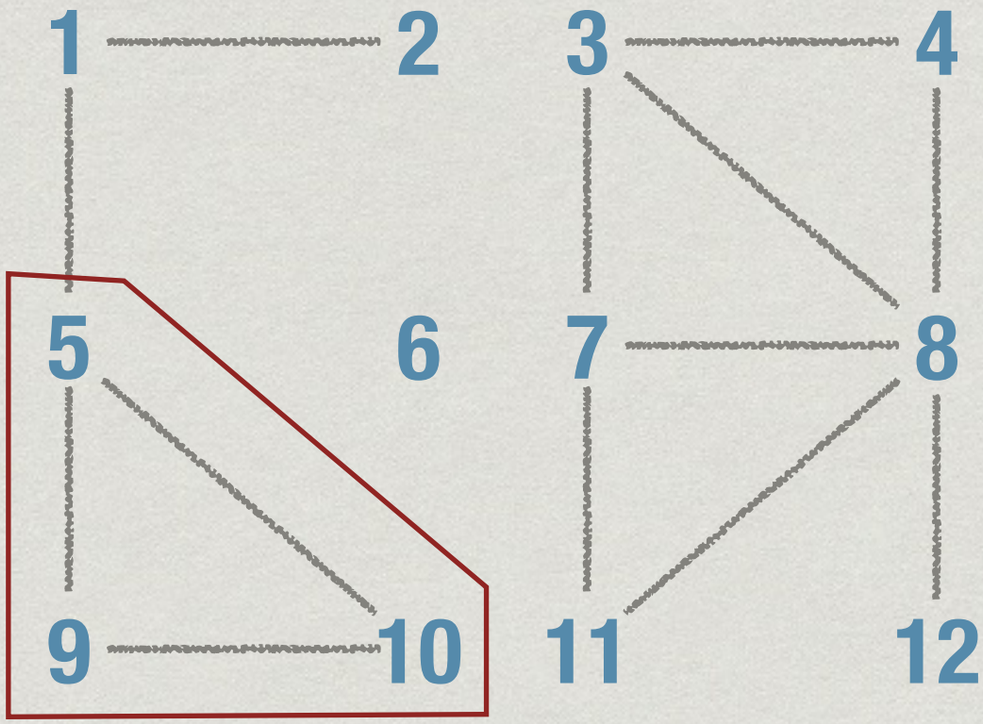
Connected components



- Add a counter `comp` to number components
- Increment counter each time a fresh BFS/DFS starts
- Label each visited node `j` with `component[j] = comp`

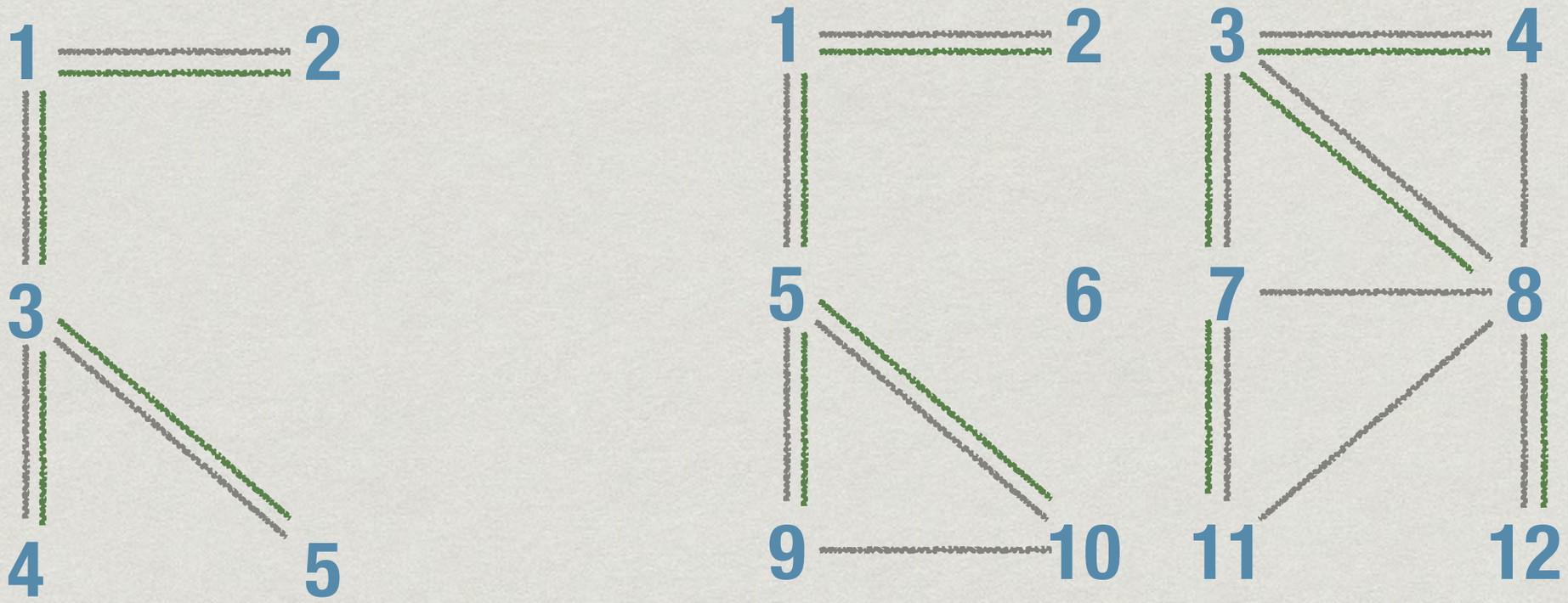
Cycles

Acyclic graph



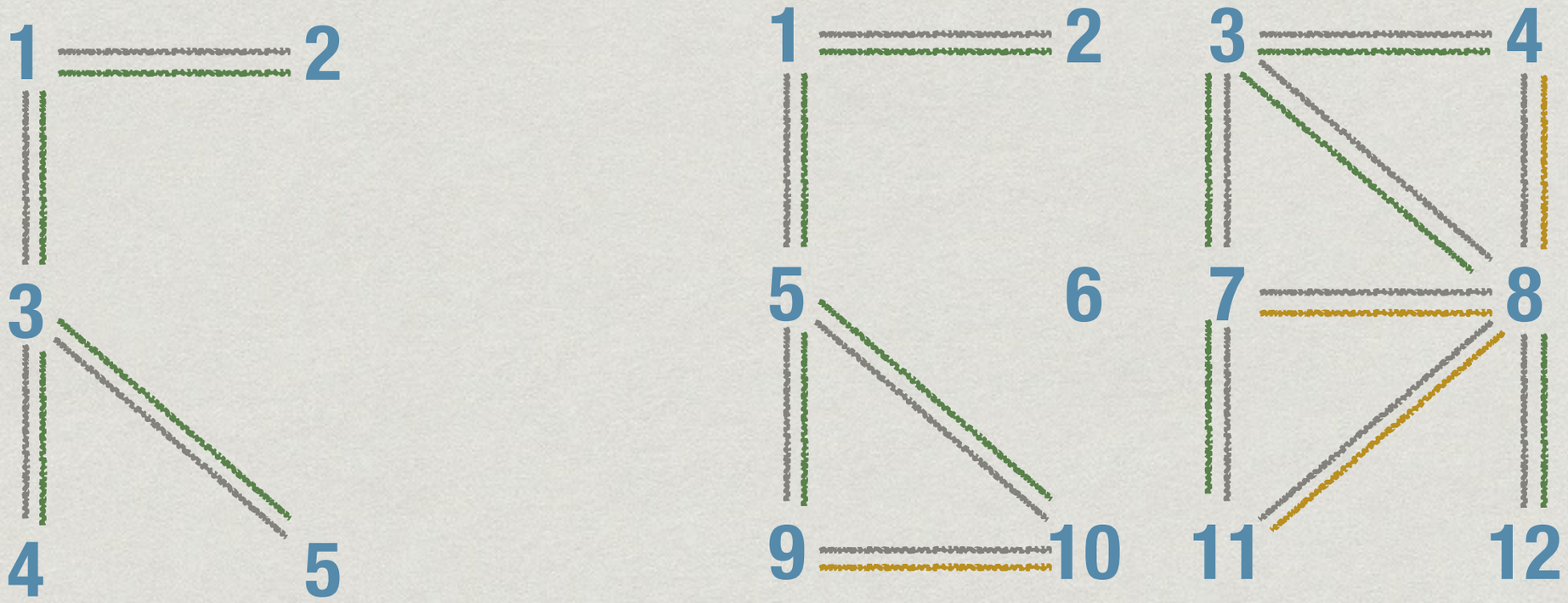
Graph with cycles

BFS tree



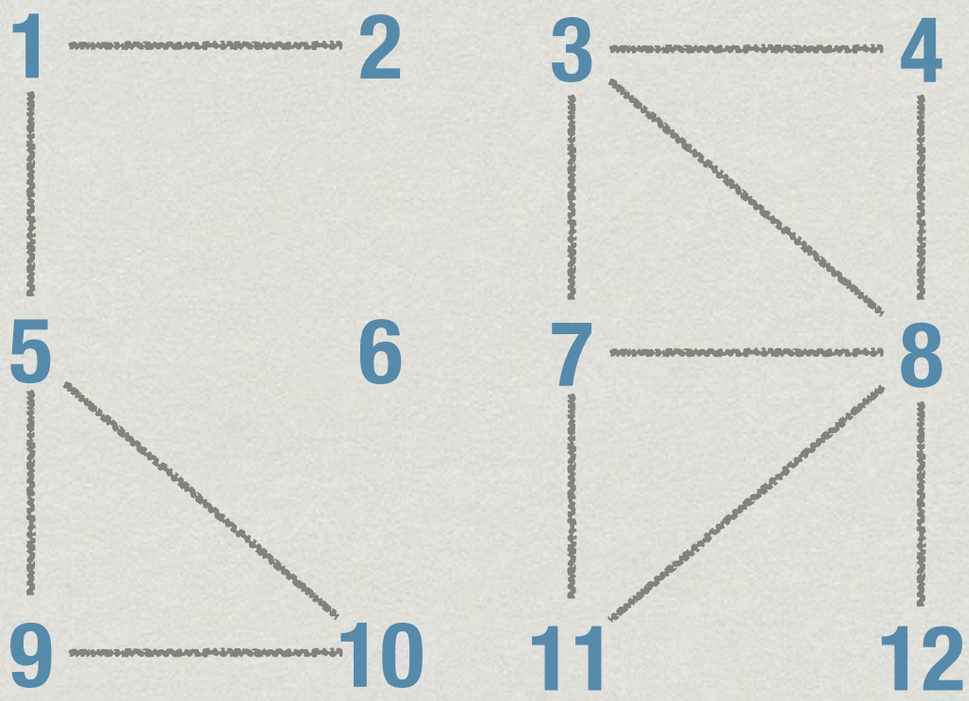
- Edges explored by BFS form a tree
  - Acyclic graph = connected, with n-1 edges

BFS tree

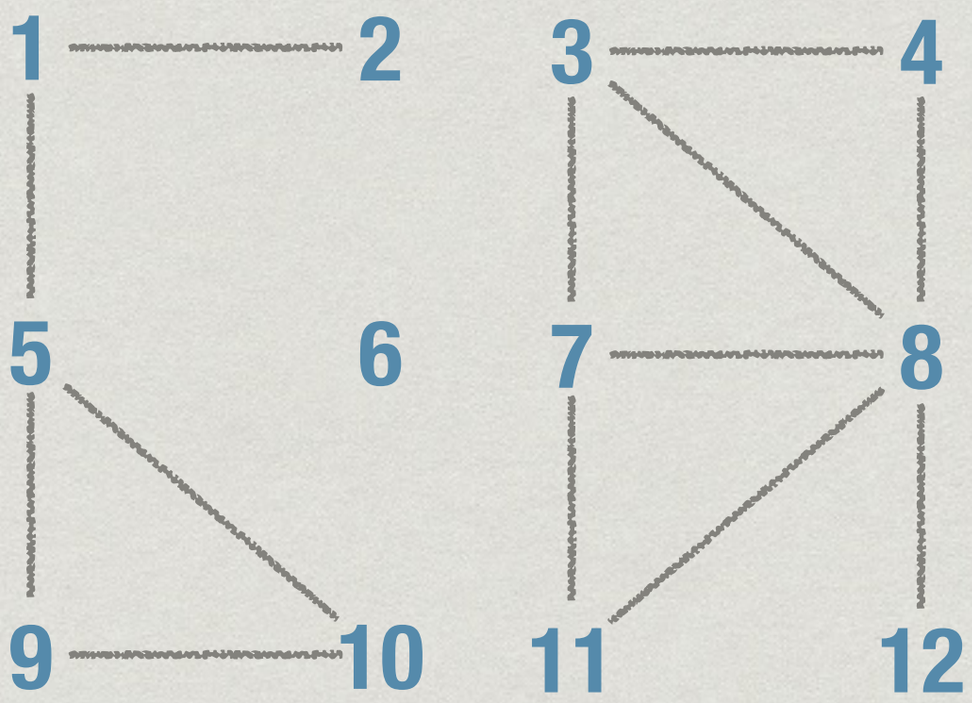


- Edges explored by BFS form a tree
  - Acyclic graph = connected, with n-1 edges
- Any non-tree edge generates a cycle

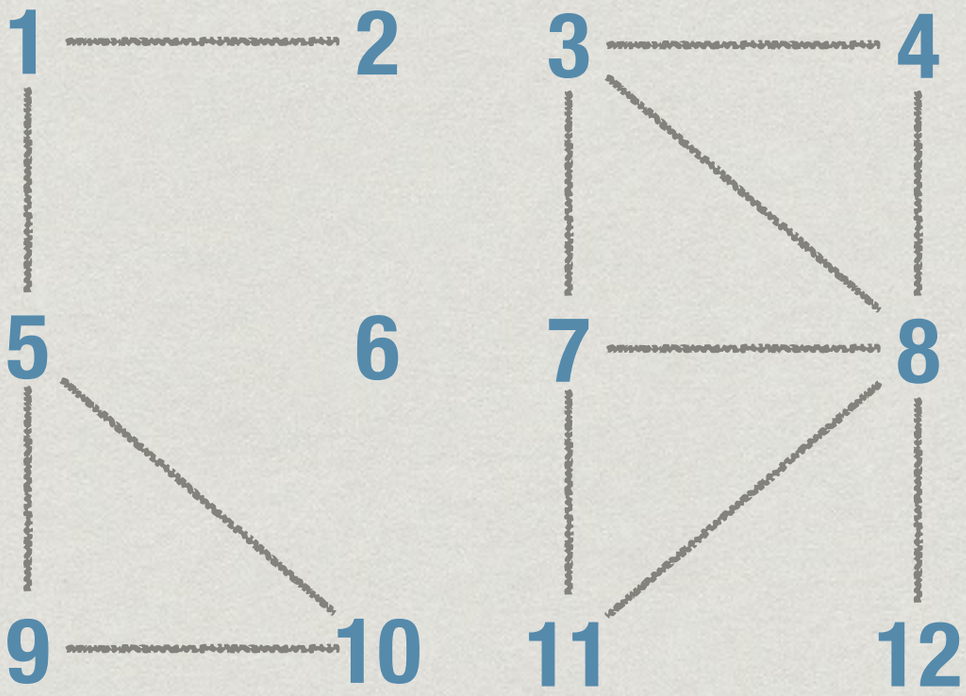
DFS tree



DFS tree



DFS tree

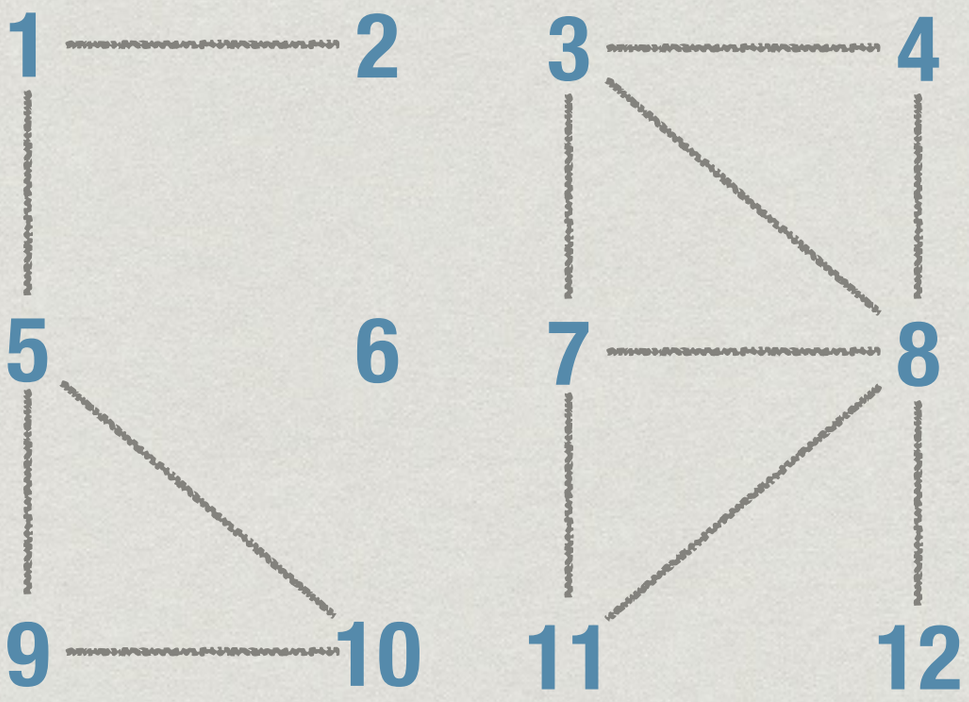


pre

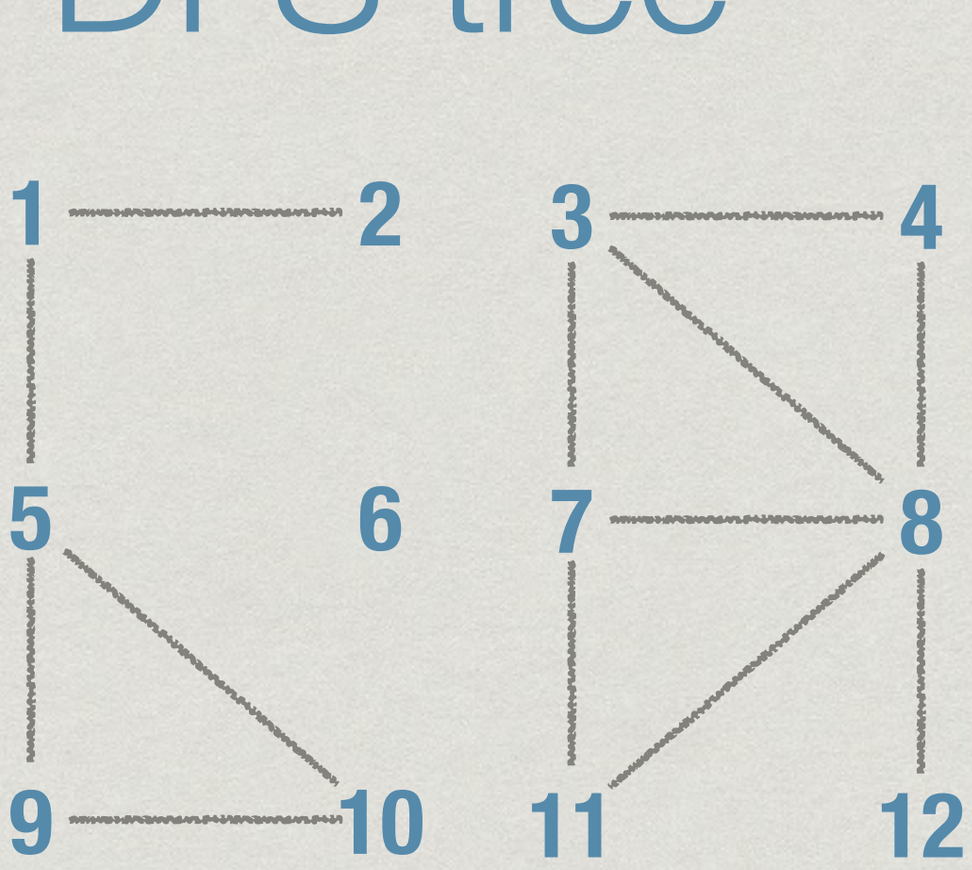
0

1

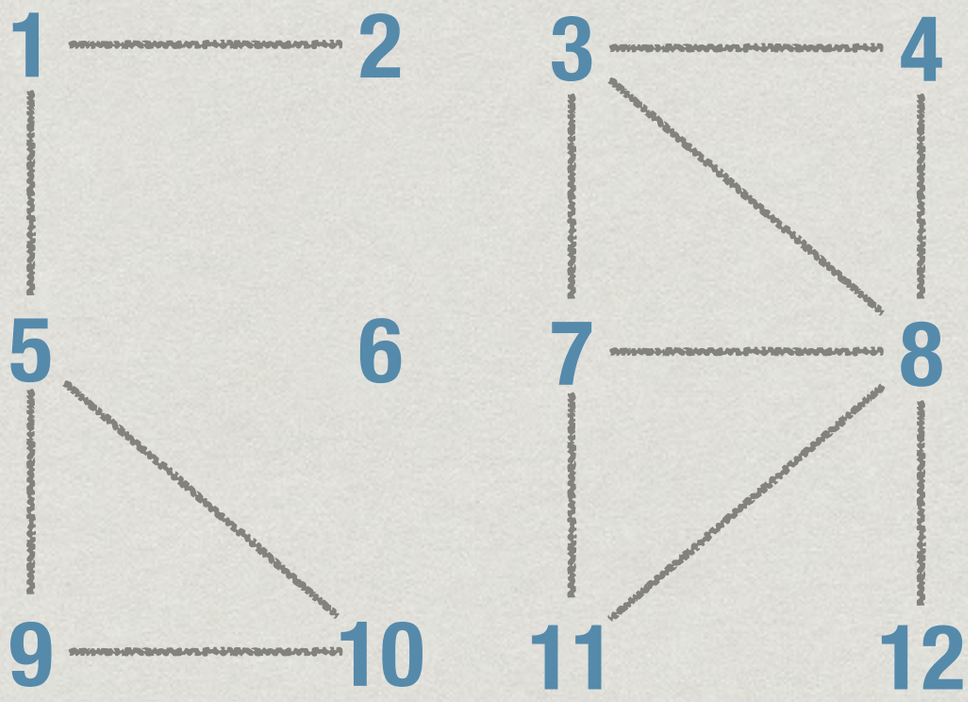
DFS tree

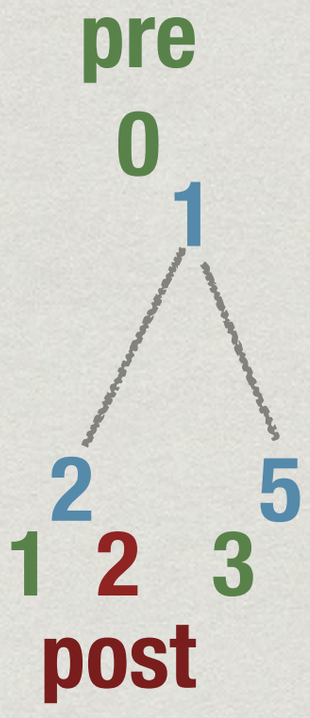


DFS tree

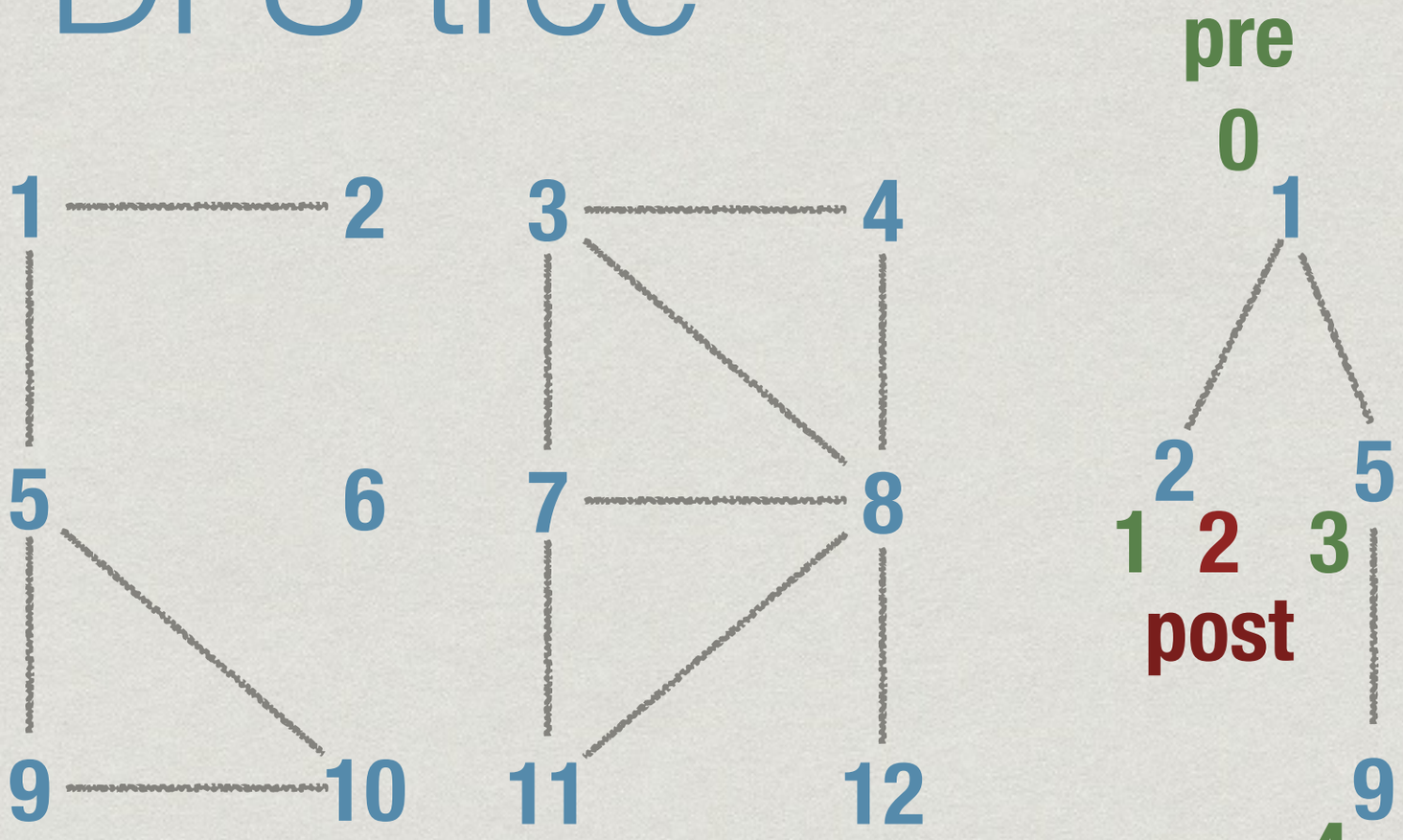


DFS tree

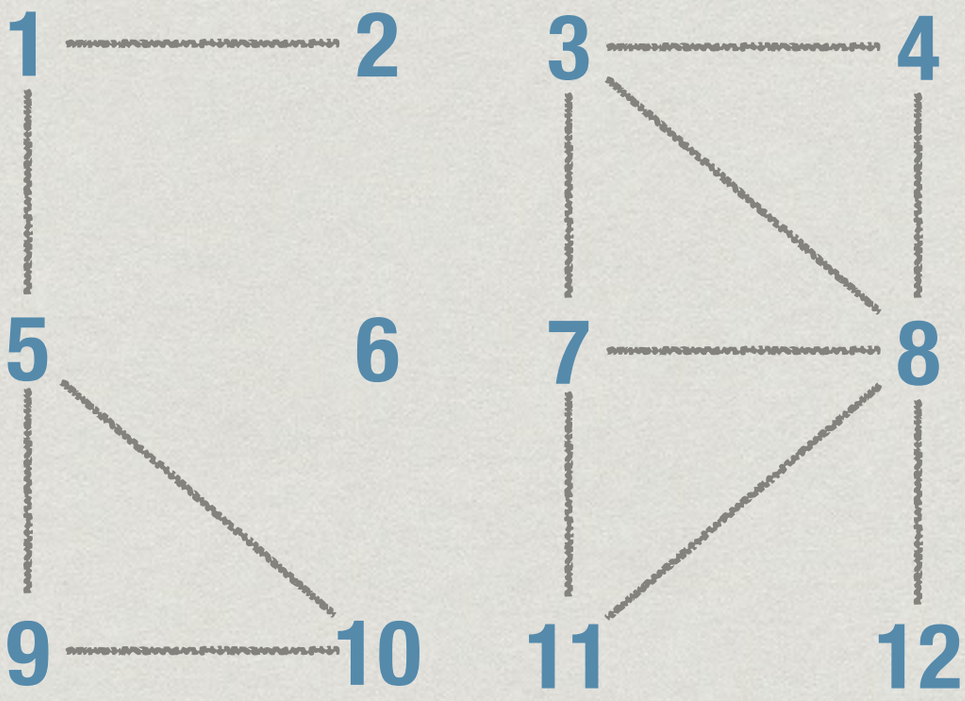




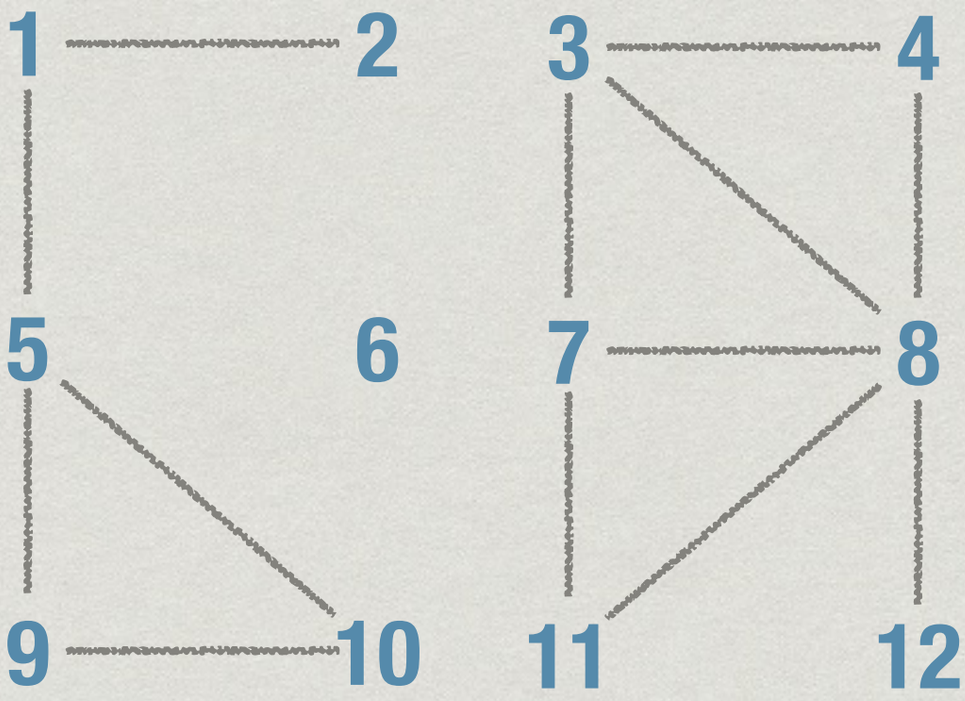
DFS tree



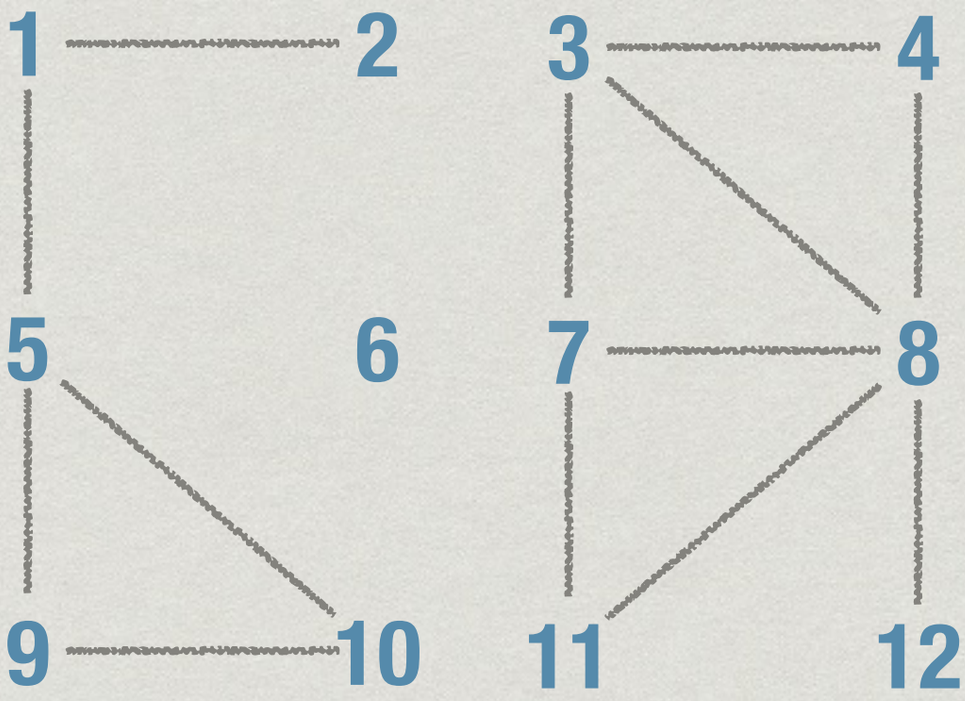
DFS tree



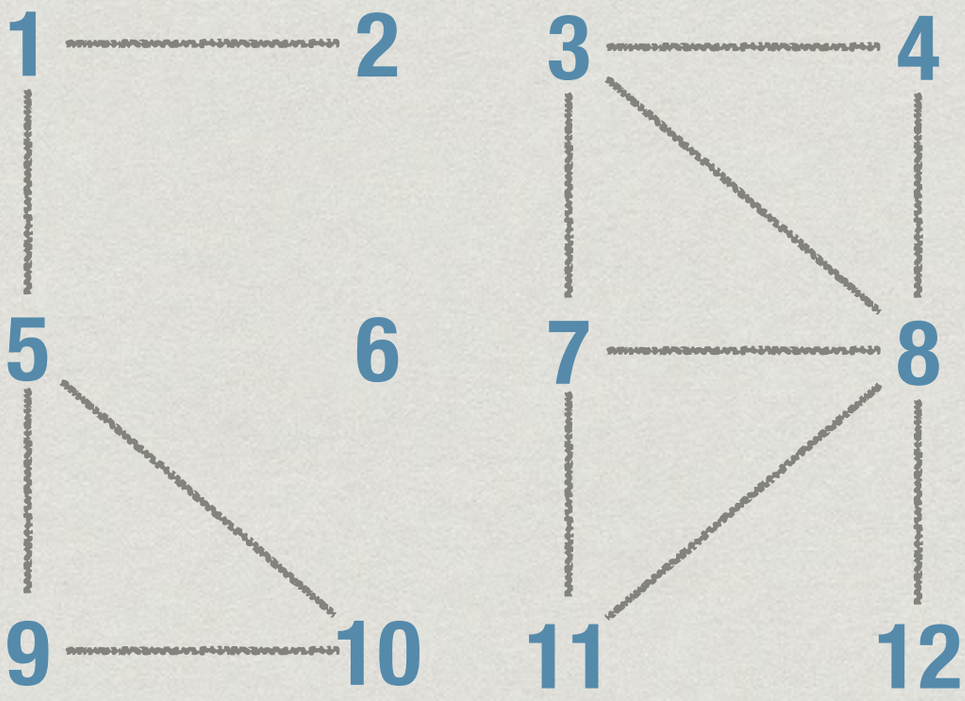
DFS tree



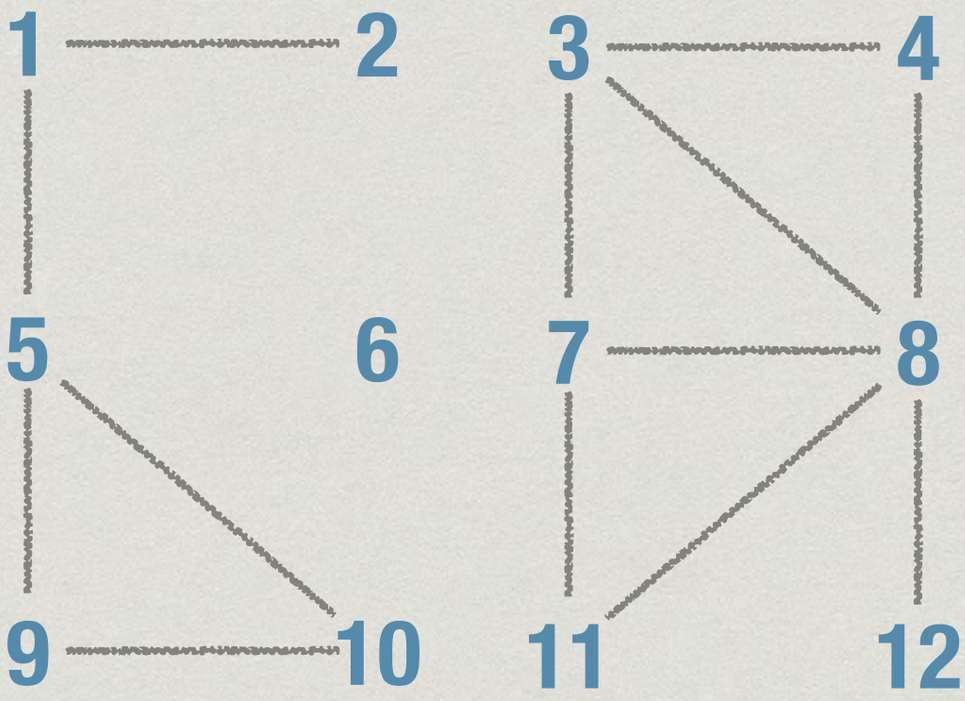
DFS tree



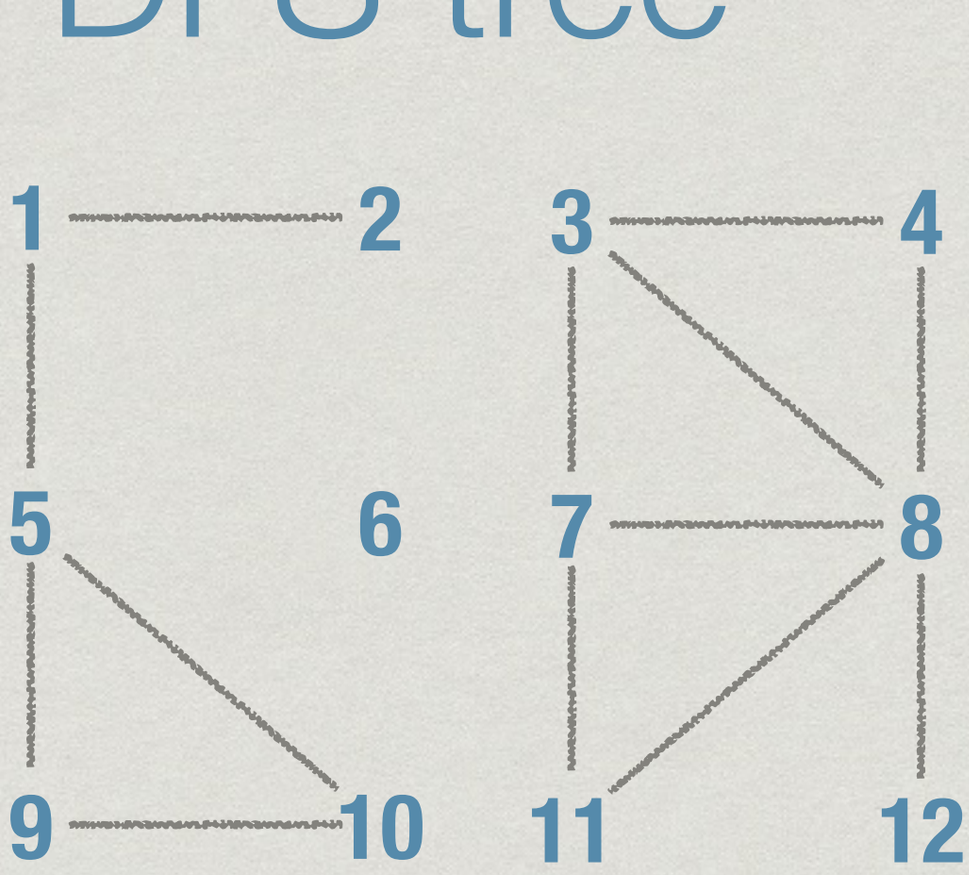
DFS tree

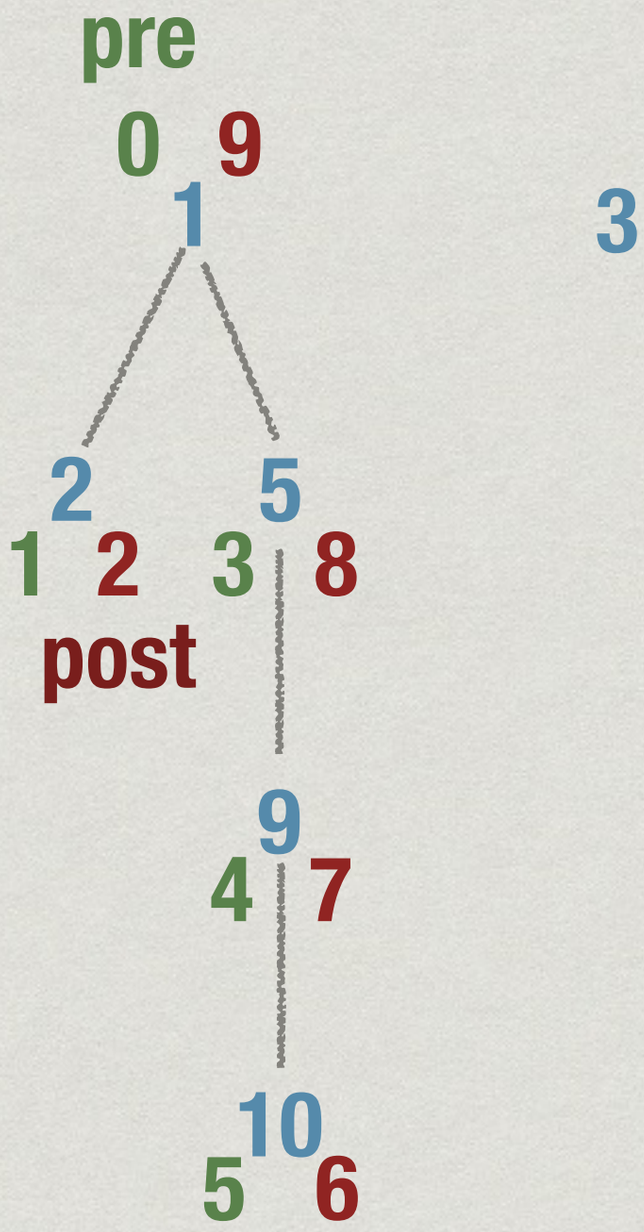


DFS tree

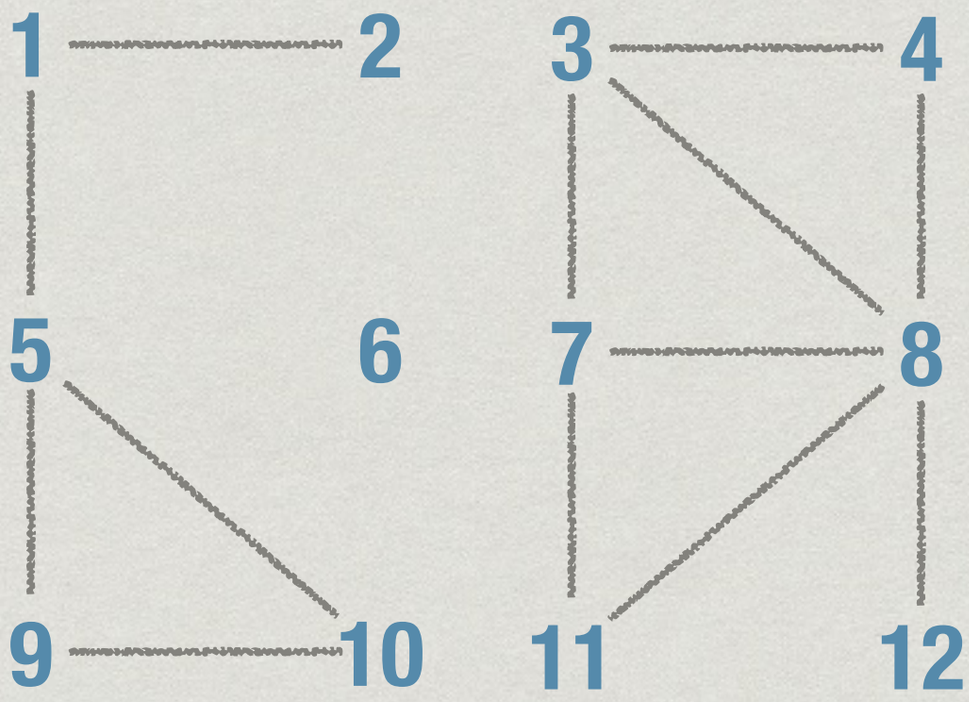


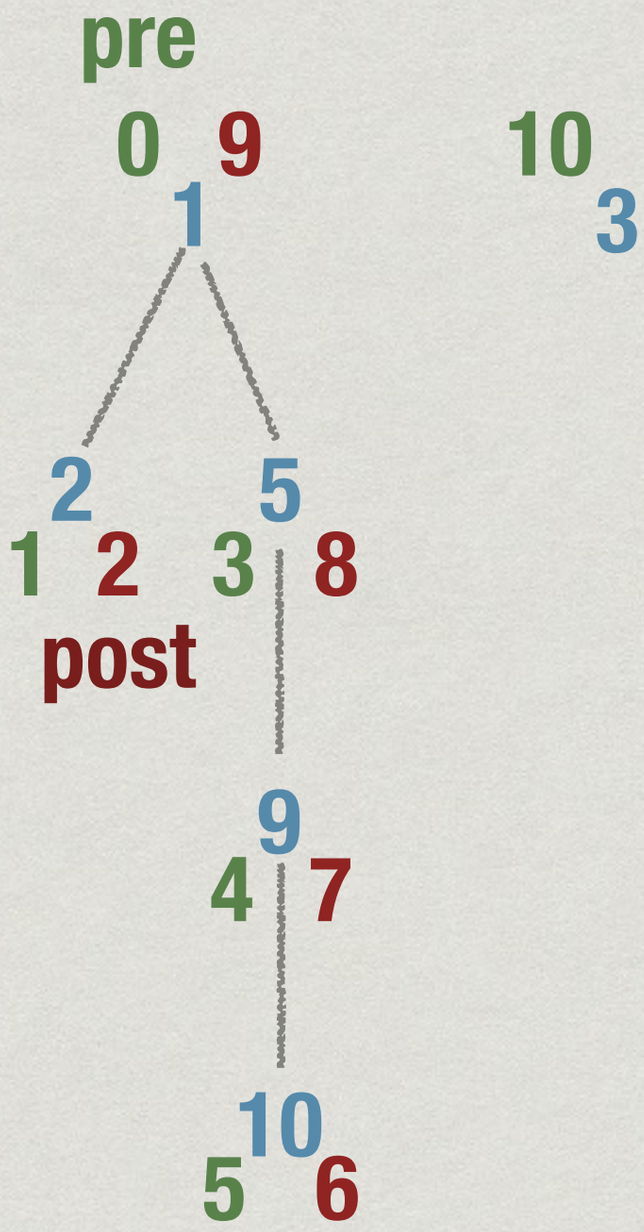
DFS tree



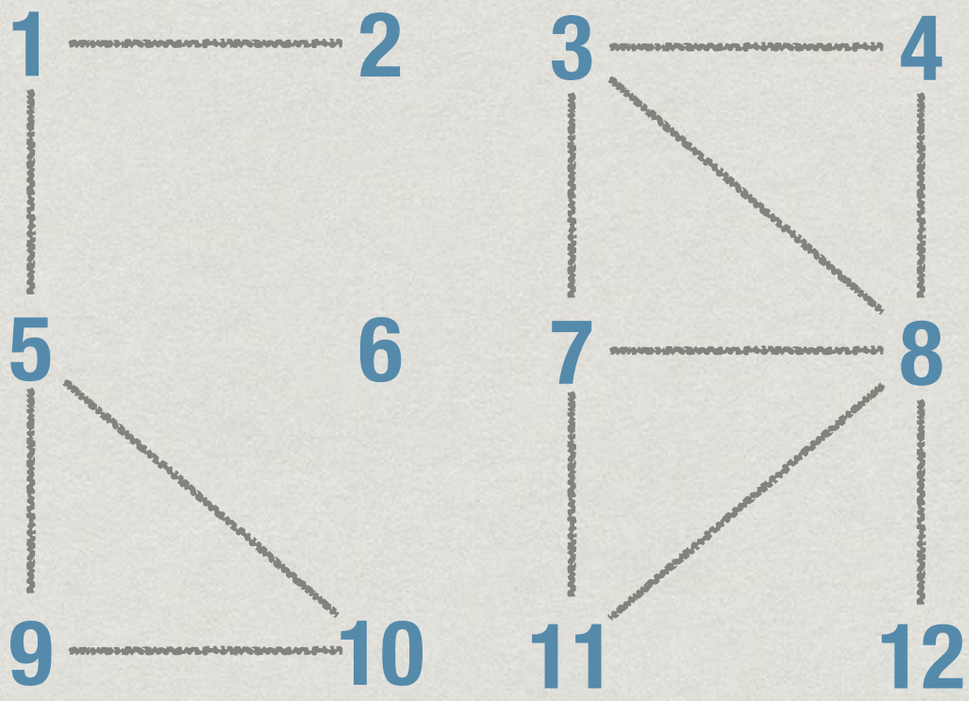


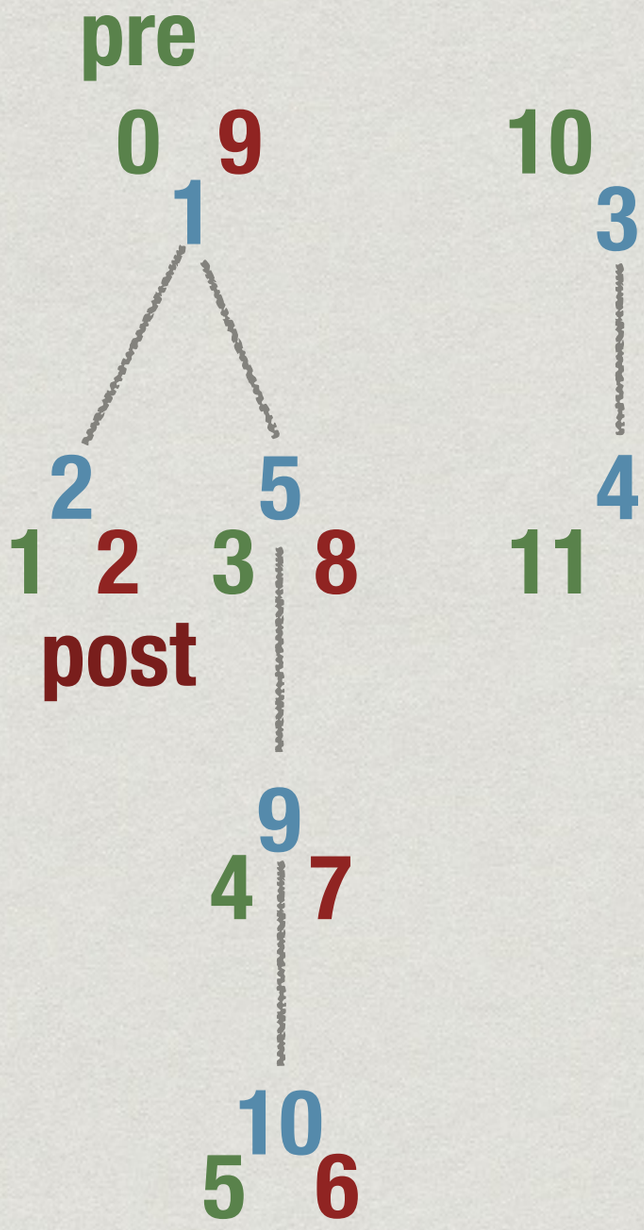
DFS tree



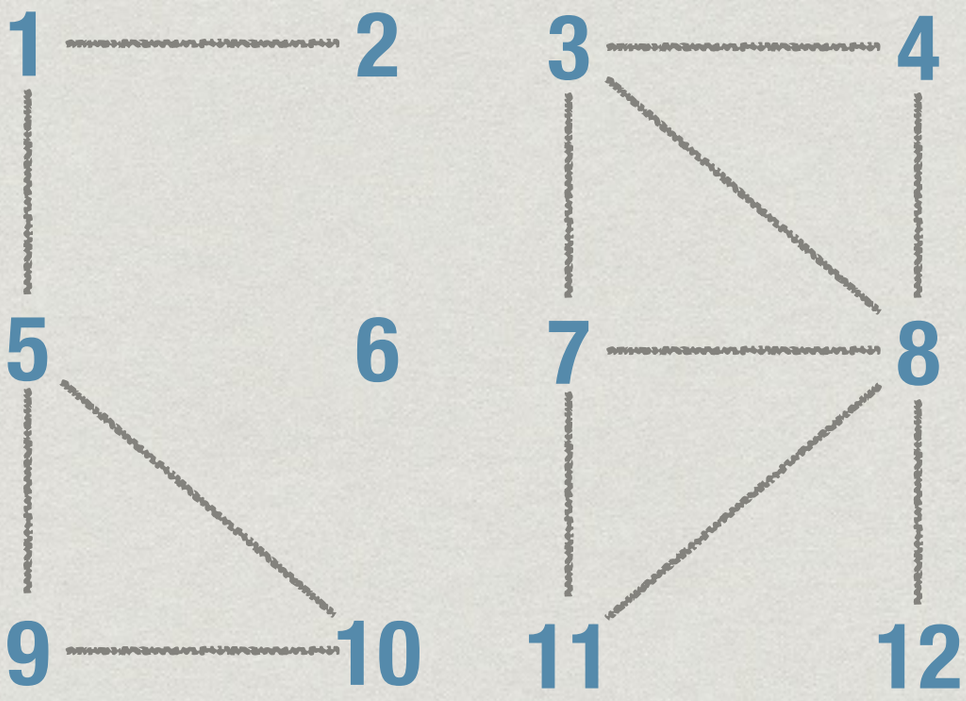


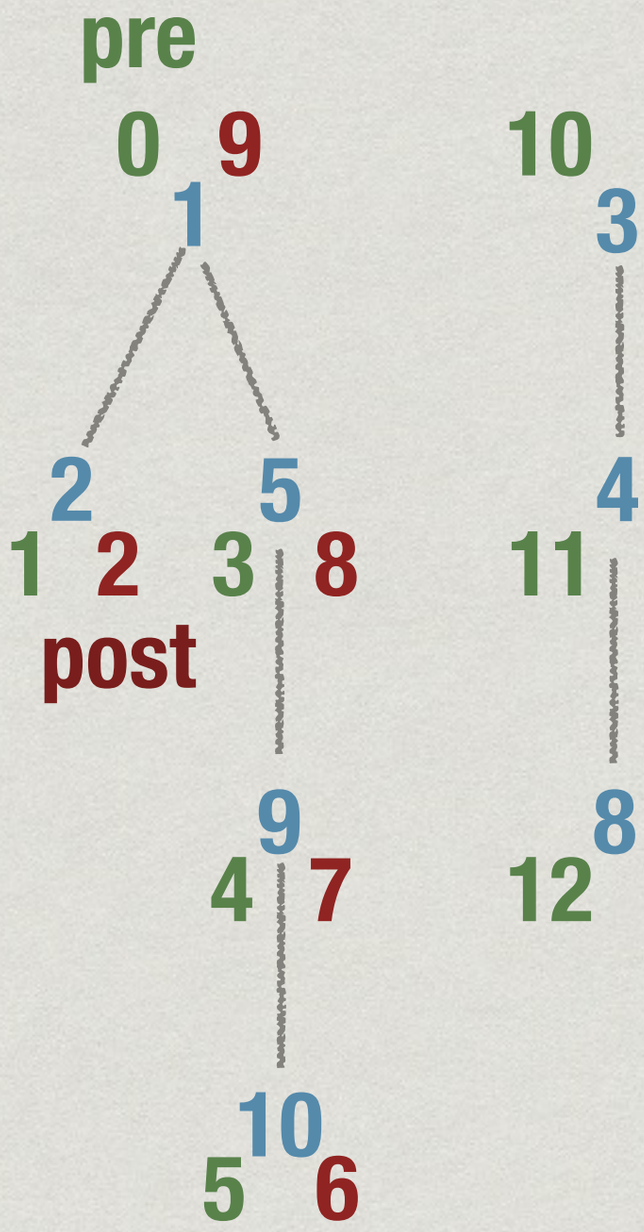
DFS tree



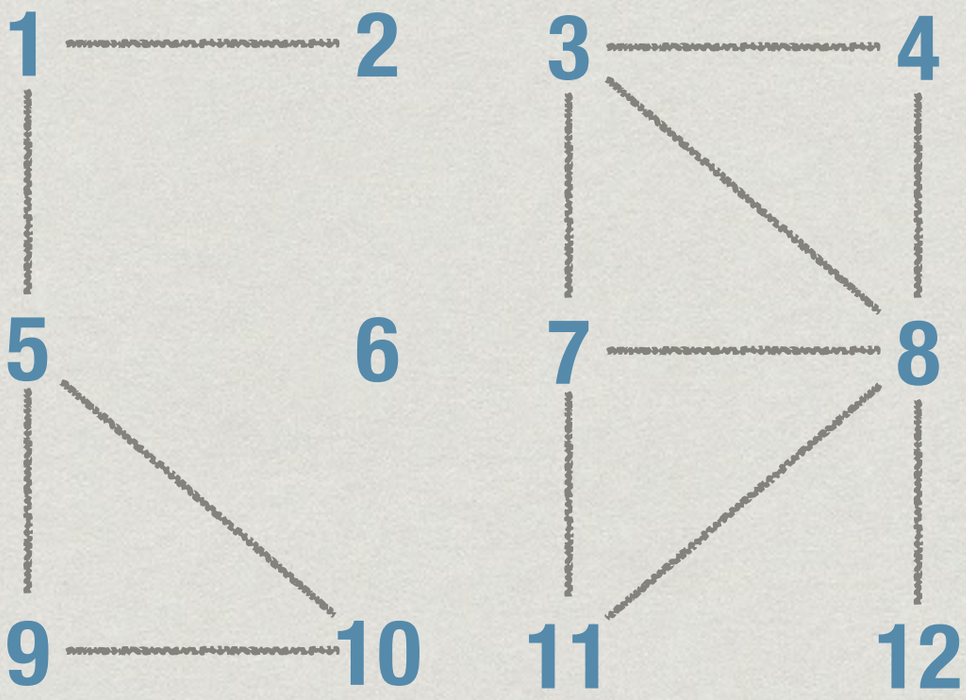


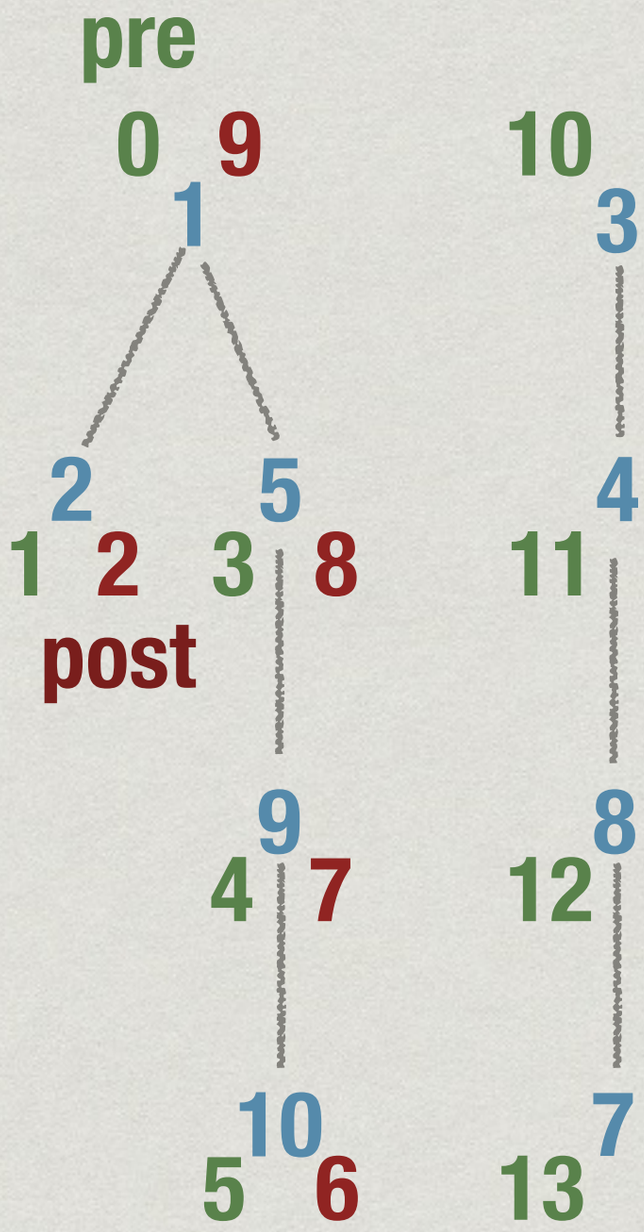
DFS tree





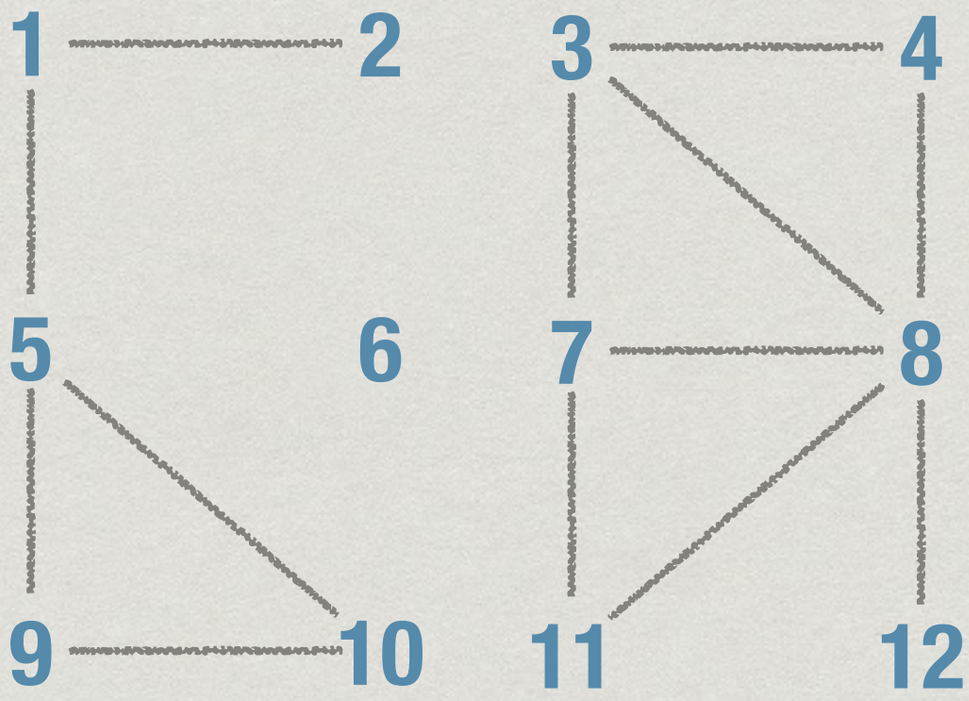
DFS tree


1
2
3
4
5
6
7
8
9
10
11
12


pre
0
9
1
2
5
1
2
3
8
post
9
4
7
10
5
6

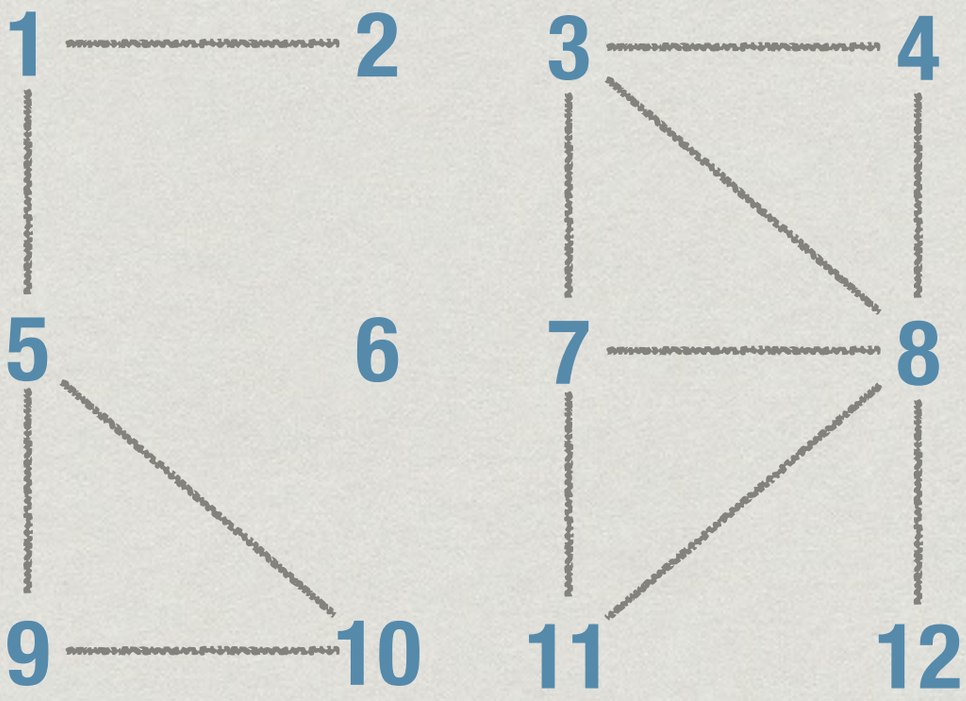
10
3
4
11
8
12
7
13

DFS tree


1
2
3
4
5
6
7
8
9
10
11
12


pre
0
9
1
2
5
1
2
3
8
post
9
4
7
10
5
6
10
3
4
11
8
12
7
13
11
14

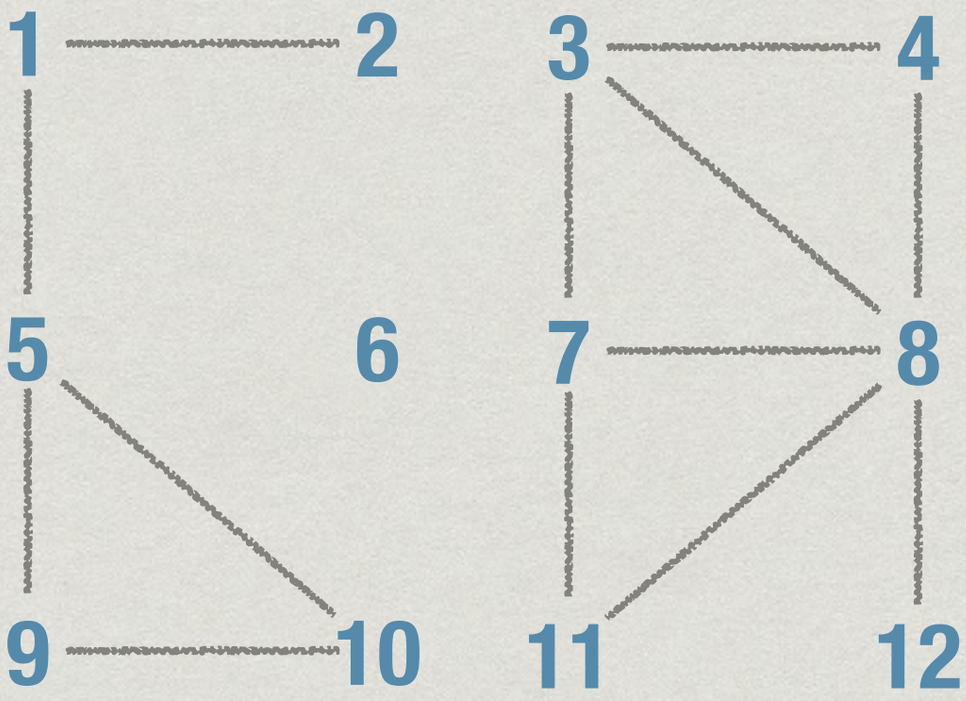
DFS tree


1
2
3
4
5
6
7
8
9
10
11
12


pre
0 9
1
2 5
1 2 3 8
post
9
4 7
10
5 6

10
3
4
11
8
12
7
13
11
14 15

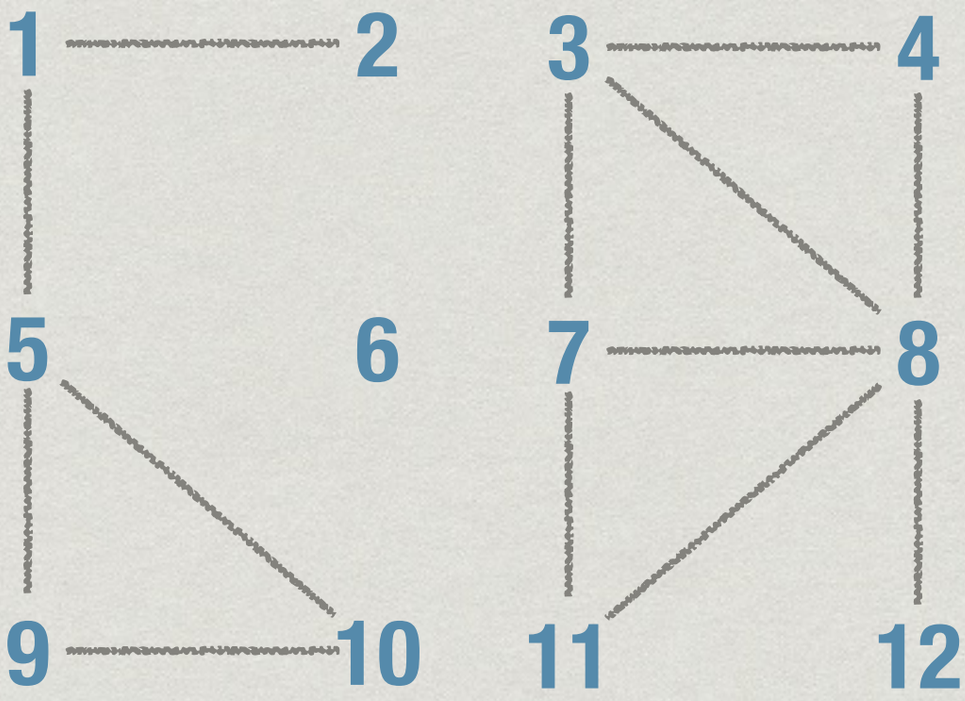
DFS tree

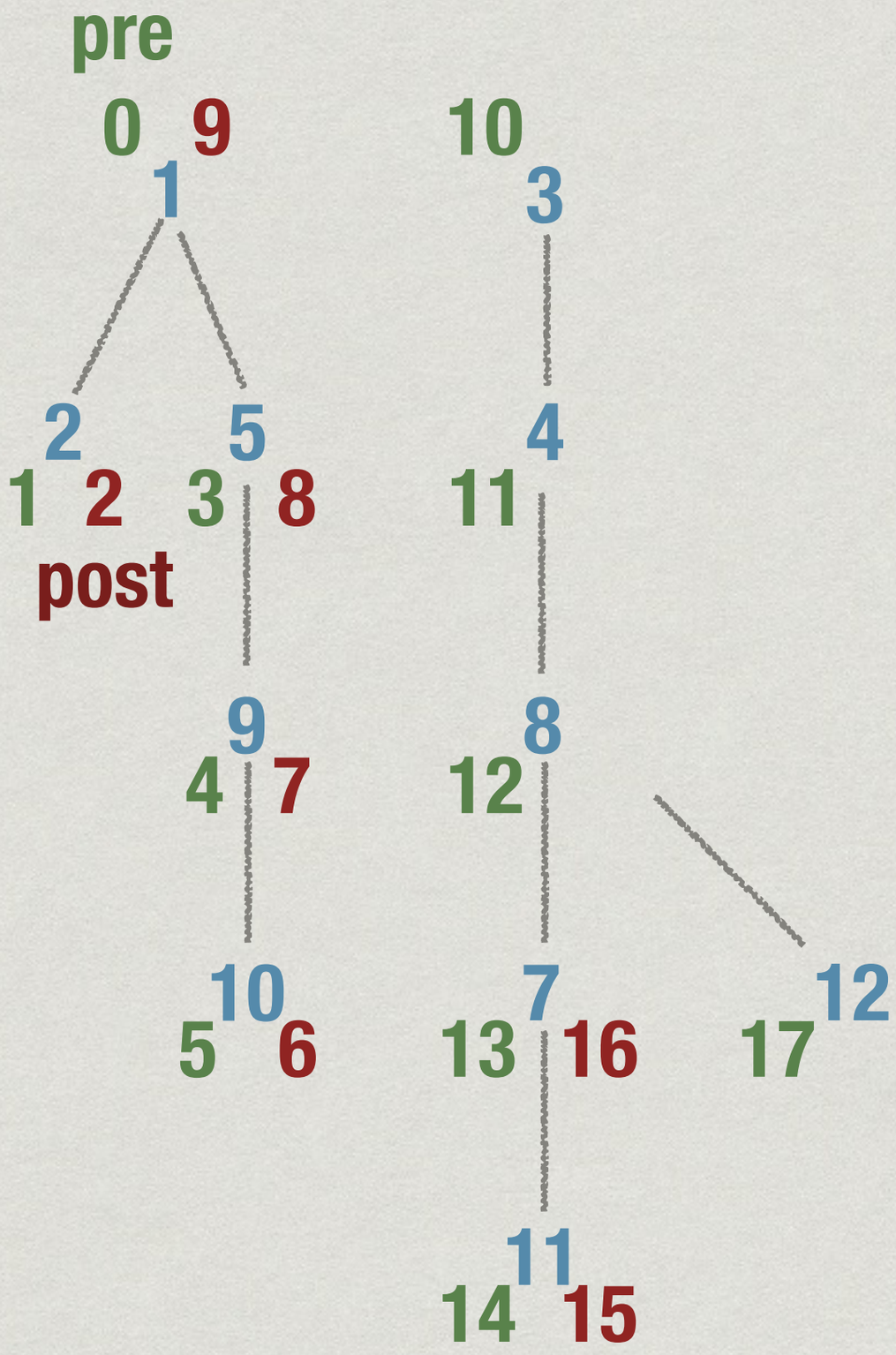

1
2
3
4
5
6
7
8
9
10
11
12


pre
0 9
1
2 5
1 2 3 8
post
9
4 7
10
5 6

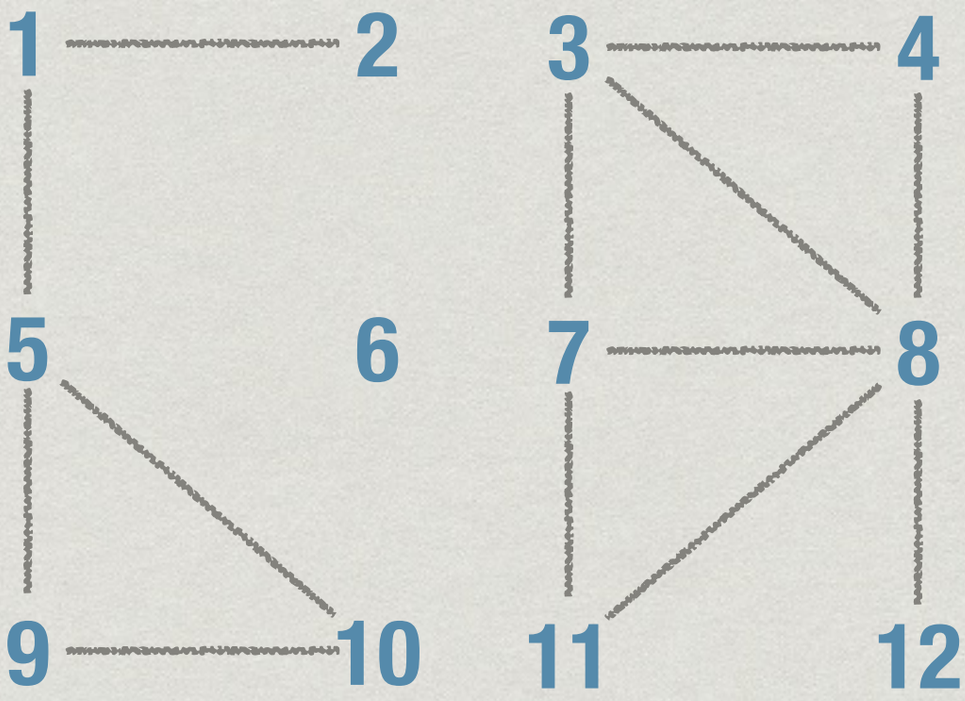
10
3
4
11
8
12
7
13 16
11
14 15

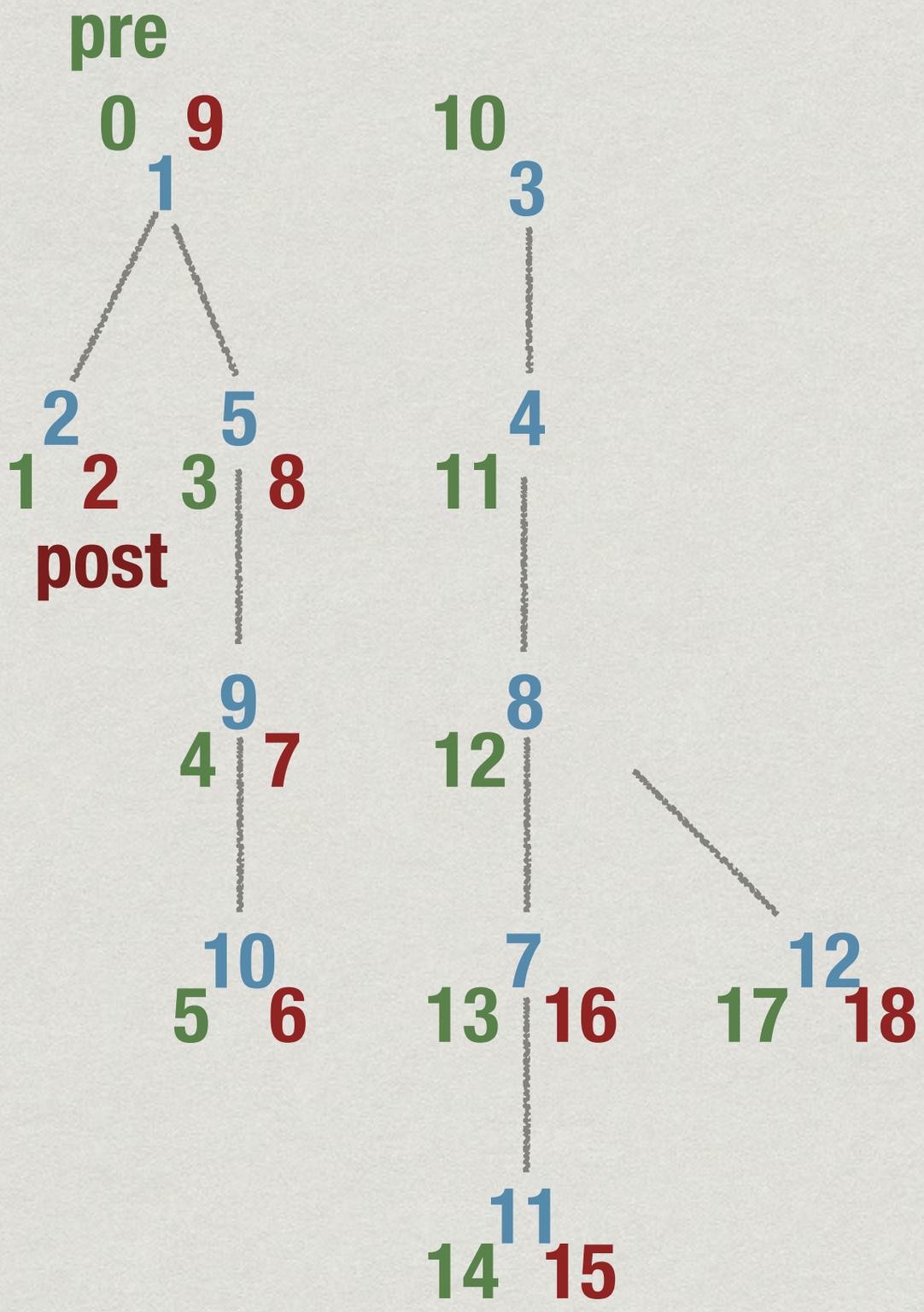
DFS tree


1
2
3
4
5
6
7
8
9
10
11
12

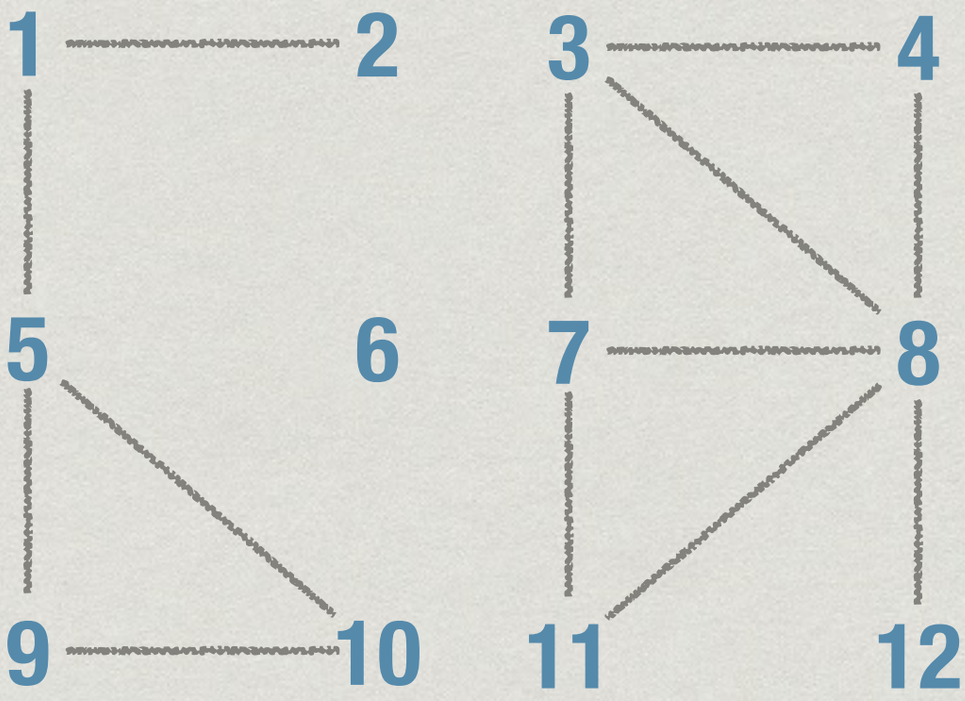

pre
0 9
1
10
3
2
5
4
1 2 3 8
11
post
9
8
4 7
12
10
7
12
5 6
13 16
17
11
14 15

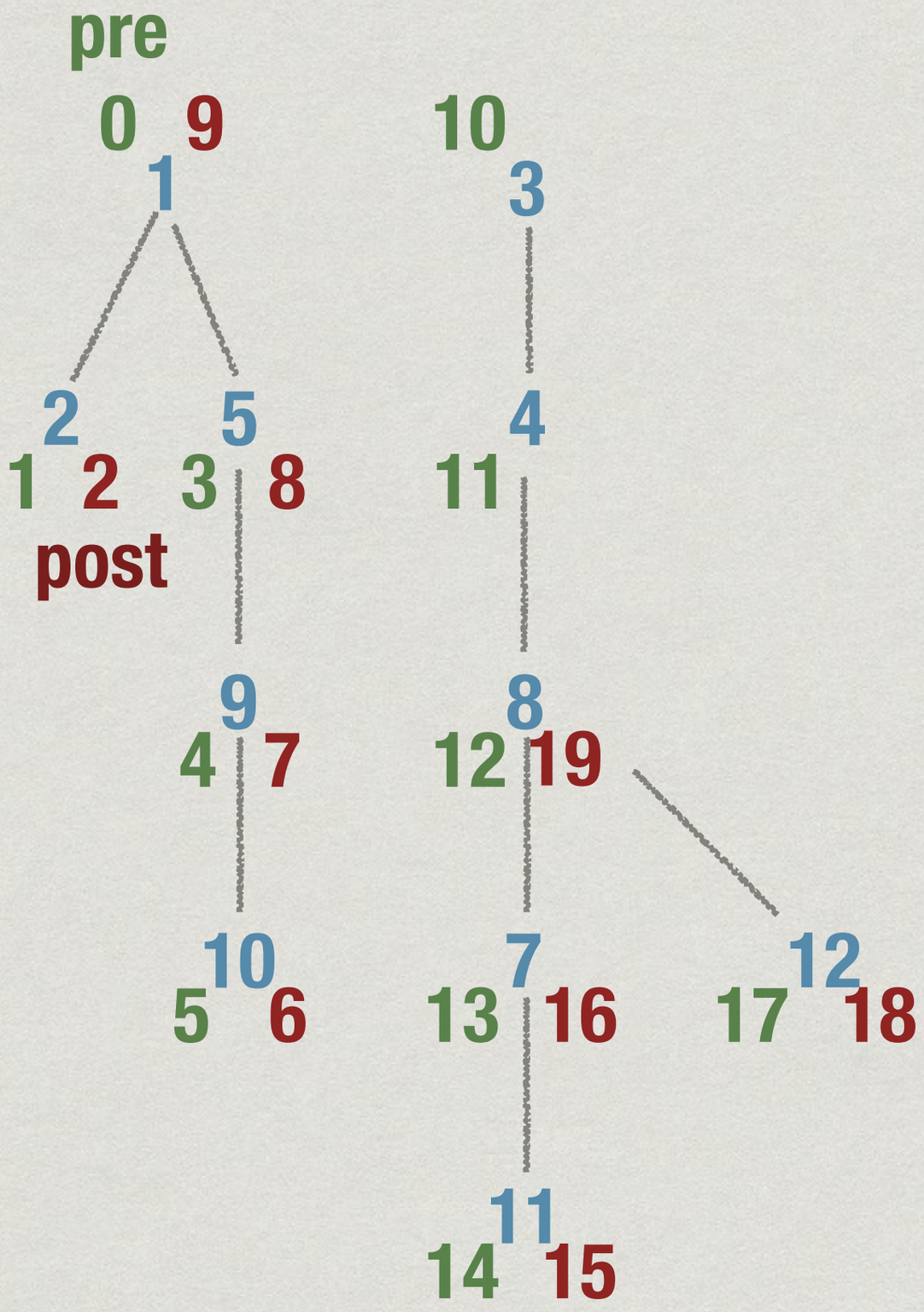
DFS tree


1
2
3
4
5
6
7
8
9
10
11
12

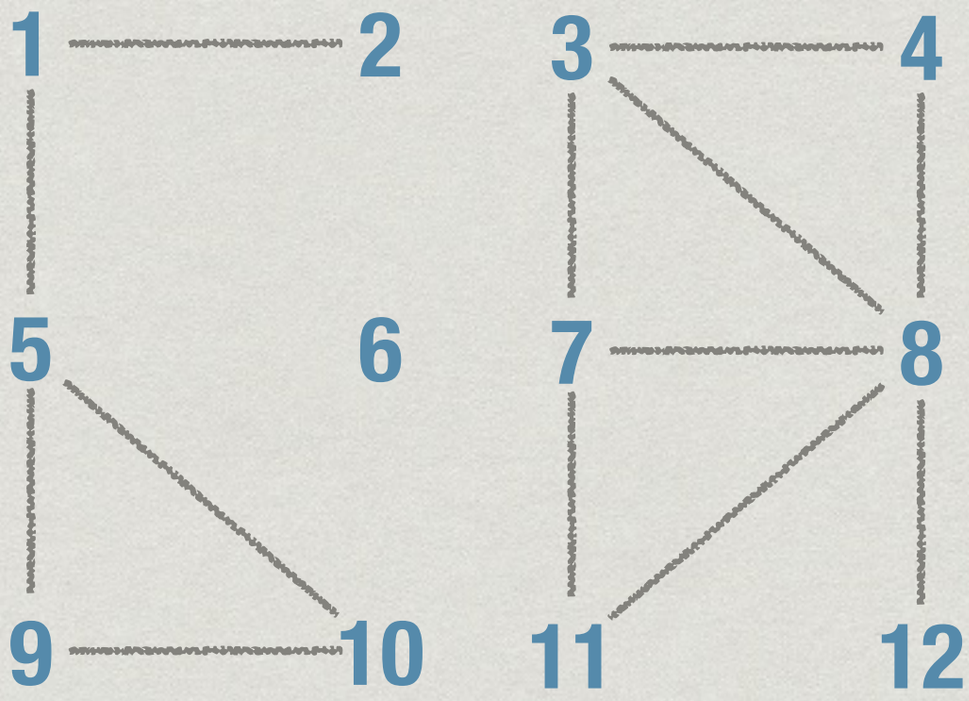

pre
0 9
1
10
3
2
5
4
1 2 3 8
11
post
9
8
4 7
12
10
7
12
5 6
13 16
17 18
11
14 15

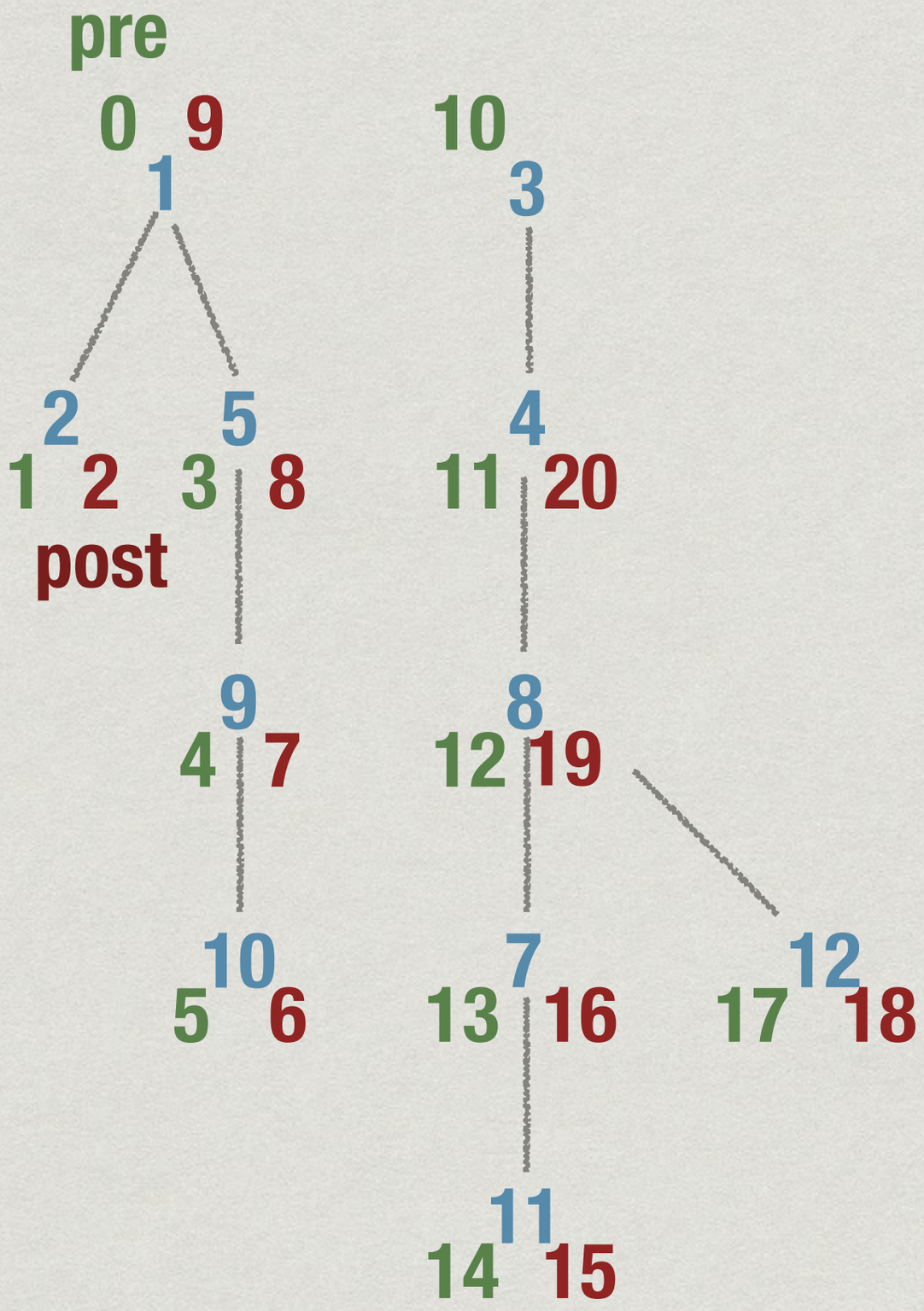
DFS tree


1
2
3
4
5
6
7
8
9
10
11
12

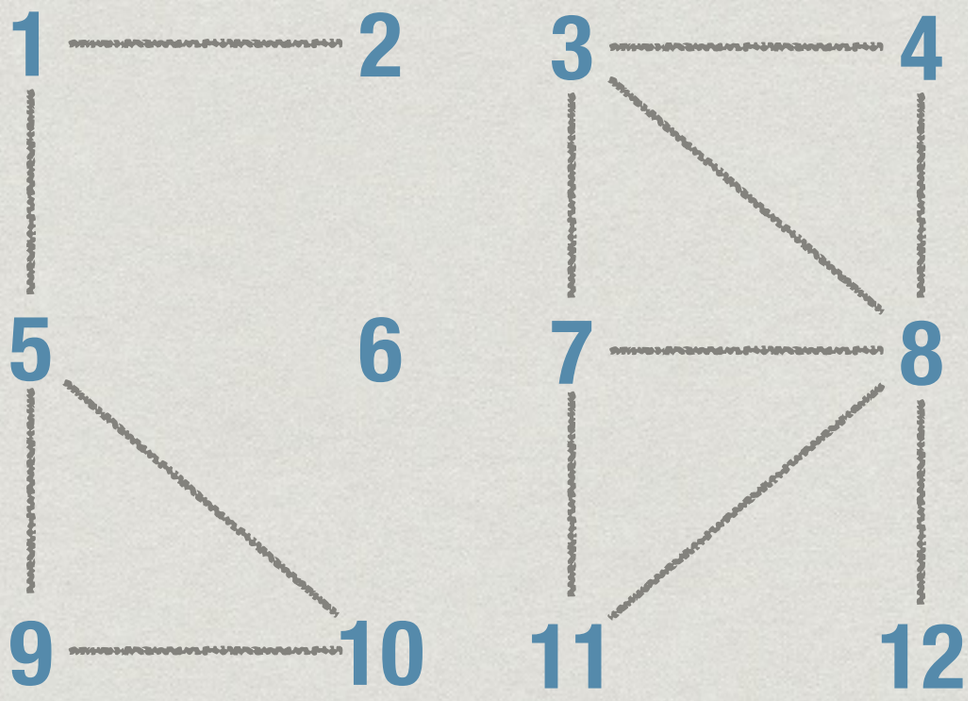

pre
0 9
1
10
3
2
5
4
1 2 3 8
11
post
9
4 7
8
12 19
10
5 6
7
13 16
12
17 18
11
14 15

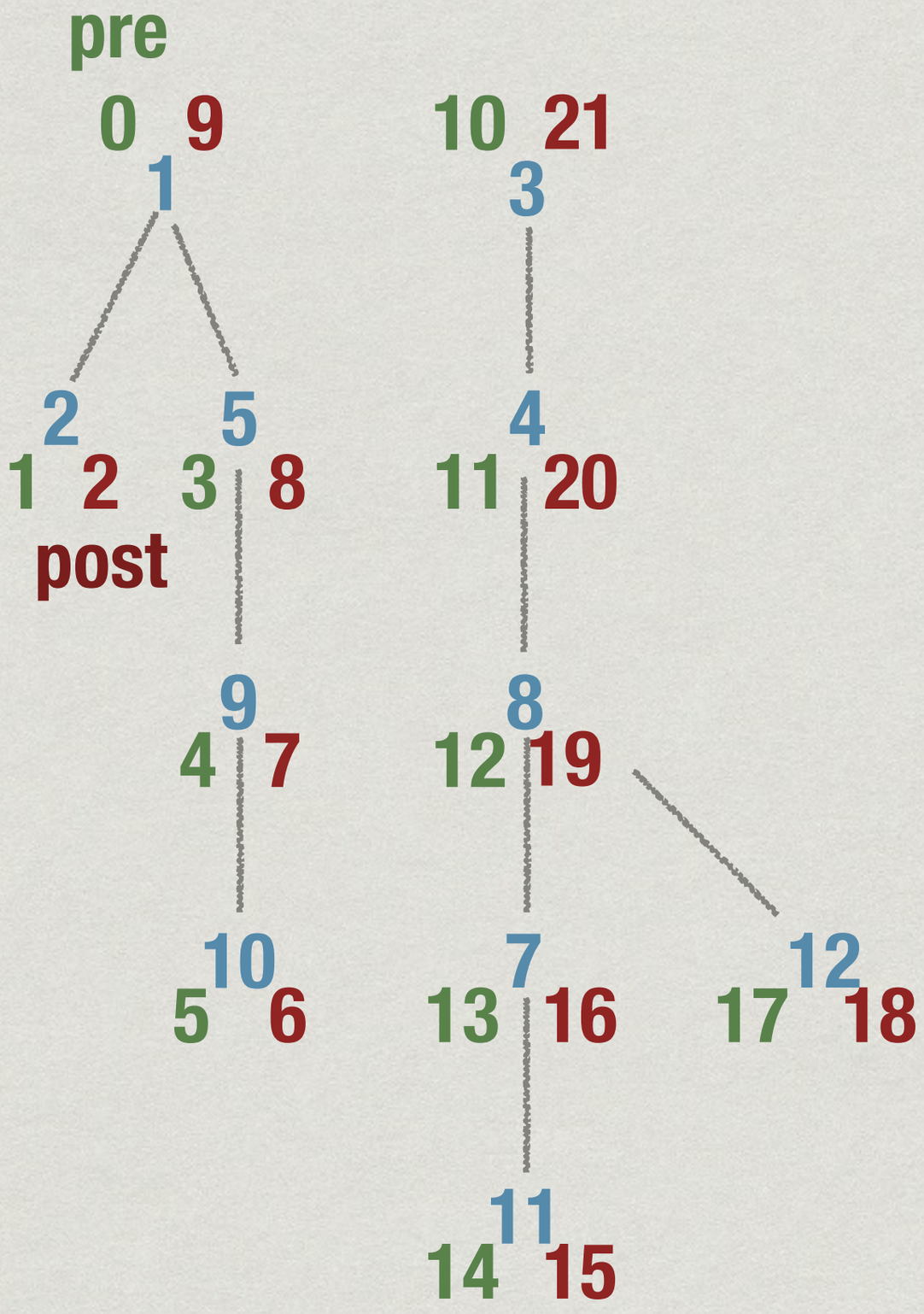
DFS tree


1
2
3
4
5
6
7
8
9
10
11
12

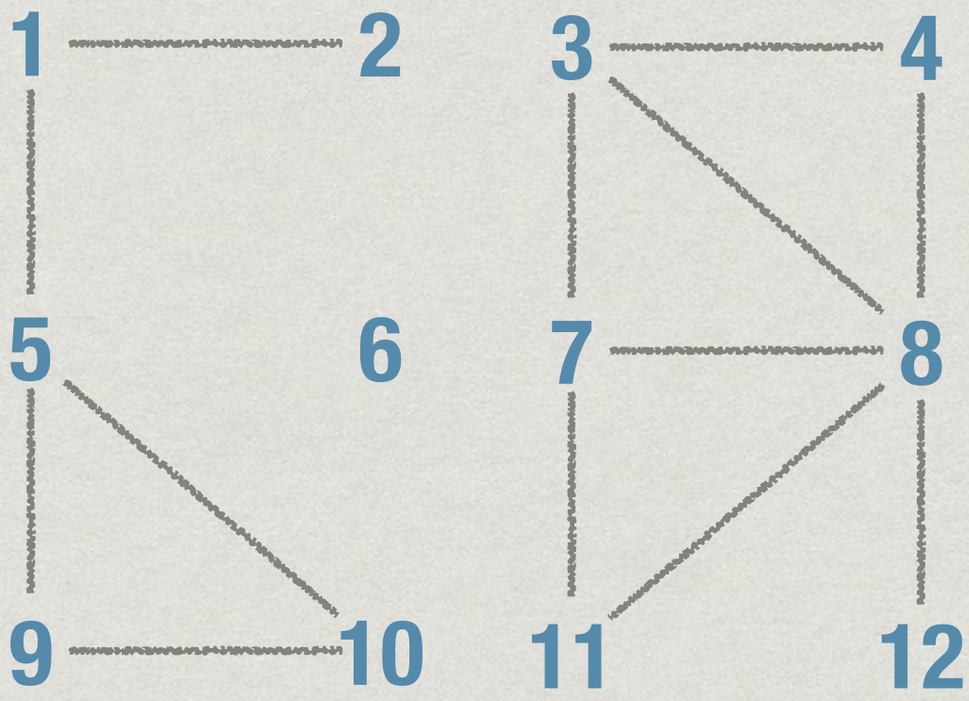

pre
0 9
1
2 5
1 2 3 8
post
9
4 7
10
5 6
10
3
4
11 20
8
12 19
7
13 16
11
14 15
12
17 18

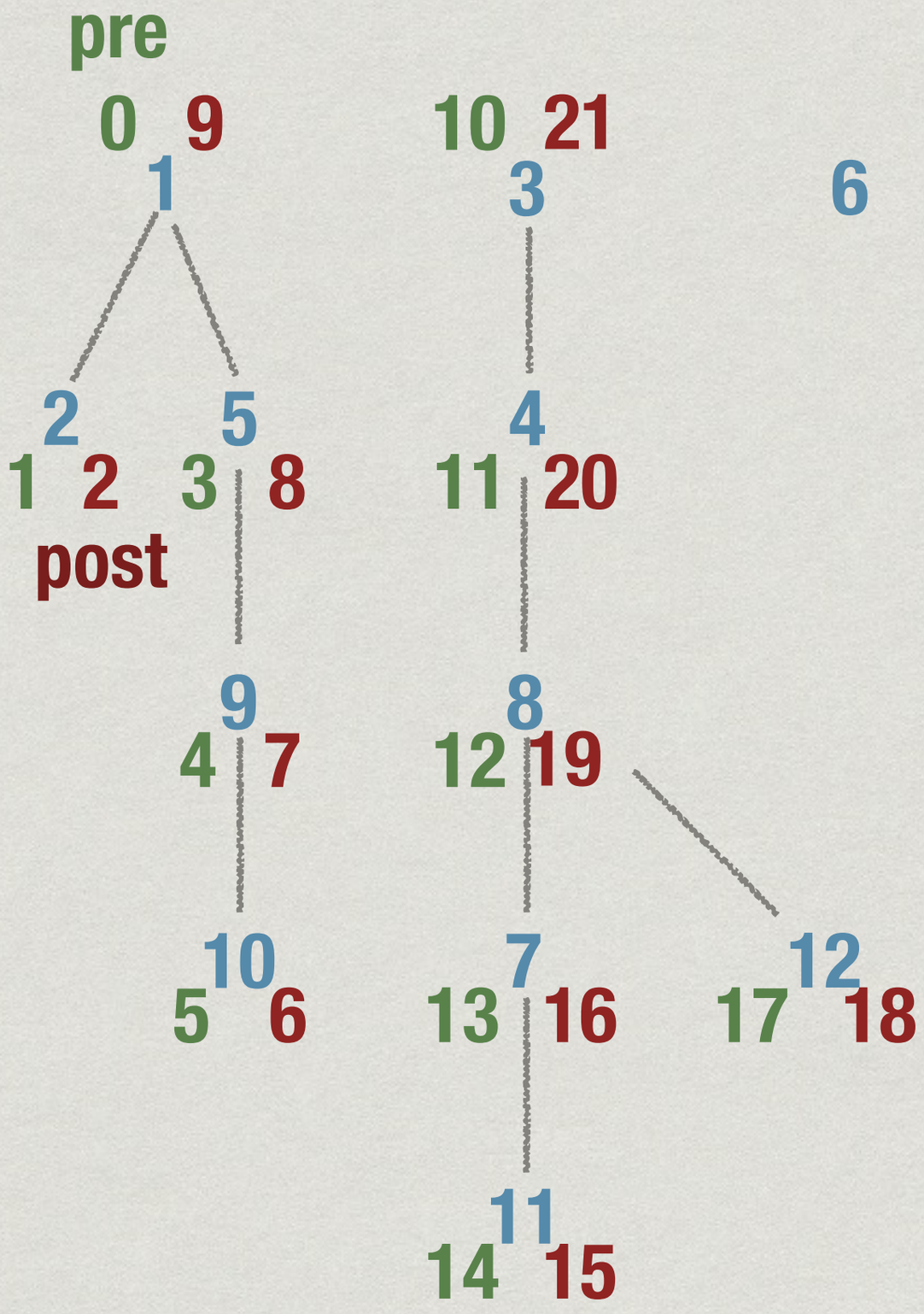
DFS tree


1
2
3
4
5
6
7
8
9
10
11
12

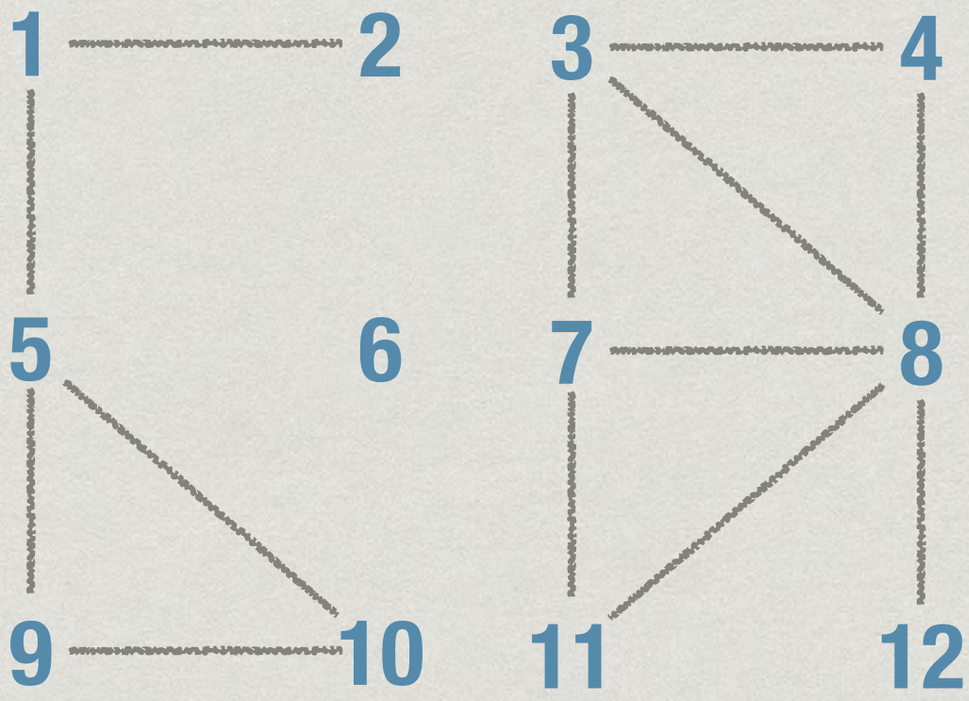

pre
0 9
1
2 5
1 2 3 8
post
9
4 7
10
5 6
10 21
3
4
11 20
8
12 19
7
13 16
11
14 15
12
17 18

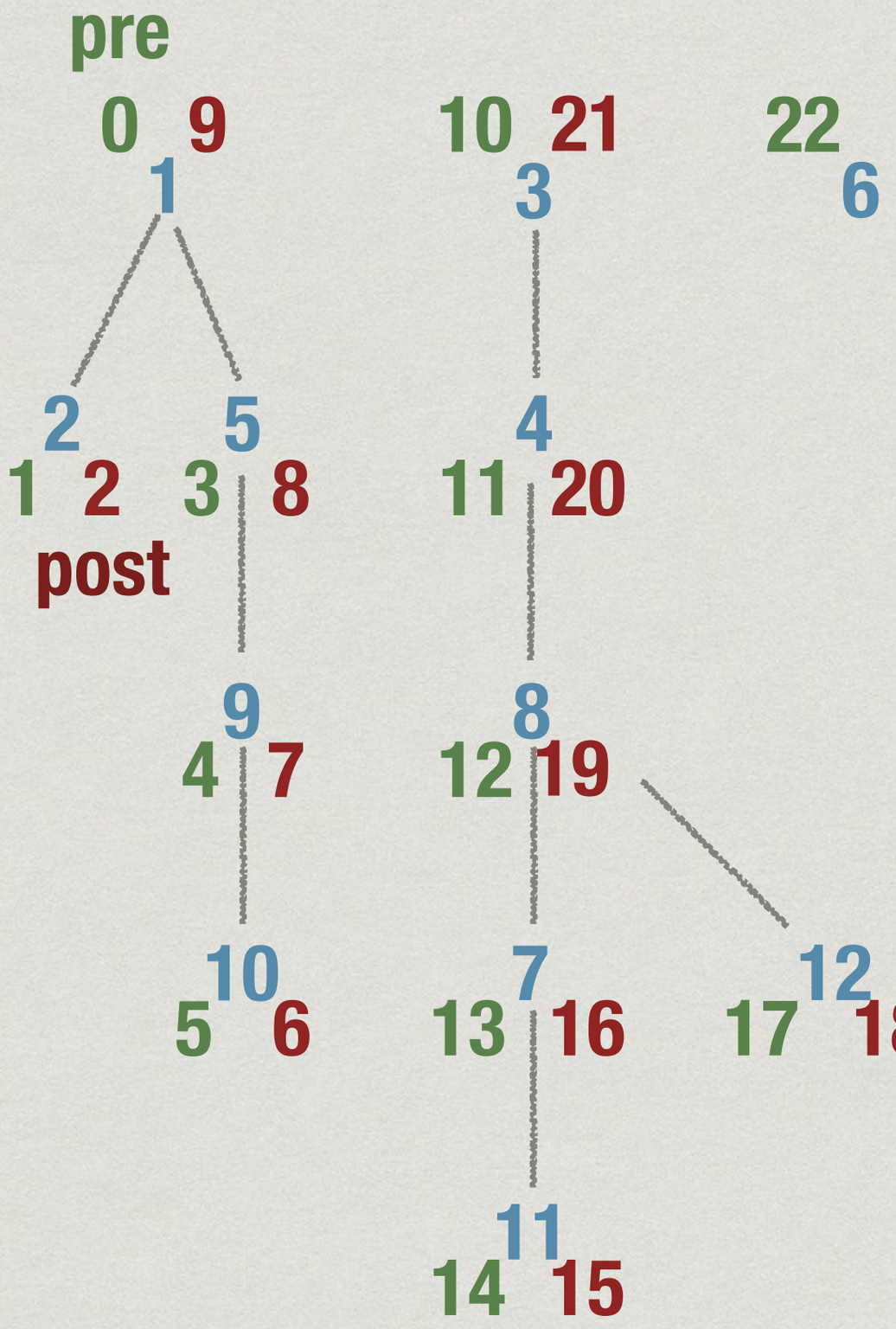
DFS tree


1
2
3
4
5
6
7
8
9
10
11
12

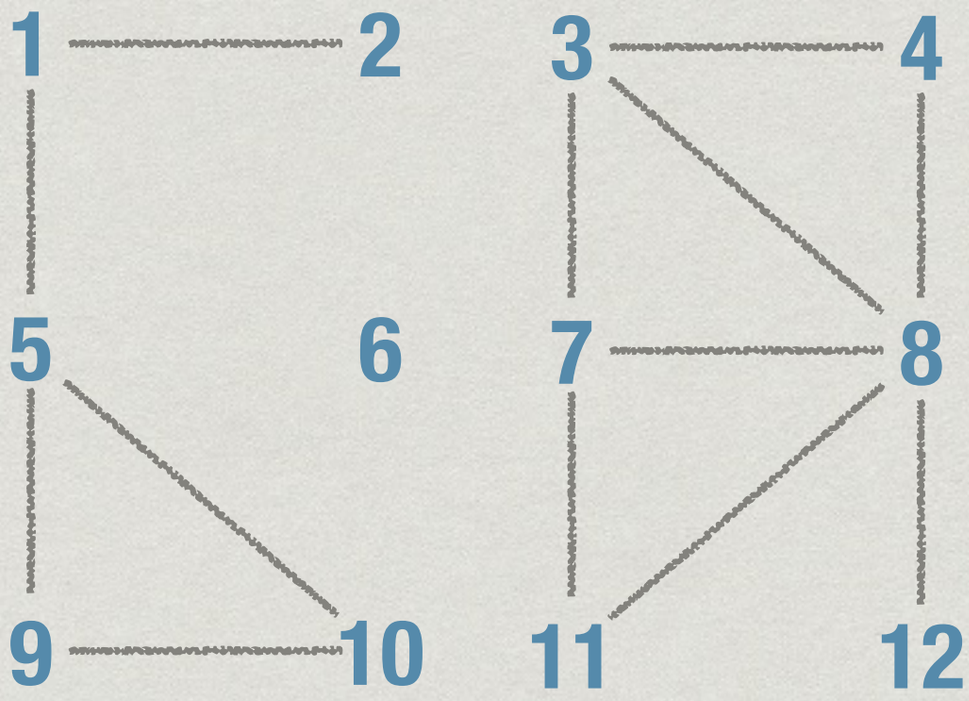

pre
post
0 9
1
10 21
3
6
2
5
4
1 2 3 8
11 20
9
8
4 7
12 19
10
7
12
5 6
13 16
17 18
11
14 15

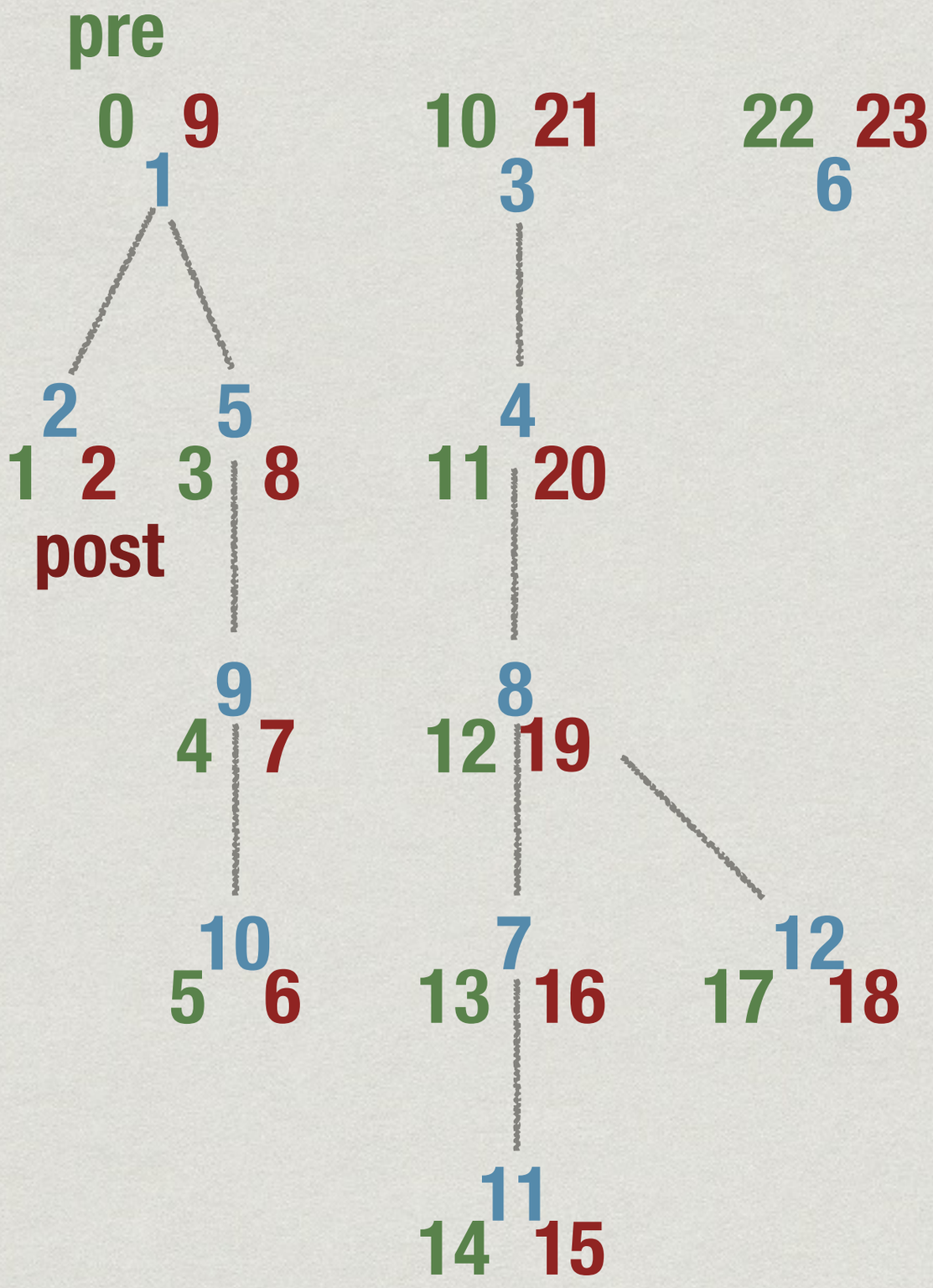
DFS tree


1
2
3
4
5
6
7
8
9
10
11
12

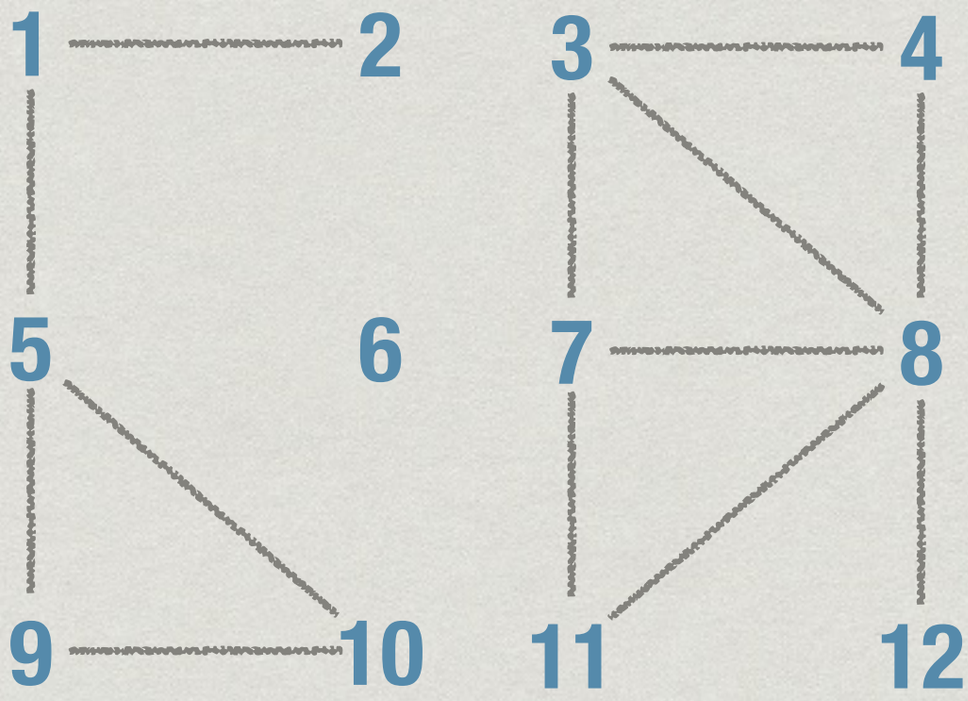

pre
0 9
1
10 21
3
22
6
2
5
4
1 2 3 8
11 20
post
9
4 7
8
12 19
10
5 6
7
13 16
12
17 18
11
14 15

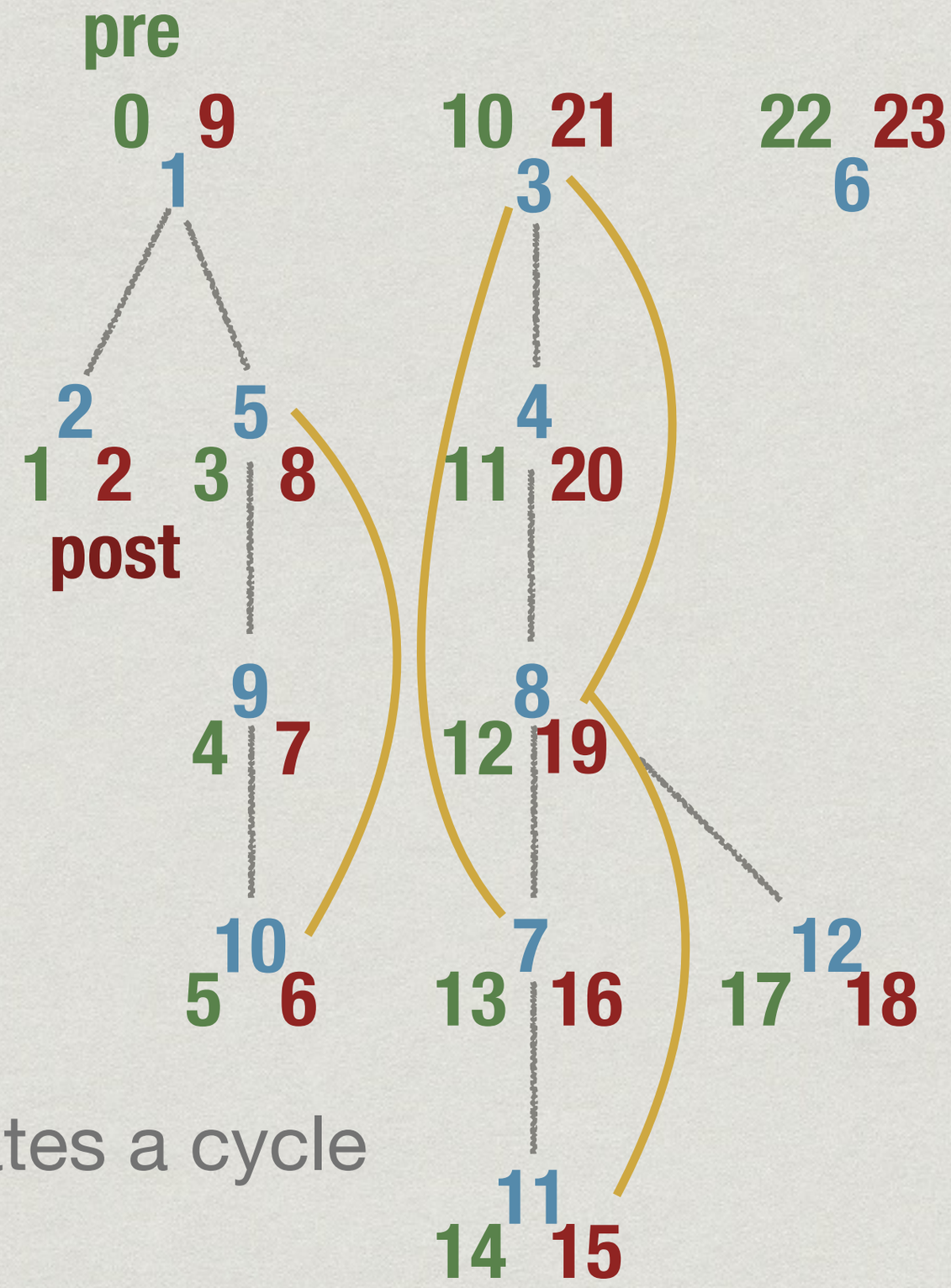
DFS tree


1
2
3
4
5
6
7
8
9
10
11
12


pre
0 9
1
10 21
3
22 23
6
2
5
4
1 2 3 8
11 20
post
9
8
4 7
12 19
10
7
12
5 6
13 16
17 18
11
14 15

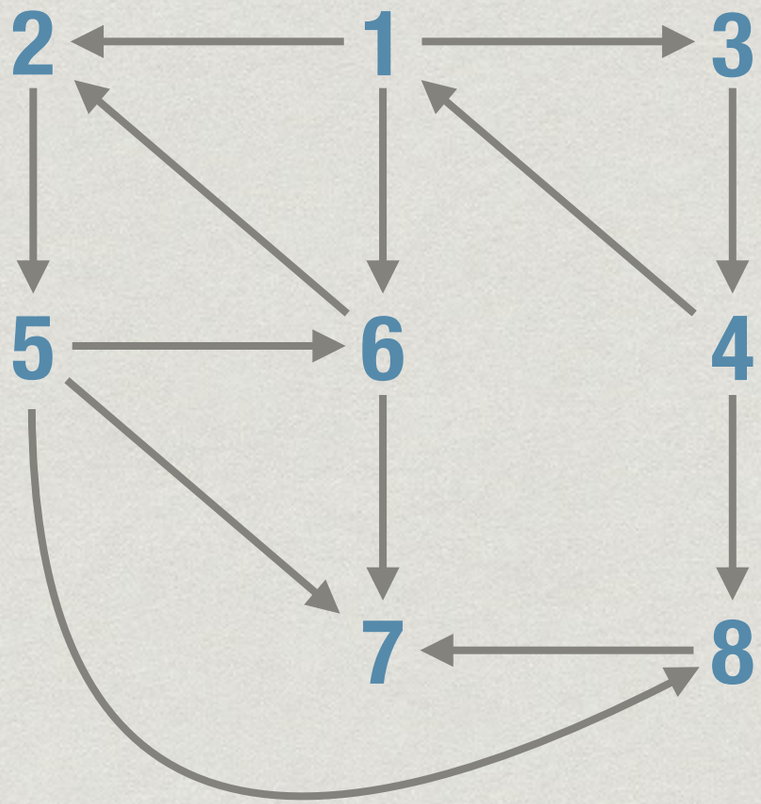
DFS tree



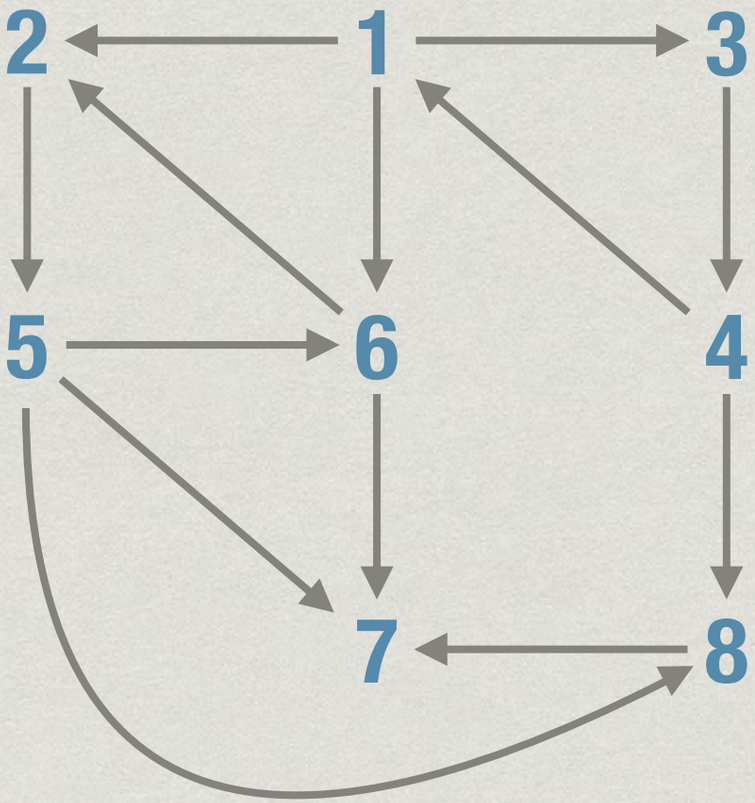


- Any non-tree edge generates a cycle

Directed cycles

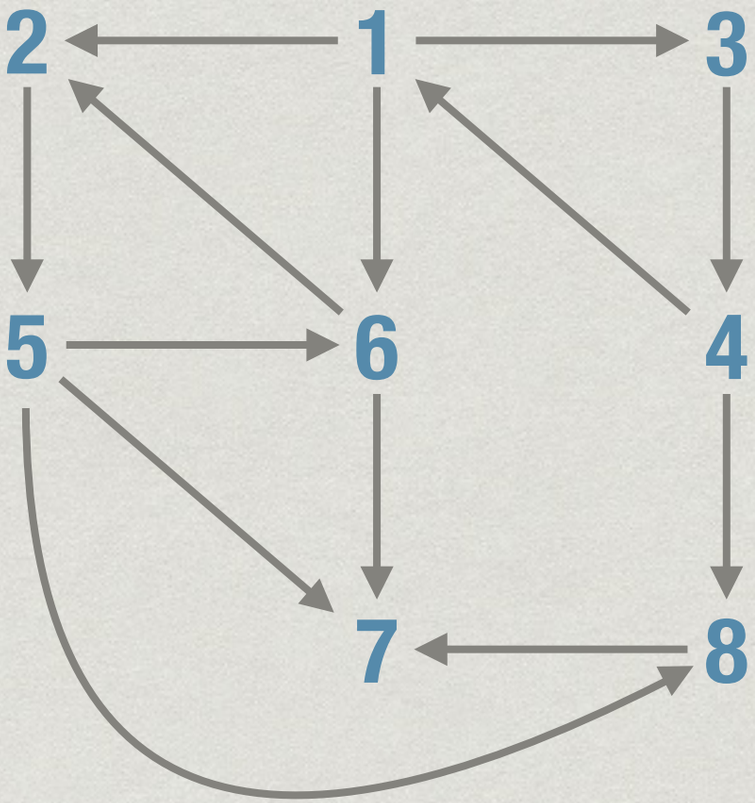


Directed cycles

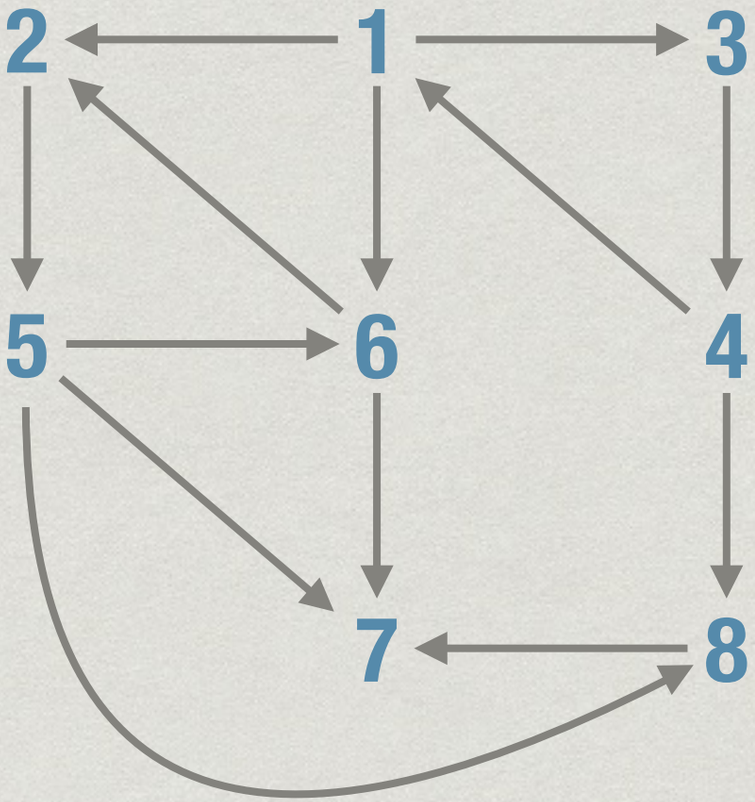


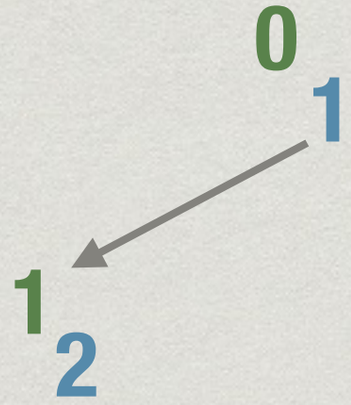
1

Directed cycles

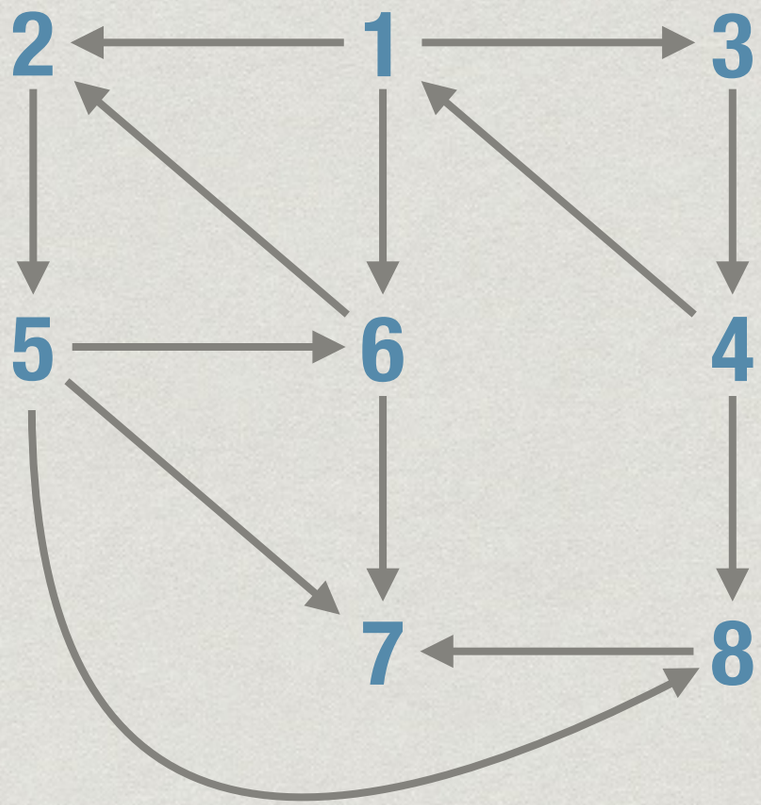


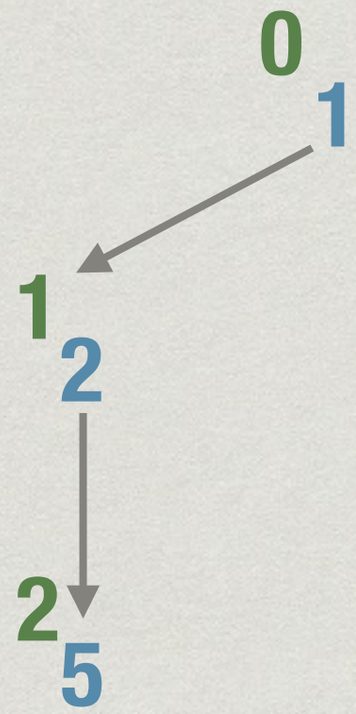
Directed cycles



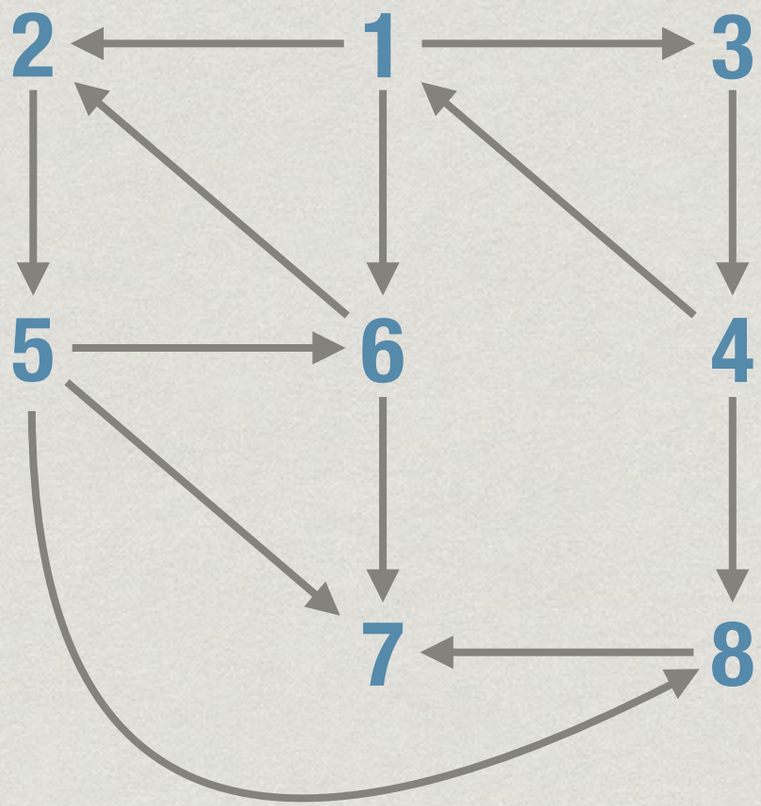


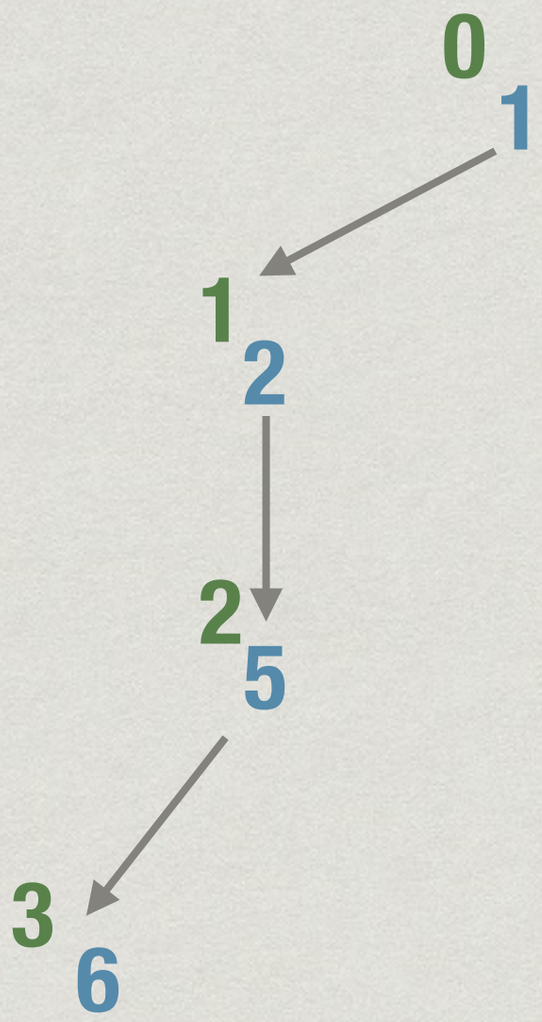
Directed cycles



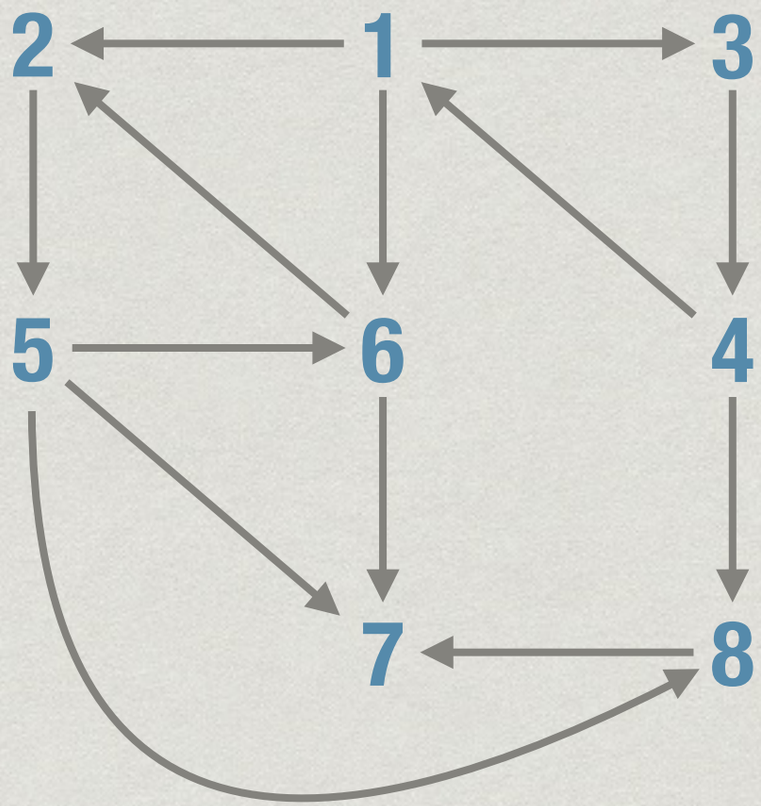


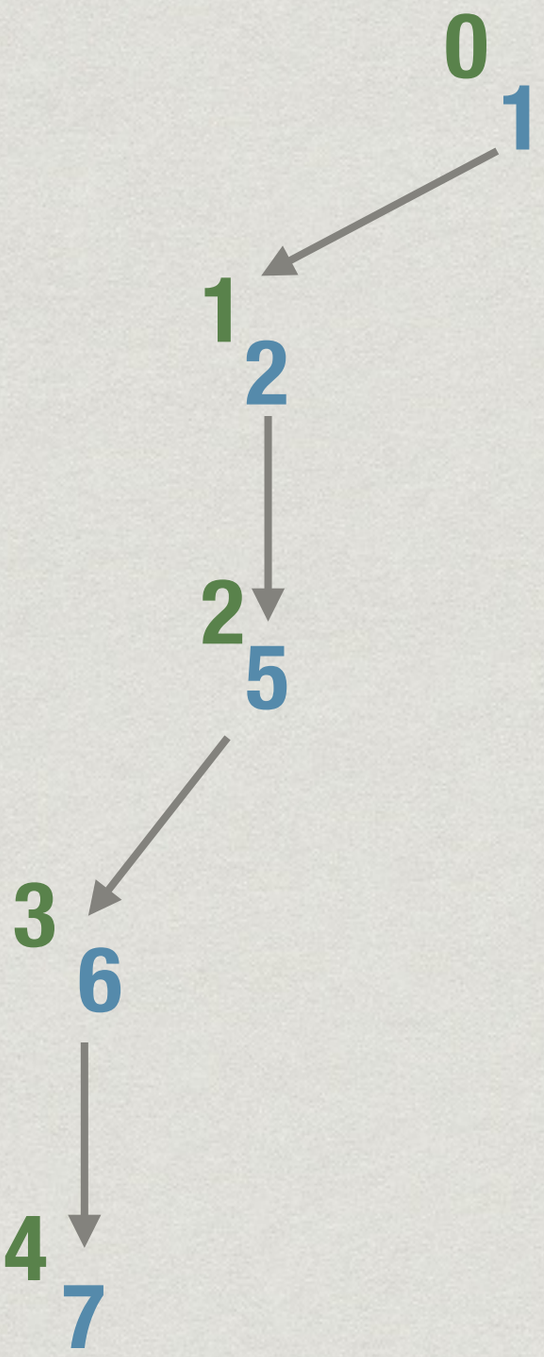
Directed cycles



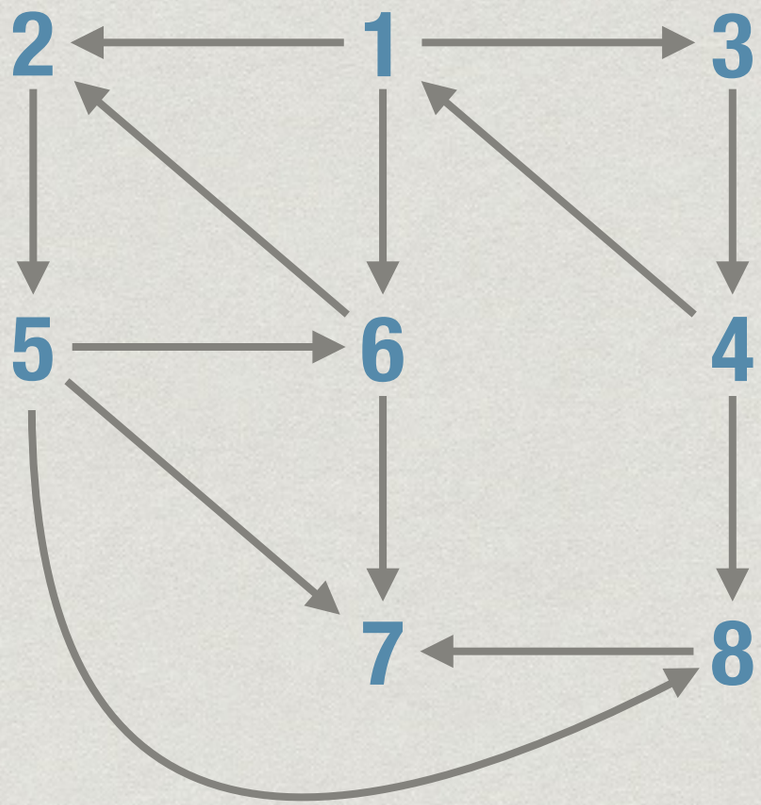


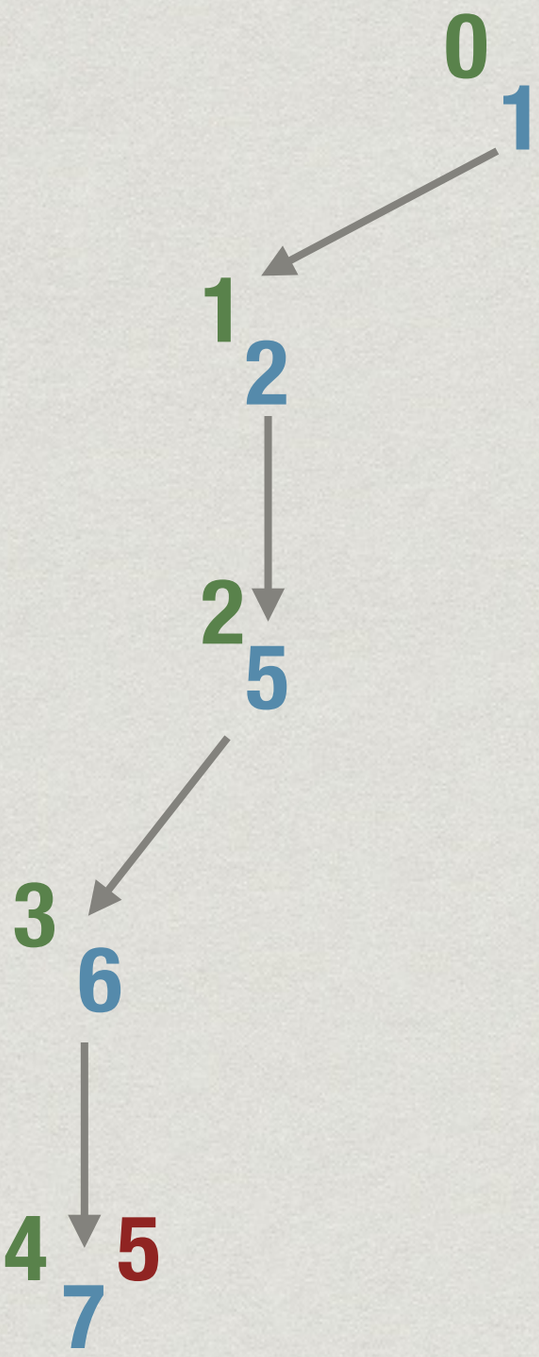
Directed cycles



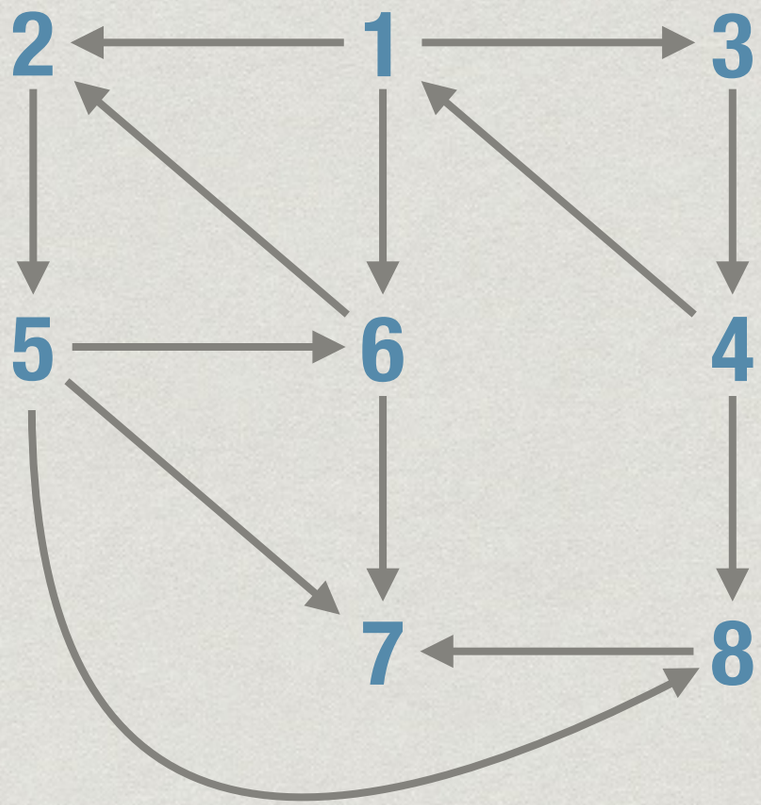


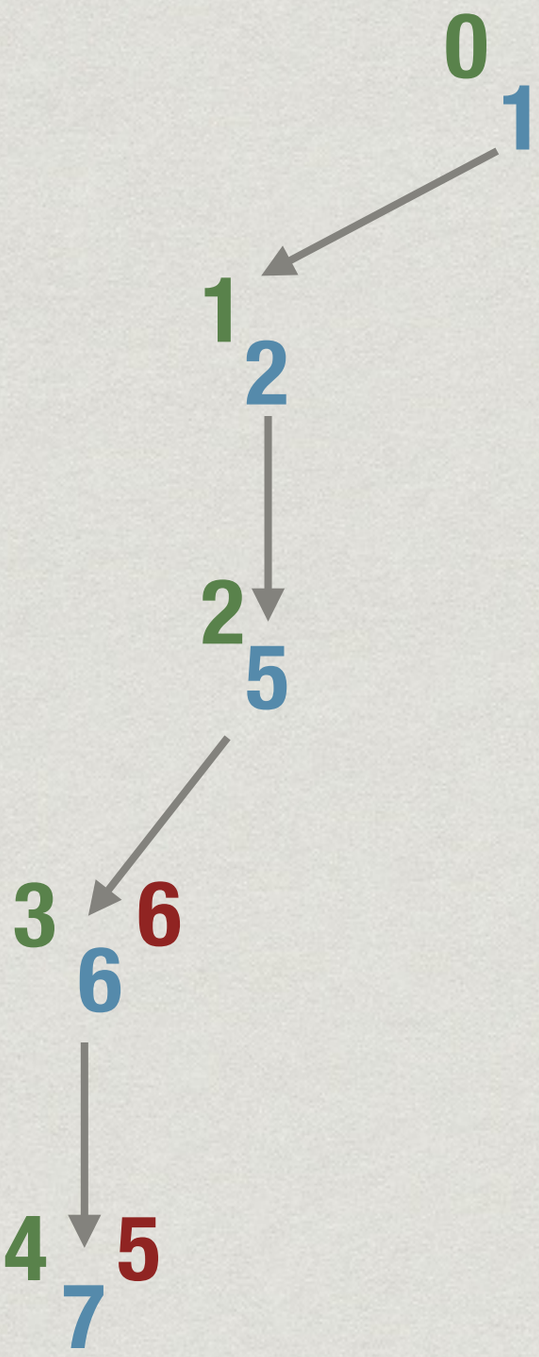
Directed cycles



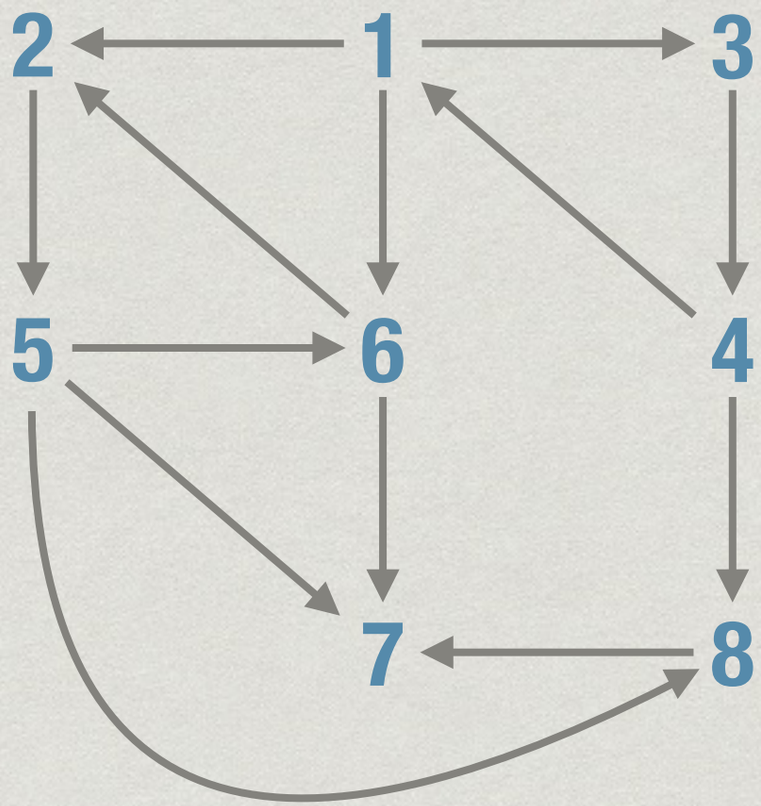


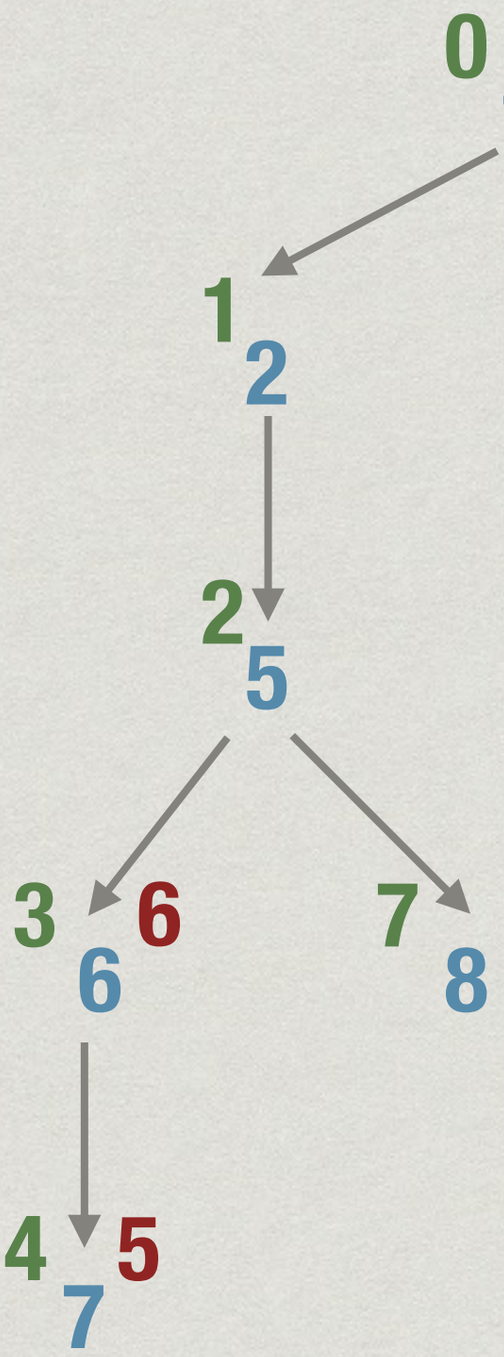
Directed cycles



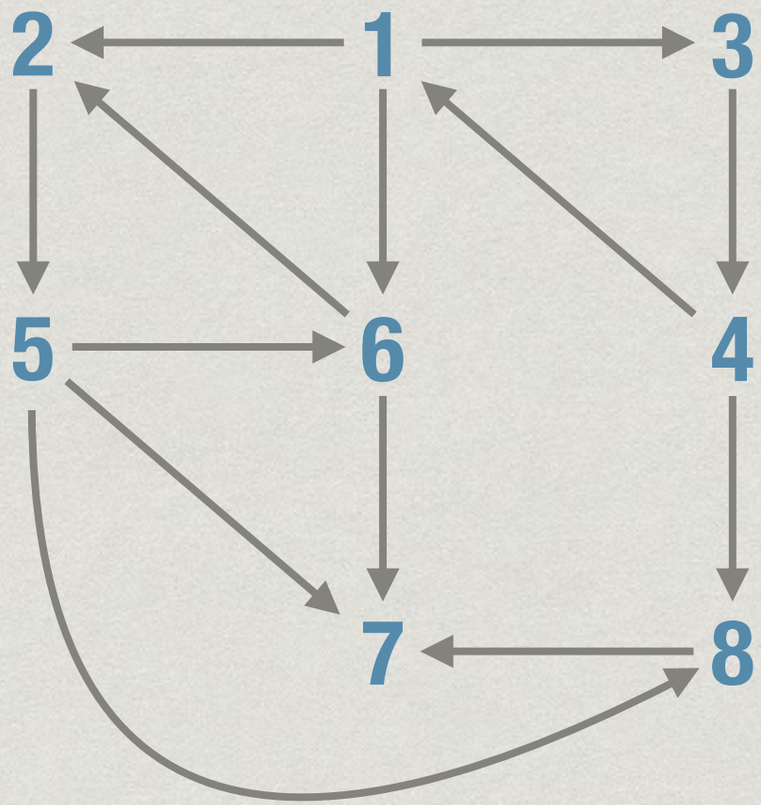


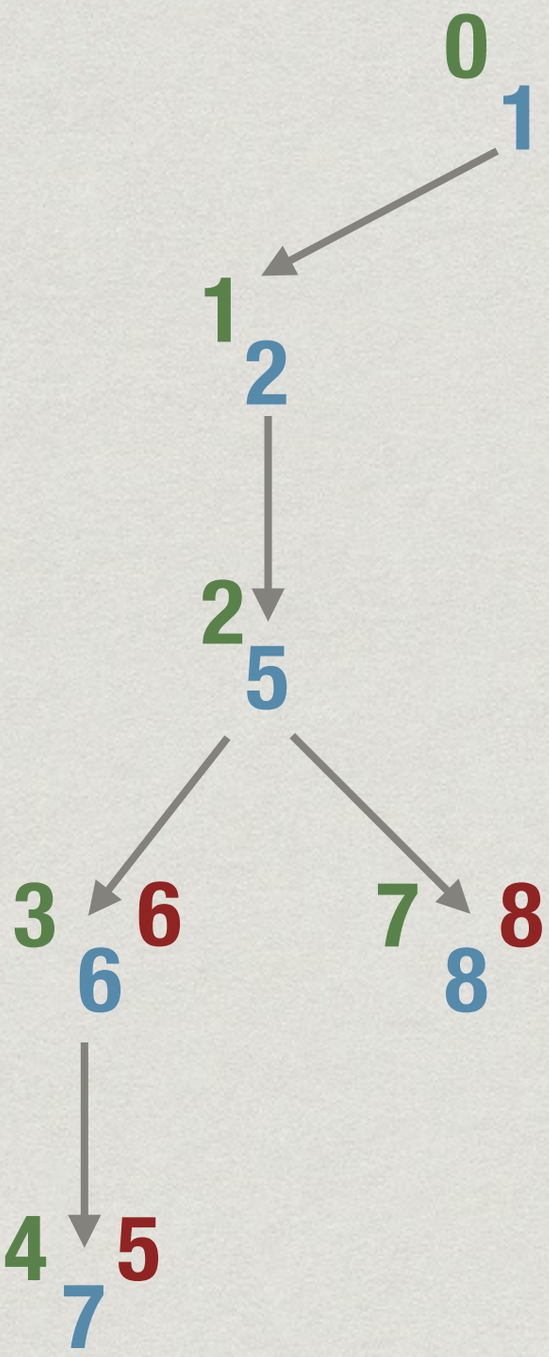
Directed cycles



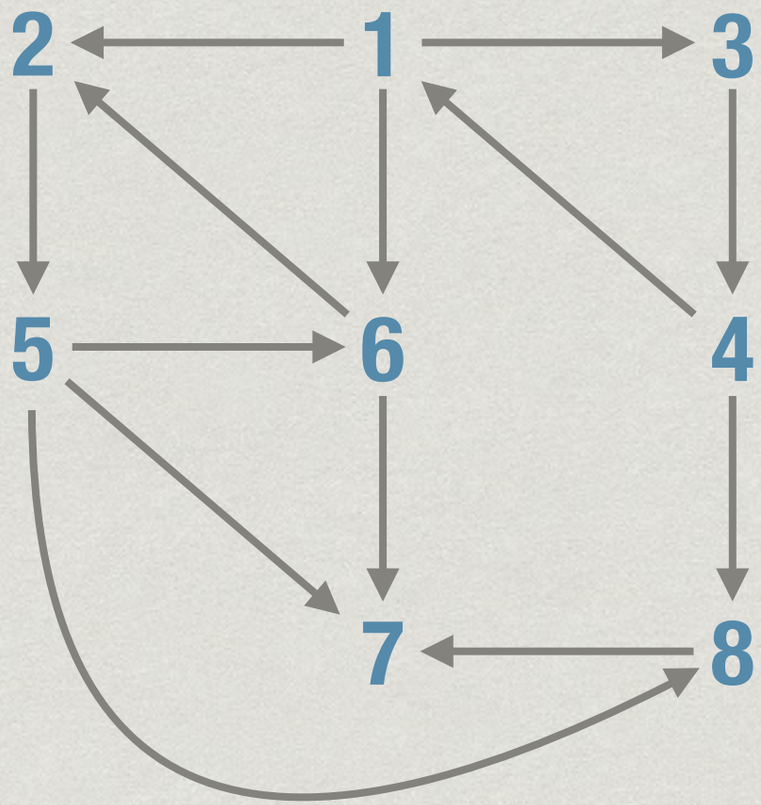


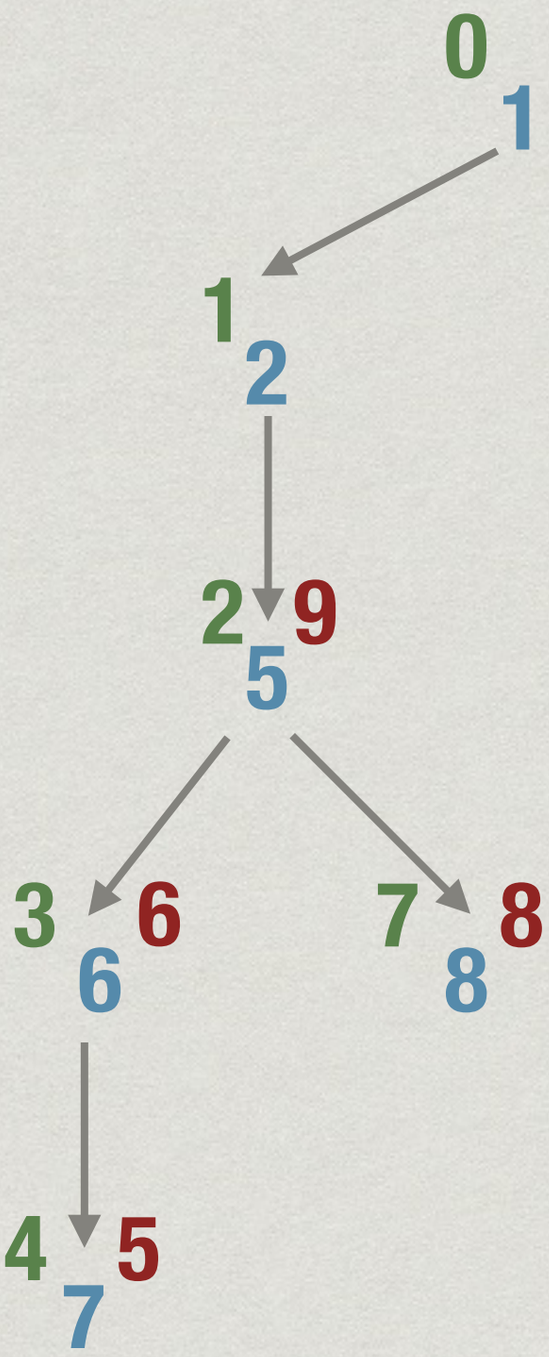
Directed cycles



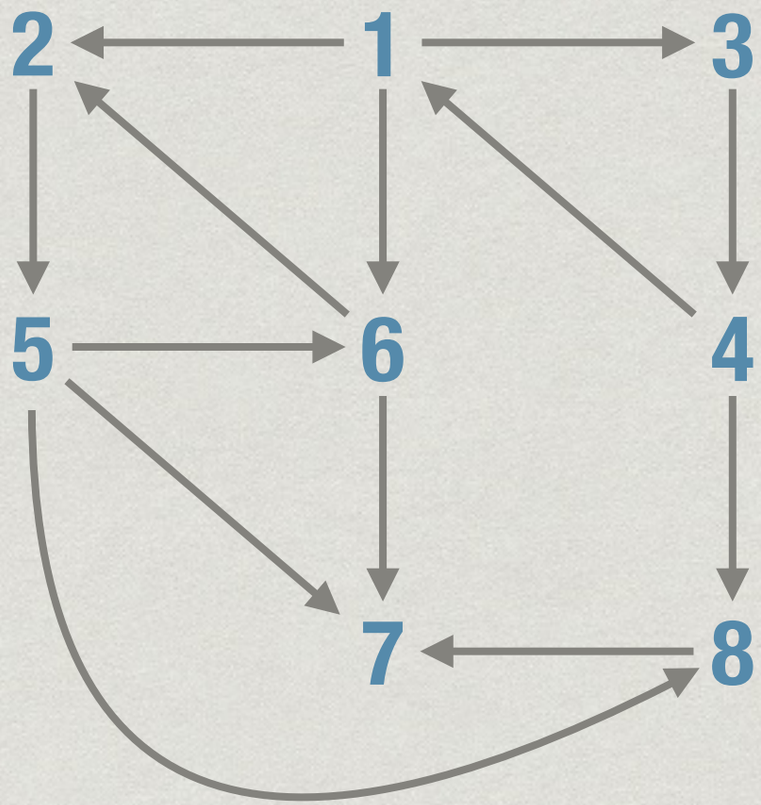


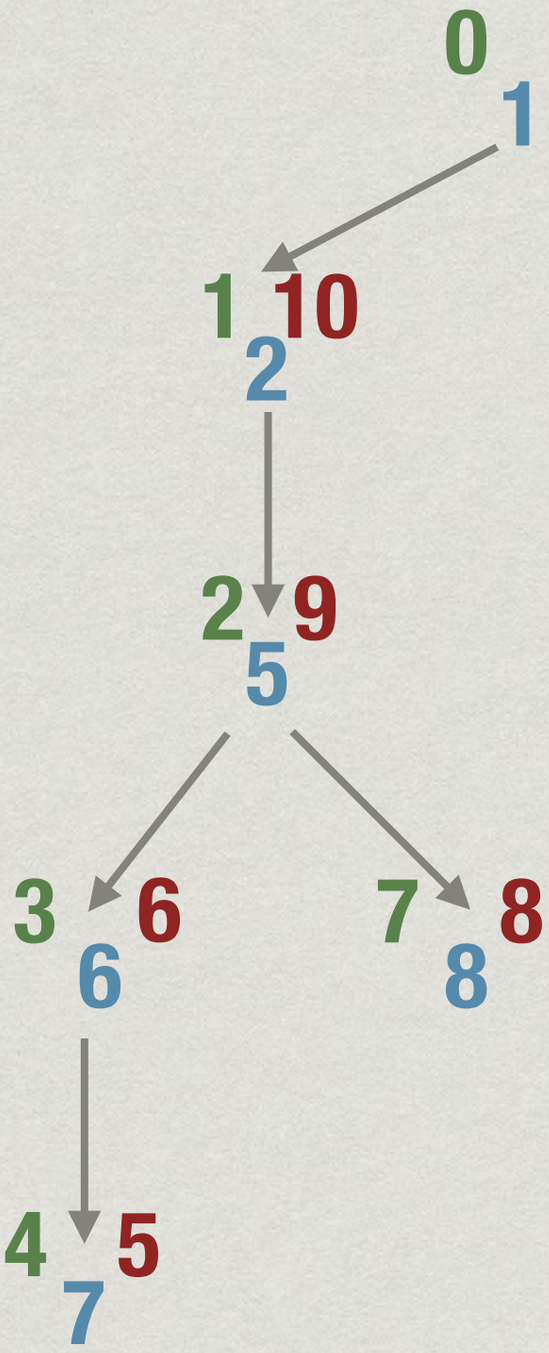
Directed cycles



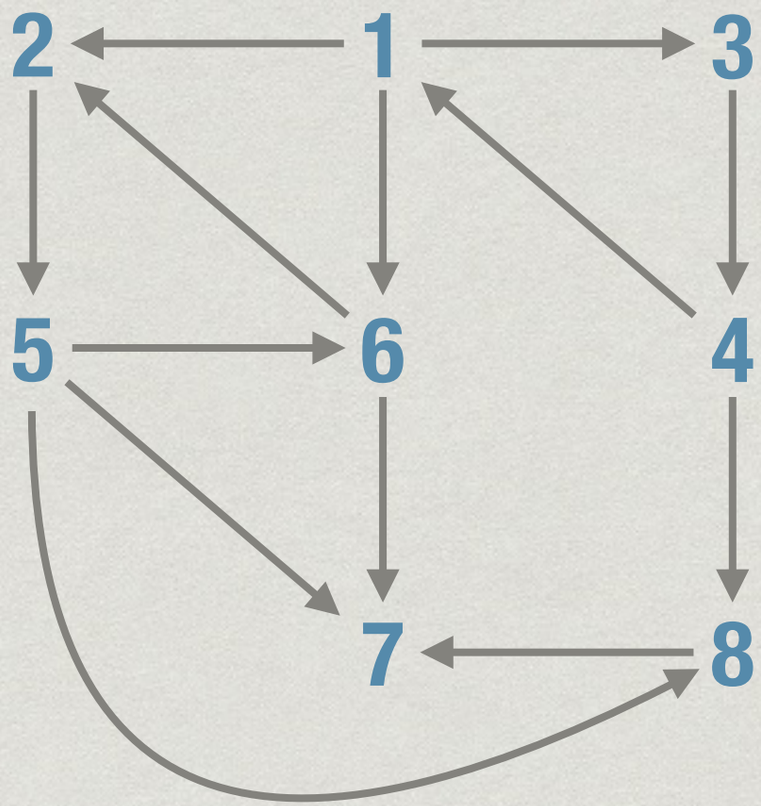


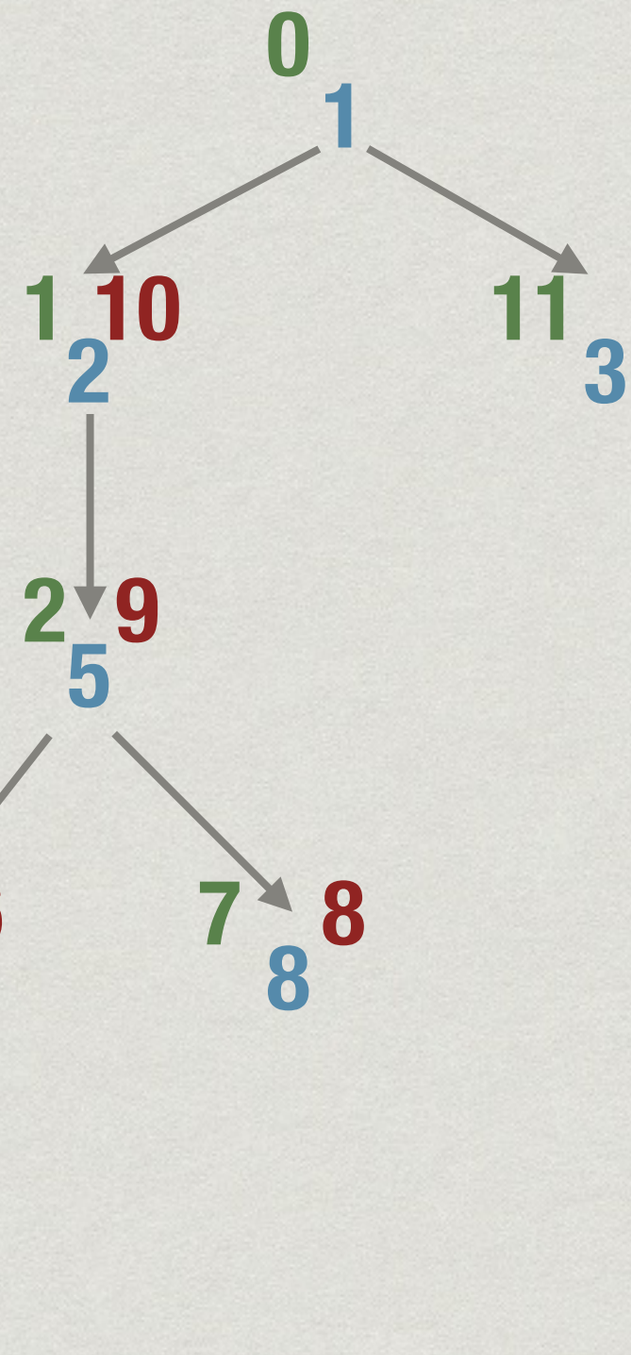
Directed cycles



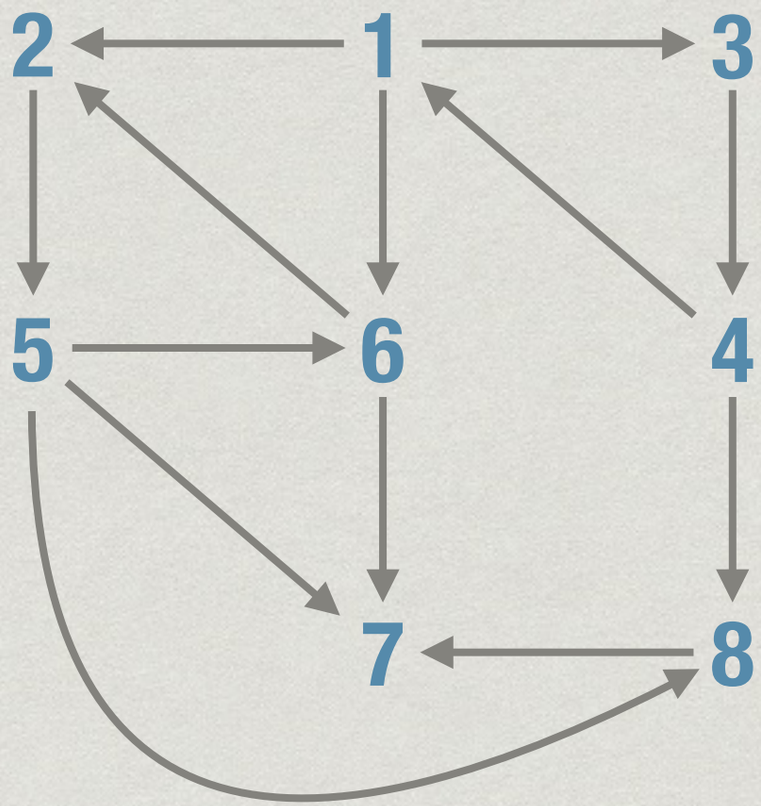


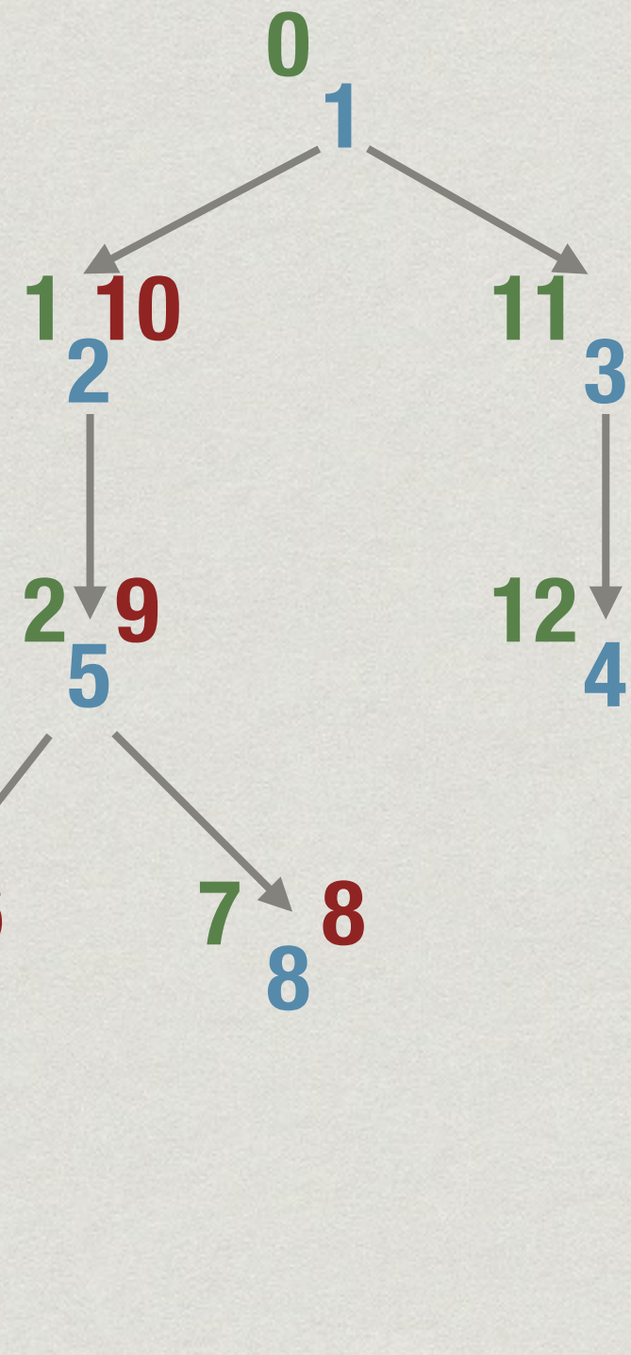
Directed cycles



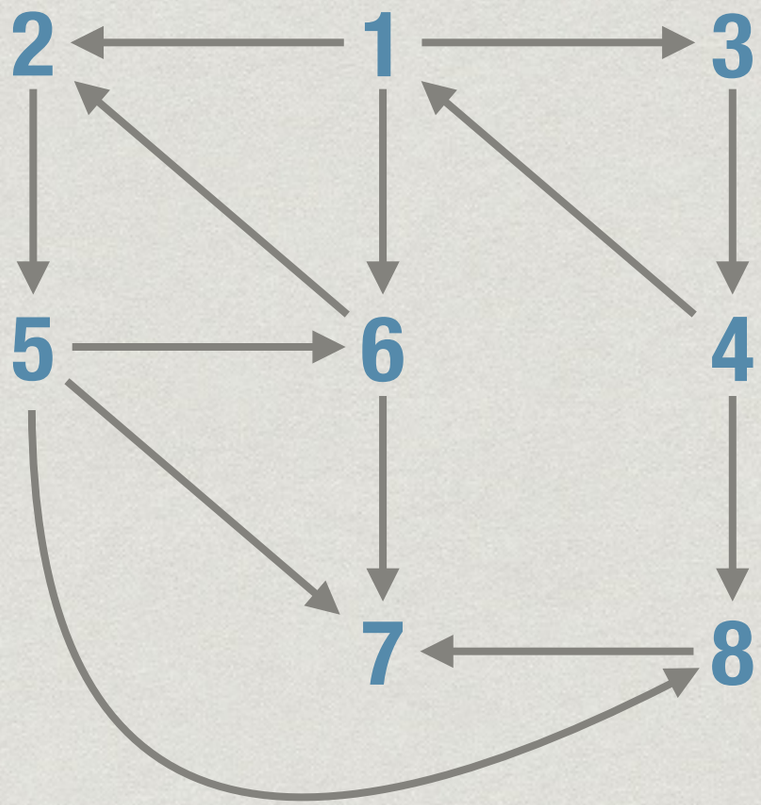


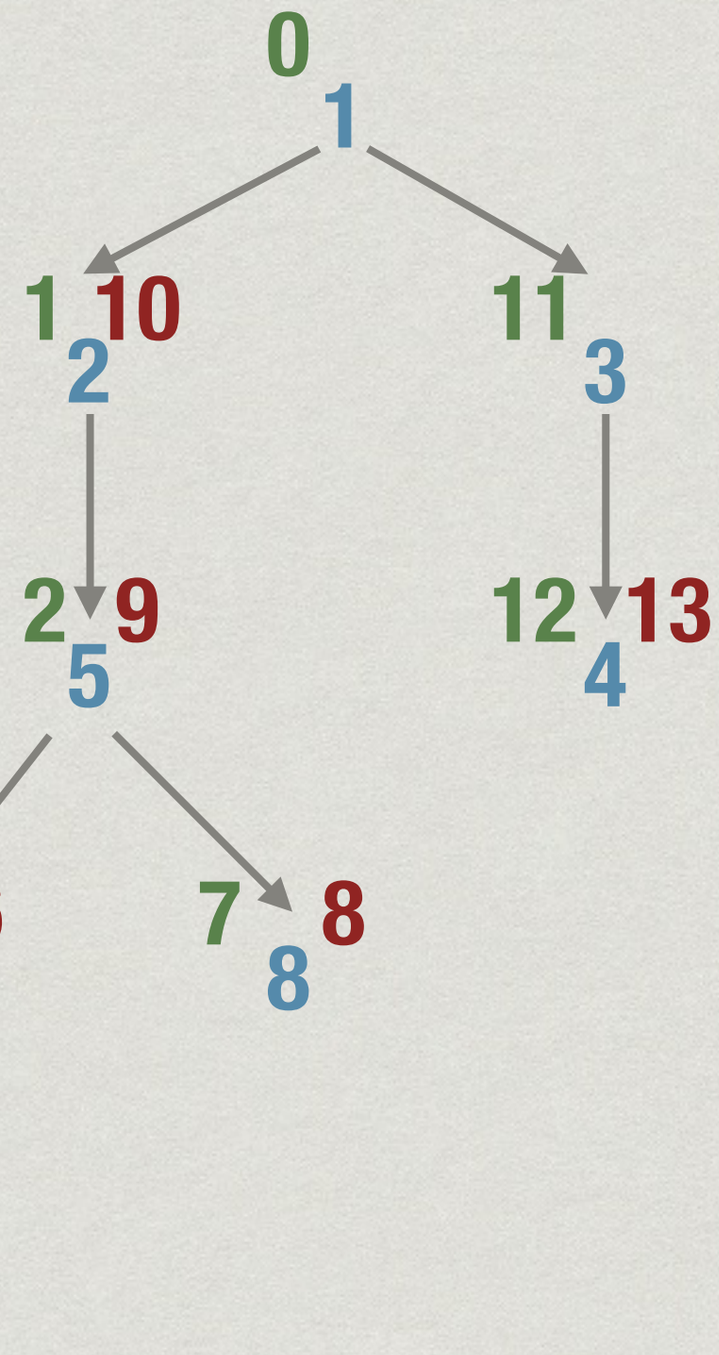
Directed cycles



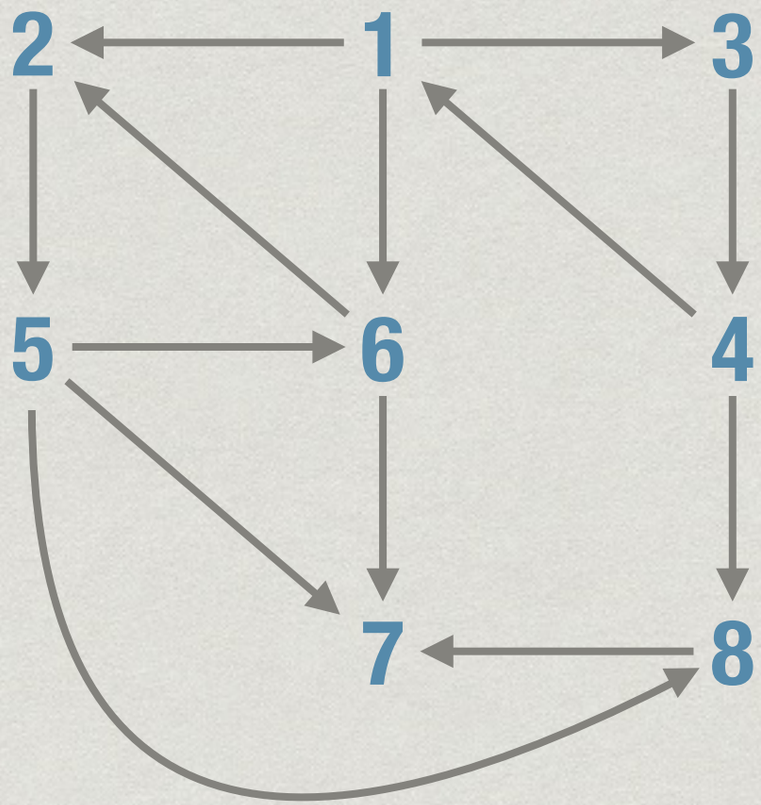


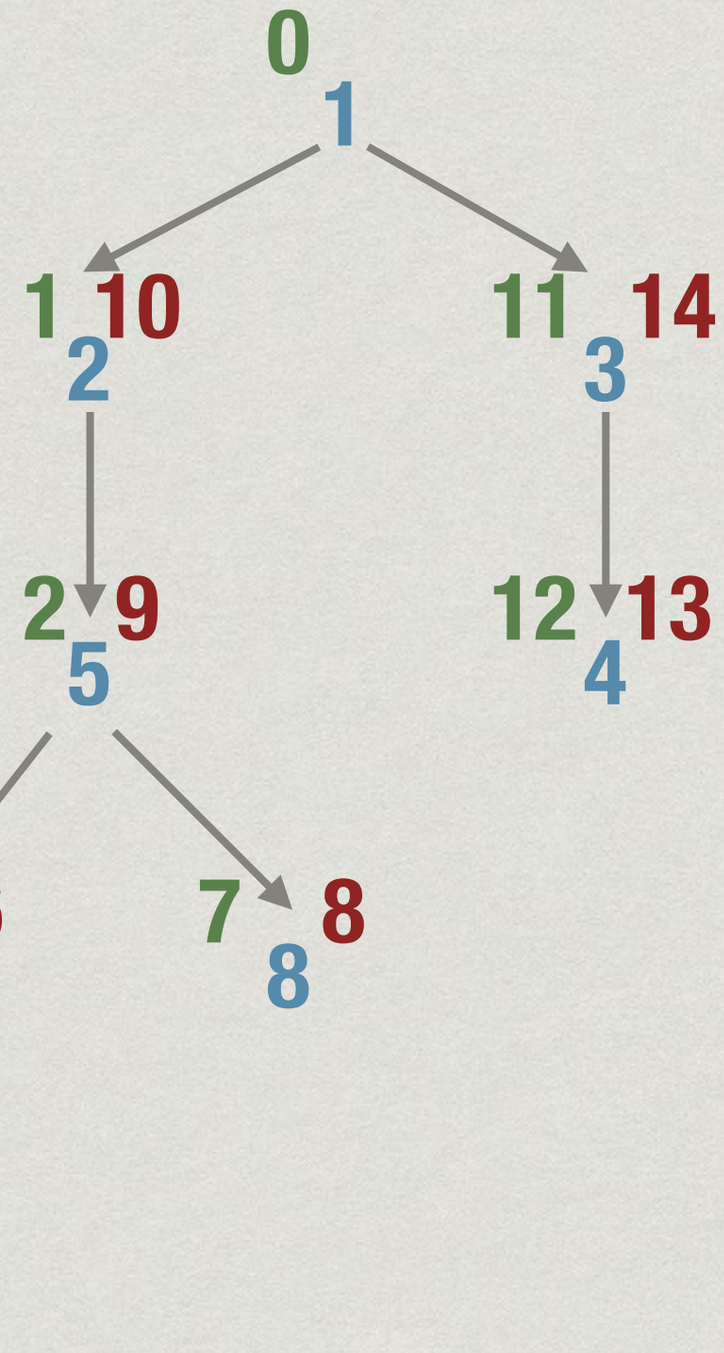
Directed cycles



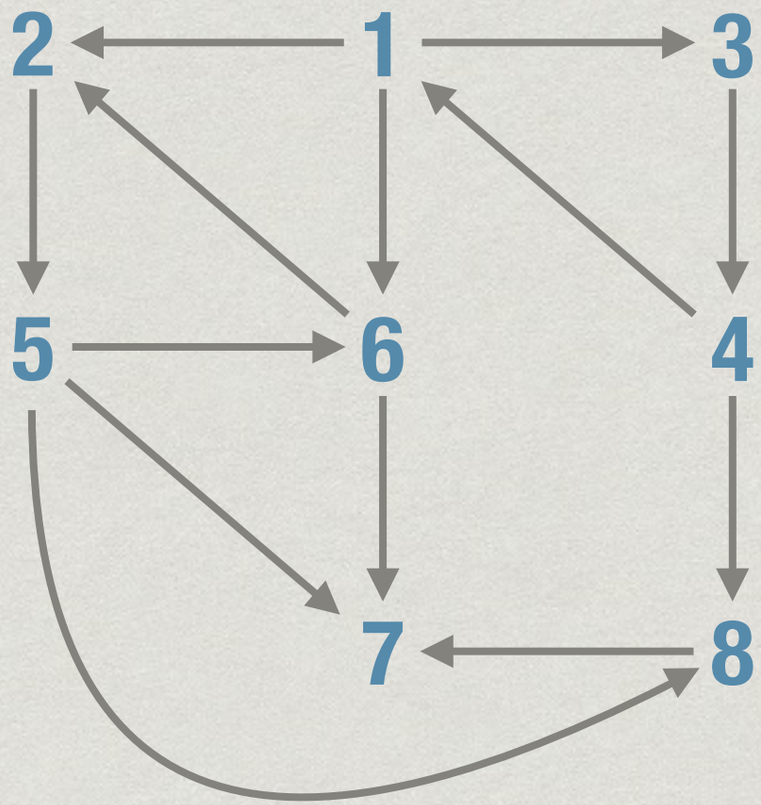


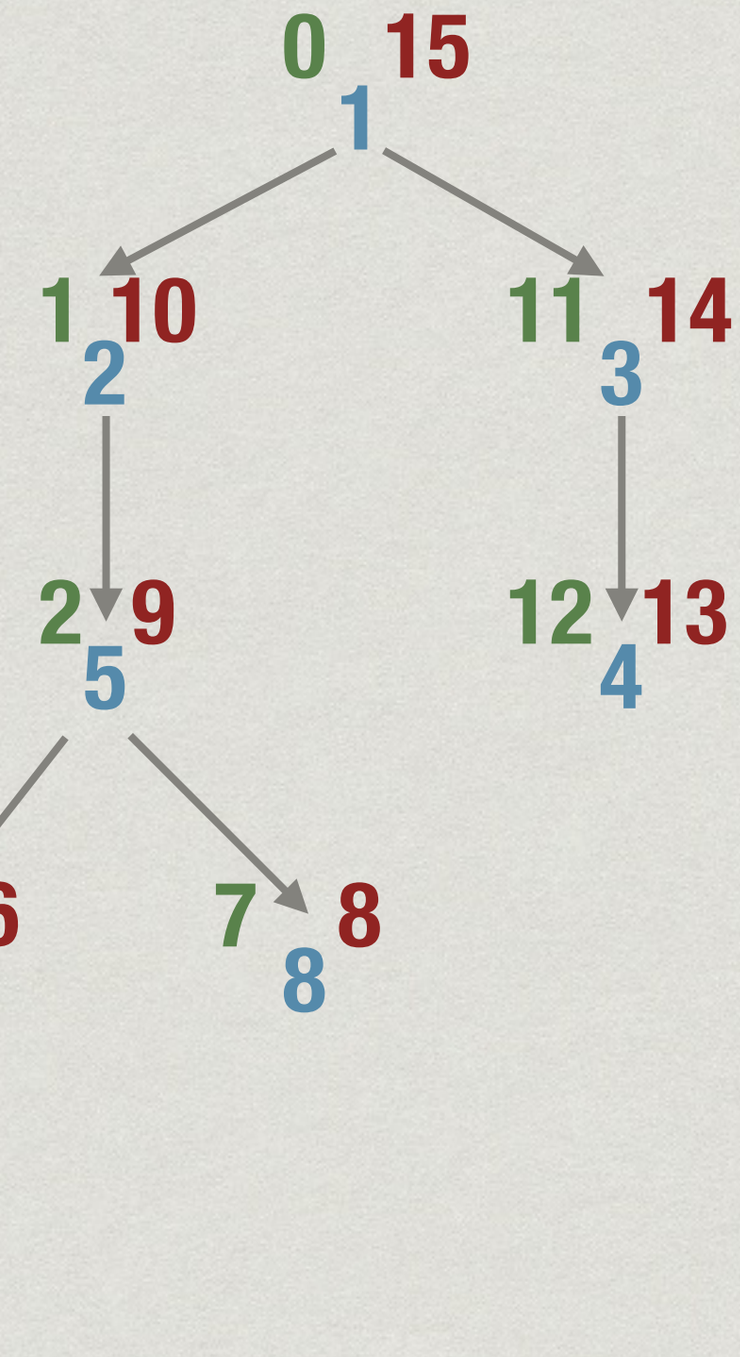
Directed cycles



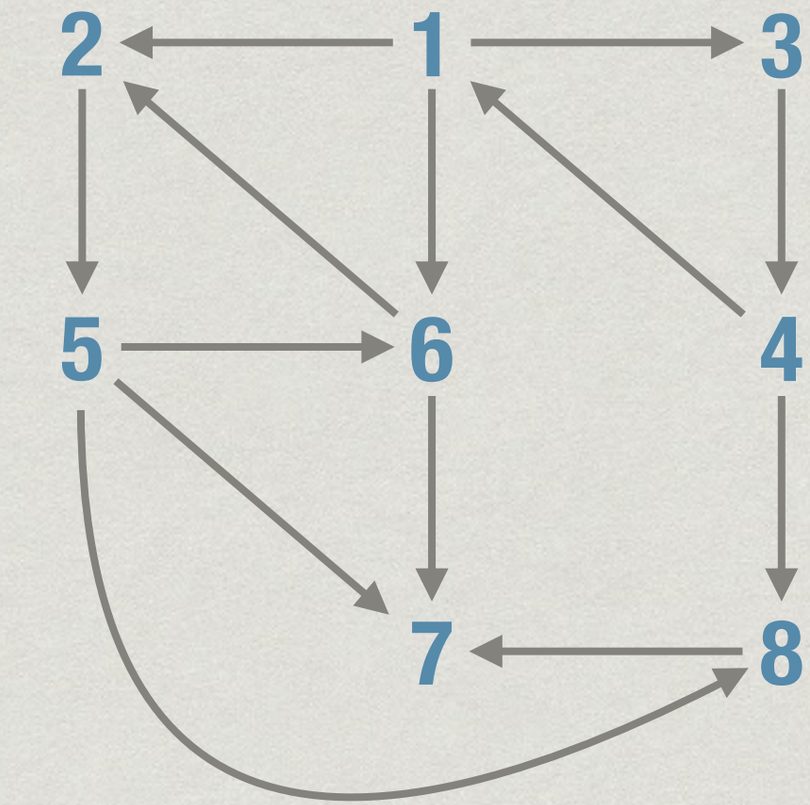


Directed cycles

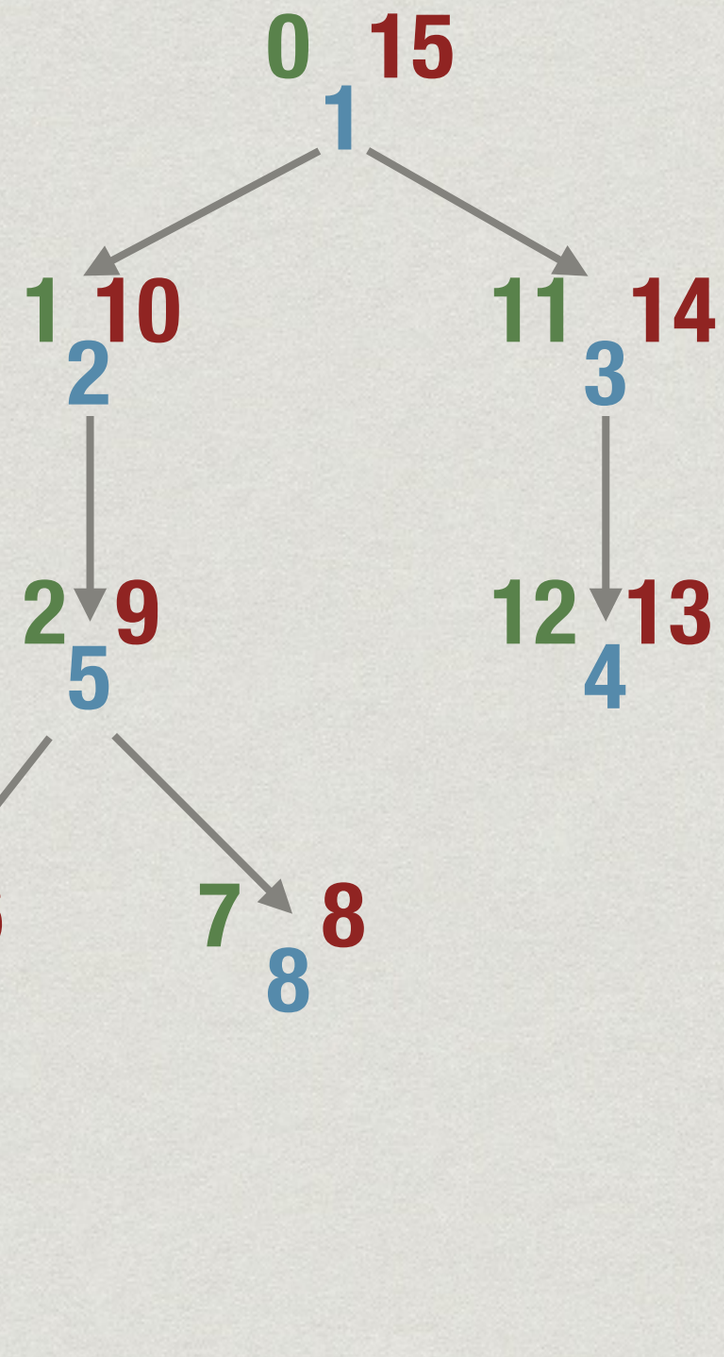




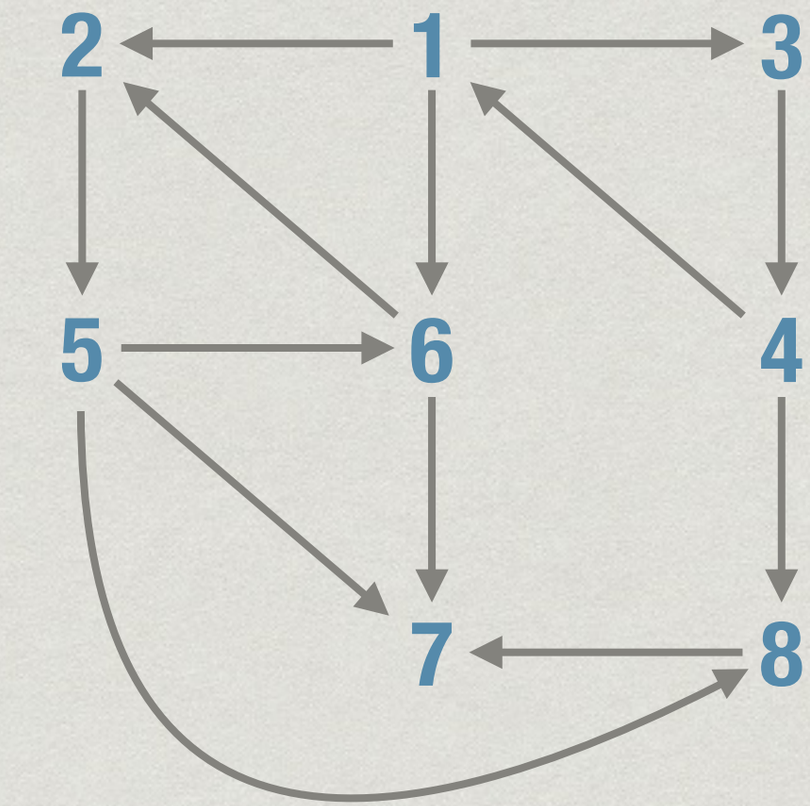
Directed cycles



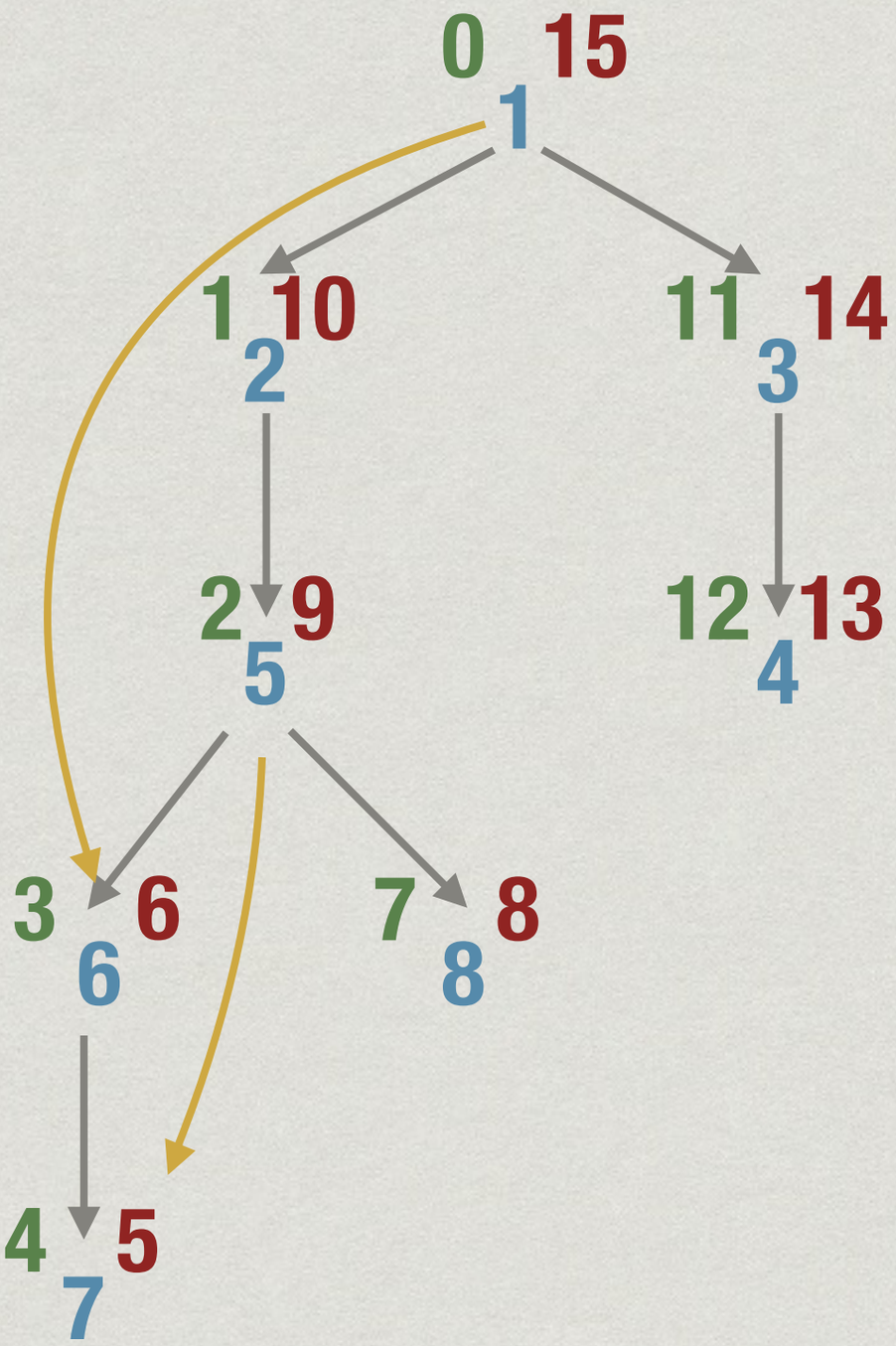
Tree edge



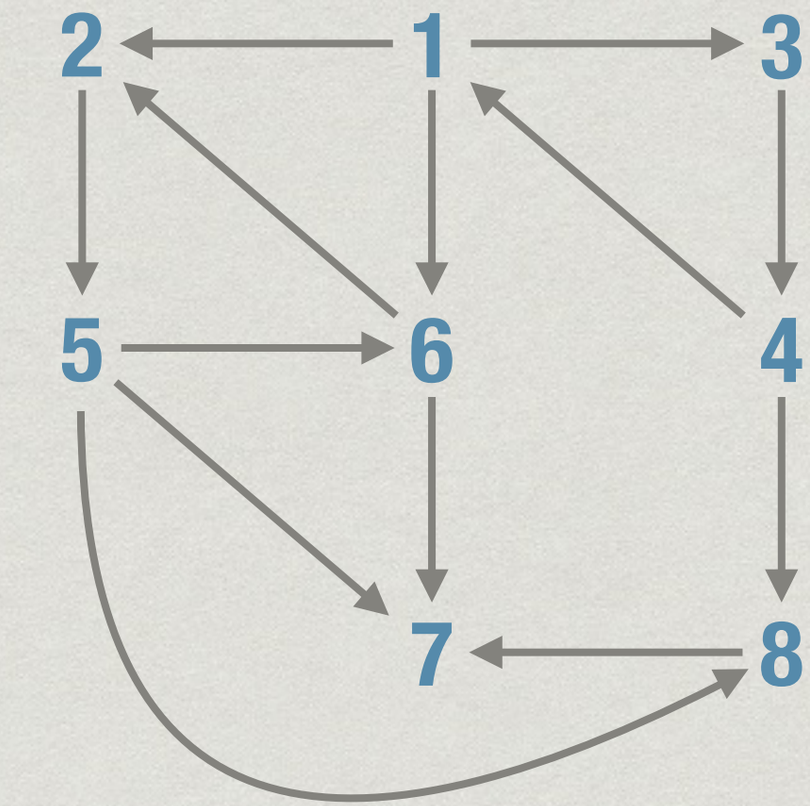
Directed cycles

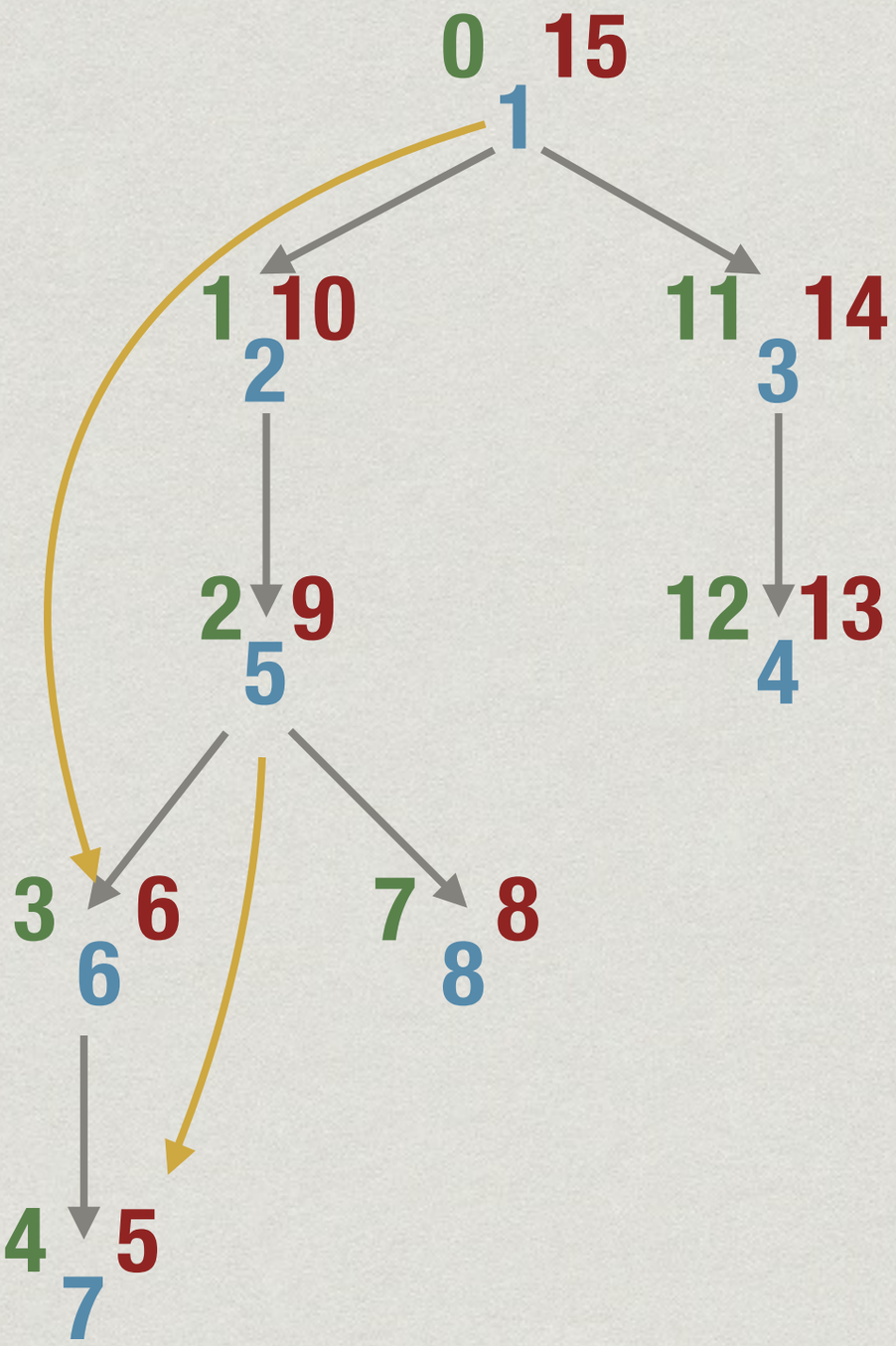


Tree edge



Directed cycles

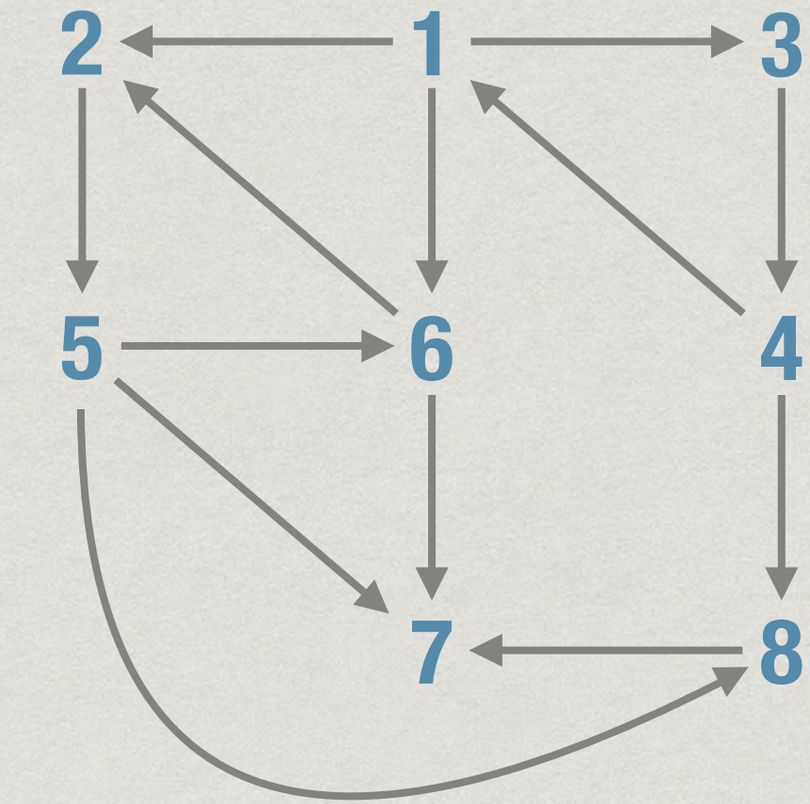


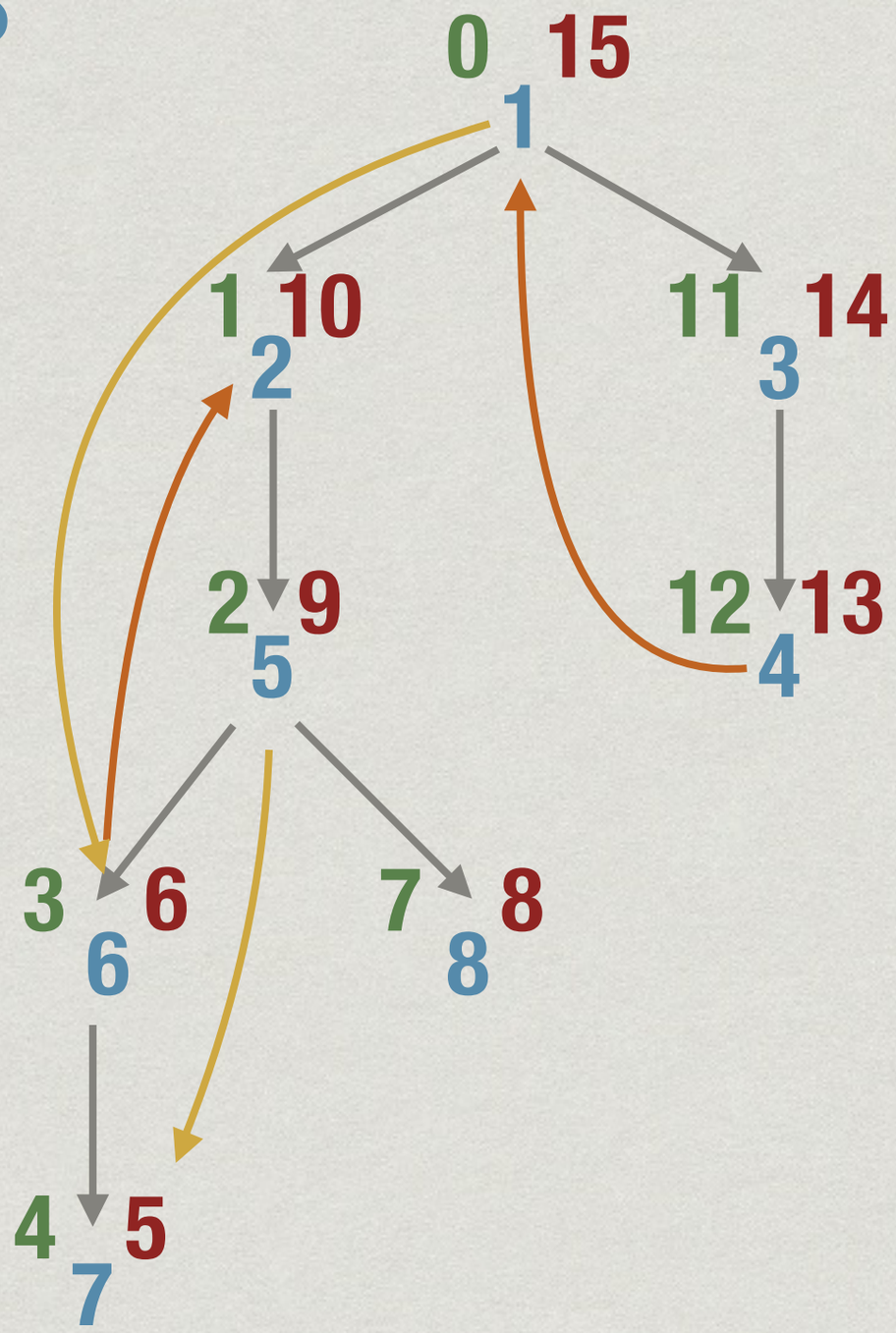


Tree edge

Forward edge

Directed cycles

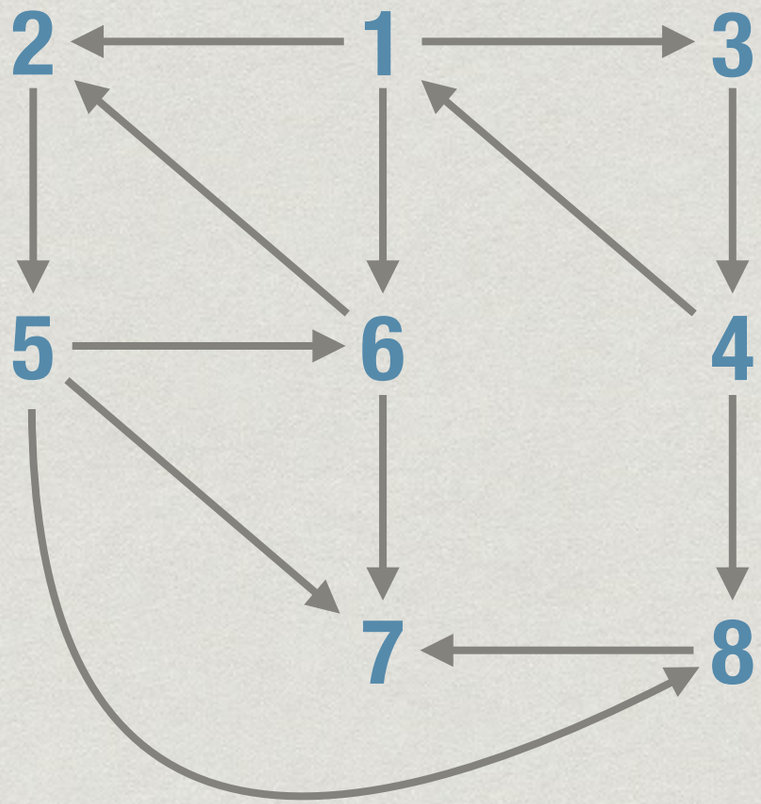


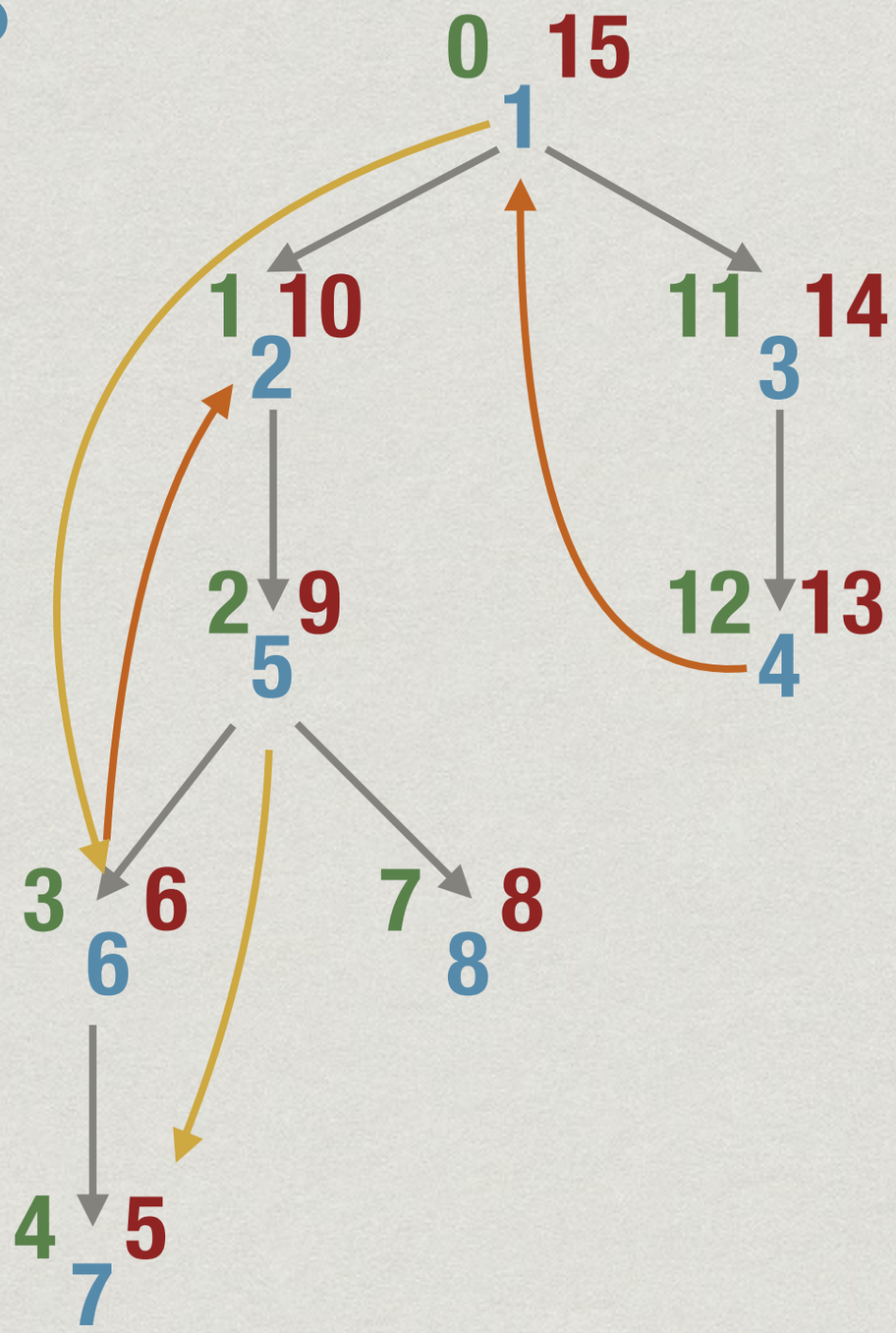


Tree edge

Forward edge

Directed cycles



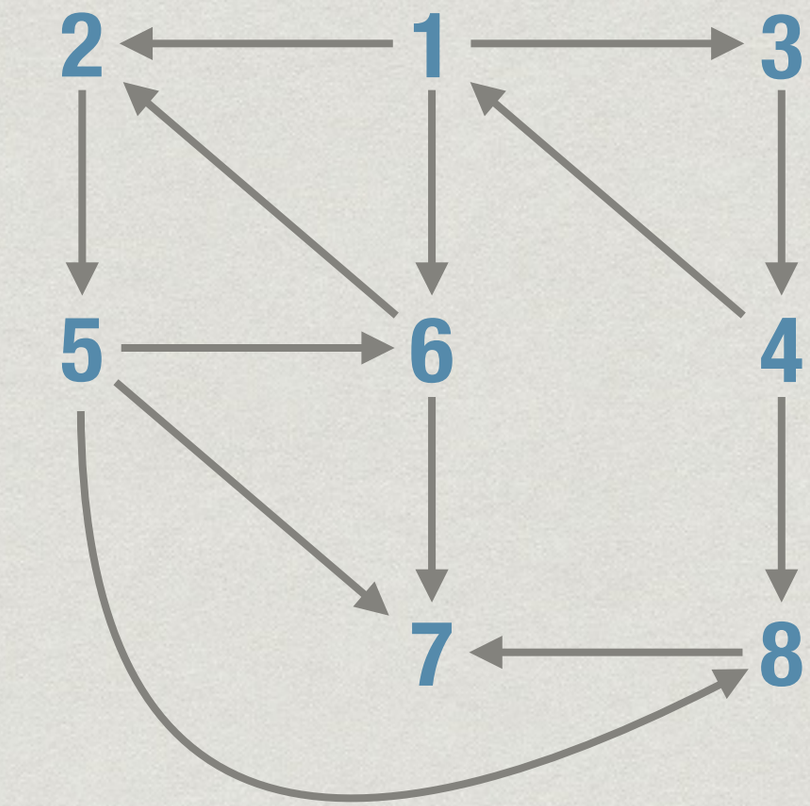


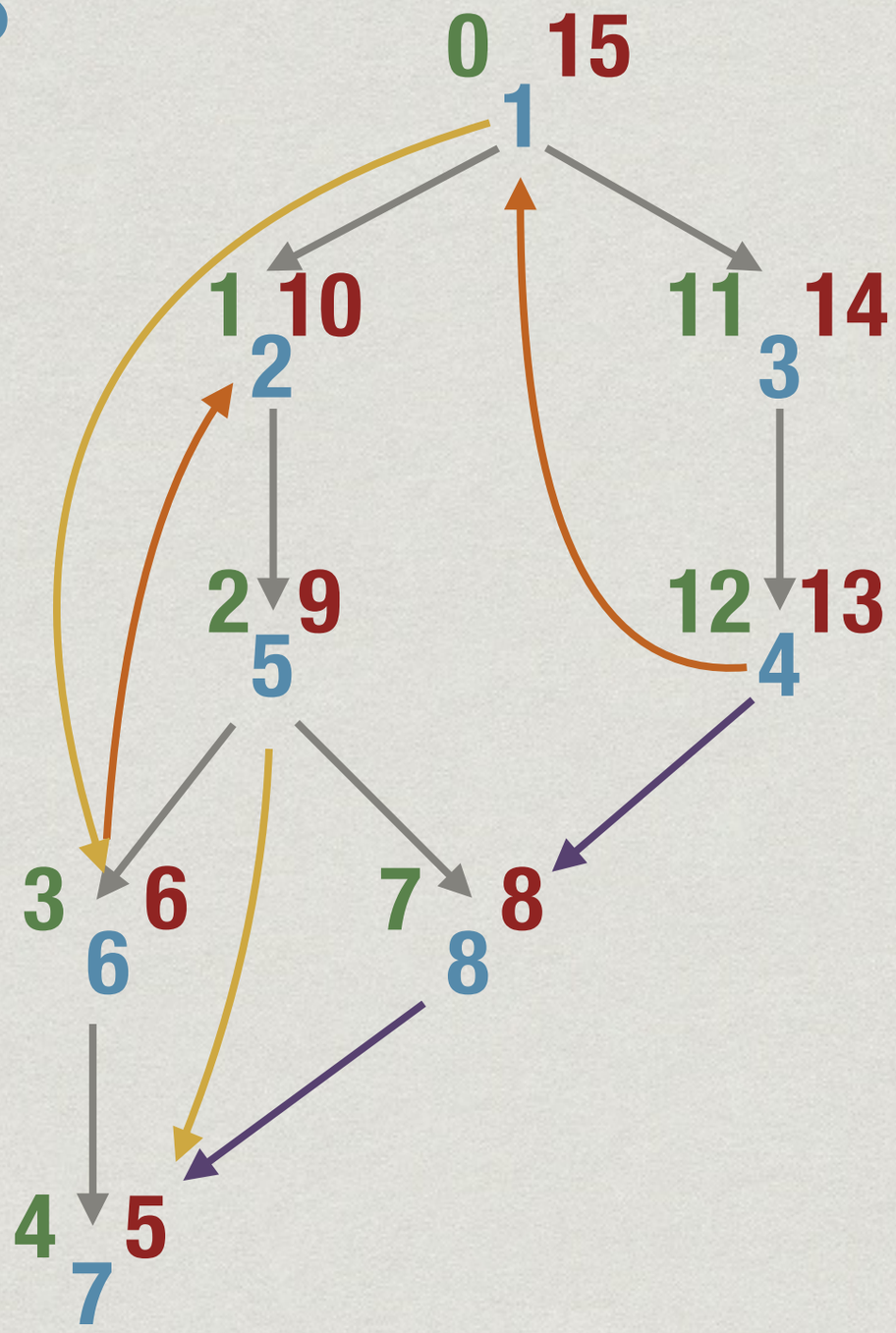
Tree edge

Forward edge

Back edge

Directed cycles



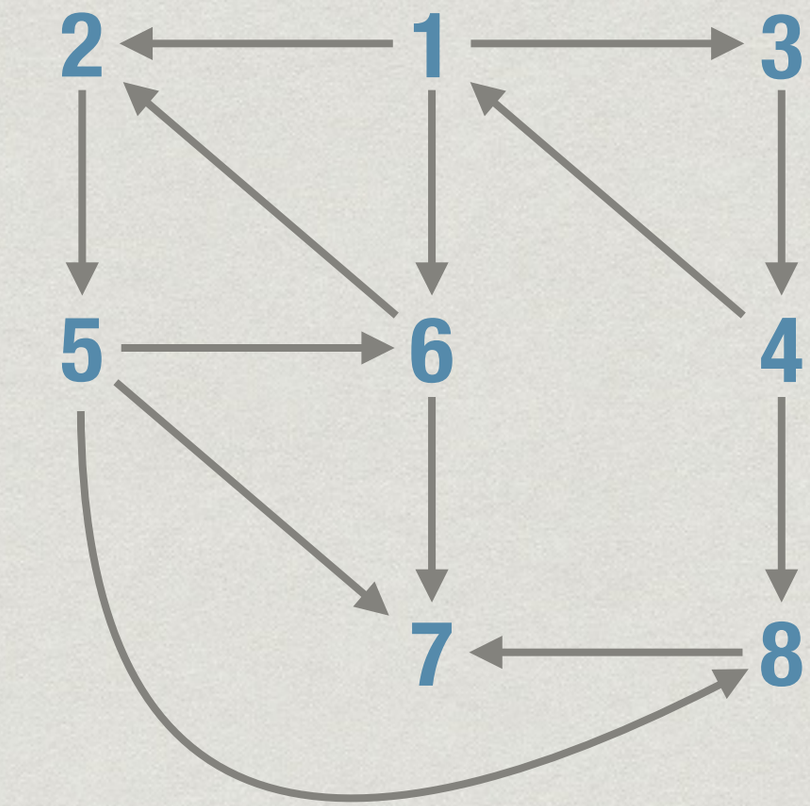


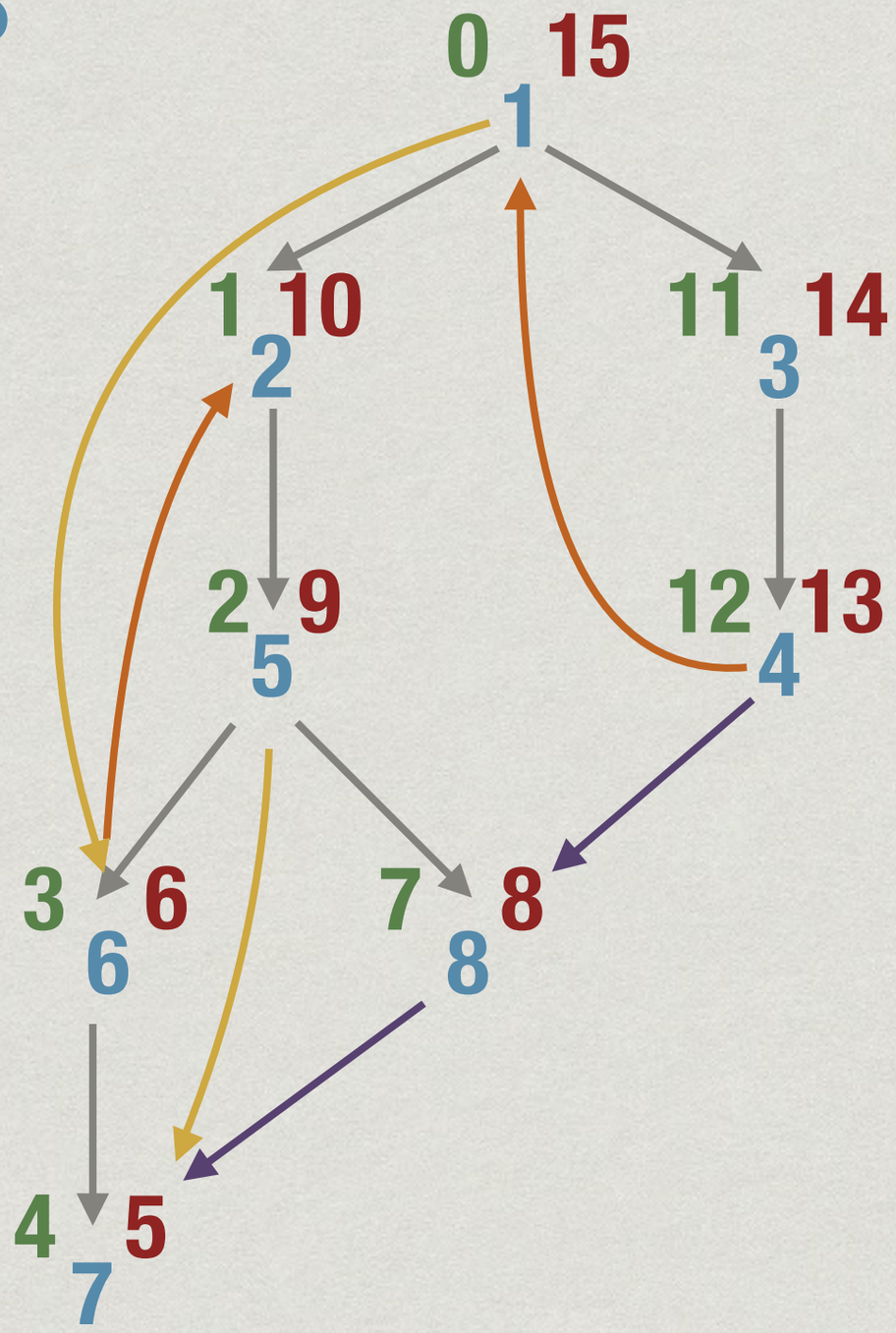
Tree edge

Forward edge

Back edge

Directed cycles





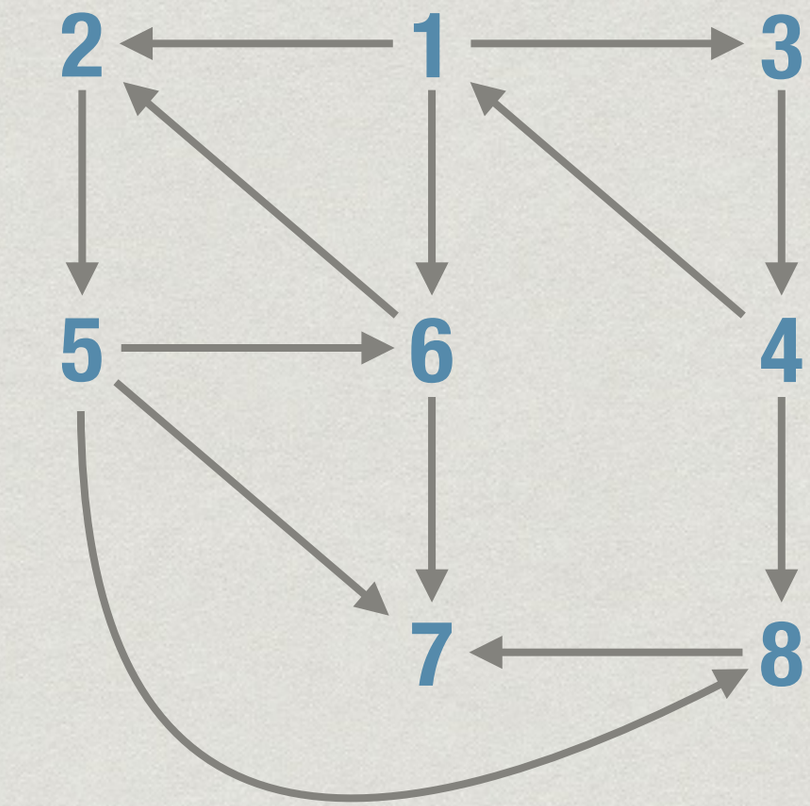
Tree edge

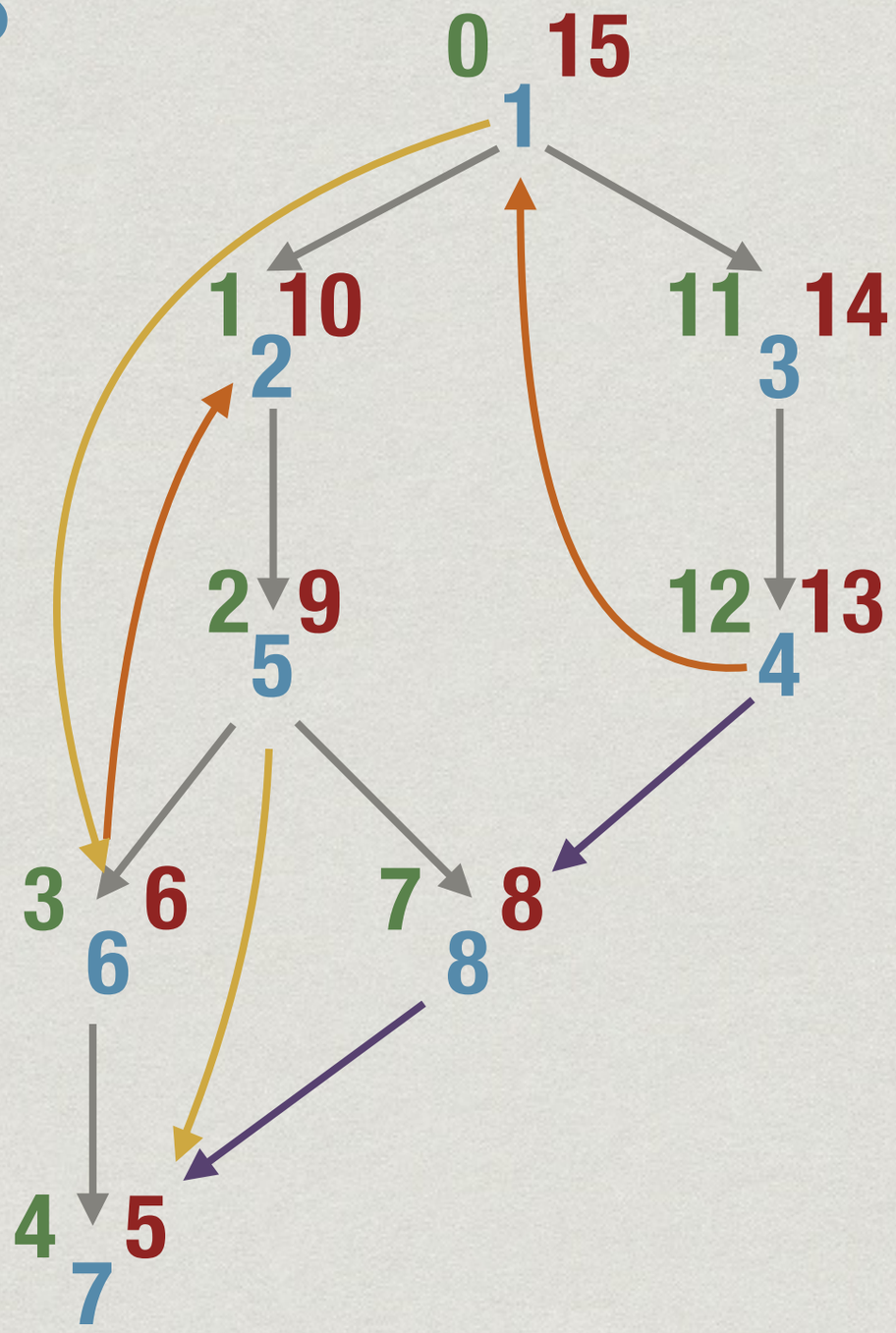
Forward edge

Back edge

Cross edge

Directed cycles





Tree edge

Forward edge

Back edge

Cross edge

Directed cycles

- A directed graph has a cycle if and only if DFS reveals a back edge
- Can classify edges using pre and post numbers
  - Tree/Forward edge (u,v) :
    Interval [pre(u),post(u)] contains [(pre(v),post(v)]
  - Backward edge (u,v):
    Interval [pre(v),post(v)] contains [(pre(u),post(u)]
  - Cross edge (u,v):
    Intervals [(pre(u),post(u)] and [(pre(v),post(v)] disjoint

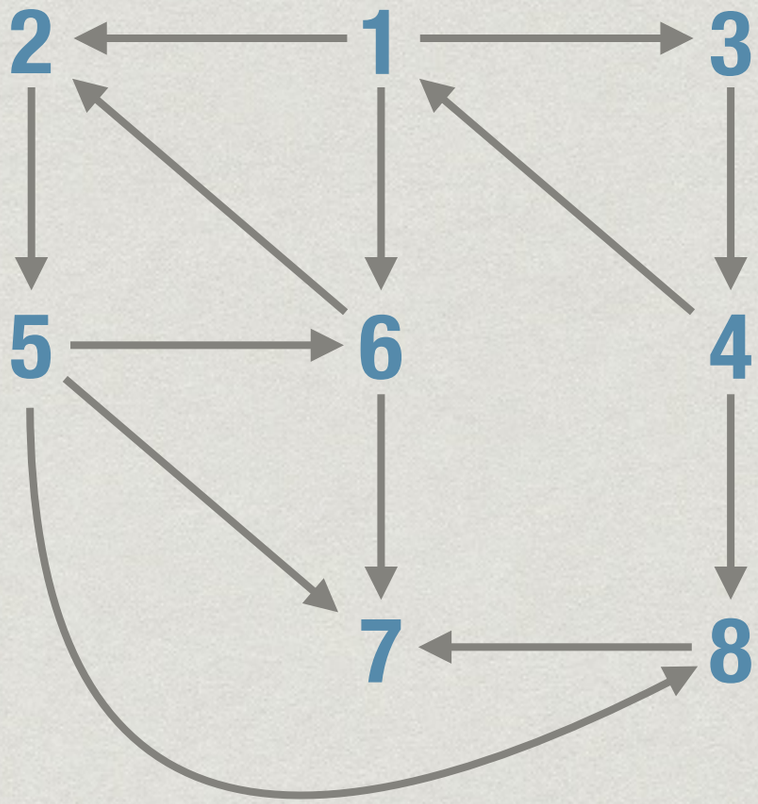
Directed acyclic graphs

- Directed graphs without cycles are useful for modelling dependencies
  - Courses with prerequisites
  - Edge (Algebra,Calculus) indicates that Algebra is a prerequisite for Calculus
- Will look at Directed Acyclic Graphs (DAGs) soon

Connectivity in directed graphs

- Need to take directions into account
- Nodes i and j are strongly connected if there is a path from i to j and a path from j to i
- Directed graph can be decomposed into strongly connected components (SCCs)
  - All pairs of nodes in an SCC are strongly connected

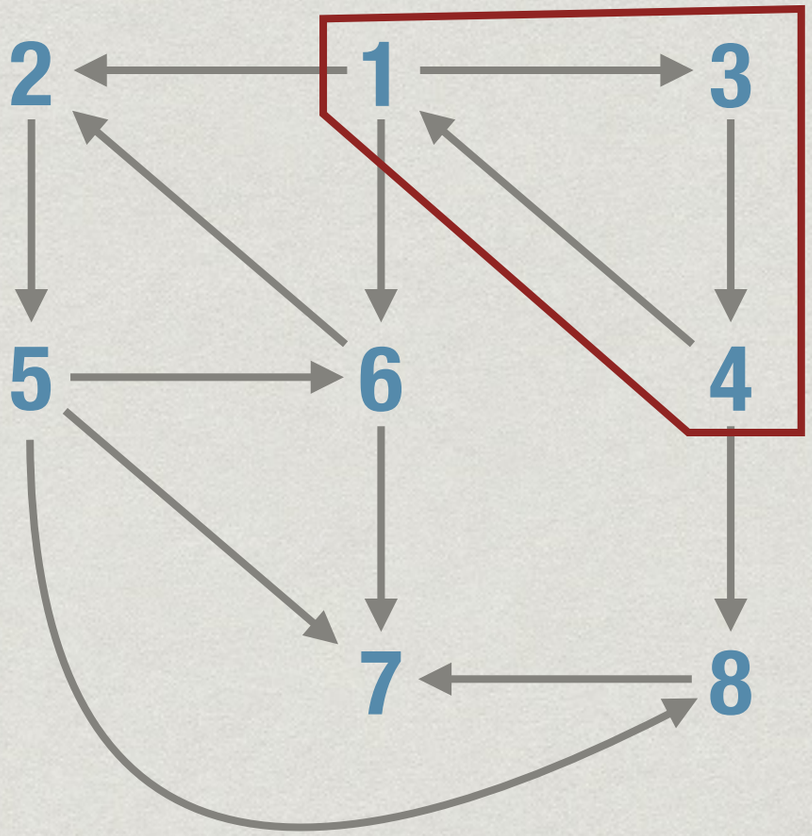
Computing SCCs



- DFS numbering (pre and post) can be used to compute SCCs

[Dasgupta, Papadimitriou,Vazirani]

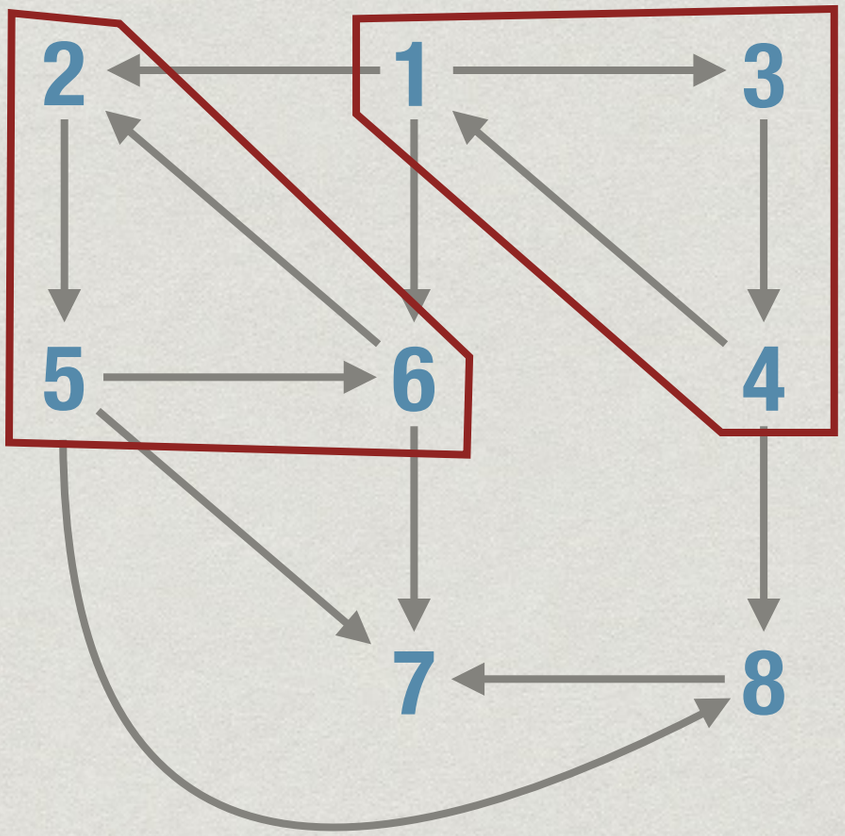
Computing SCCs



- DFS numbering (pre and post) can be used to compute SCCs

[Dasgupta, Papadimitriou,Vazirani]

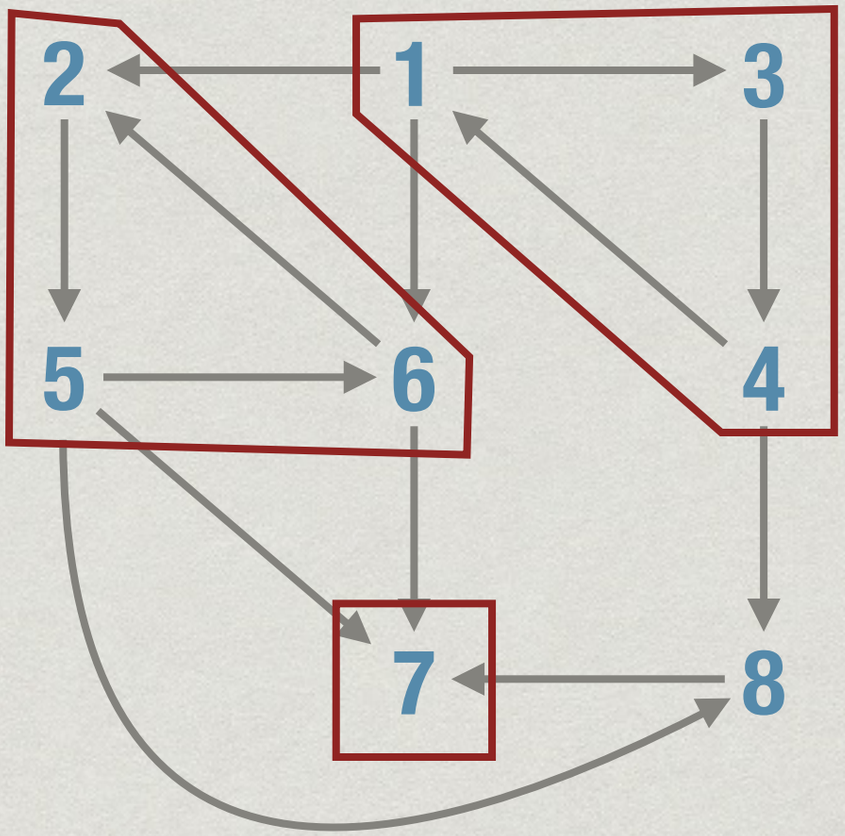
Computing SCCs



- DFS numbering (pre and post) can be used to compute SCCs

[Dasgupta, Papadimitriou,Vazirani]

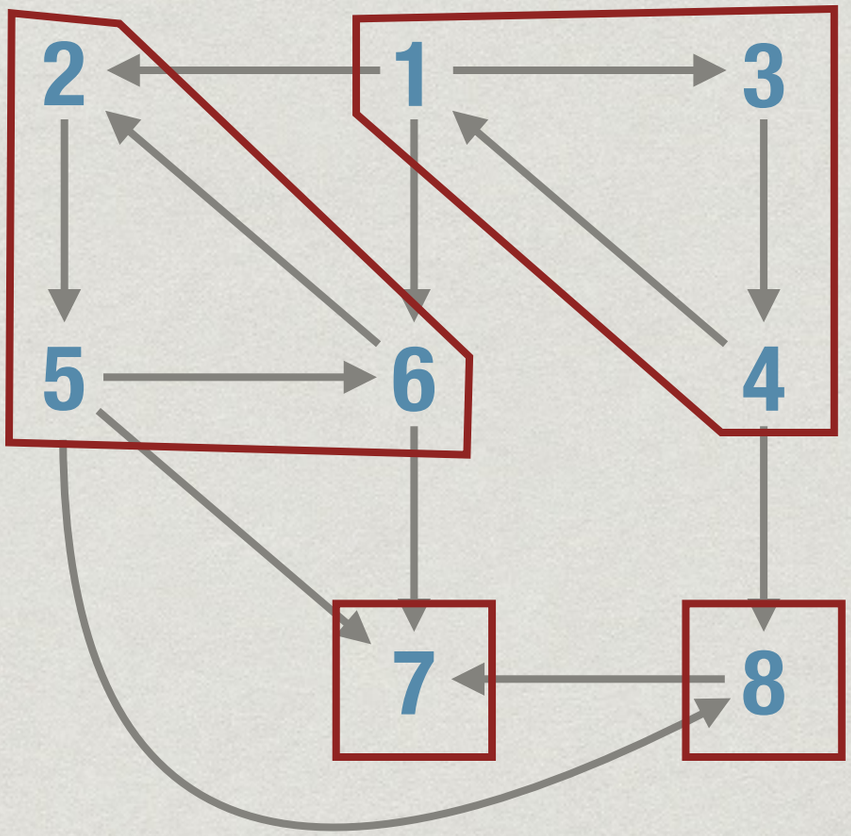
Computing SCCs



- DFS numbering (pre and post) can be used to compute SCCs

[Dasgupta, Papadimitriou,Vazirani]

Computing SCCs



- DFS numbering (pre and post) can be used to compute SCCs

[Dasgupta, Papadimitriou,Vazirani]

Other properties

- A number of other structural properties can be inferred from DFS numbering
- Articulation points (vertices)
  - Removing such a vertex disconnects the graph
- Bridges (edges)
  - Removing such an edge disconnects the graph