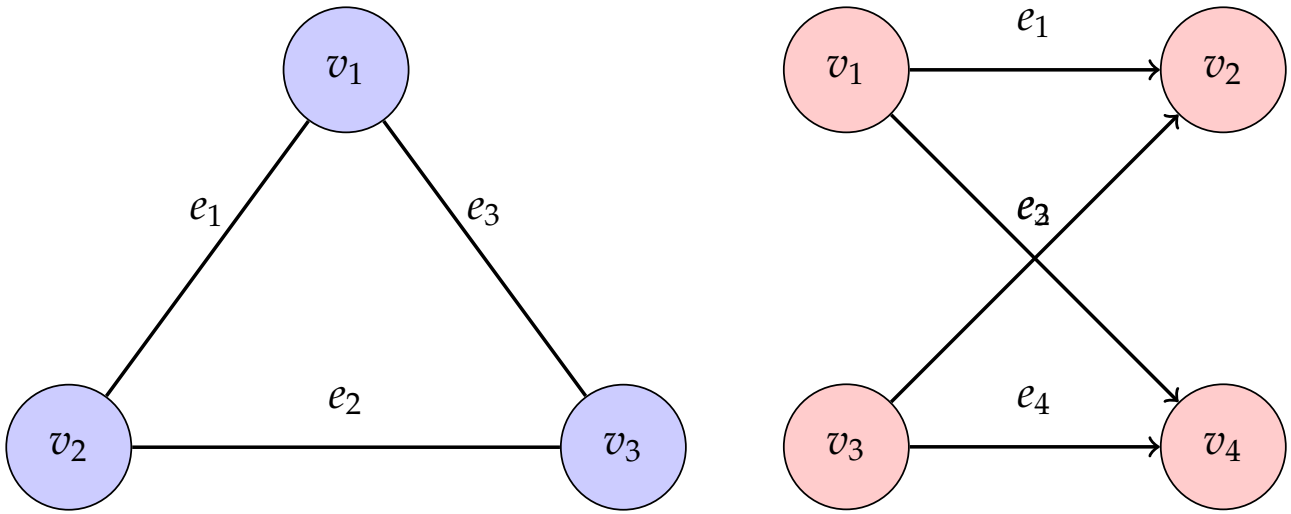
CSCI 2270: Data Structures

Lecture 28–30: Graph Traversal

Ashutosh Trivedi



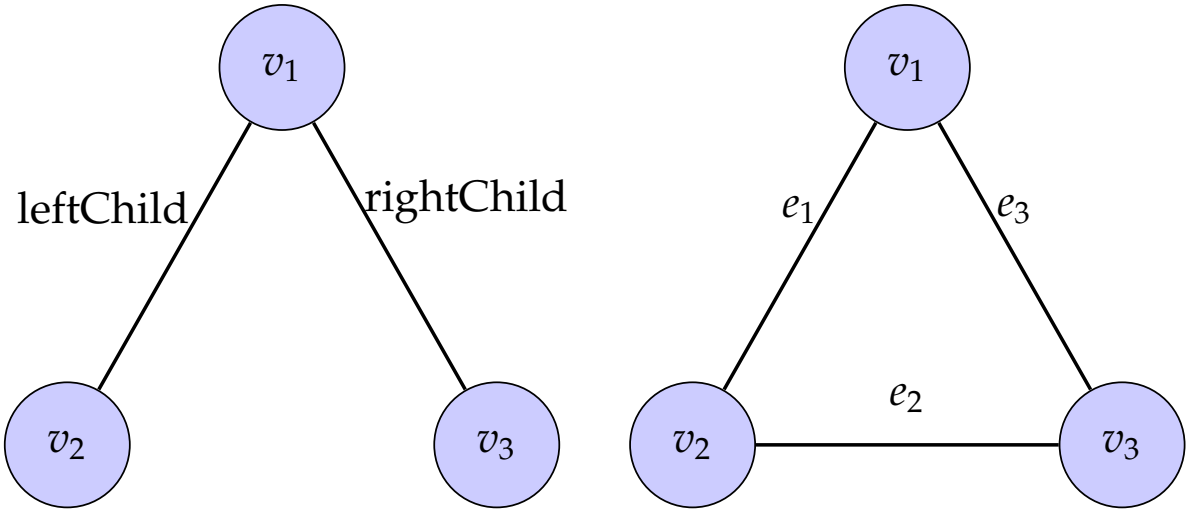
Department of Computer Science
UNIVERSITY OF COLORADO BOULDER

Graphs

Graph Traversal

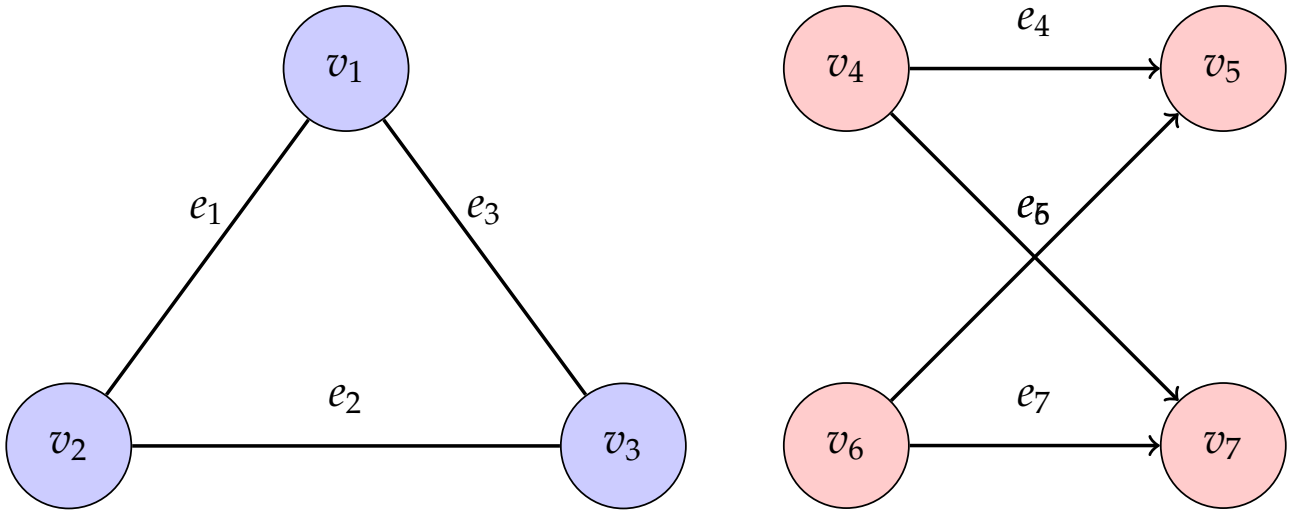
C++ Standard Template Library: Containers

Trees Vs Graph



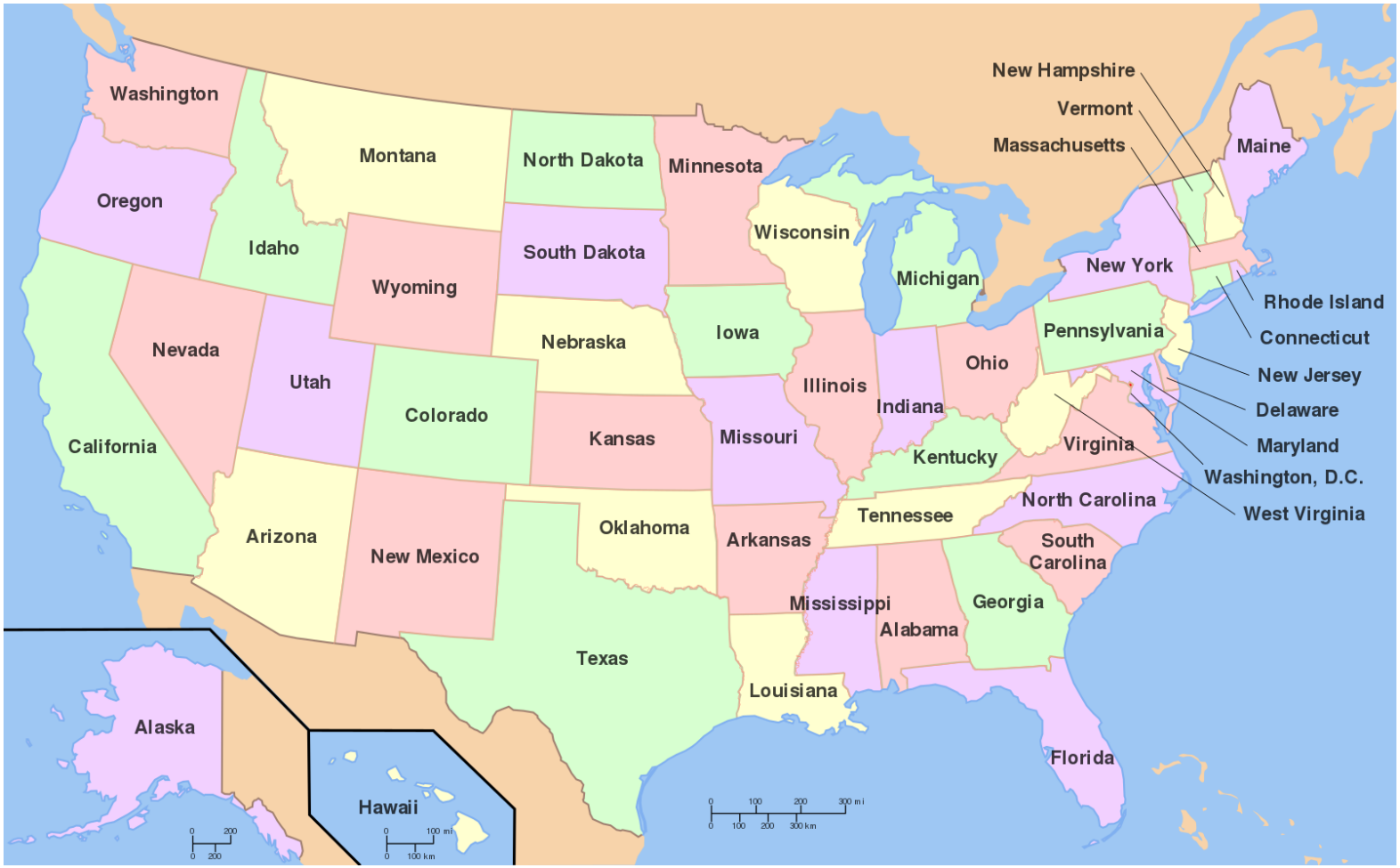
- Trees: *parent, child, sibling, ancestors, successors*
- Graphs: *adjacent node, nodes distance k apart, path, shortest path*

Undirected vs directed graphs

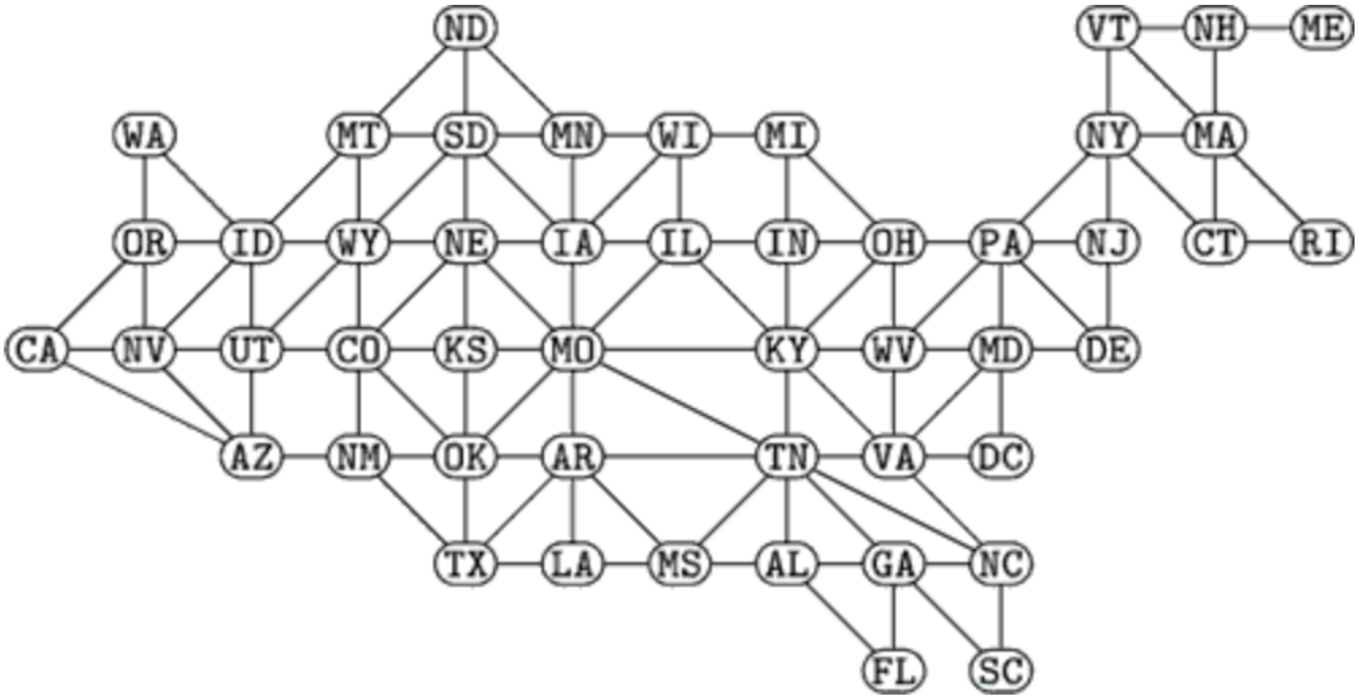


- $G = (V, E)$ where V is the set of *vertices* and E is the set of edges.
- Example: $G_1 = (V_1, E_1)$ where $V_1 = \{v_1, v_2, v_3\}$ and $E_1 = \{e_1, e_2, e_3\}$.
- Undirected Graph: edges are bi-directional.
- Directed Graphs: edges are one-directional.

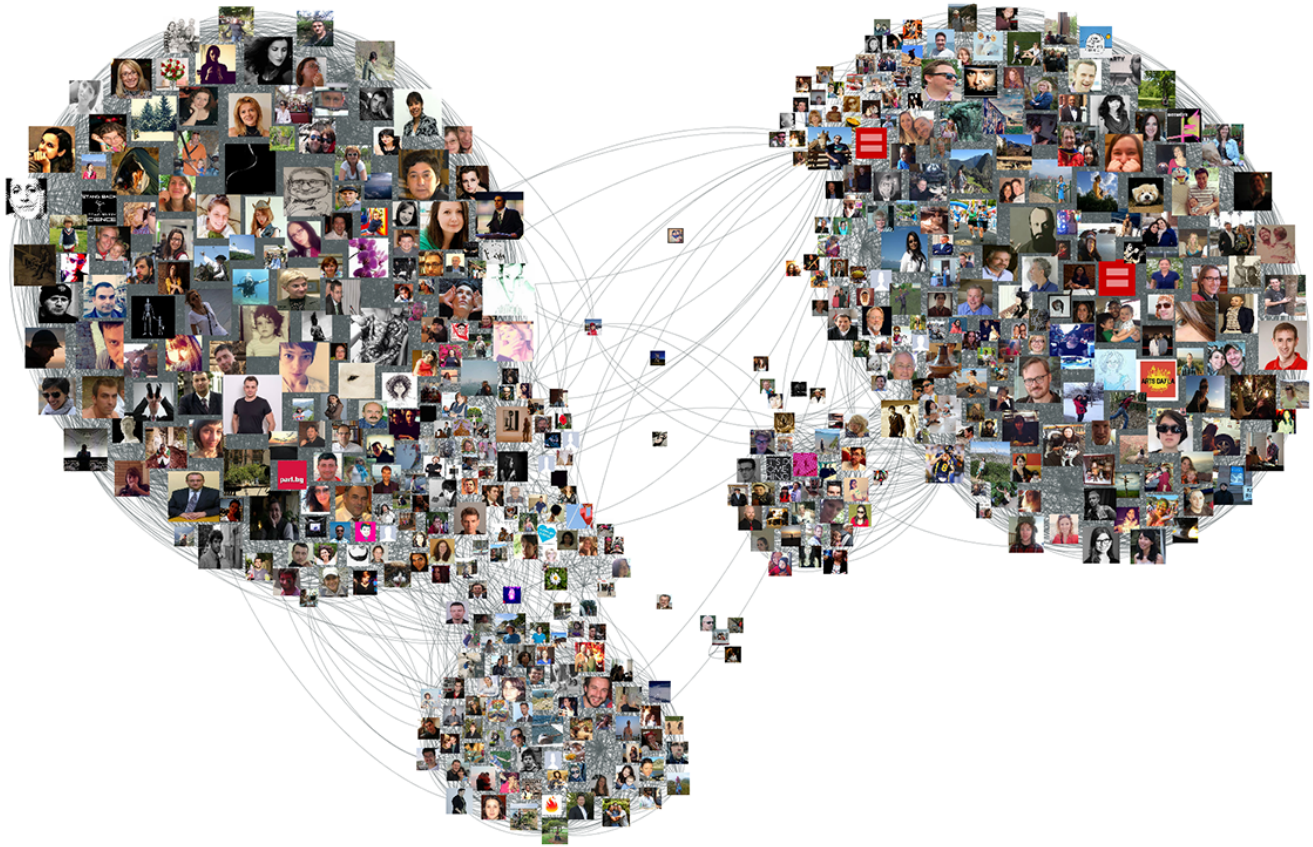
Graphs


Washington
Oregon
California
Idaho
Nevada
Montana
Wyoming
Utah
Arizona
Colorado
New Mexico
North Dakota
South Dakota
Nebraska
Kansas
Oklahoma
Texas
Minnesota
Iowa
Missouri
Arkansas
Louisiana
Wisconsin
Illinois
Michigan
Indiana
Ohio
Kentucky
Tennessee
Mississippi
Alabama
Georgia
Florida
South Carolina
North Carolina
Virginia
West Virginia
Pennsylvania
New York
Maryland
Delaware
New Jersey
Connecticut
Rhode Island
Washington, D.C.
Massachusetts
Vermont
New Hampshire
Maine
Alaska
Hawaii

Graphs

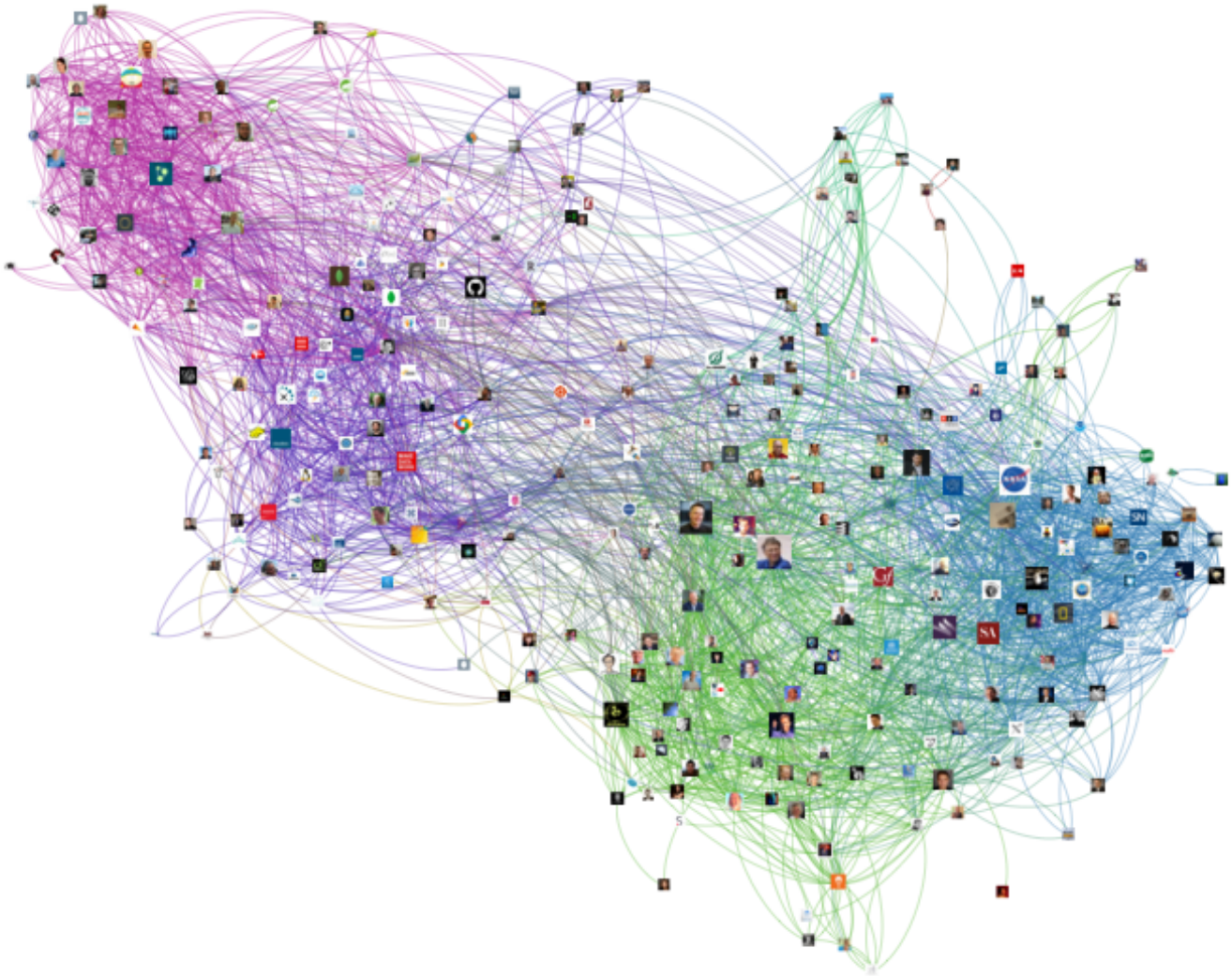

ND
VT
NH
ME
WA
MT
SD
MN
WI
MI
NY
MA
OR
ID
WY
NE
IA
IL
IN
OH
PA
NJ
CT
RI
CA
NV
UT
CO
KS
MO
KY
WV
MD
DE
AZ
NM
OK
AR
TN
VA
DC
TX
LA
MS
AL
GA
NC
FL
SC

Graphs-Facebook Friends Graph



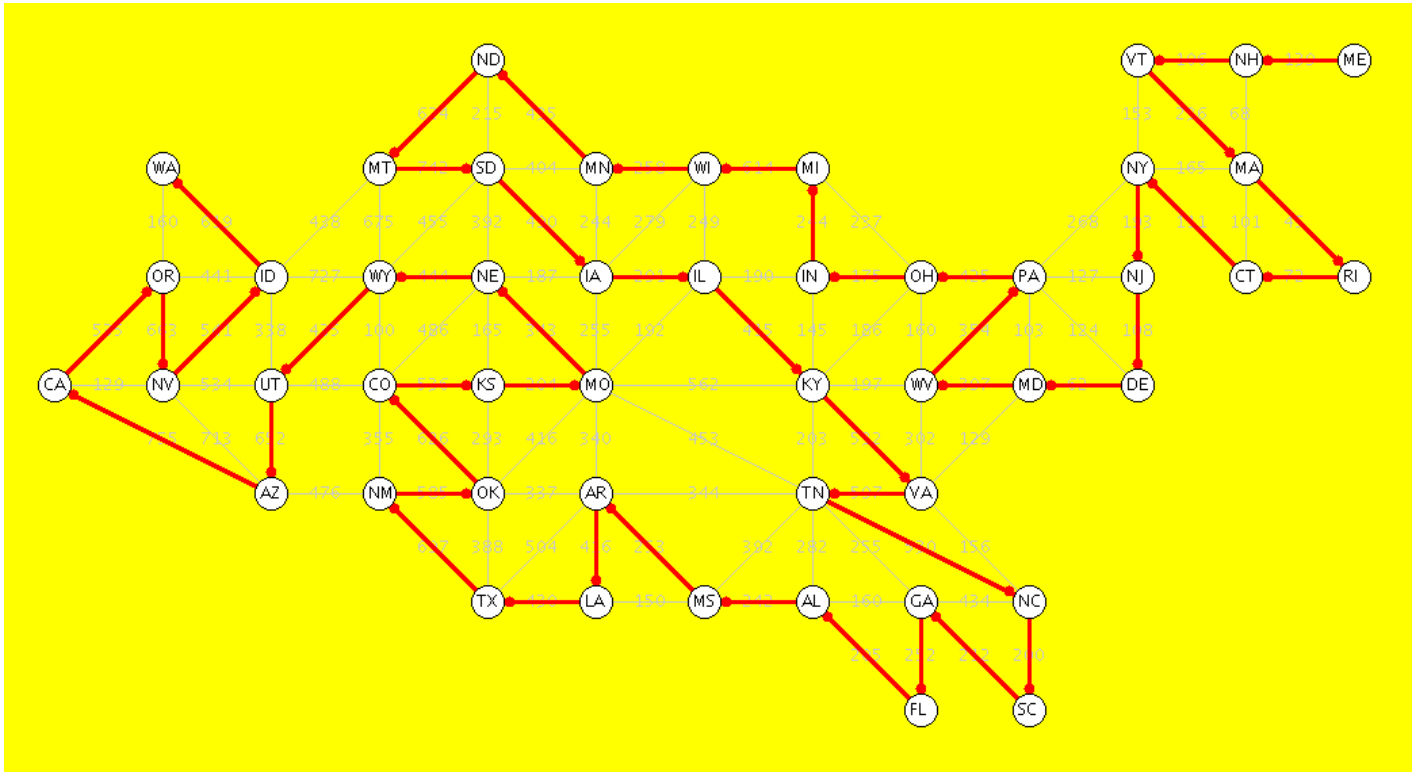
https://gephi.org

Graphs—Twitter Follower Graph

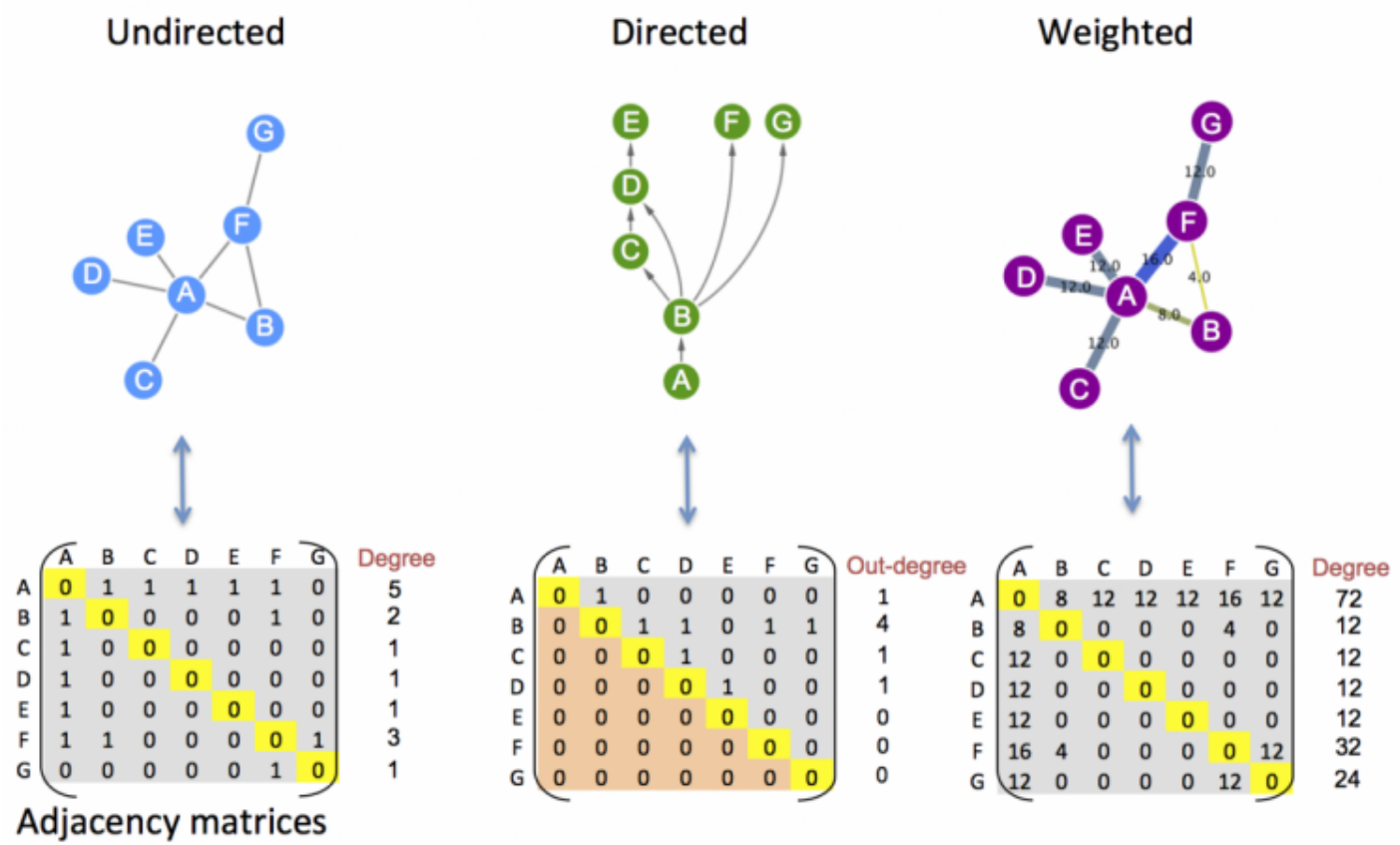


http://allthingsgraphed.com/2014/11/02/twitter-friends-network/

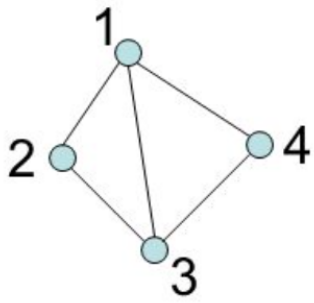
Weighted Graphs—Distance Graph



Graph: Adjacency Matrix

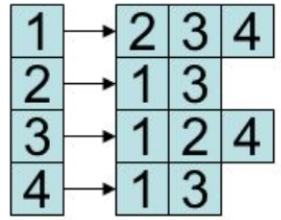


Graph: Adjacency List



	1	2	3	4
1	0	1	1	1
2	1	0	1	0
3	1	1	0	1
4	1	0	1	0

Adjacency matrix



Adjacency list

- Adjacency matrix can be represented as a $2D$ array.
- Adjacency list can be represented as an array of lists.

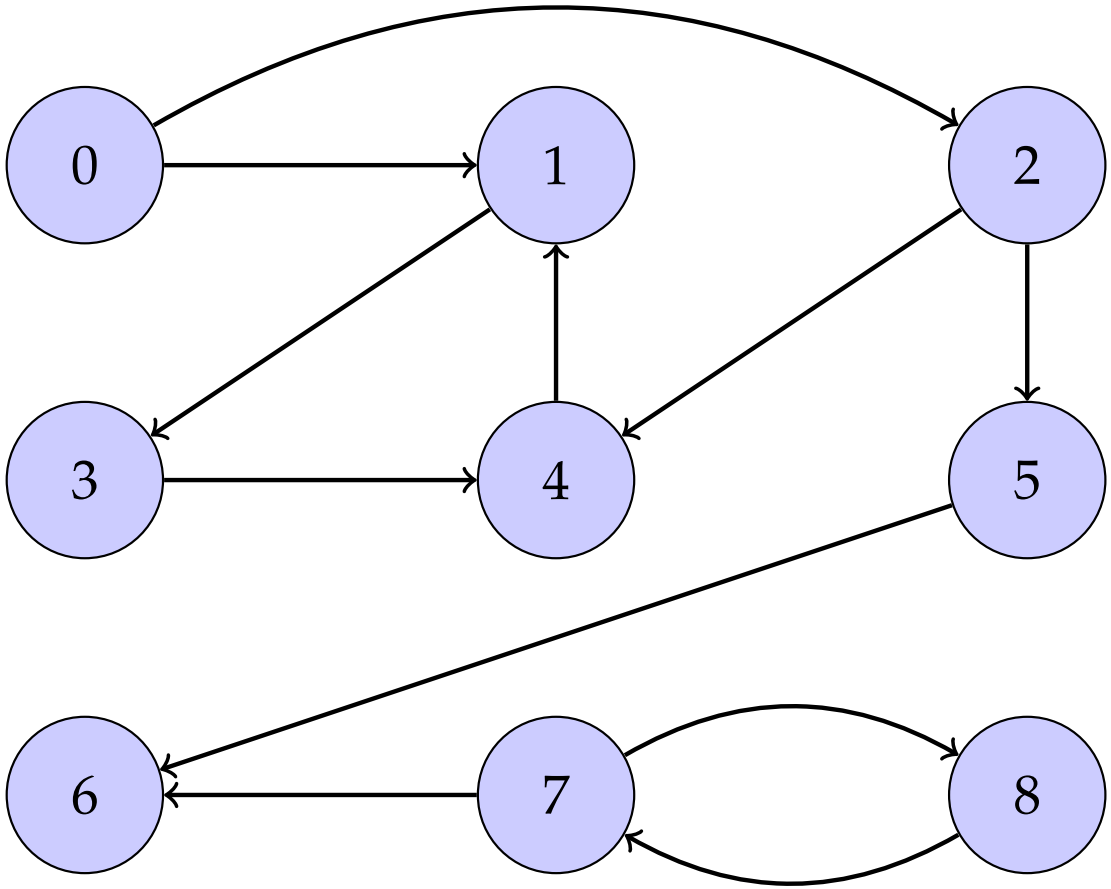
Graphs

Graph Traversal

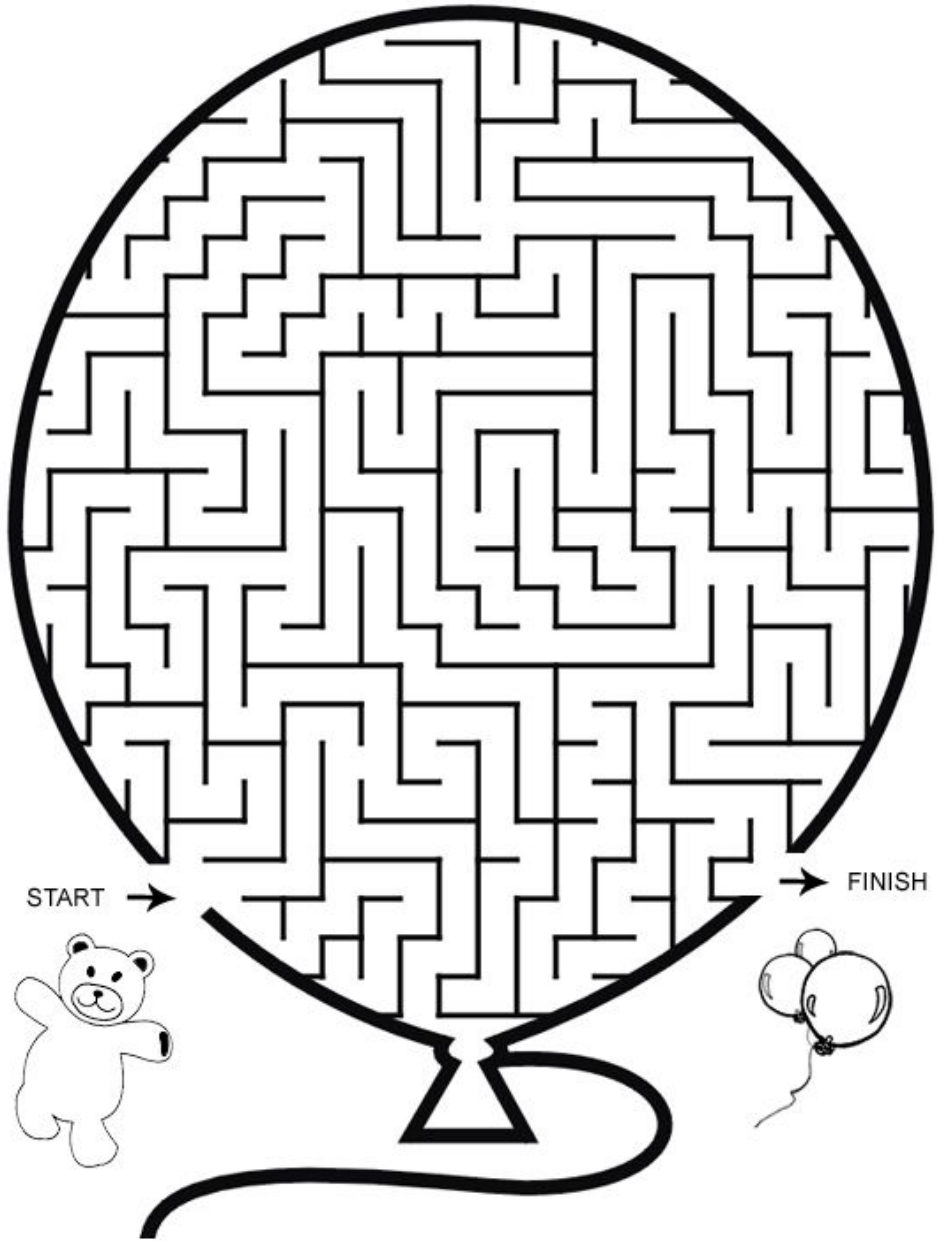
C++ Standard Template Library: Containers

Graph Traversal

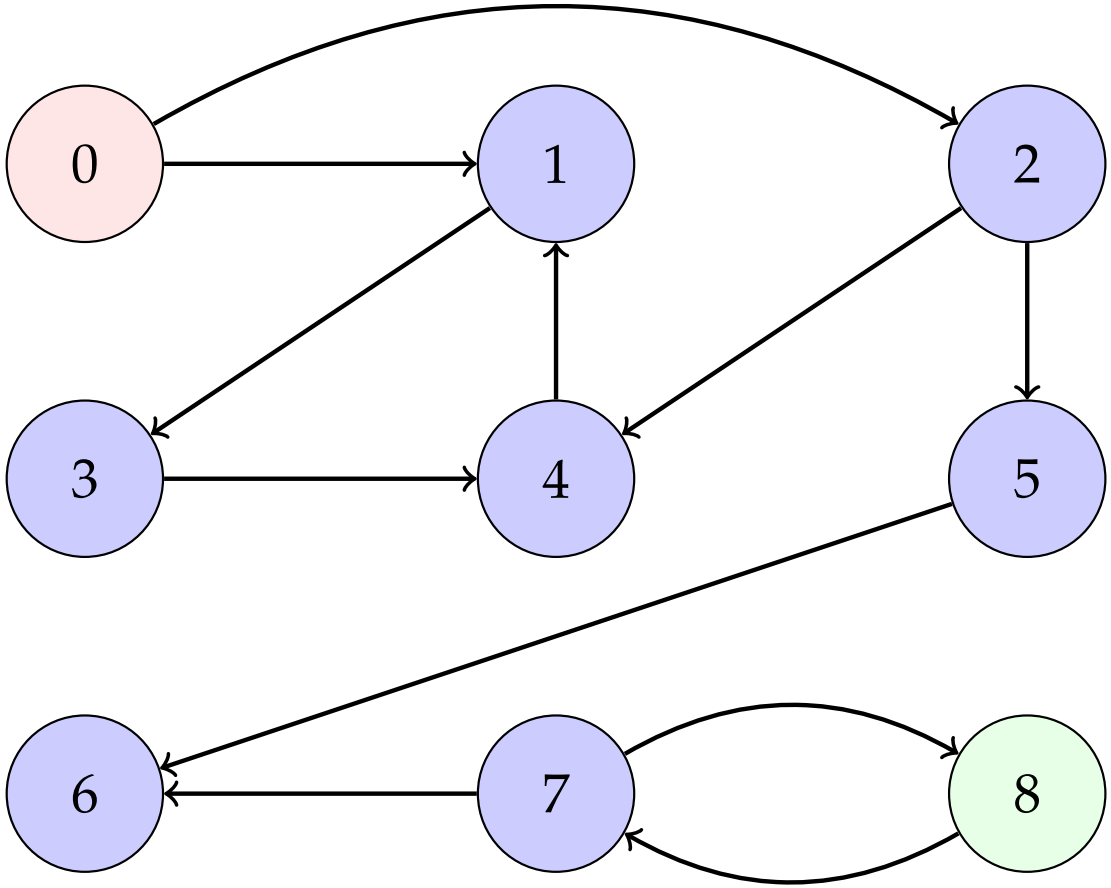
1. *Is there a sequence of edges (path) following which we can reach from s to t?*
2. *What is a shortest such path?*



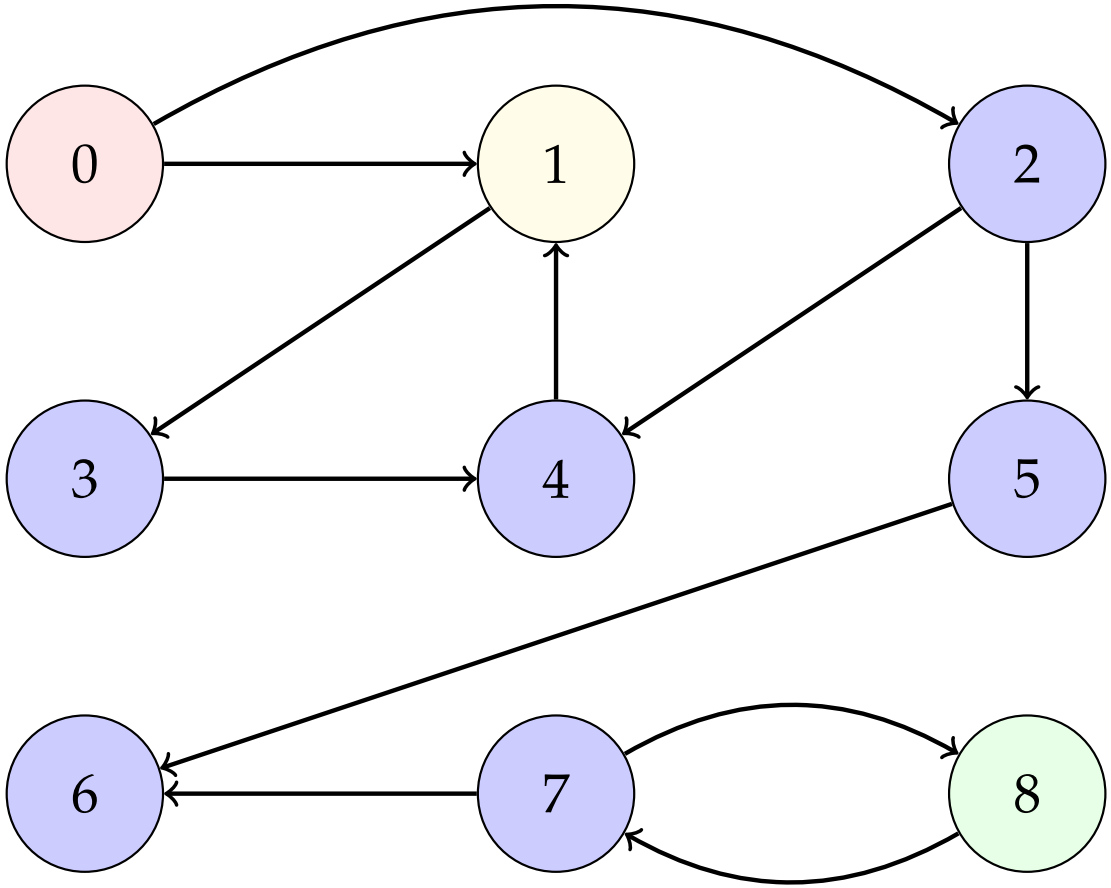
Show Benny the bear the way to his balloon



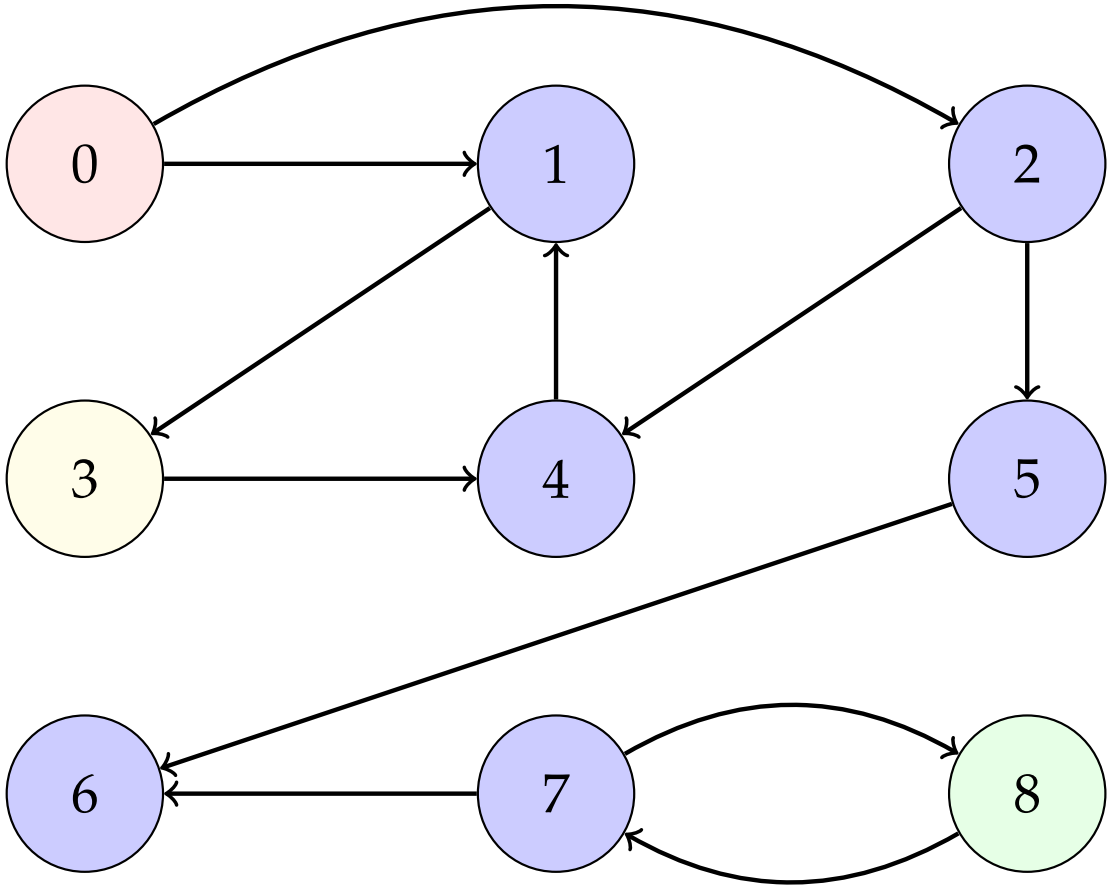
Graph Traversal: Naïve



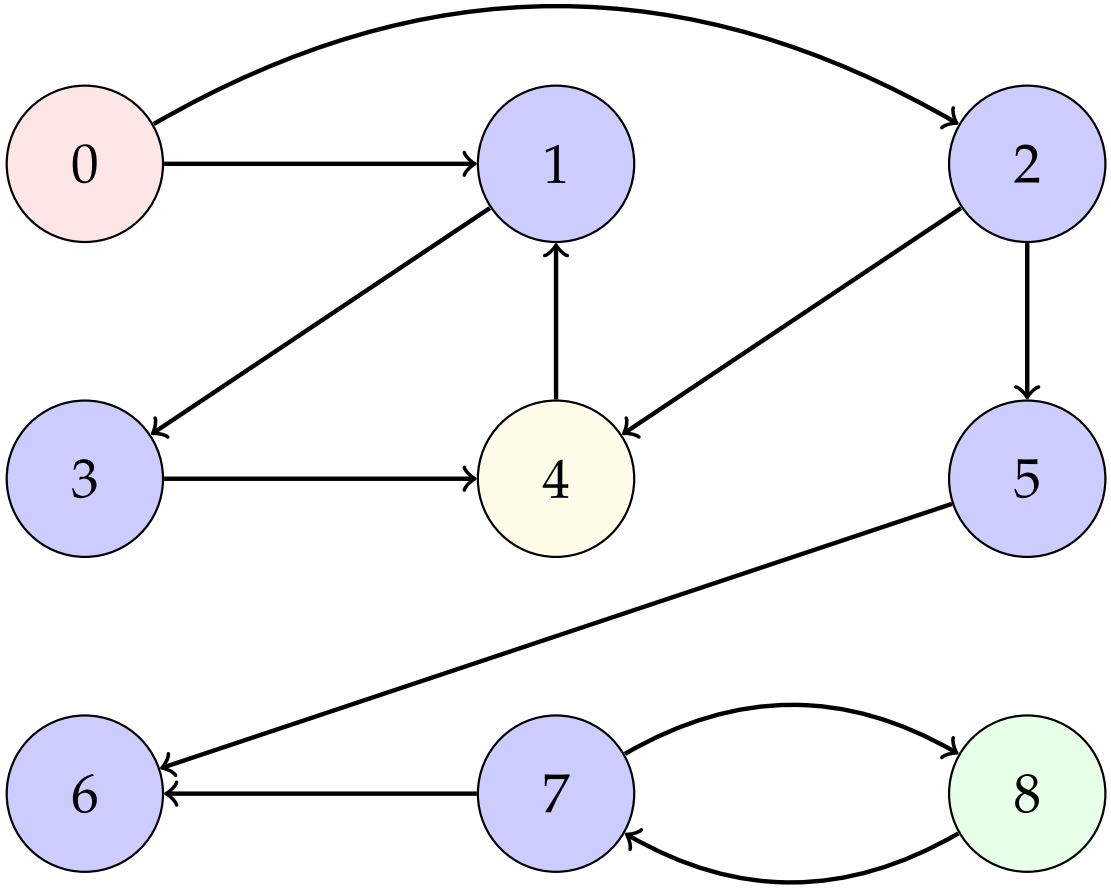
Graph Traversal: Naïve



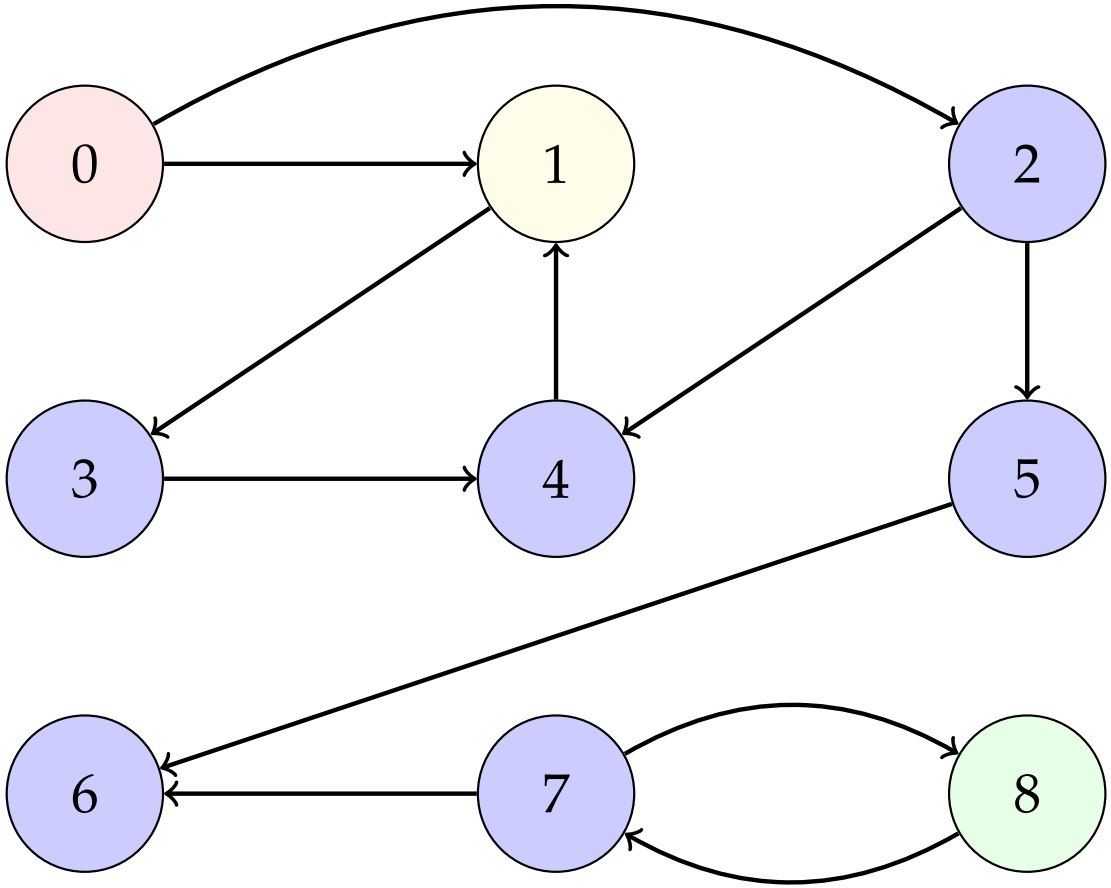
Graph Traversal: Naïve



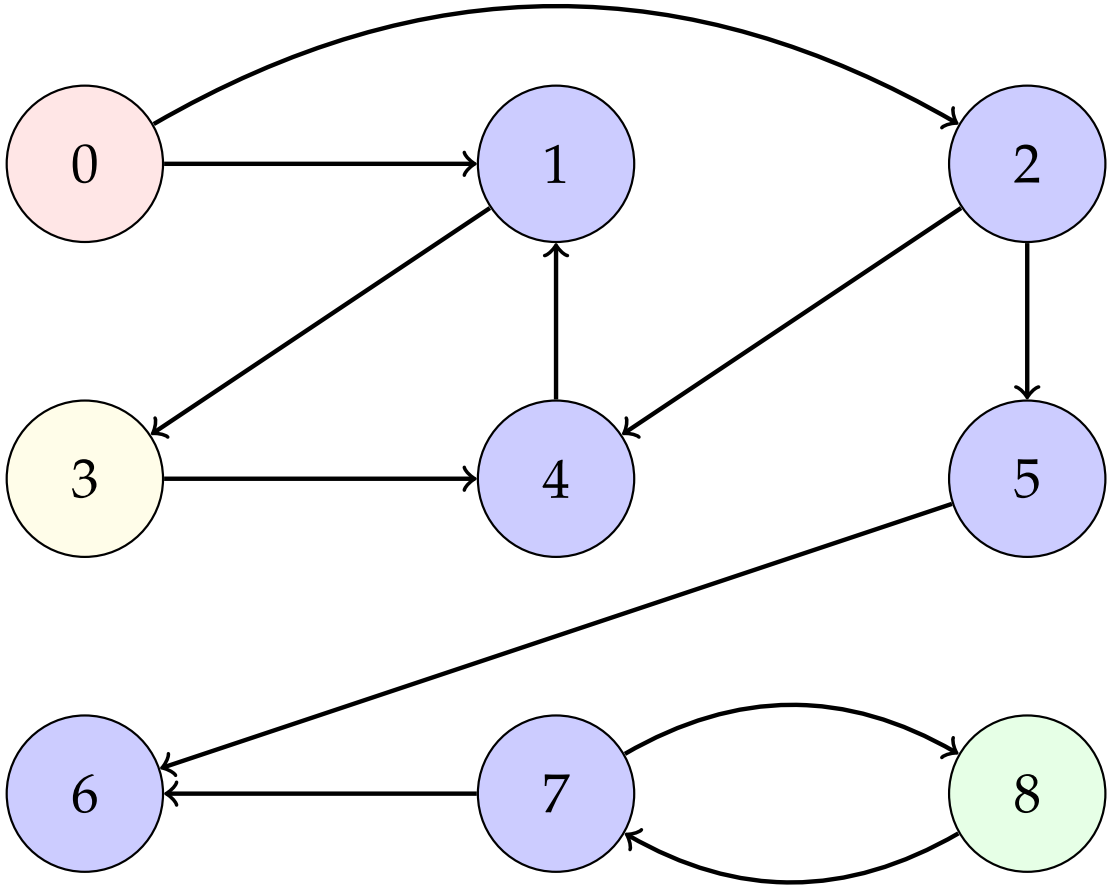
Graph Traversal: Naïve



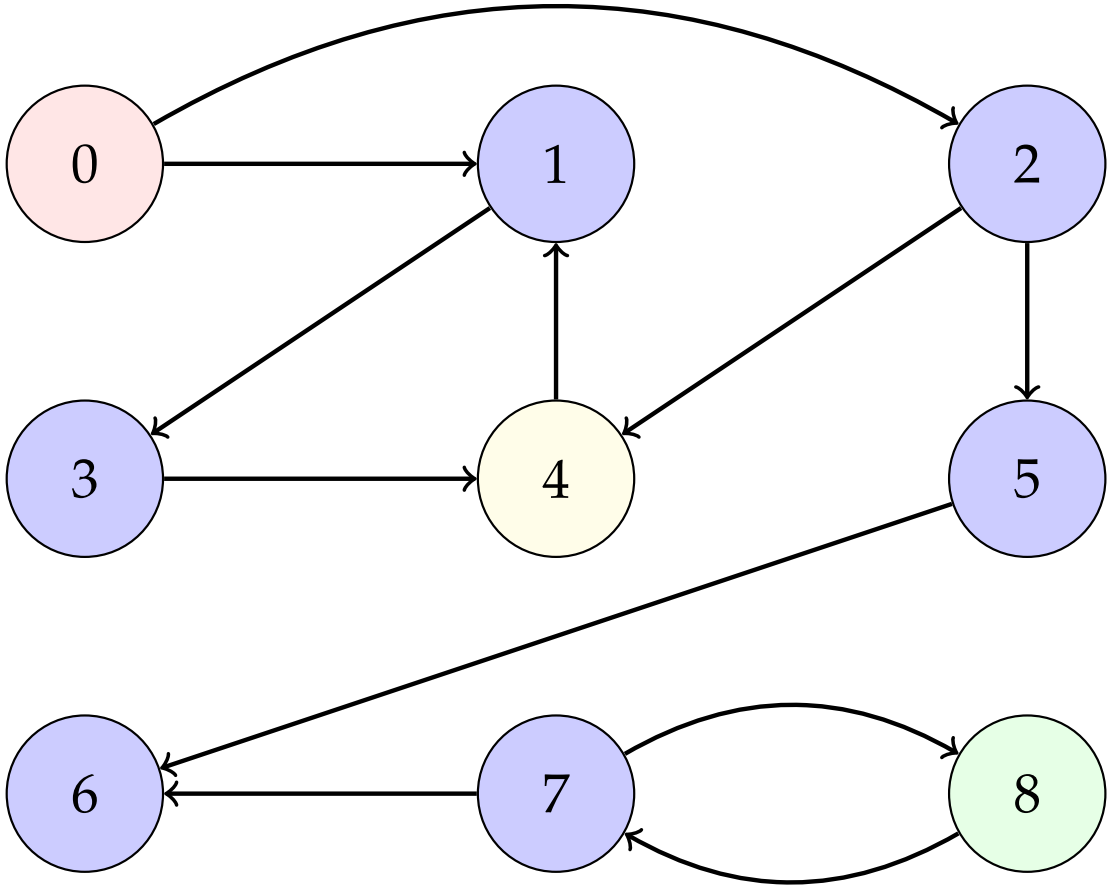
Graph Traversal: Naïve



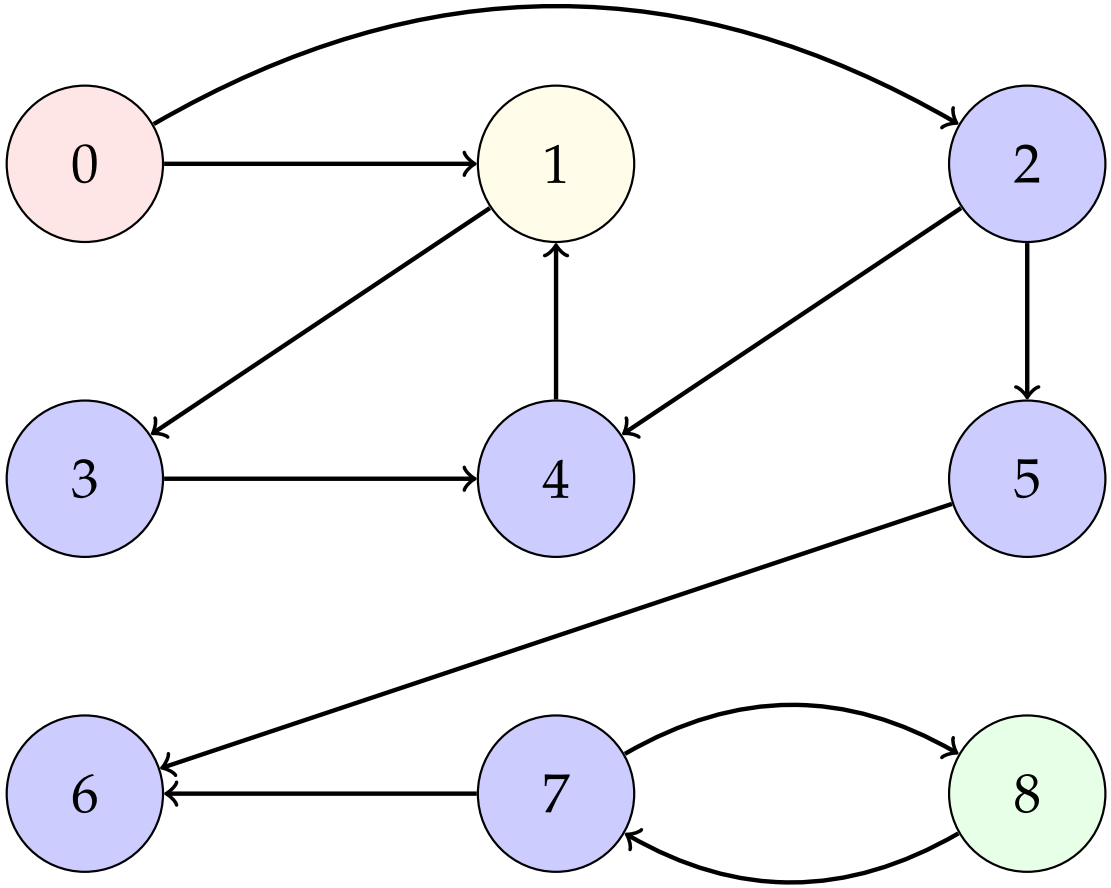
Graph Traversal: Naïve



Graph Traversal: Naïve



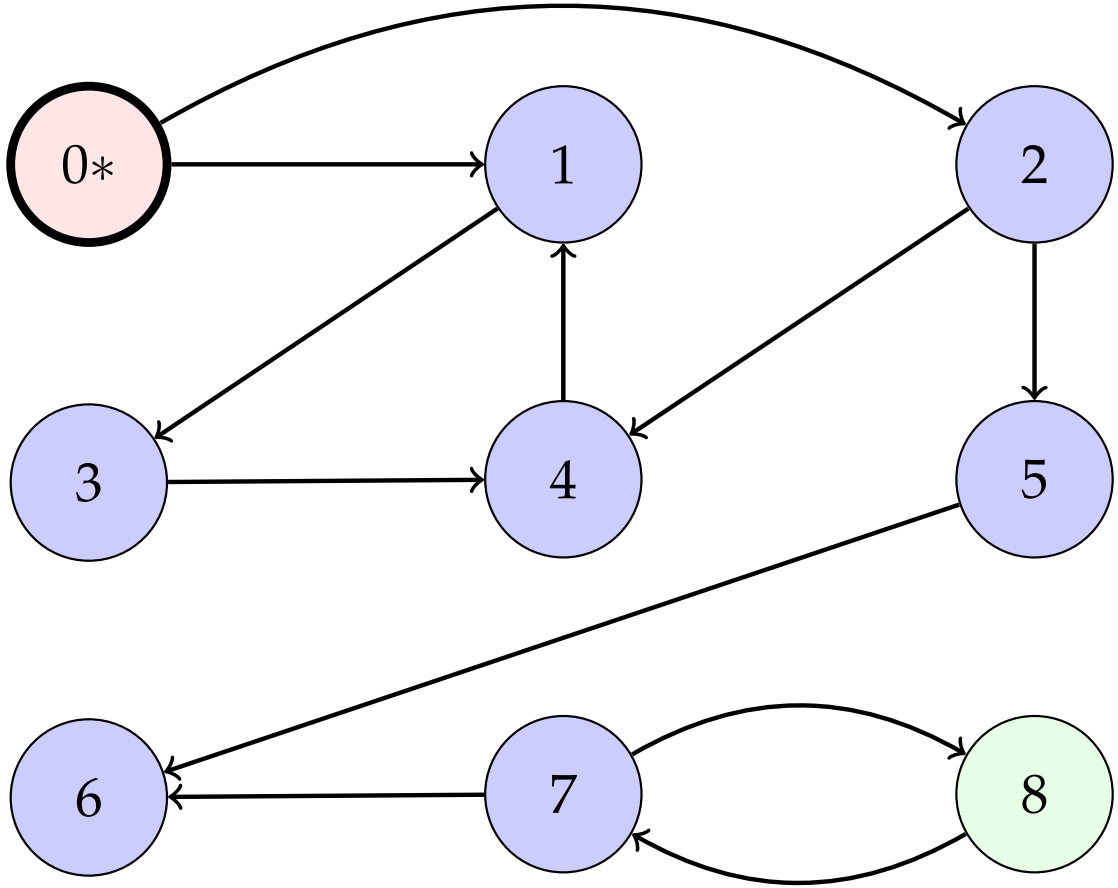
Graph Traversal: Naïve



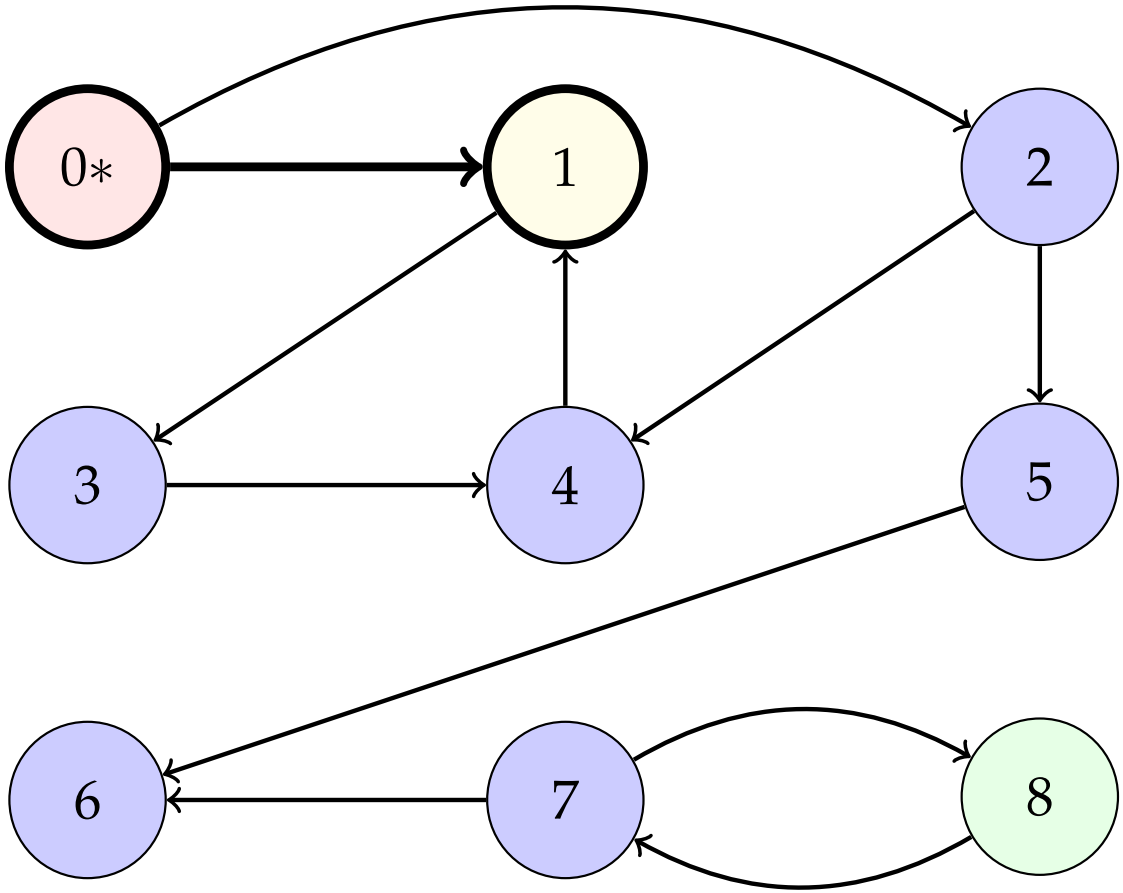
Graph Traversal: Depth-first search (DFS)

Depth-first Traversal: *Explore as far as possible along a branch before backtracking and trying another.*

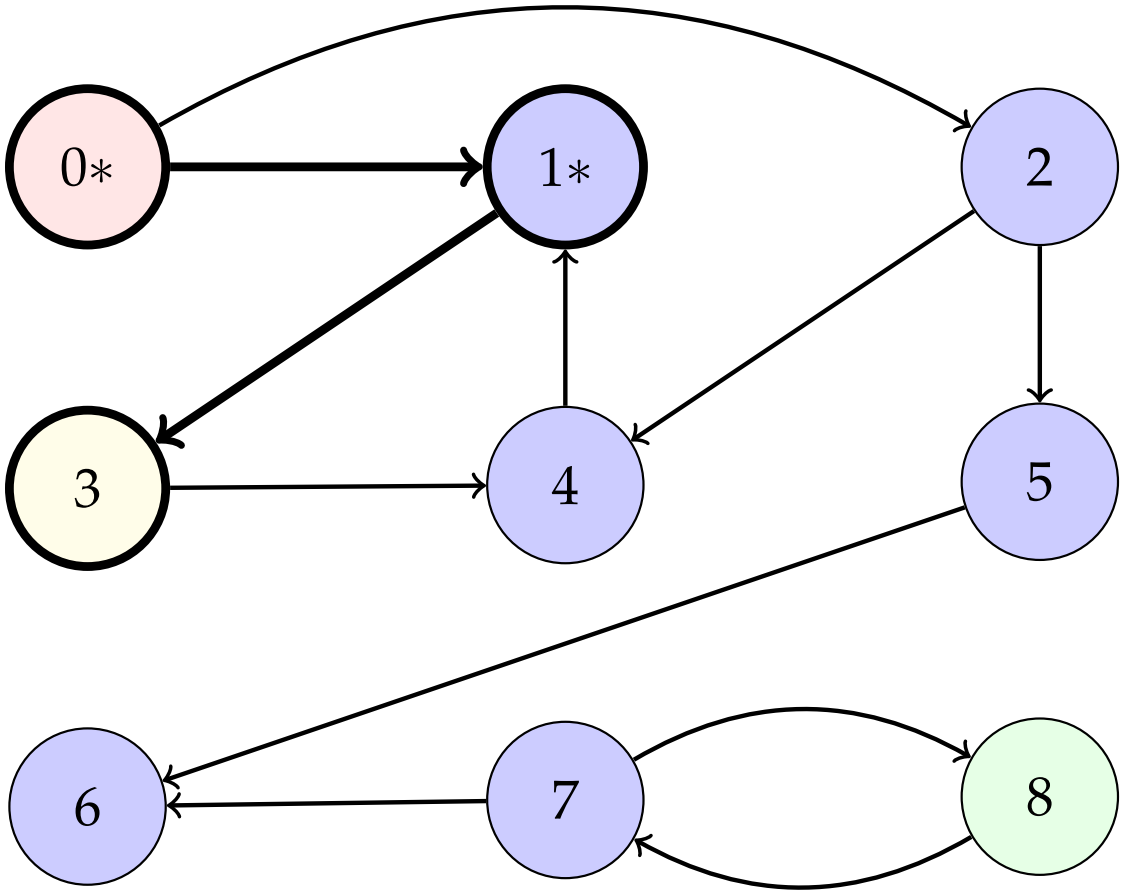
Graph Traversal: Depth-first search (DFS)



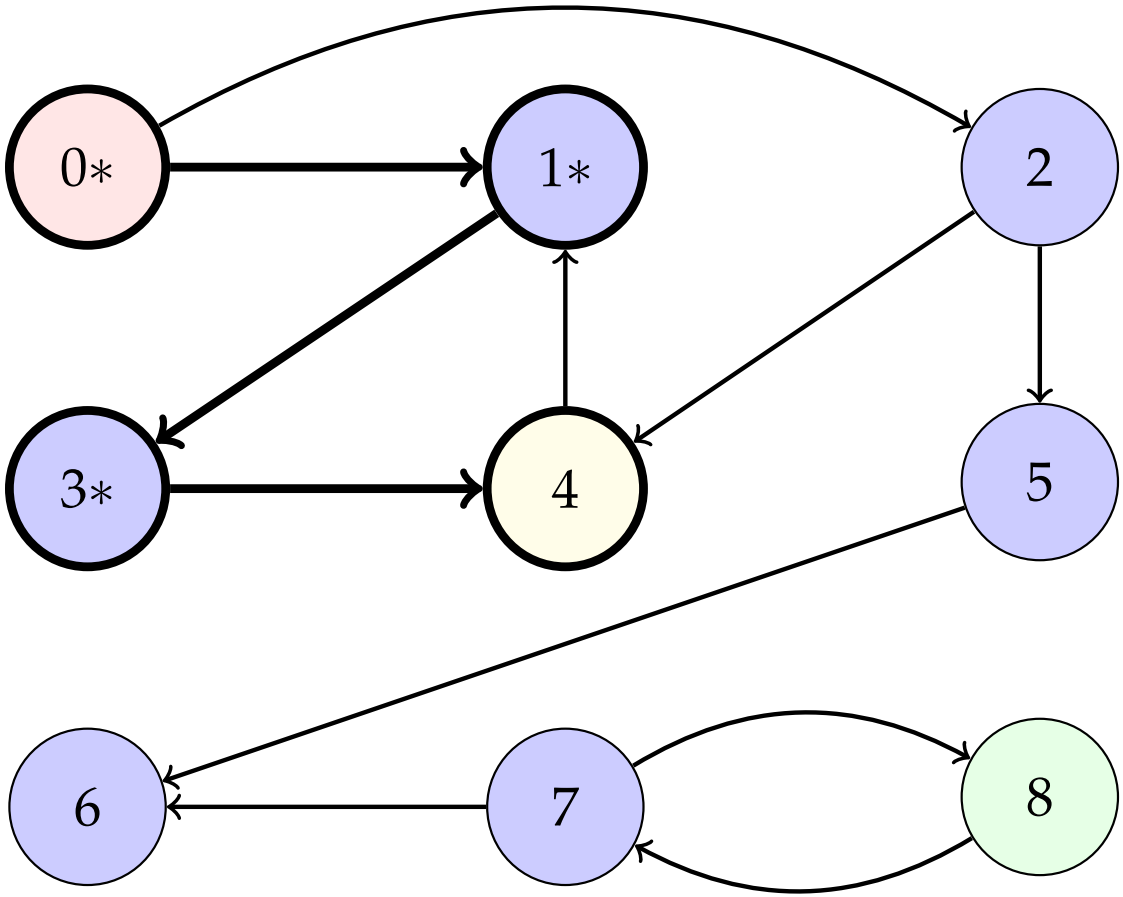
Graph Traversal: Depth-first search (DFS)



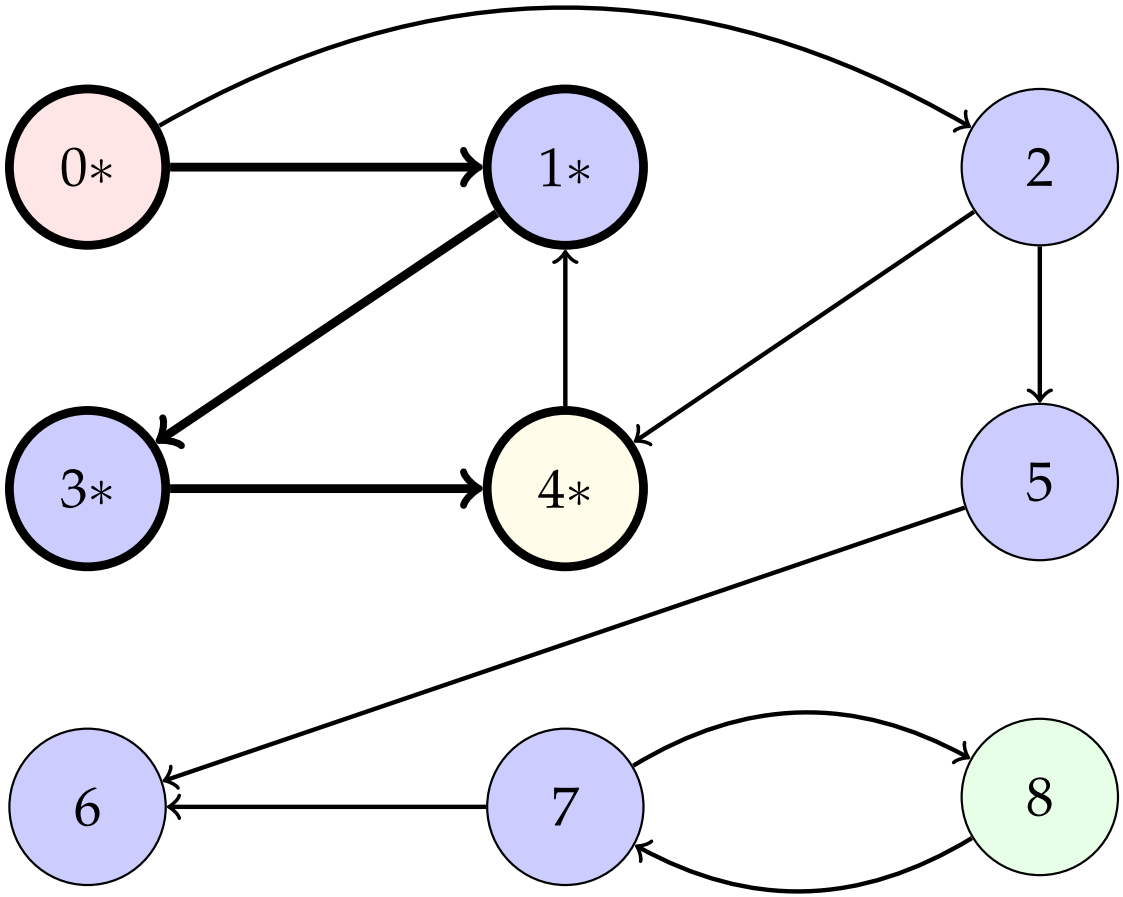
Graph Traversal: Depth-first search (DFS)



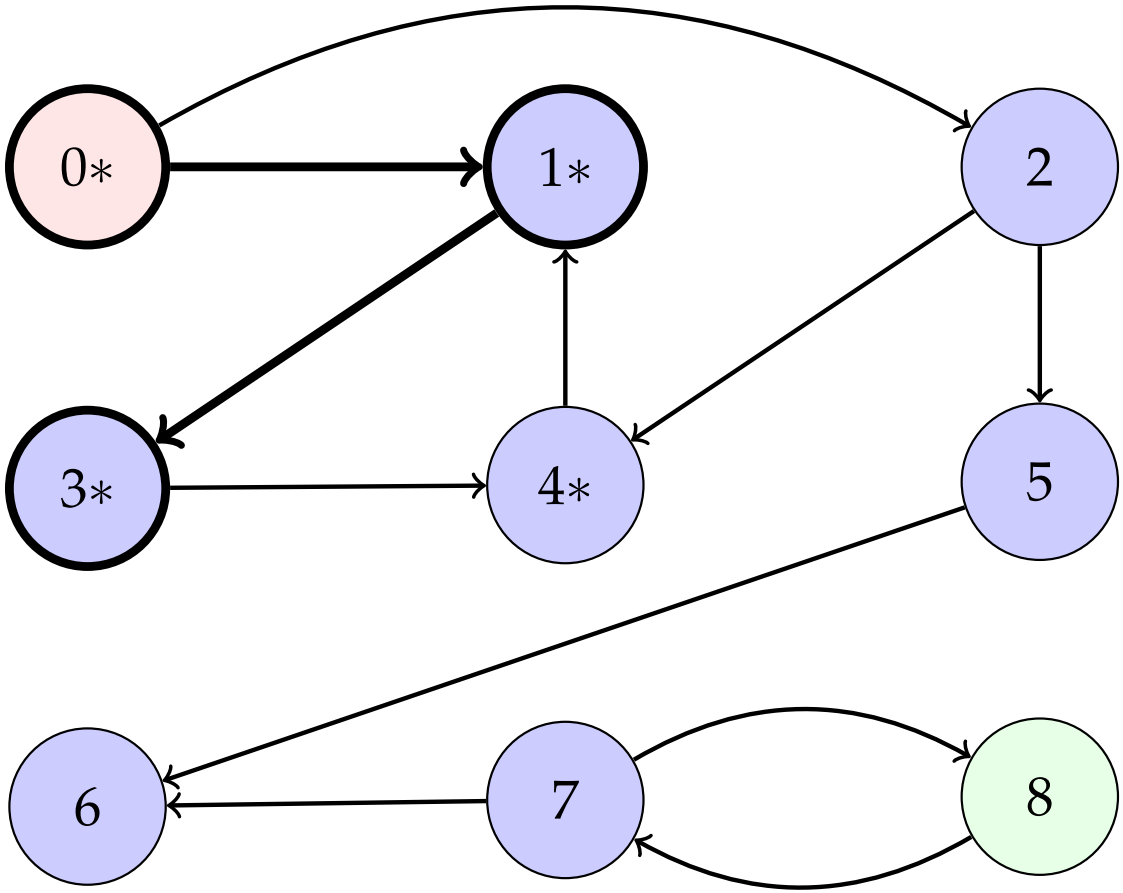
Graph Traversal: Depth-first search (DFS)



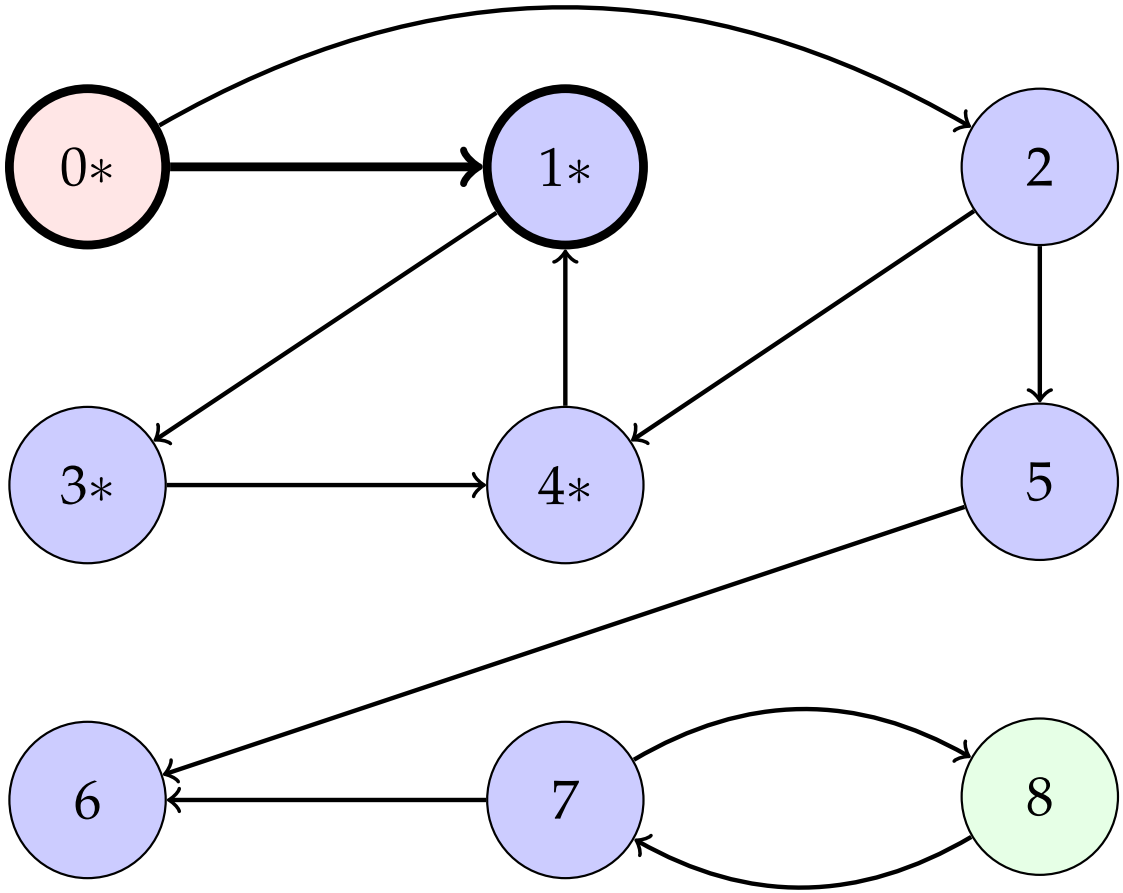
Graph Traversal: Depth-first search (DFS)



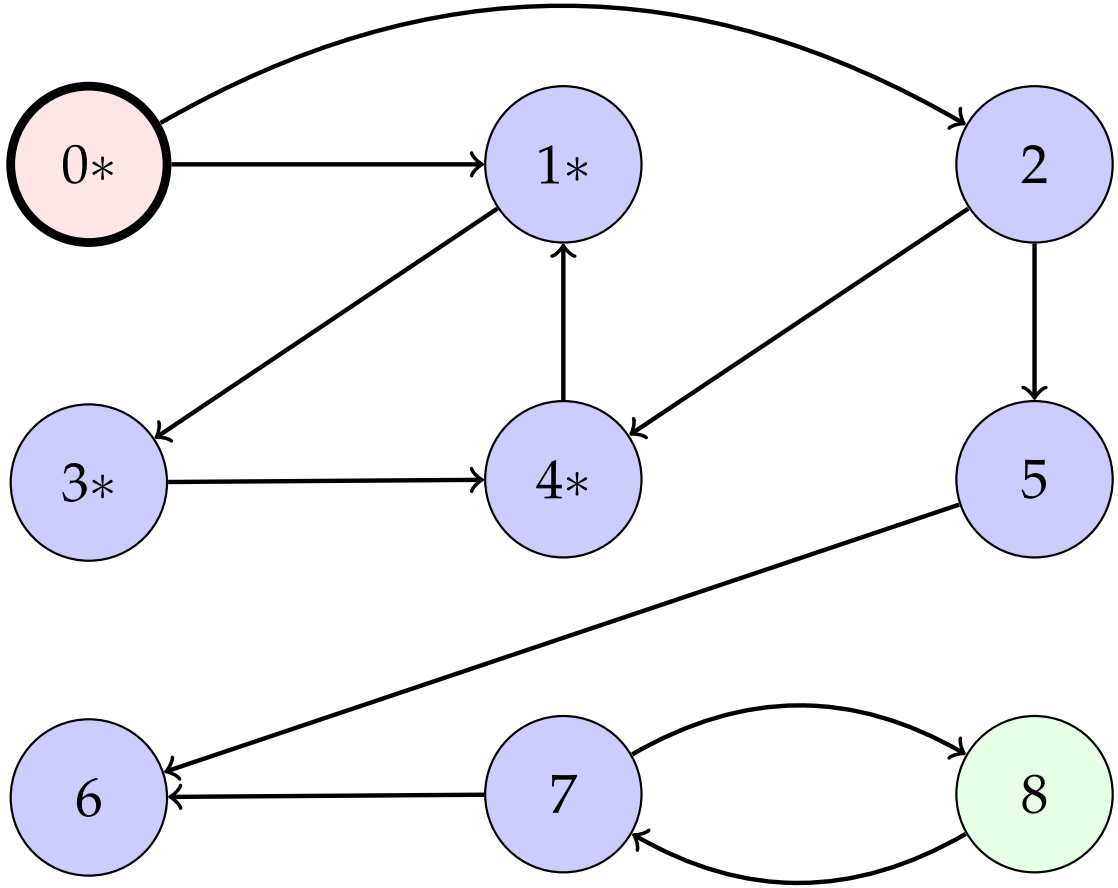
Graph Traversal: Depth-first search (DFS)



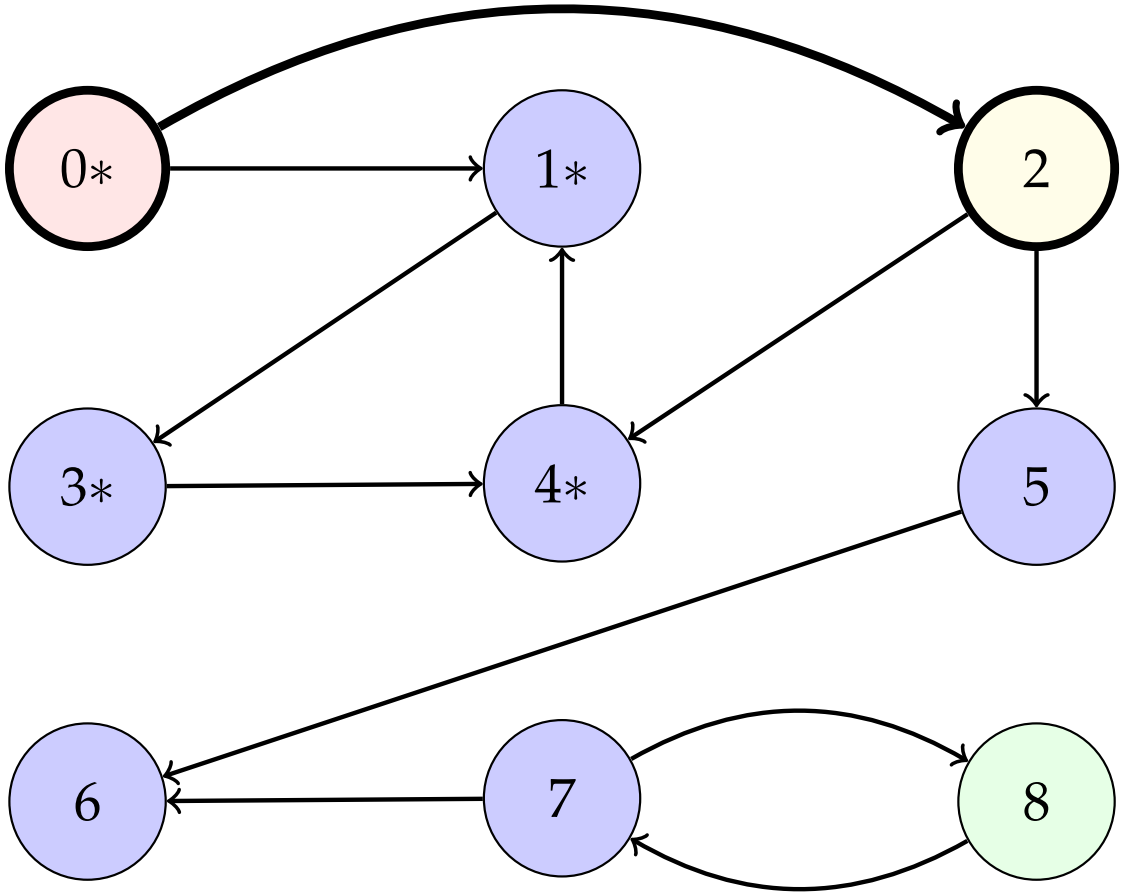
Graph Traversal: Depth-first search (DFS)



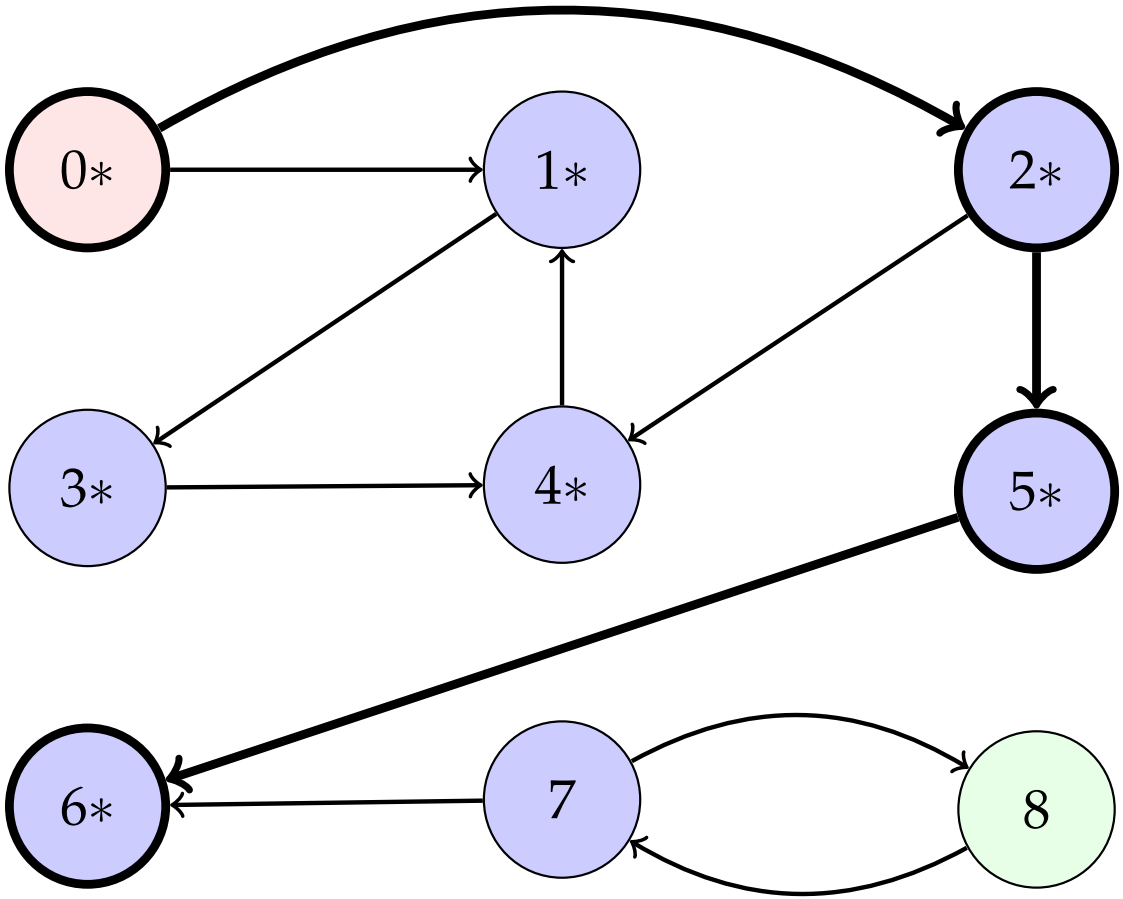
Graph Traversal: Depth-first search (DFS)



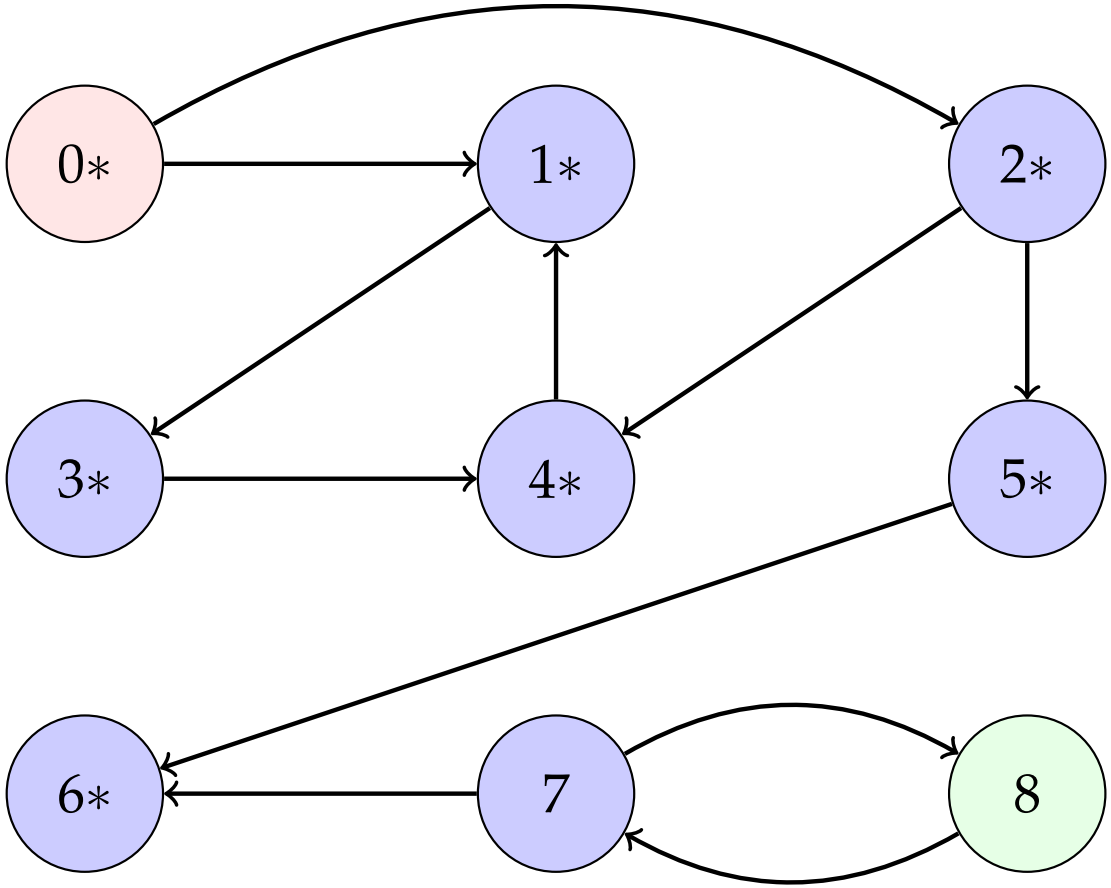
Graph Traversal: Depth-first search (DFS)



Graph Traversal: Depth-first search (DFS)



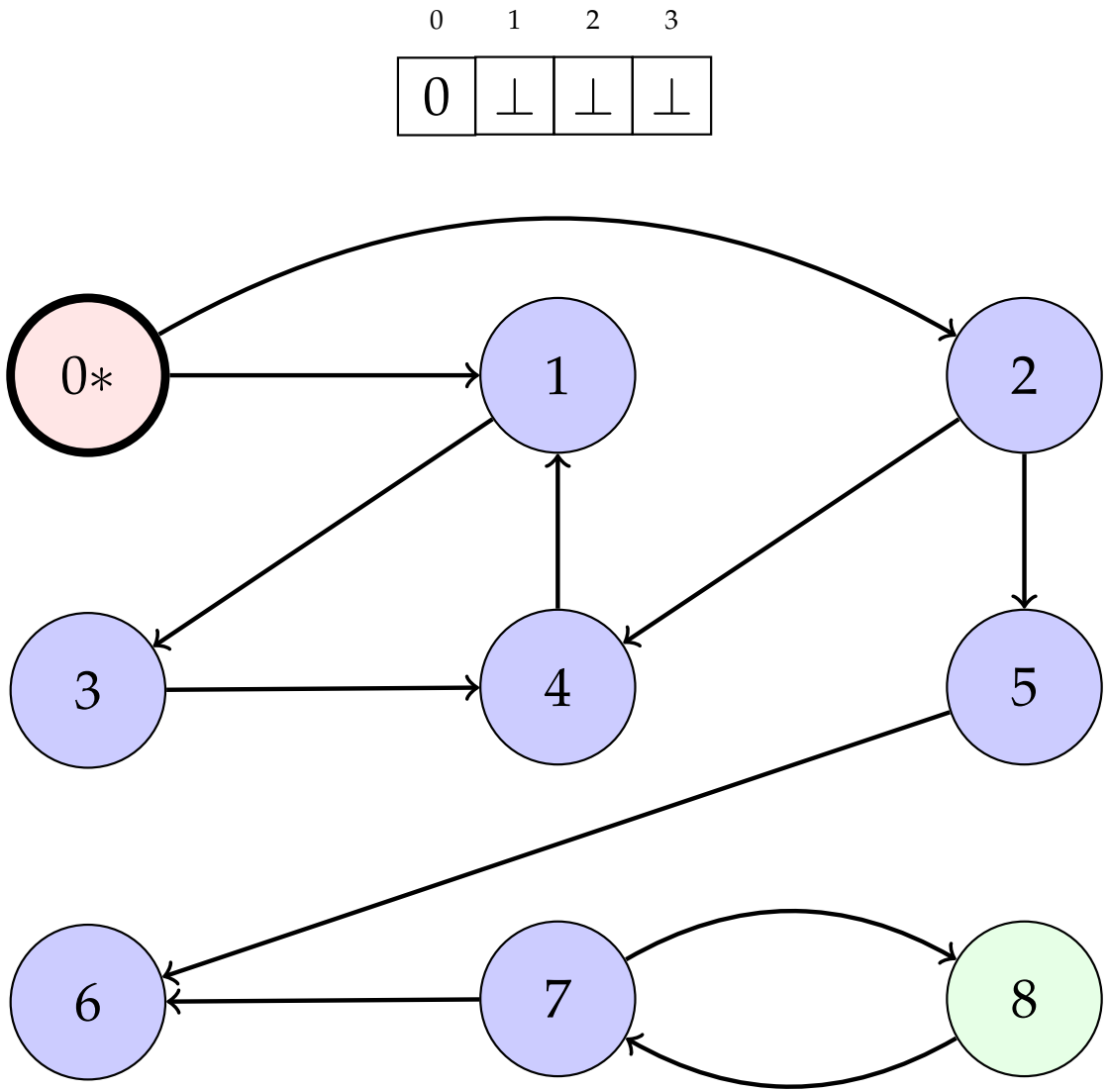
Graph Traversal: Depth-first search (DFS)



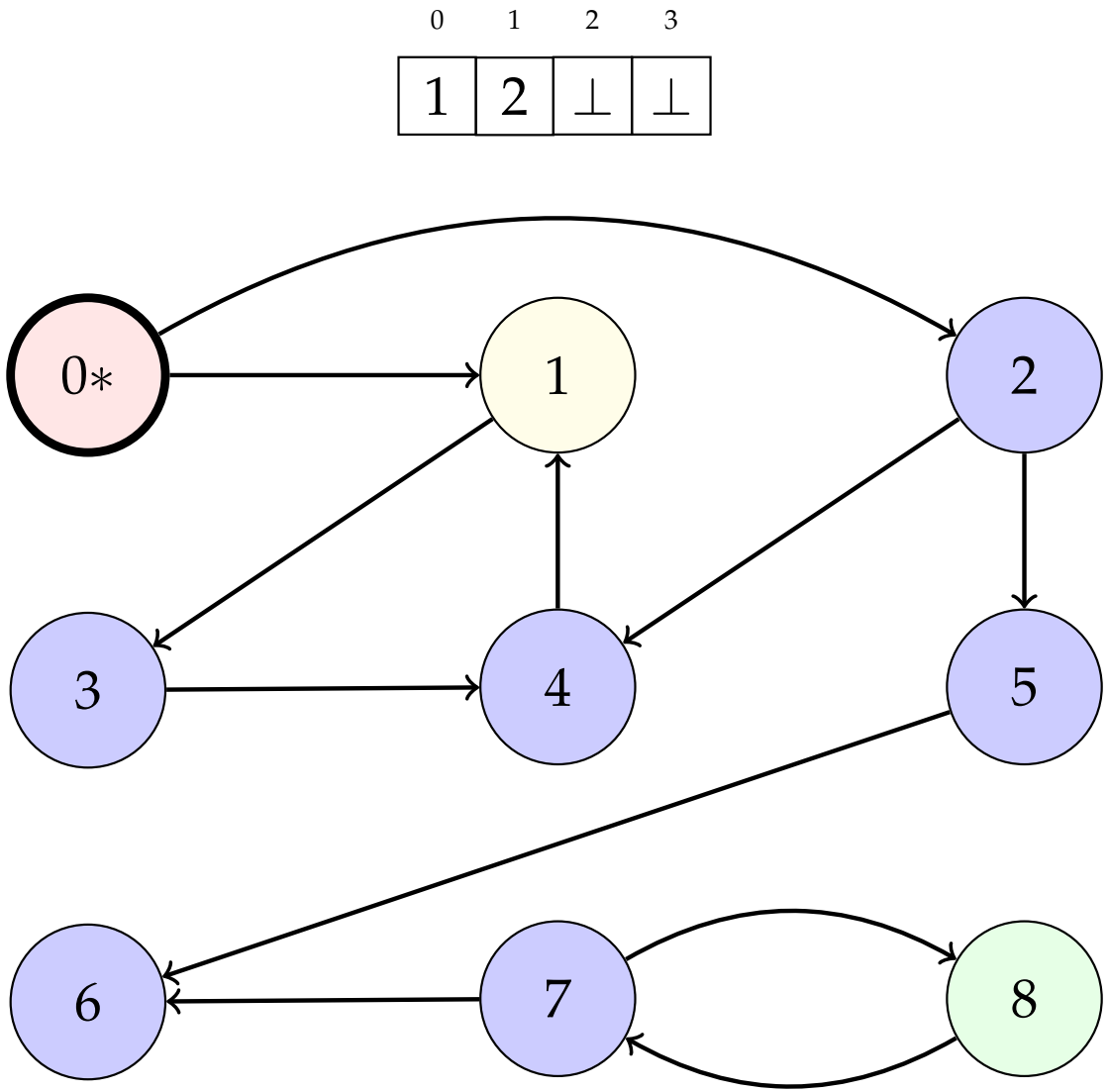
Graph Traversal: Breadth-first search (BFS)

Breadth-first Traversal: *Explore all nodes at a level (distance form the root) before visiting nodes at a greater level.*

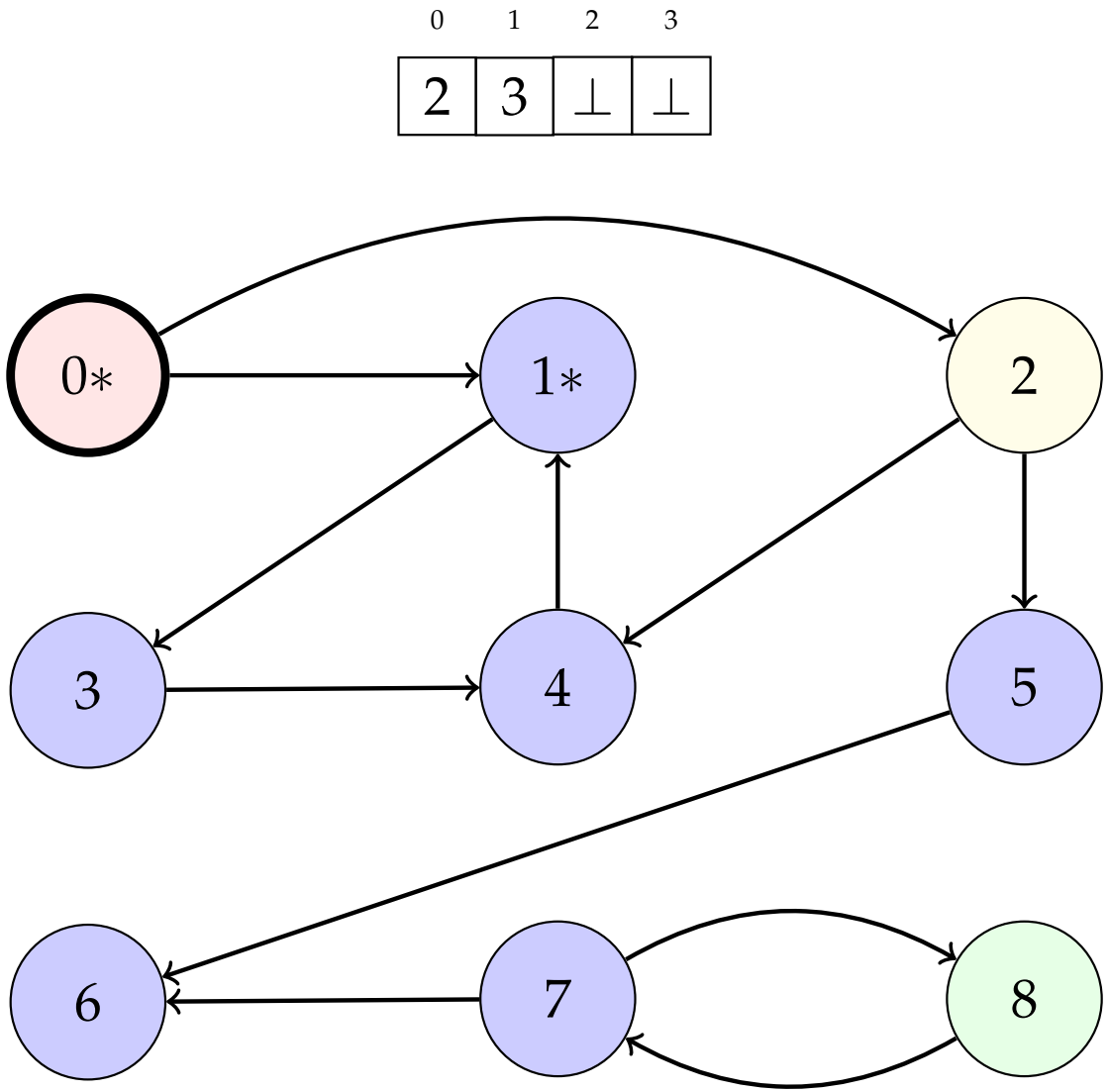
Graph Traversal: Breadth-first search (DFS)



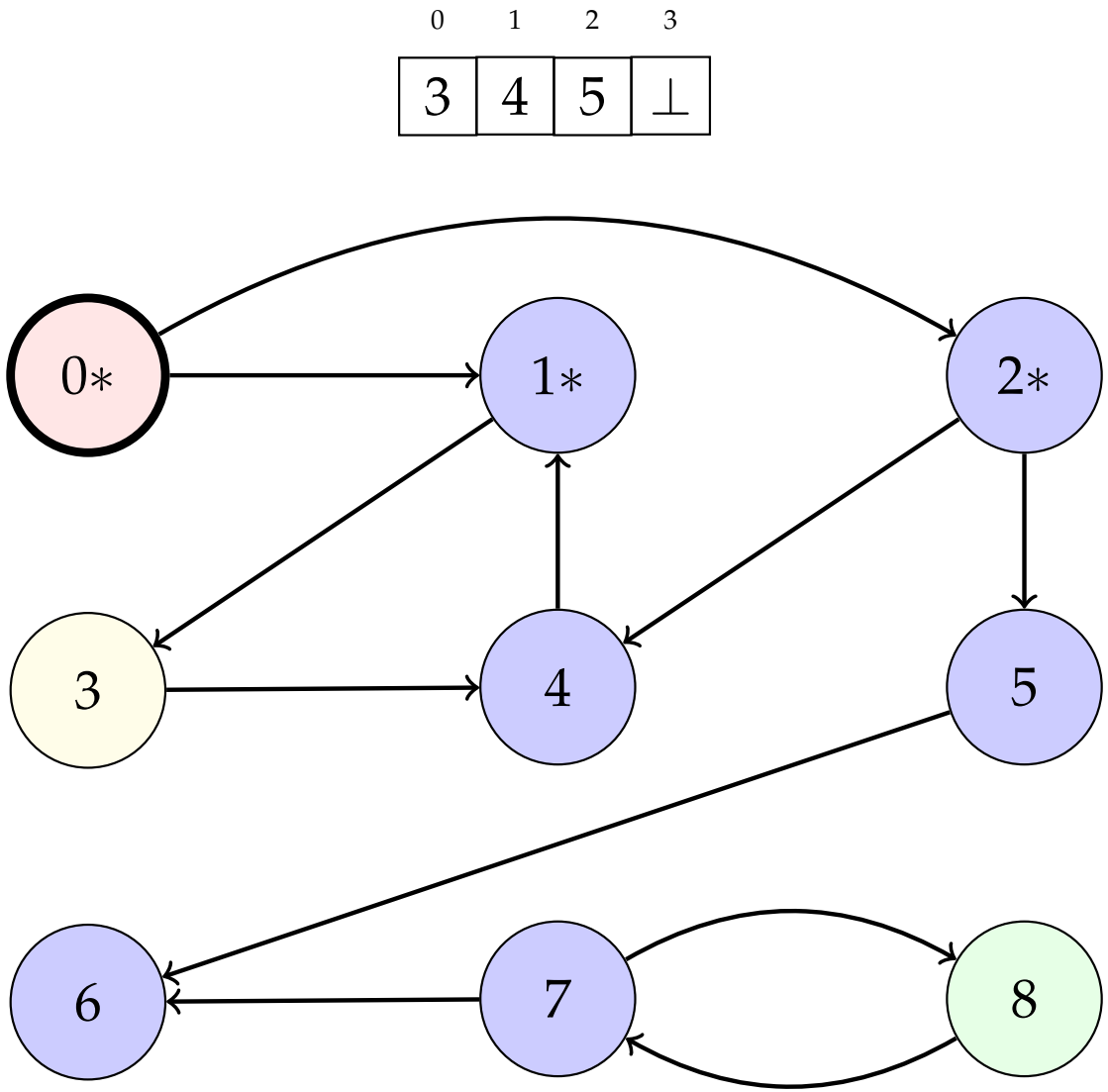
Graph Traversal: Breadth-first search (DFS)



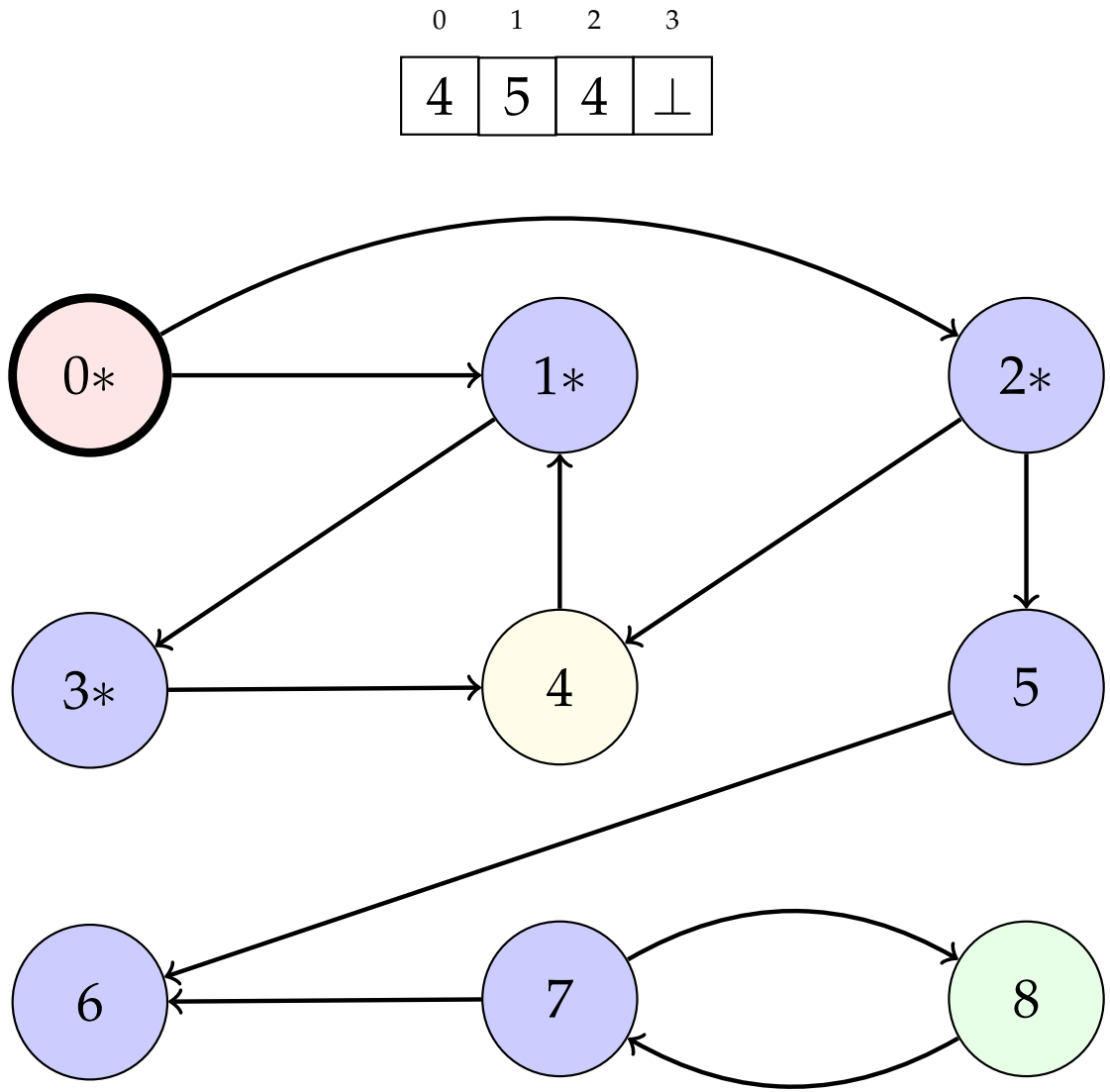
Graph Traversal: Breadth-first search (DFS)



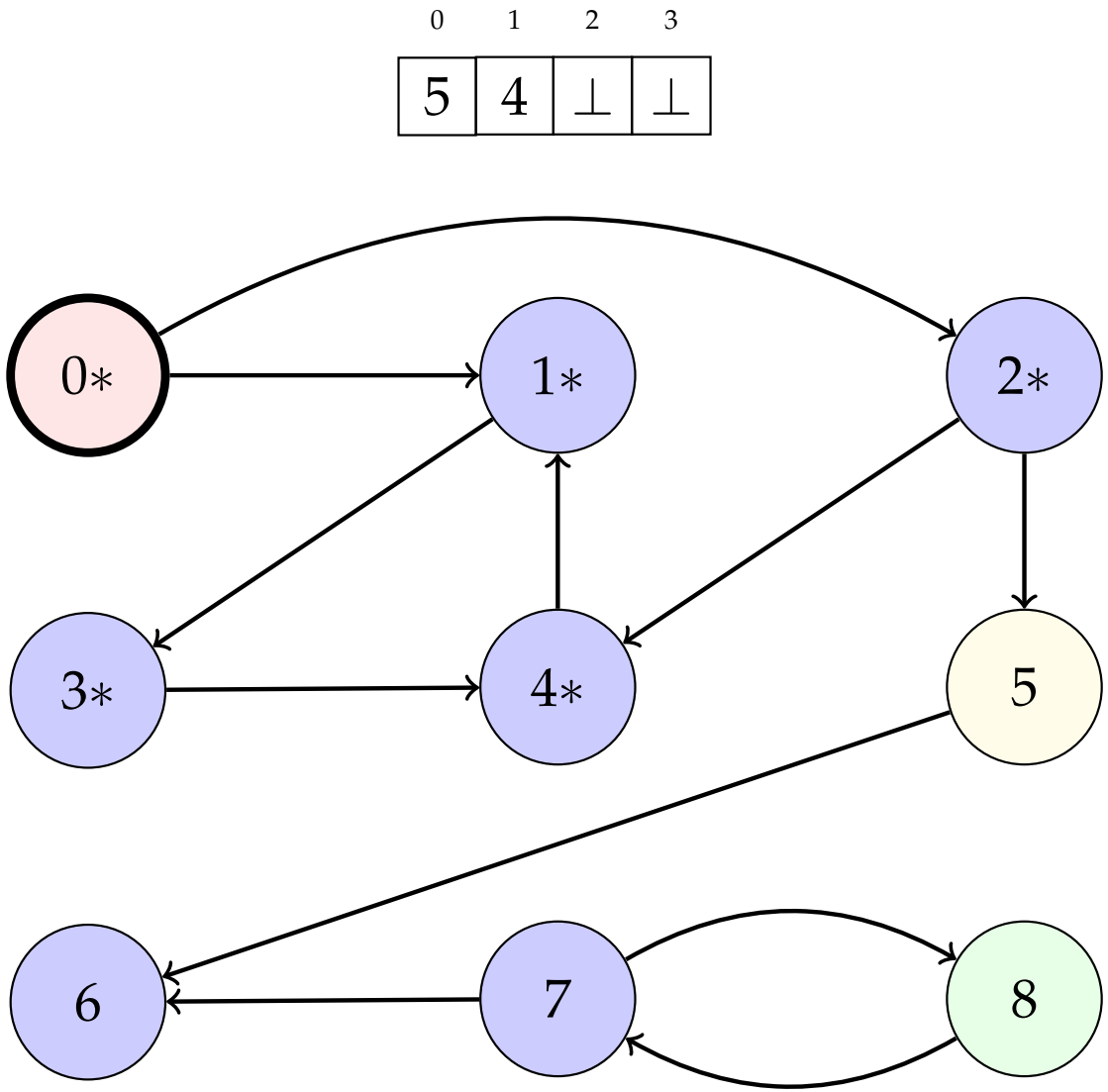
Graph Traversal: Breadth-first search (DFS)



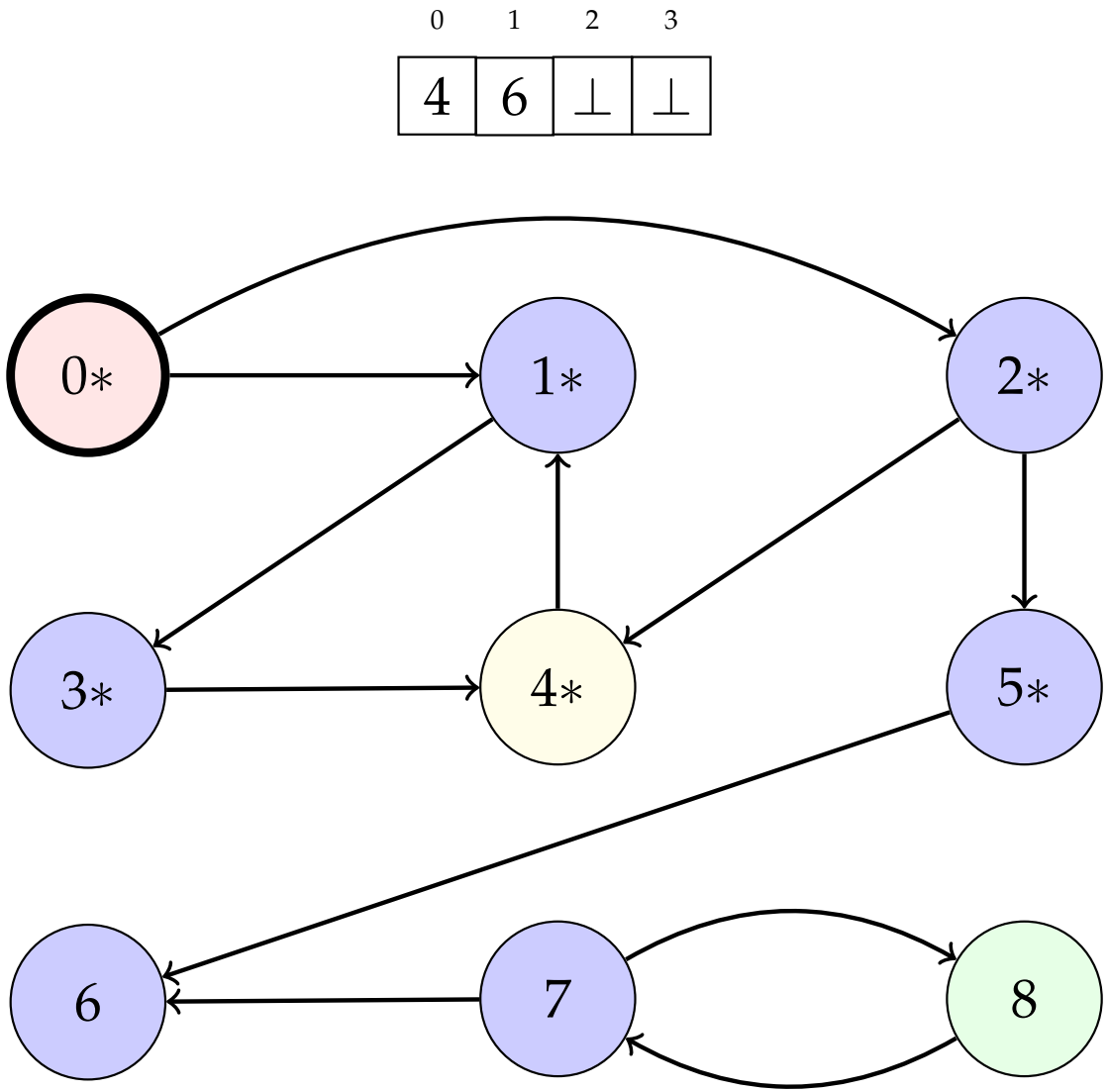
Graph Traversal: Breadth-first search (DFS)



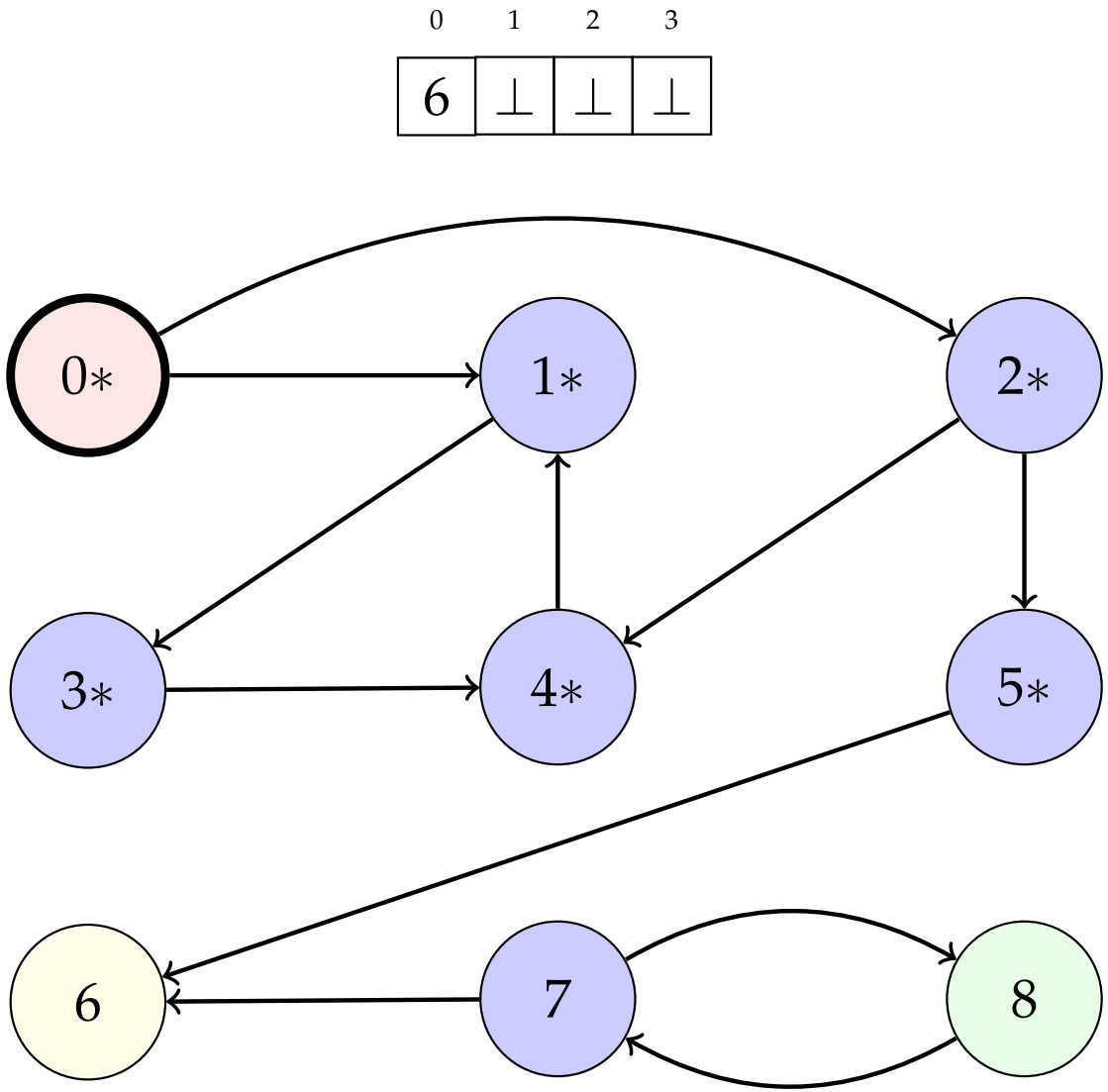
Graph Traversal: Breadth-first search (DFS)



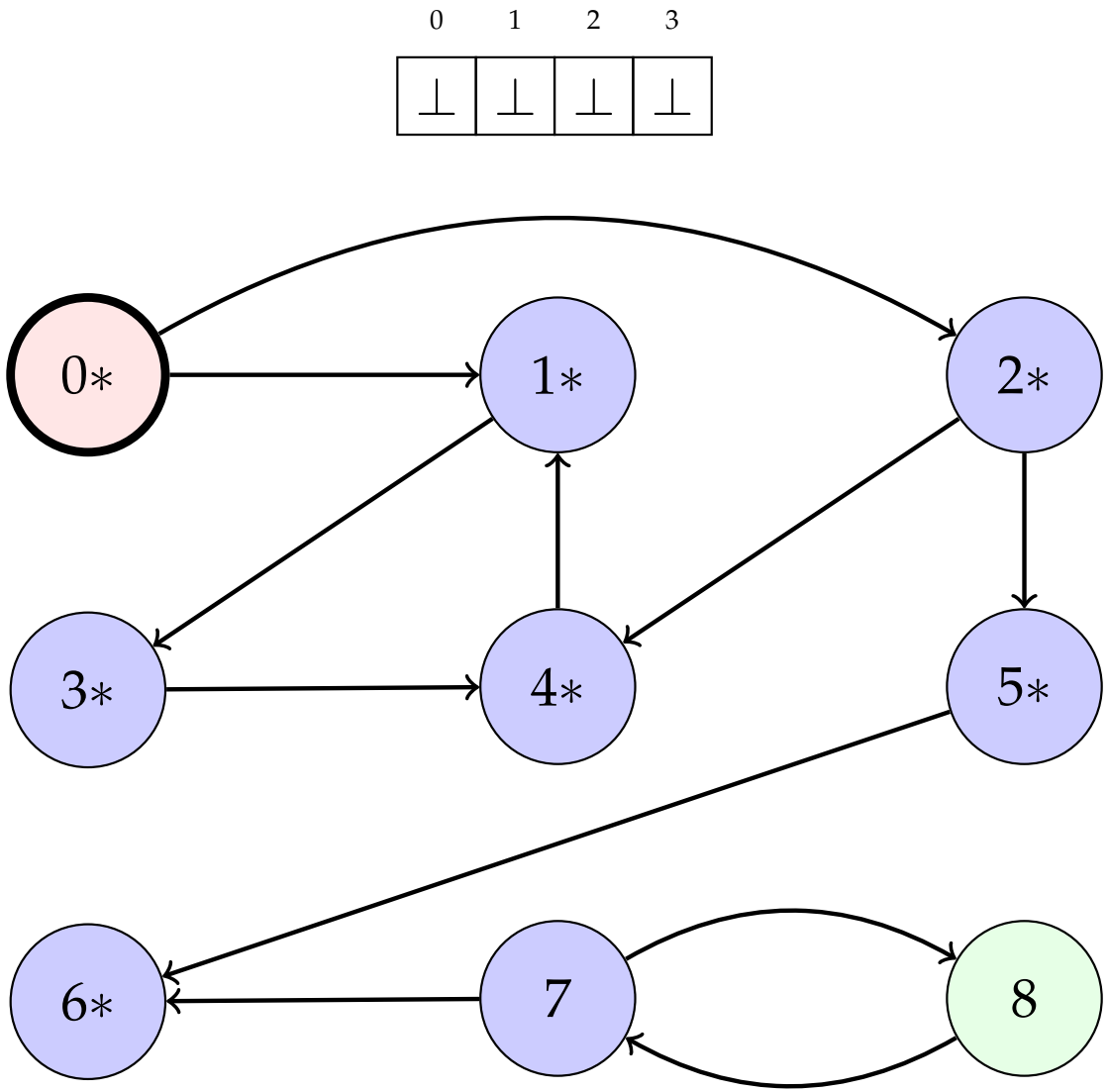
Graph Traversal: Breadth-first search (DFS)



Graph Traversal: Breadth-first search (DFS)



Graph Traversal: Breadth-first search (DFS)



Graph Traversal: BFS Vs DFS

DFS is BFS with a stack instead of a queue!

Graphs

Graph Traversal

C++ Standard Template Library: Containers

C++ Containers

- The containers library is a collection of templates and algorithms that implement the common data structures that we work with as programmers
- A container is an object that stores a collection of elements.
- Containers implement commonly used data-structures:
  - static arrays: *array*
  - dynamic arrays : *vector*
  - singly-linked list: *forward_list*
  - linked lists: *list*
  - queues : *queue*
  - double-ended queue: *dequeue*
  - stacks : *stack*
  - heaps : *priority_queue*
  - binary search trees : *set*
  - dictionary or associative arrays: *map*

Reasons to Use Standard Containers

- STL containers are implemented correctly.
- STL containers are fast and efficient.
- STL containers share common interfaces.
- STL containers are well-documented and easily understood by other developer.

Vectors

- C++ STL implementation of dynamic arrays (with doubling)
- Vector elements are placed in contiguous storage, so that they can be accessed and traversed using iterators.
- Key functions:
  - push_back — to push an element to the last free index of the array.
  - size() — returns the number of elements in the vector.
  - capacity() — returns the size of the storage space currently allocated to the vector expressed as number of elements.
  - reserve() — requests that the vector capacity be at least enough to contain n elements.
  - pop_back() — It is used to pop or remove elements from a vector from the back.

Queues

- C++ STL implementation of queues (FIFO)
- Key functions:
  - empty() — Returns whether the queue is empty.
  - size() — Returns the size of the queue.
  - queue::front() and queue::back() — front() function returns a reference to the first element of the queue and back() function returns a reference to the last element of the queue.
  - push(g) and pop() — push() function adds the element 'g' at the end of the queue. pop() function deletes the first element of the queue.
  -
- Further reading: *http://www.cplusplus.com/reference/stl/*