Machine Learning with Graphs (MLG)

Search in Graphs

Cheng-Te Li (李政德)

Institute of Data Science

National Cheng Kung University

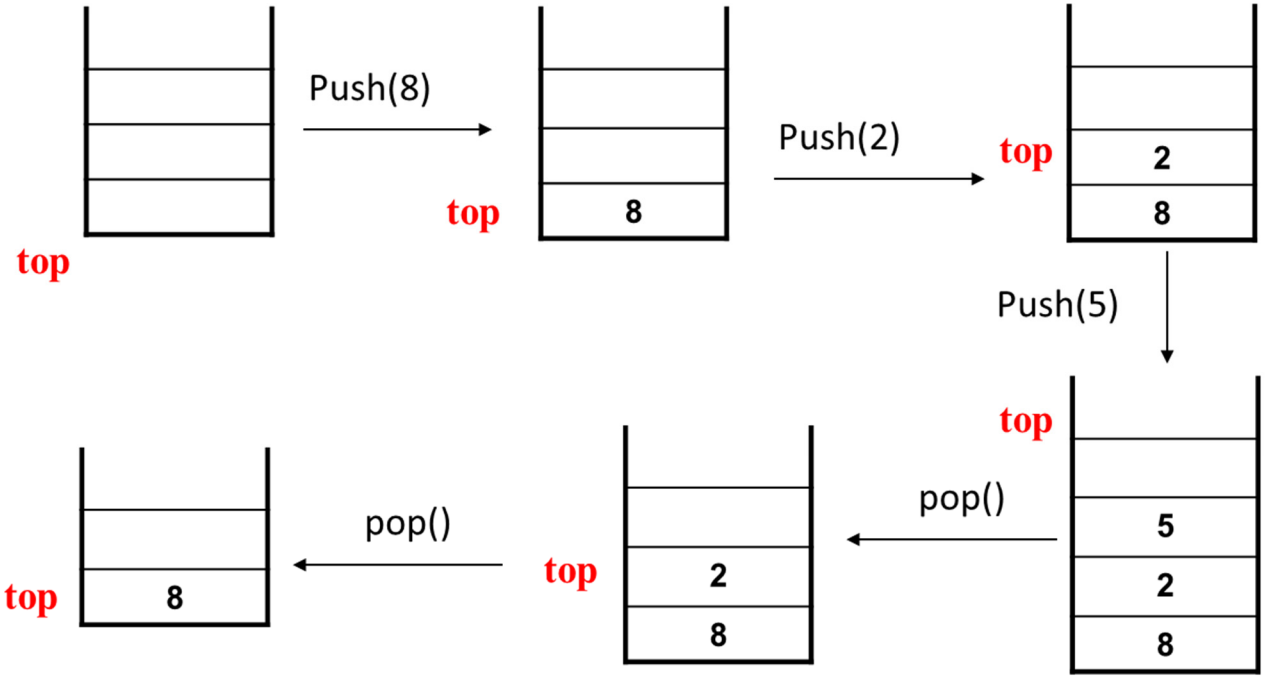
chengte@mail.ncku.edu.tw

Graph Traversals

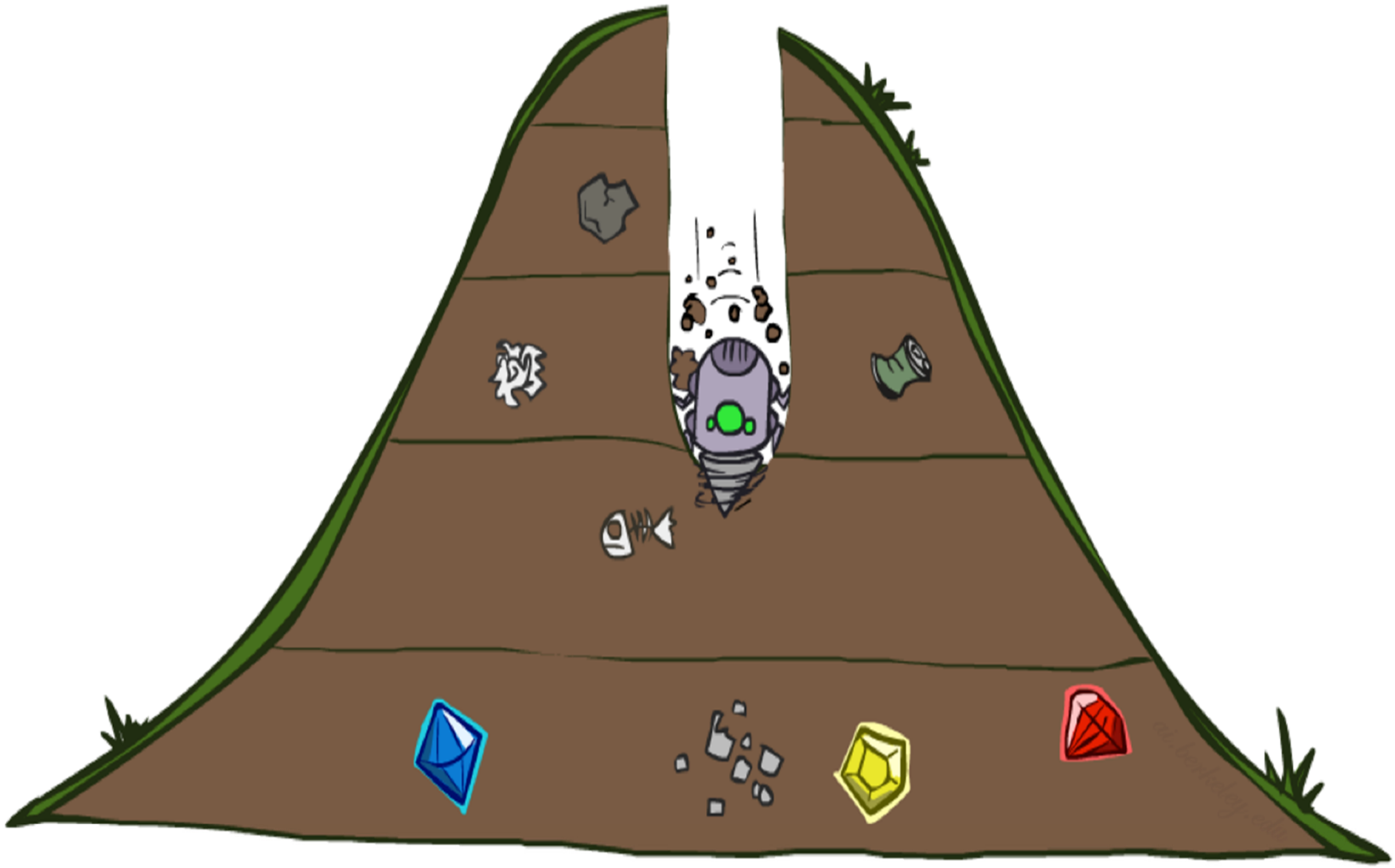
- We are interested in surveying a social network to computing the average age of its users
  - Start from one user
  - Employ some traversal technique to reach her friends and then friends' friends, …
- A traversal needs to guarantee that
  1) All users are visited; and
  2) No user is visited more than once
- Two main traversal techniques
  - Depth First Search (DFS) traversal
  - Breath First Search (BFS) traversal

DFS Traversal

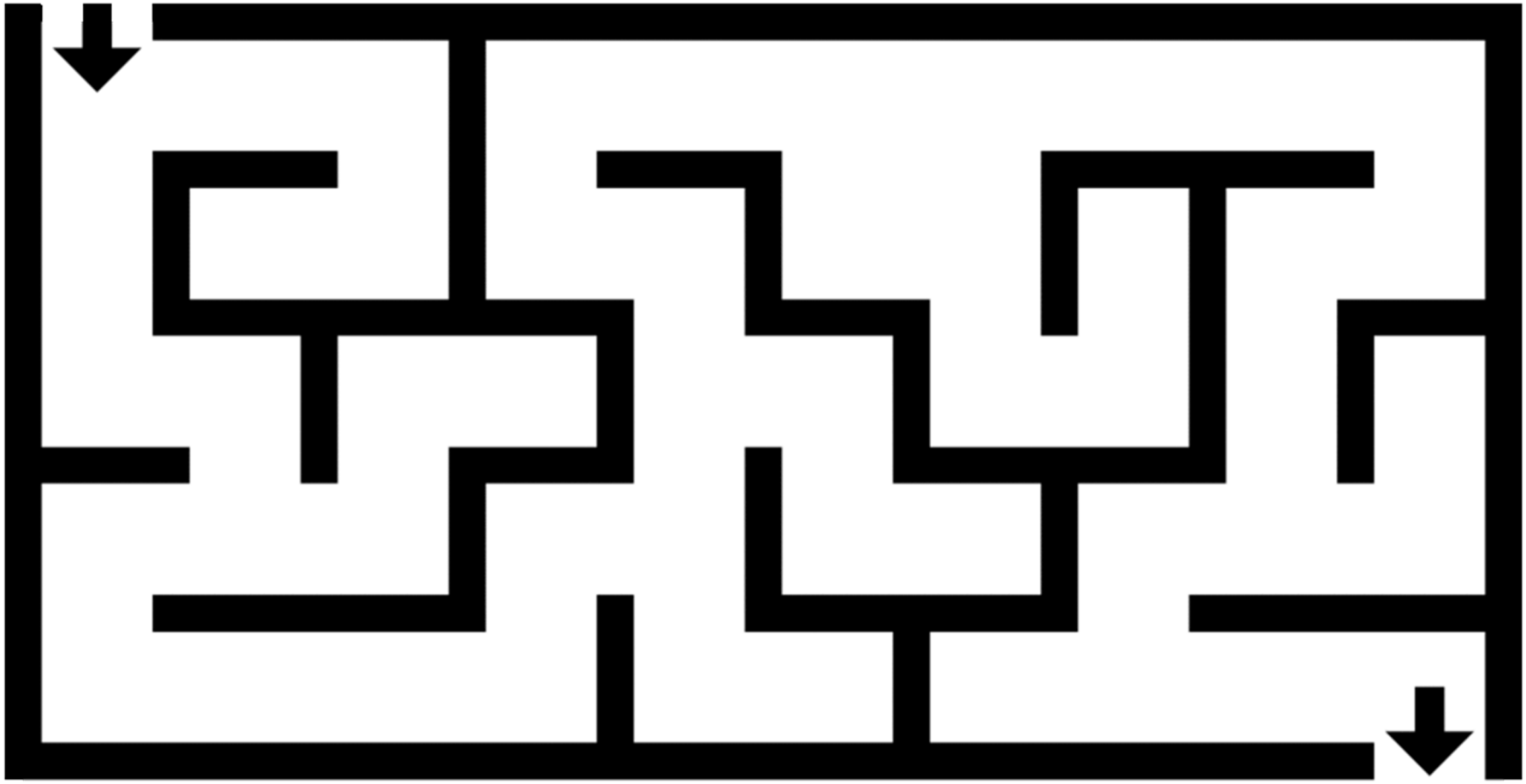
- Depth-First Search (DFS) starts from a node v_i, selects one of its neighbors v_j from $N(v_i)$ and performs Depth-First Search on v_j before visiting other neighbors in $N(v_i)$
- DFS can be implemented using **stack**
  - Last-In-First-Out (LIFO) data structure



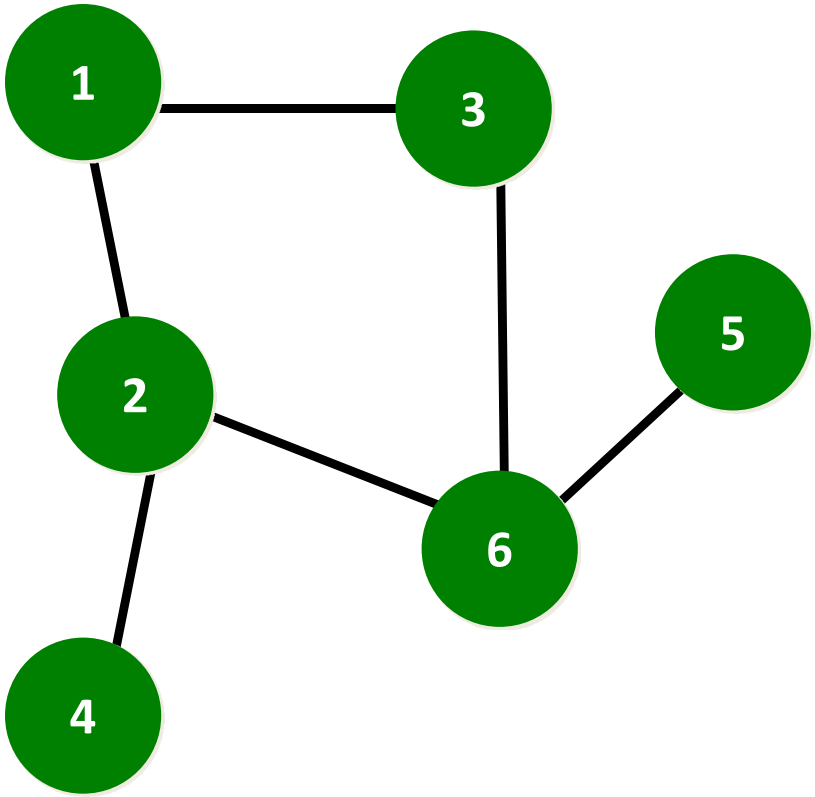
Depth First Search: Concept



Depth First Search: A Maze Example

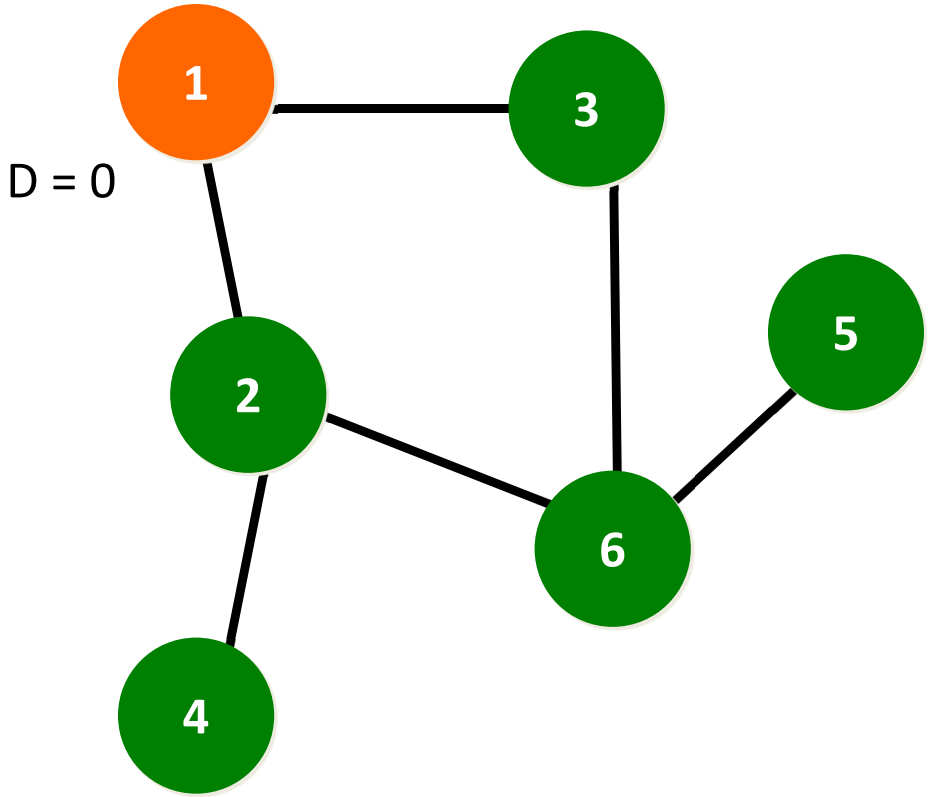


DFS Example (1/13)



DFS Example (2/13)

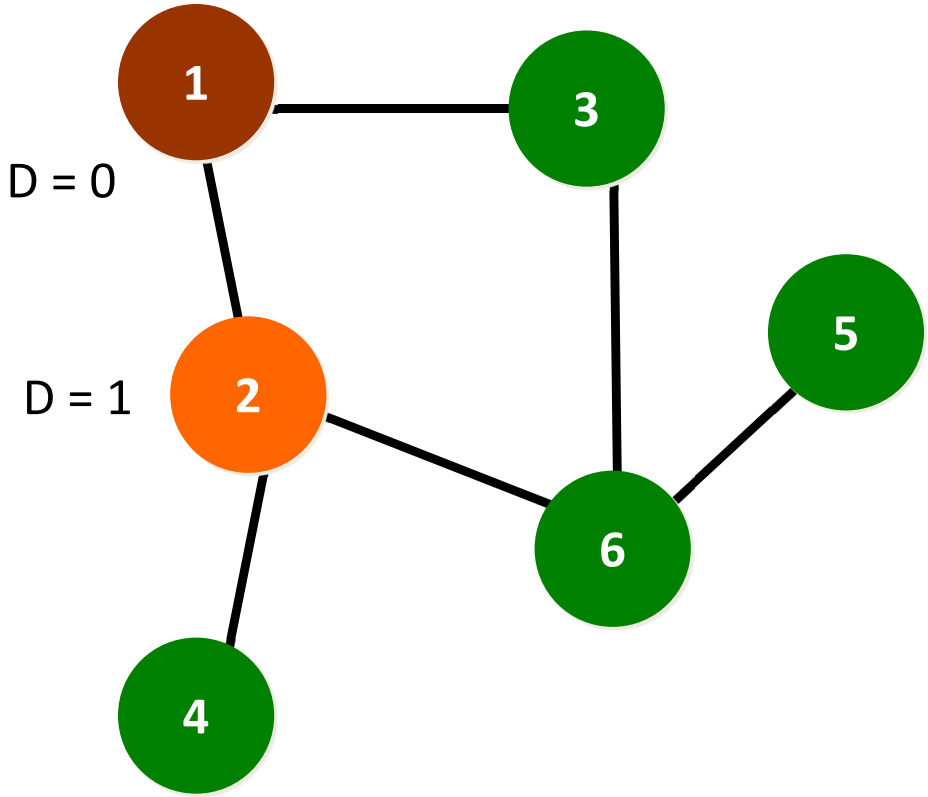
D: Depth

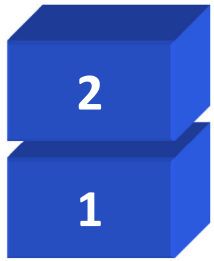


Stack

DFS Example (3/13)

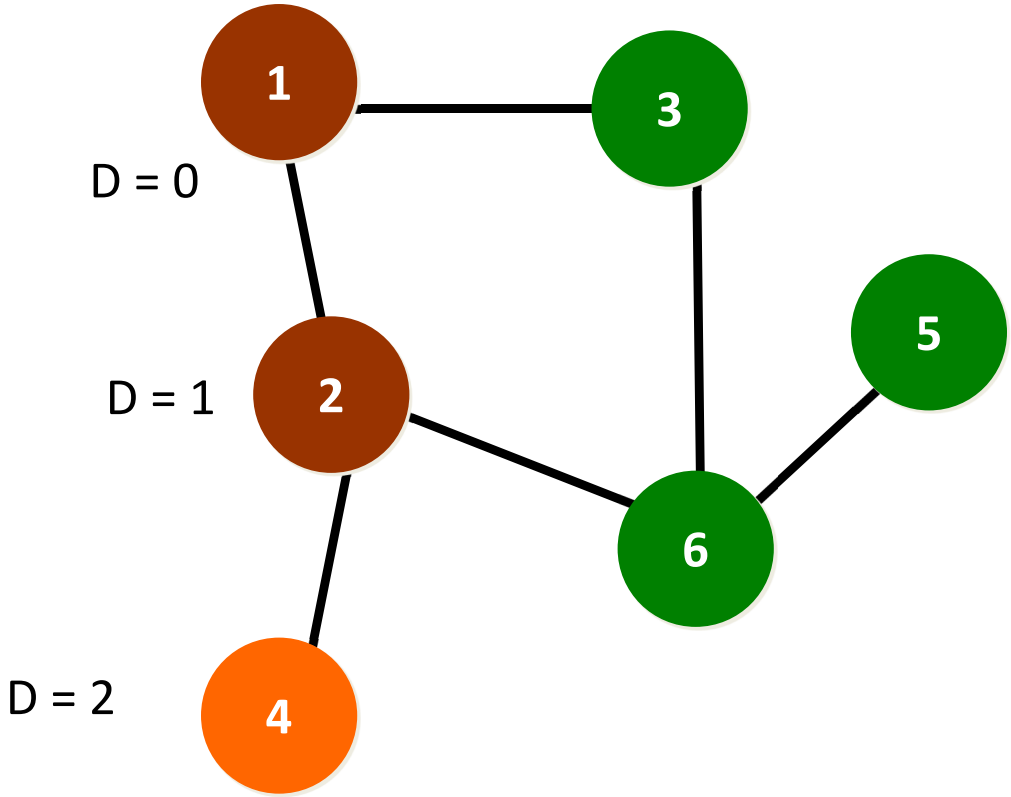
D: Depth

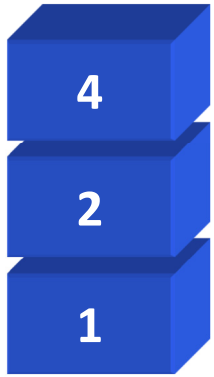




DFS Example (4/13)

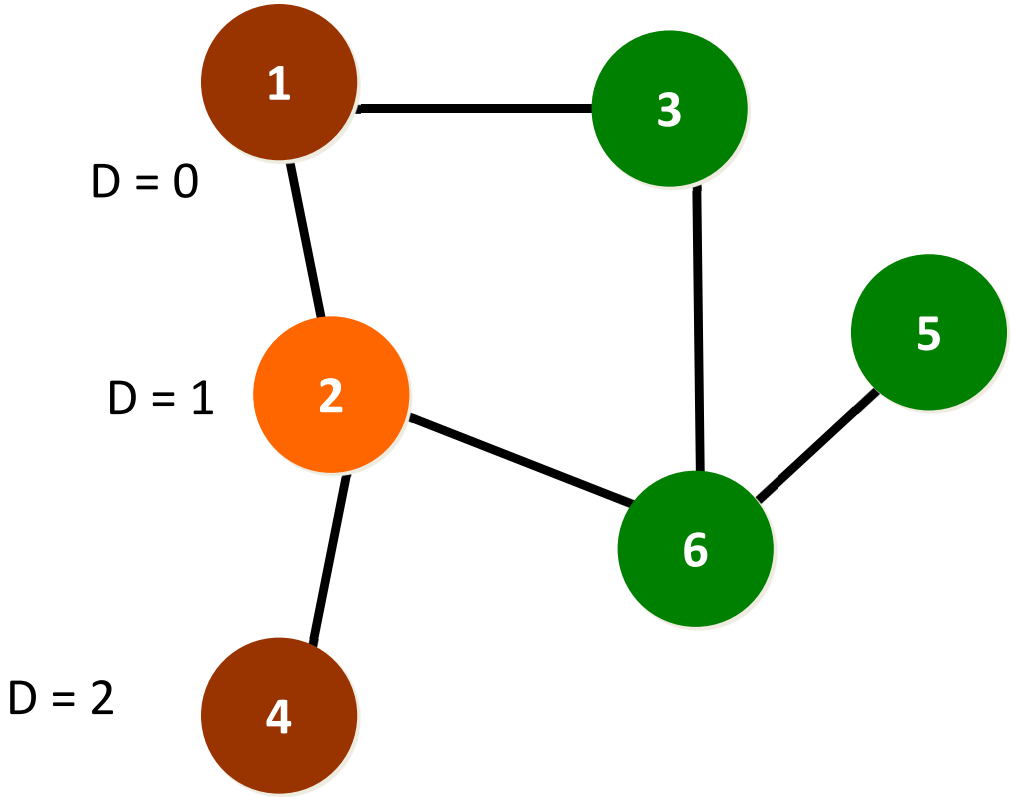
D: Depth

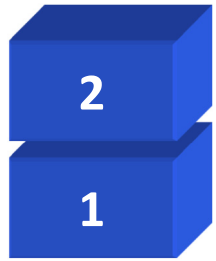




DFS Example (5/13)

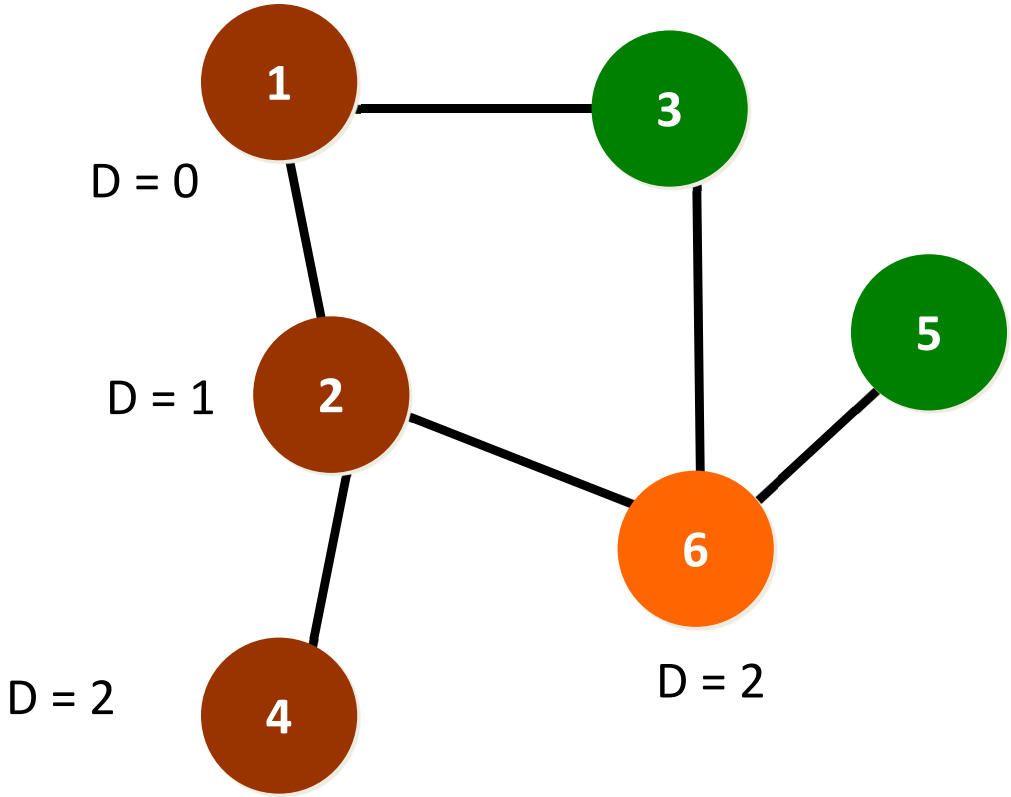
D: Depth

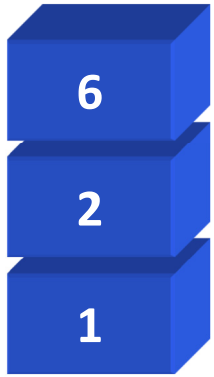




DFS Example (6/13)

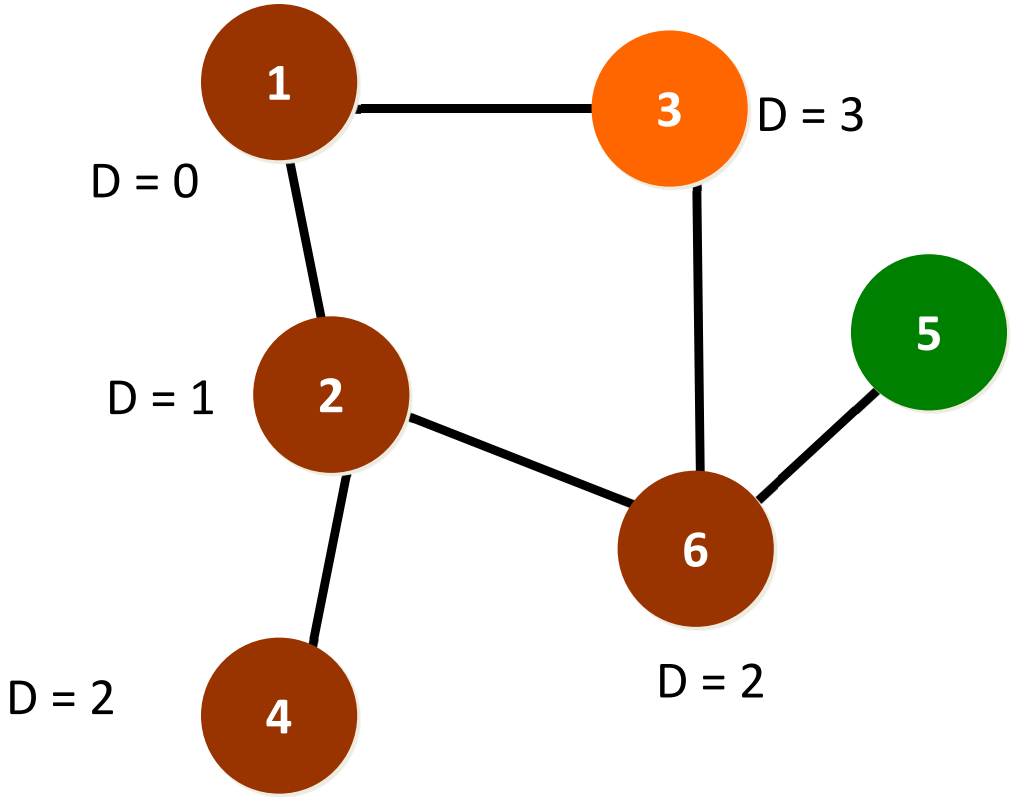
D: Depth

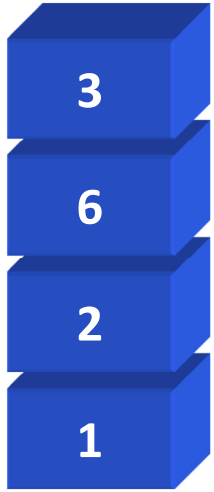




DFS Example (7/13)

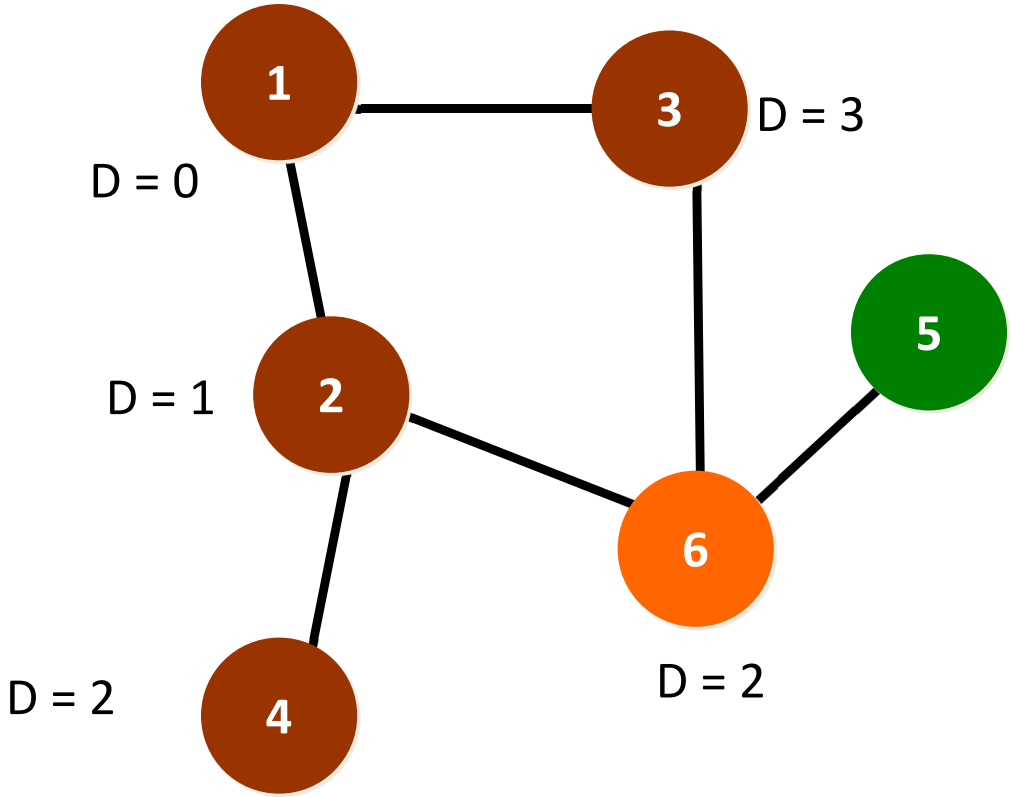
D: Depth

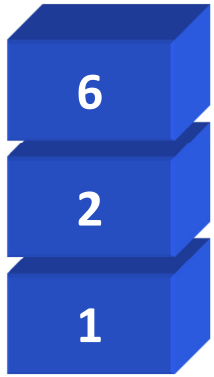




DFS Example (8/13)

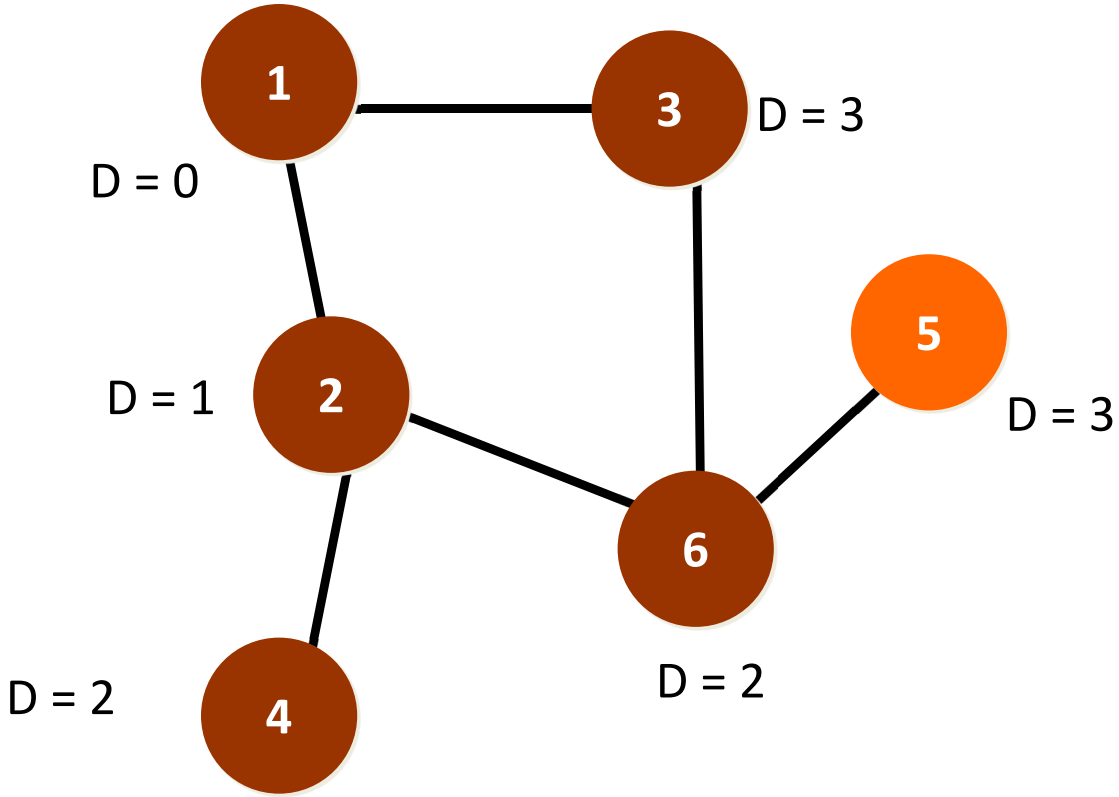
D: Depth

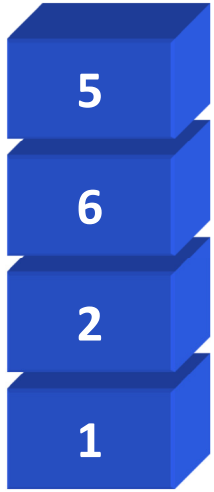




DFS Example (9/13)

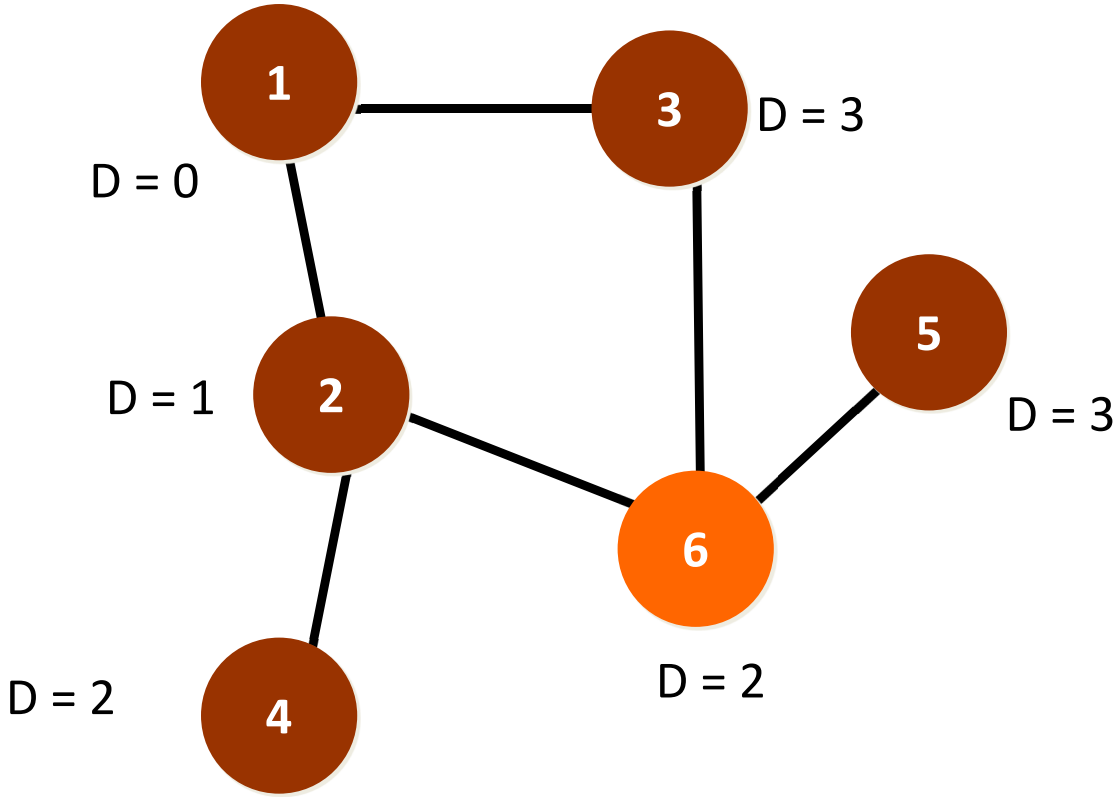
D: Depth

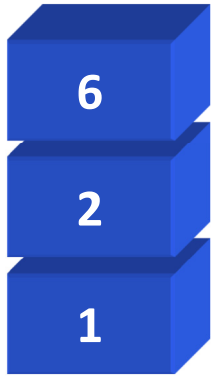




DFS Example (10/13)

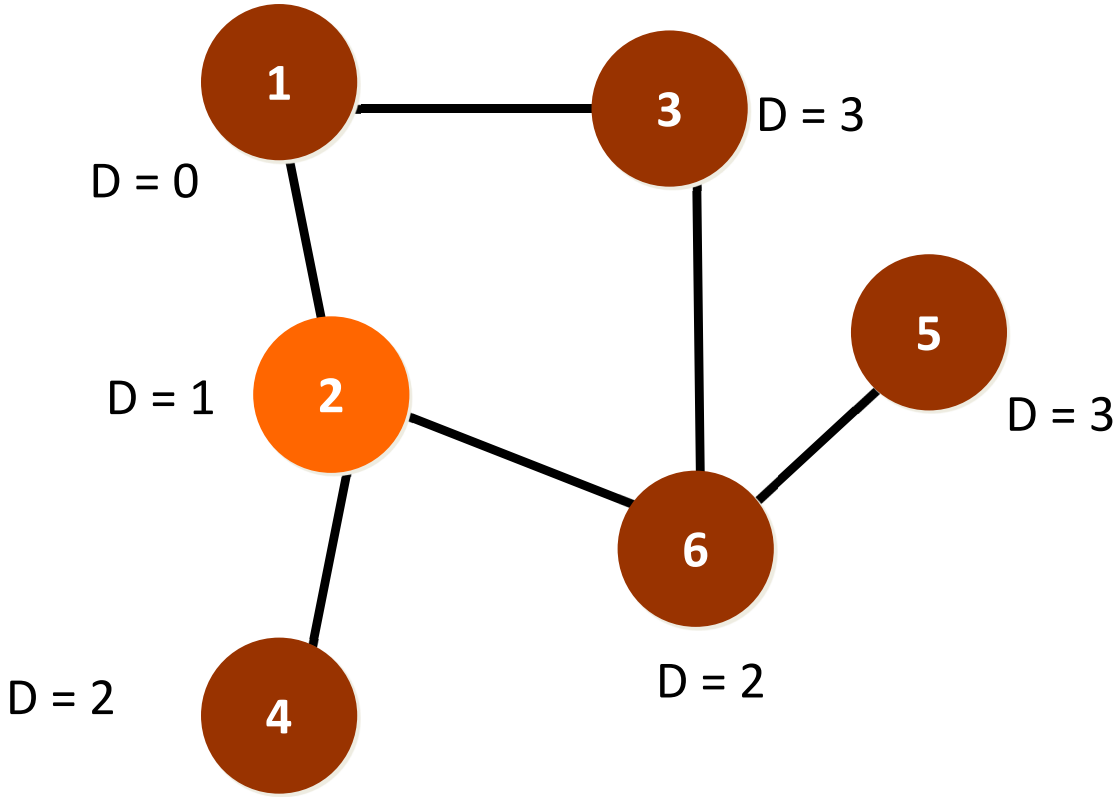
D: Depth

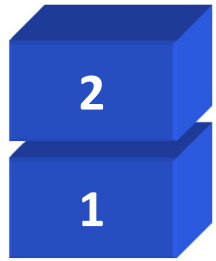




DFS Example (11/13)

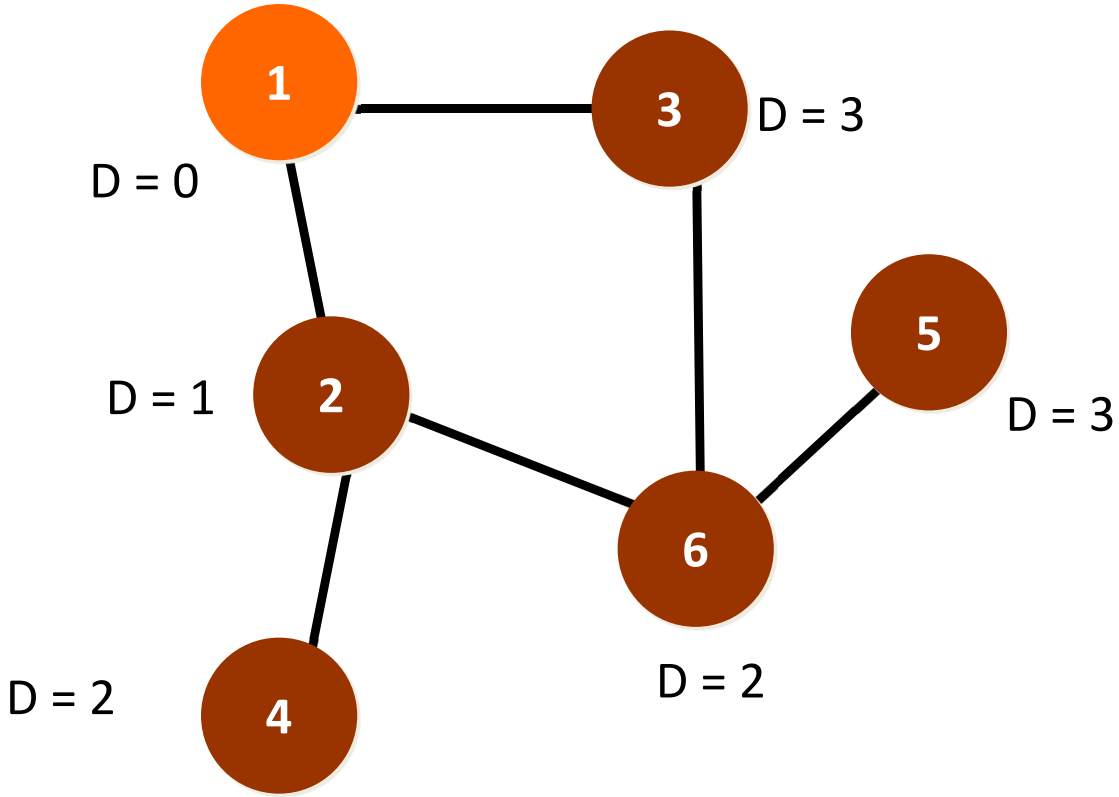
D: Depth





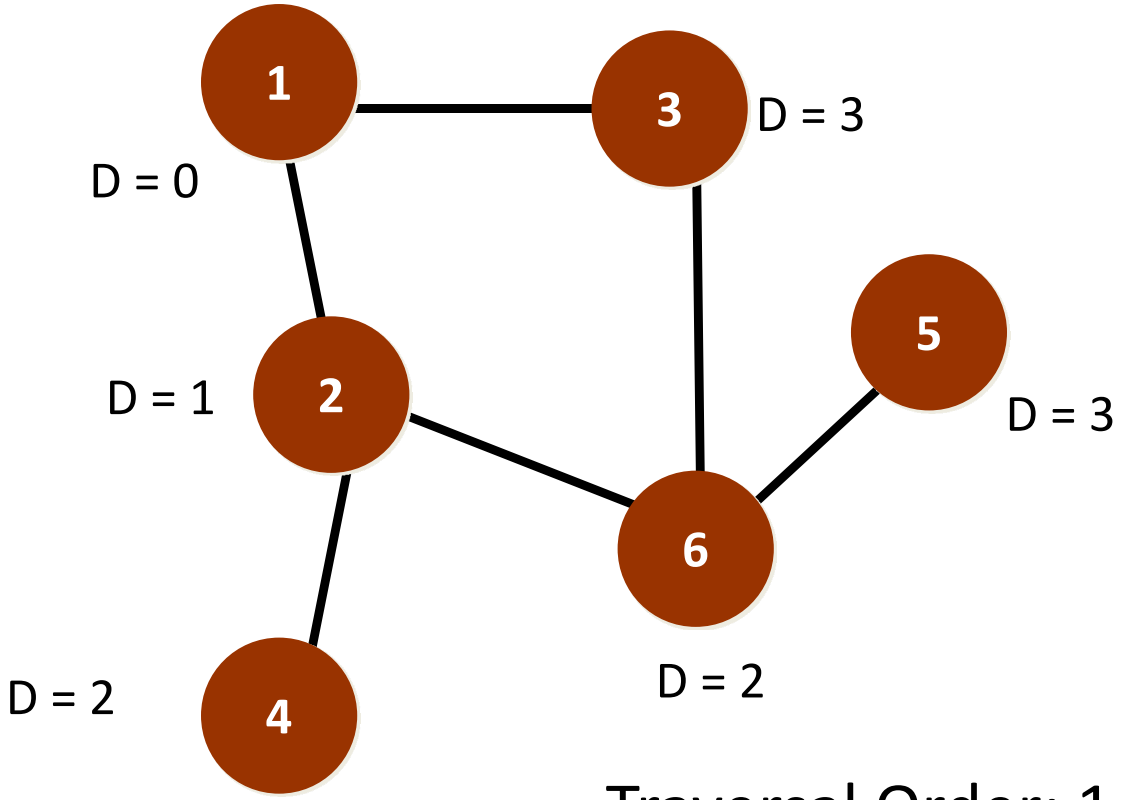
DFS Example (12/13)

D: Depth



DFS Example (13/13)

D: Depth



Traversal Order: 1 2 4 6 3 5

DFS Algorithm

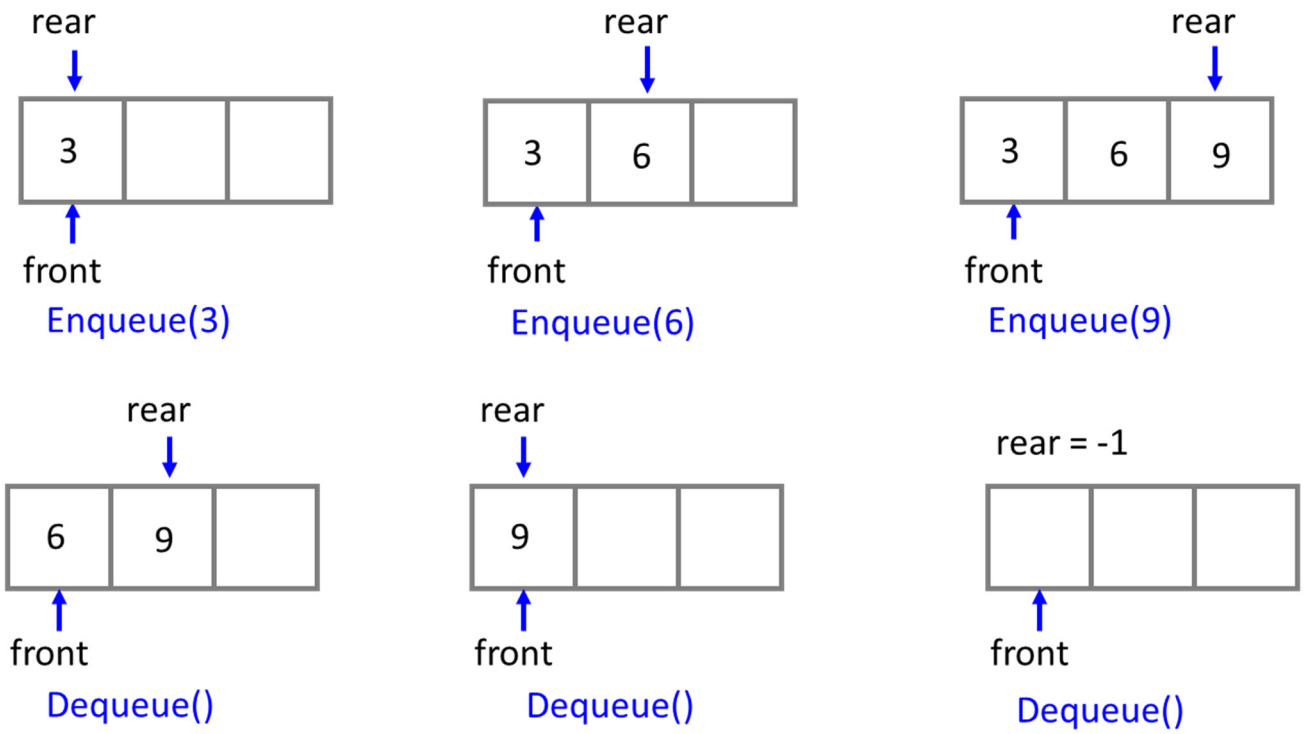
Require: Initial node v, graph/tree $G(V, E)$, stack S

1: **return** An ordering on how nodes in G are visited
2: Push v into S;
3: *visitOrder* = 0;
4: **while** S not empty **do**
5: *node* = pop from S;
6: **if** *node* not visited **then**
7: *visitOrder* = *visitOrder* +1;
8: Mark *node* as visited with order *visitOrder*; `//or print` *node*
9: Push all neighbors/children of *node* into S;
10: **end if**
11: **end while**
12: Return all nodes with their visit order.

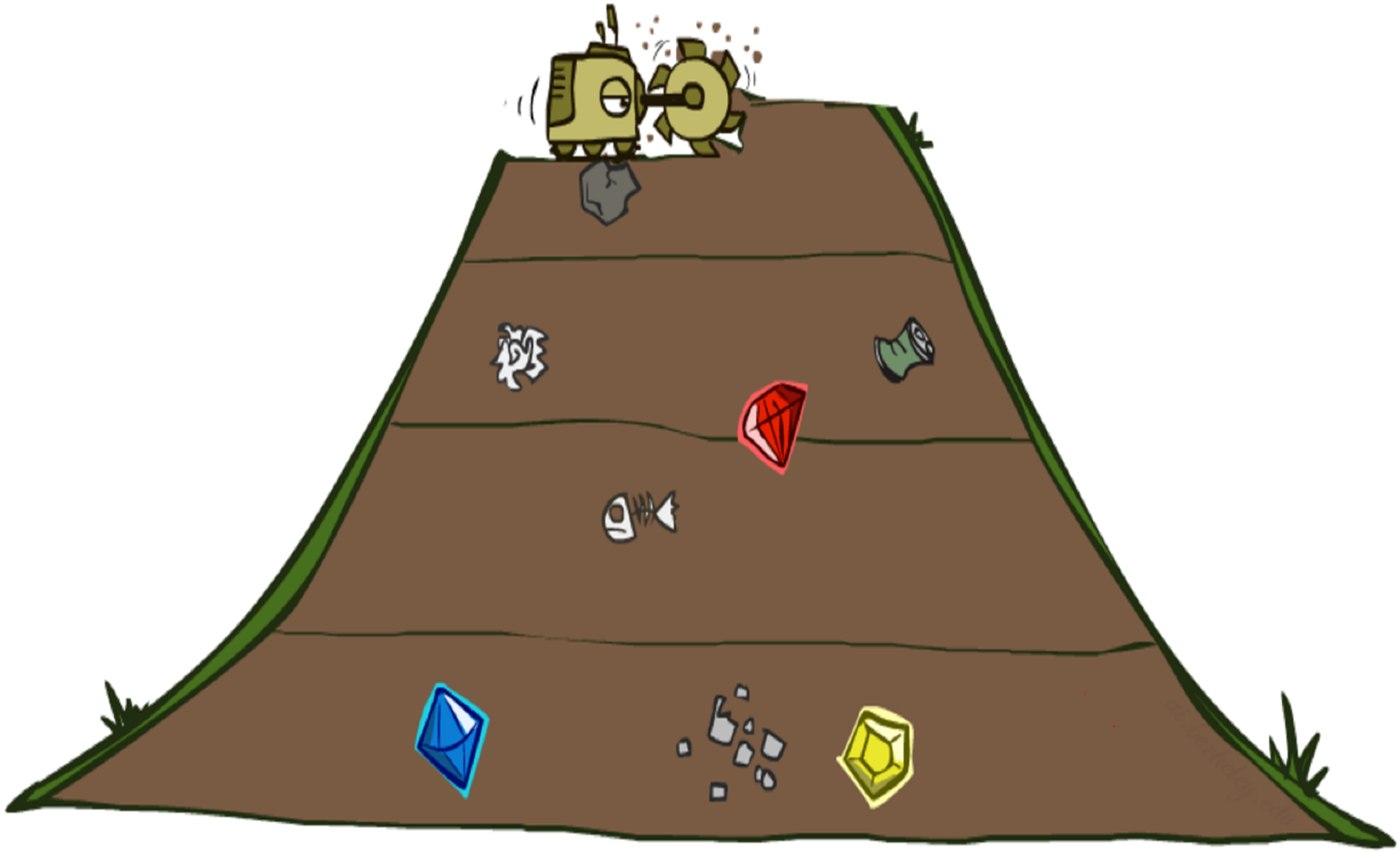
https://en.wikipedia.org/wiki/Depth-first_search

Breadth First Search Traversal

- Breadth-First-Search (BFS) starts from a node, visits all its immediate neighbors first, and then moves to the second level by traversing their neighbors
- BFS can be implemented using queue
  - First-In-First-Out (FIFO) data structure

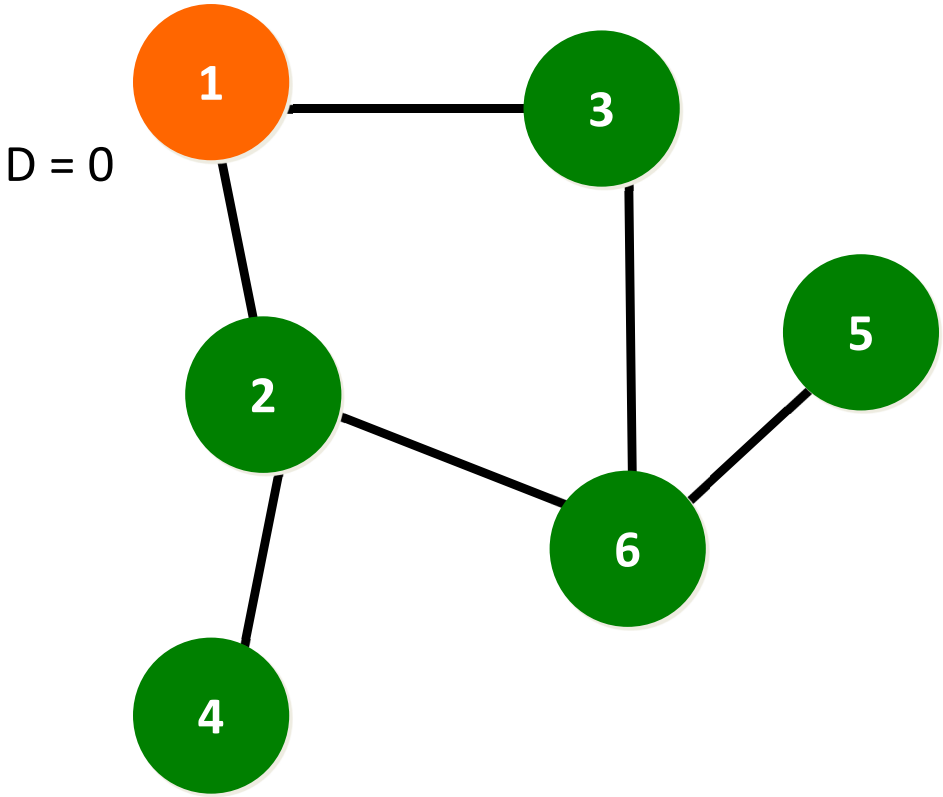


Breadth First Search



BFS Example (1/8)

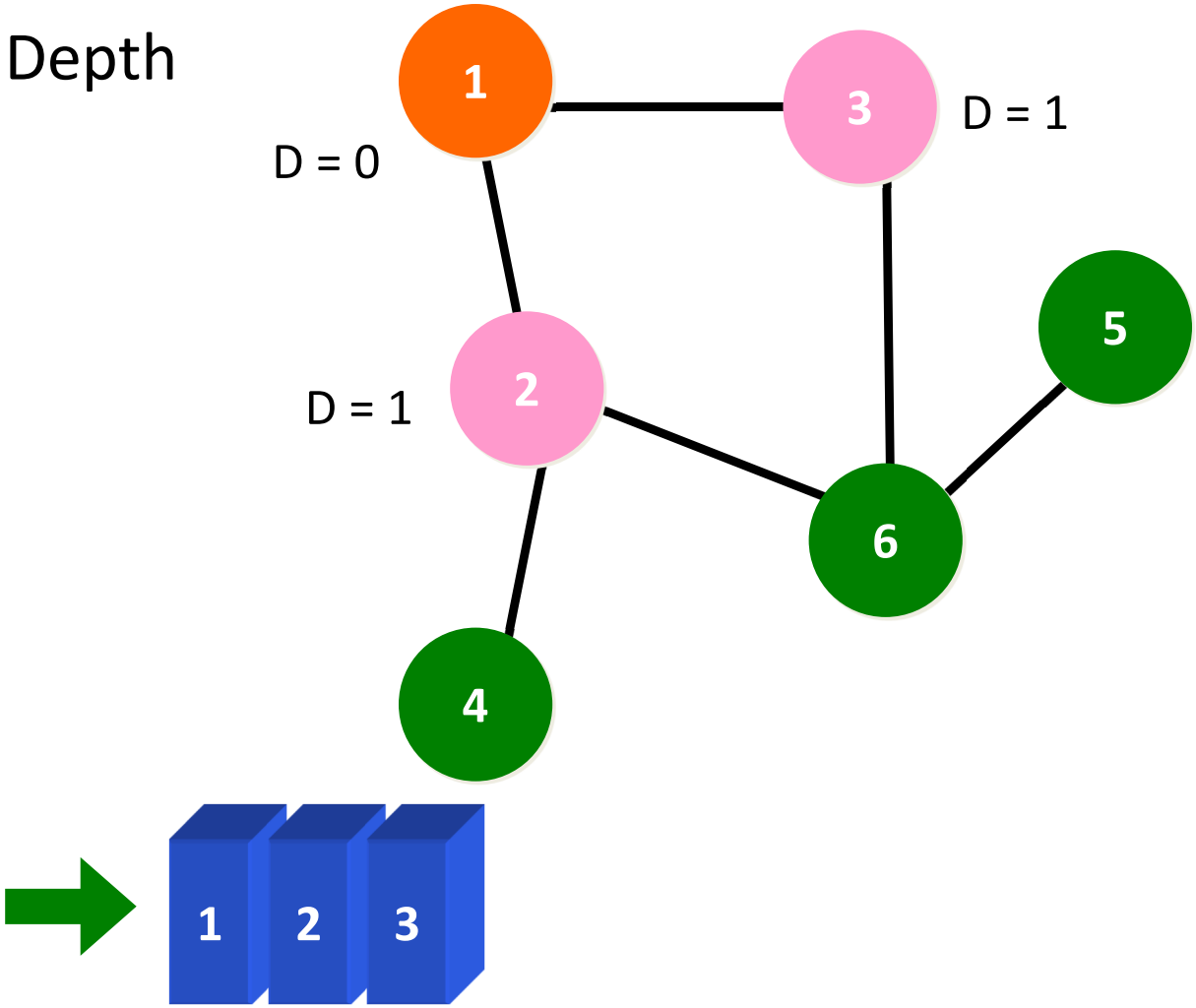
D: Depth



Queue

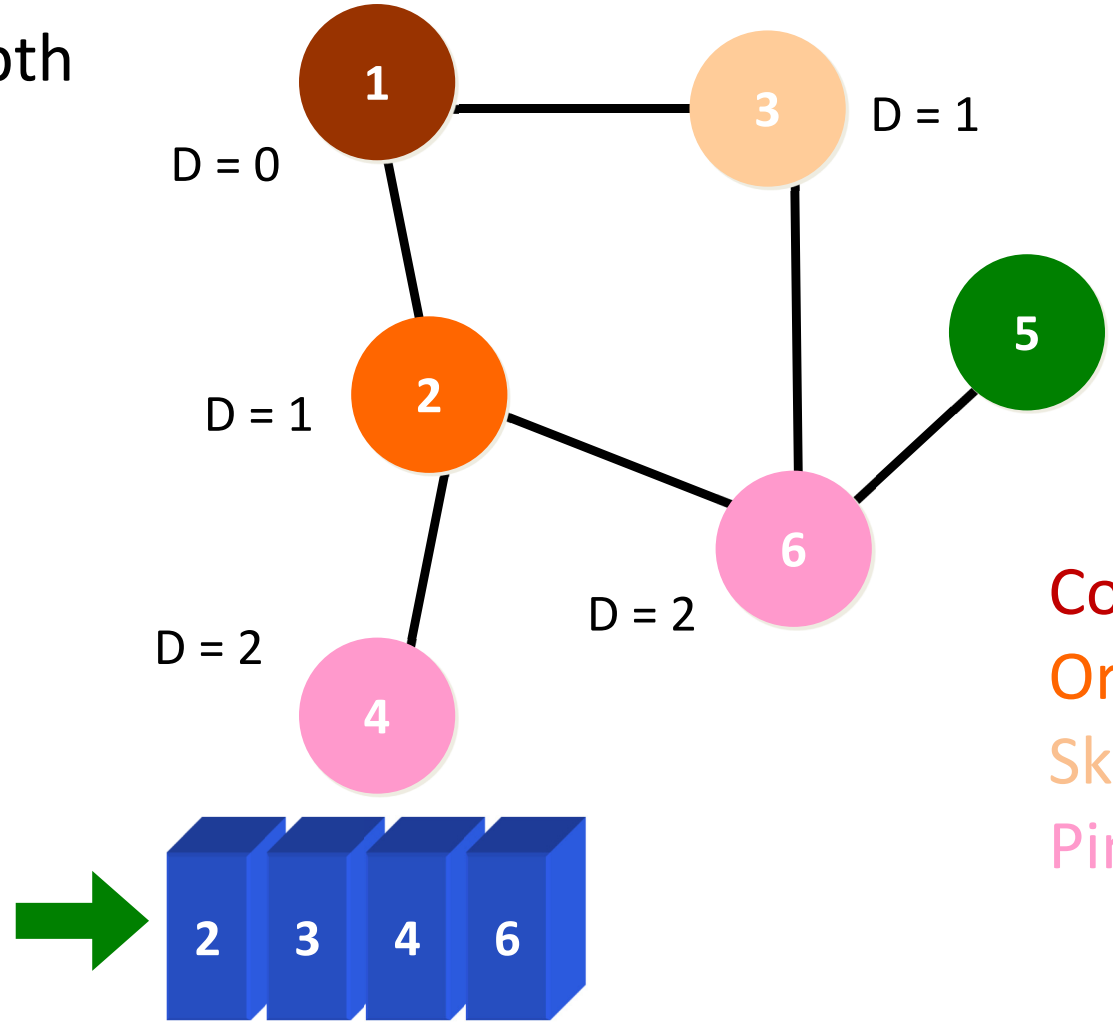
BFS Example (2/8)

D: Depth



BFS Example (3/8)

D: Depth



Coffee: visited

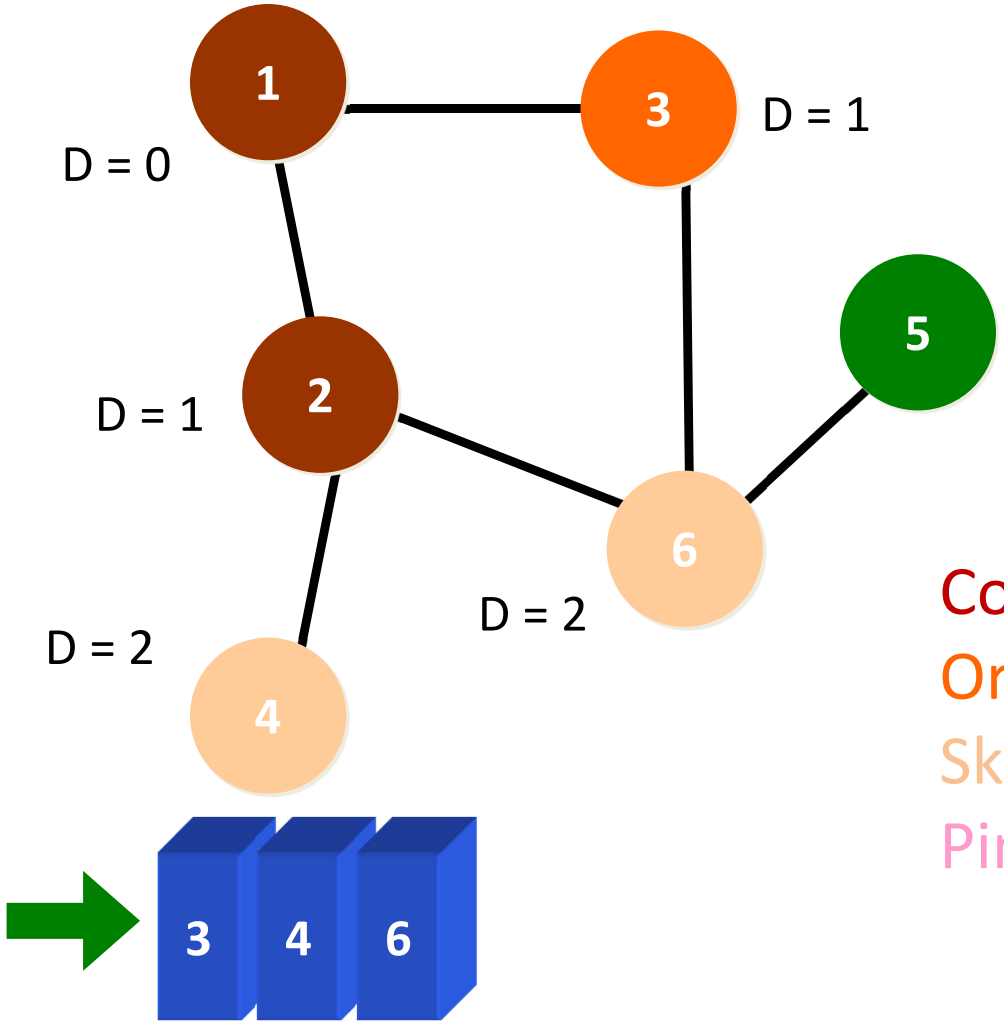
Orange: current

Skin: in queue

Pink: add into queue

BFS Example (4/8)

D: Depth



Coffee: visited

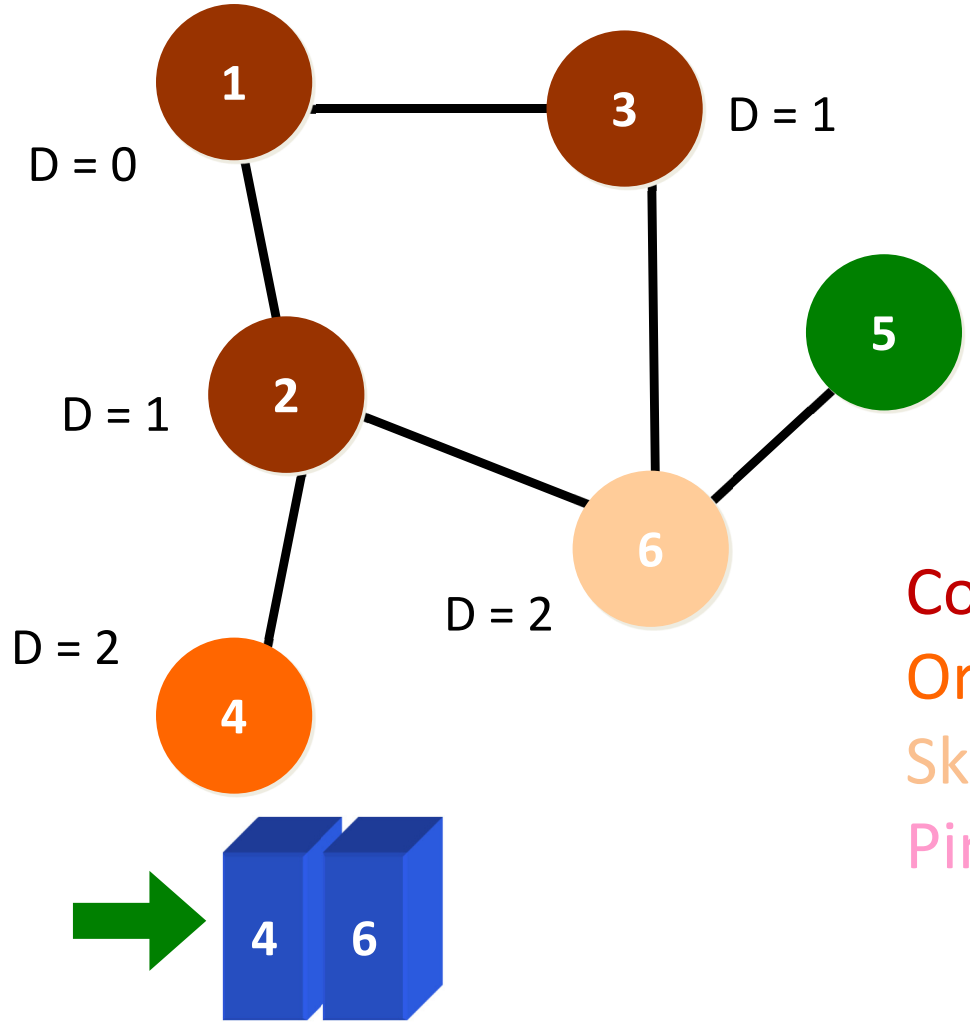
Orange: current

Skin: in queue

Pink: add into queue

BFS Example (5/8)

D: Depth



Coffee: visited

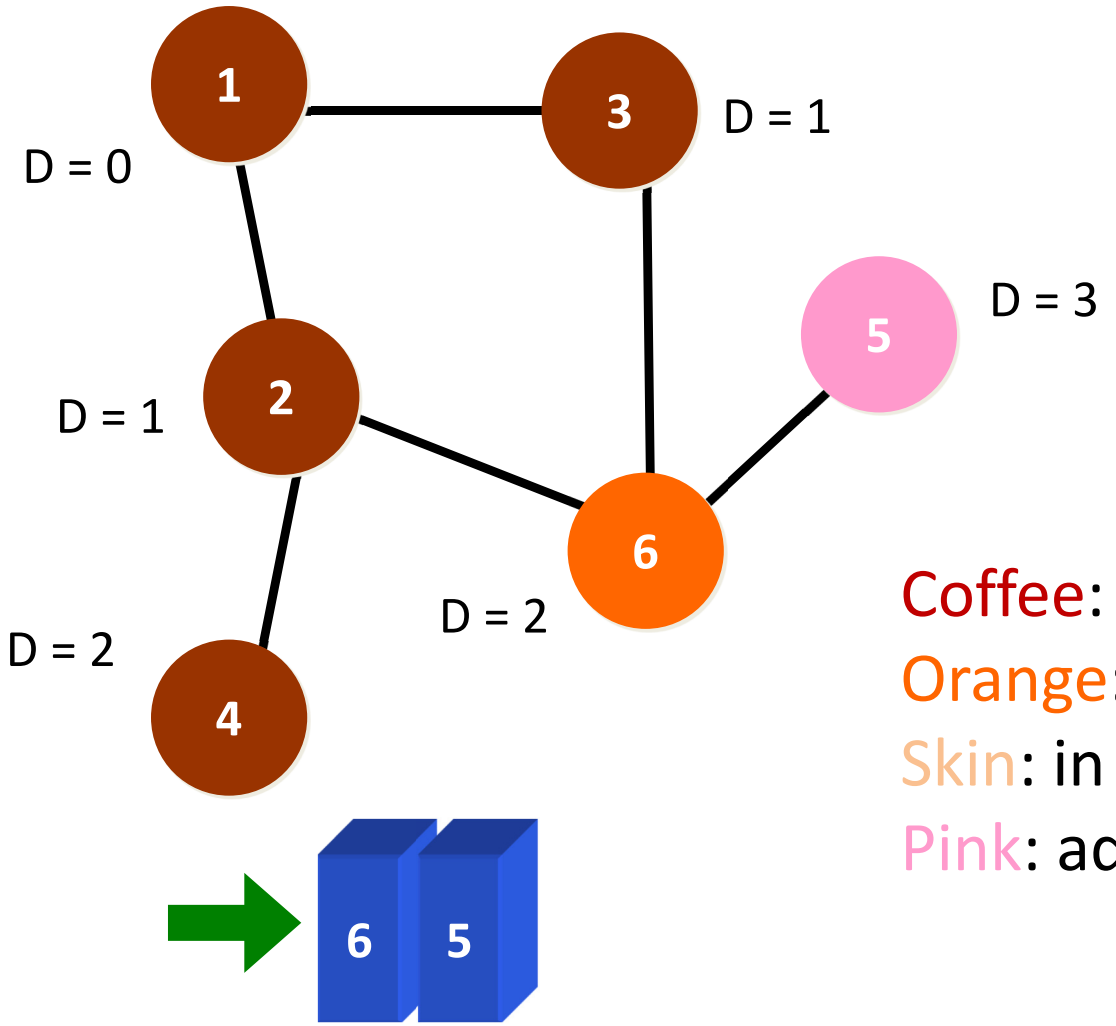
Orange: current

Skin: in queue

Pink: add into queue

BFS Example (6/8)

D: Depth



Coffee: visited

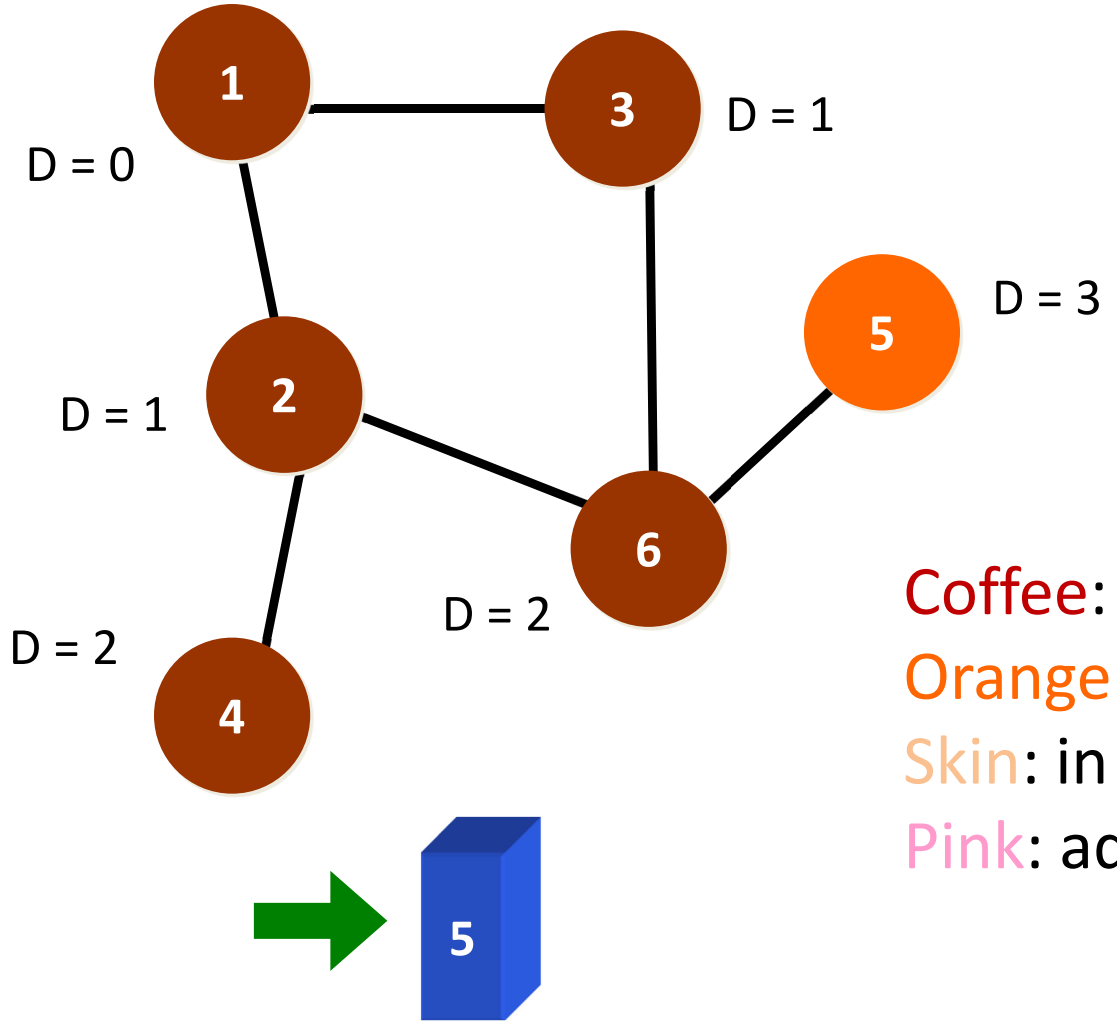
Orange: current

Skin: in queue

Pink: add into queue

BFS Example (7/8)

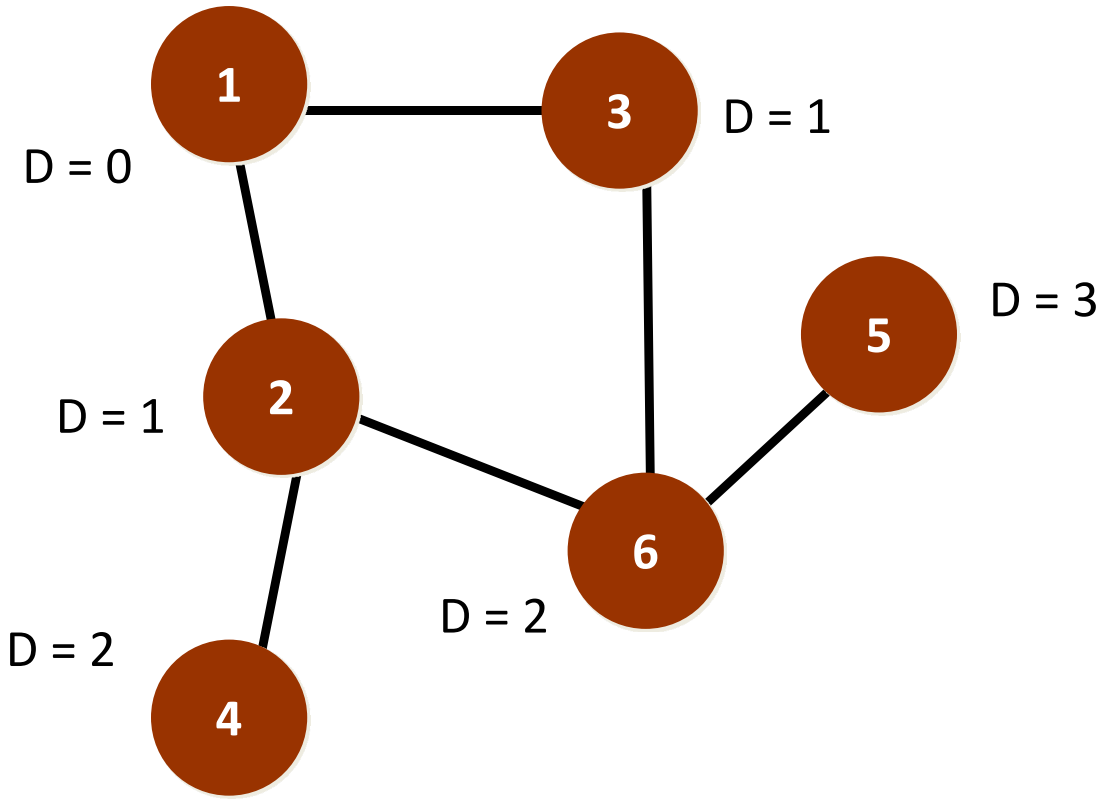
D: Depth



Coffee: visited
Orange: current
Skin: in queue
Pink: add into queue

BFS Example (8/8)

D: Depth



Traversal Order: 1 2 3 4 6 5

BFS Algorithm

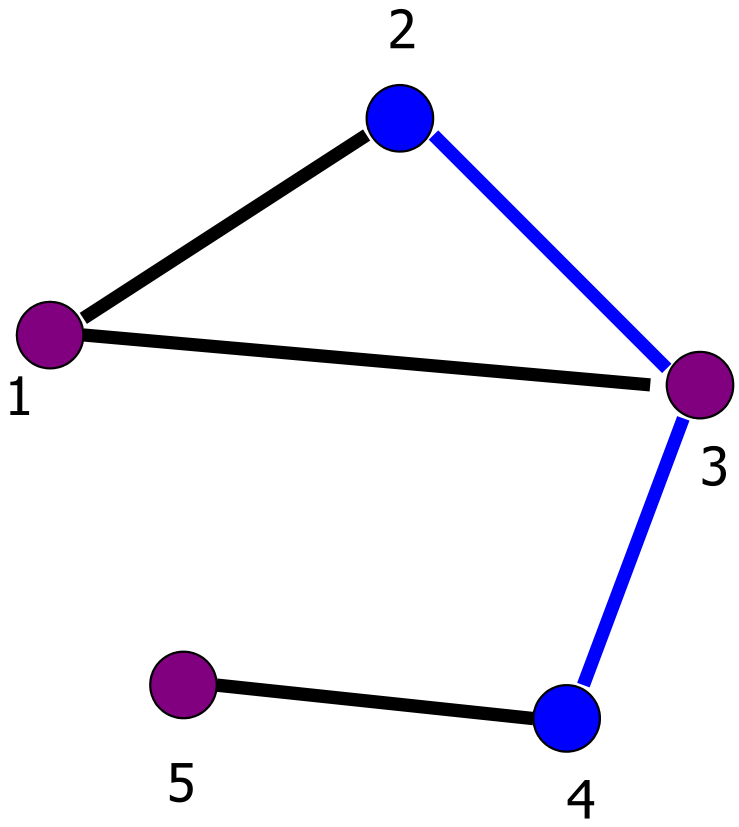
Require: Initial node v, graph/tree $G(V, E)$, queue Q

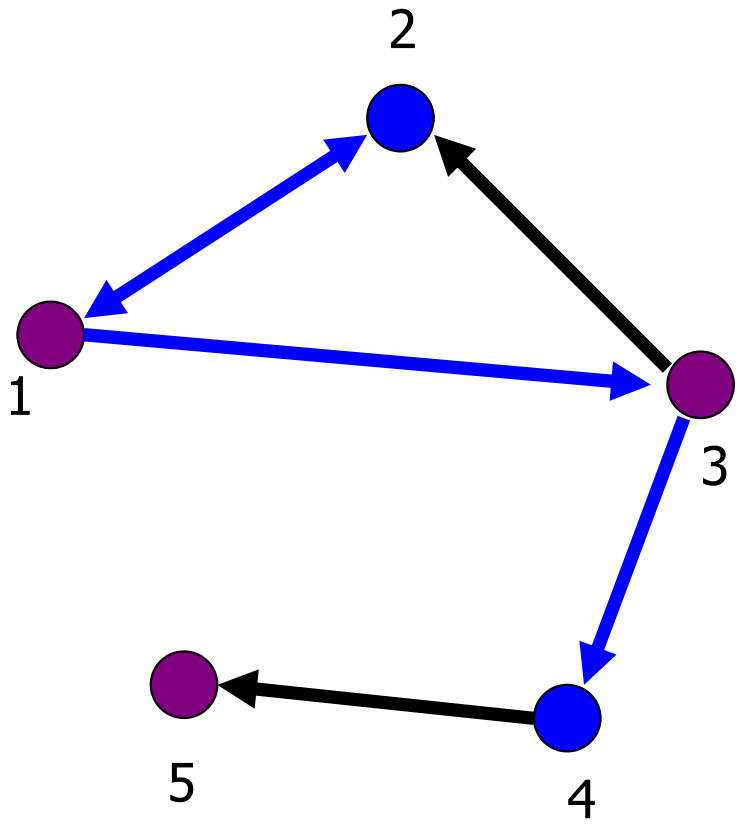
1: **return** An ordering on how nodes are visited
2: Enqueue v into queue Q;
3: $visitOrder = 0$;
4: **while** Q not empty **do**
5: $node$ = dequeue from Q;
6: **if** $node$ not visited **then**
7: $visitOrder = visitOrder + 1$;
8: Mark $node$ as visited with order $visitOrder$; `//or print` $node$
9: Enqueue all neighbors/children of $node$ into Q;
10: **end if**
11: **end while**

https://en.wikipedia.org/wiki/Breadth-first_search

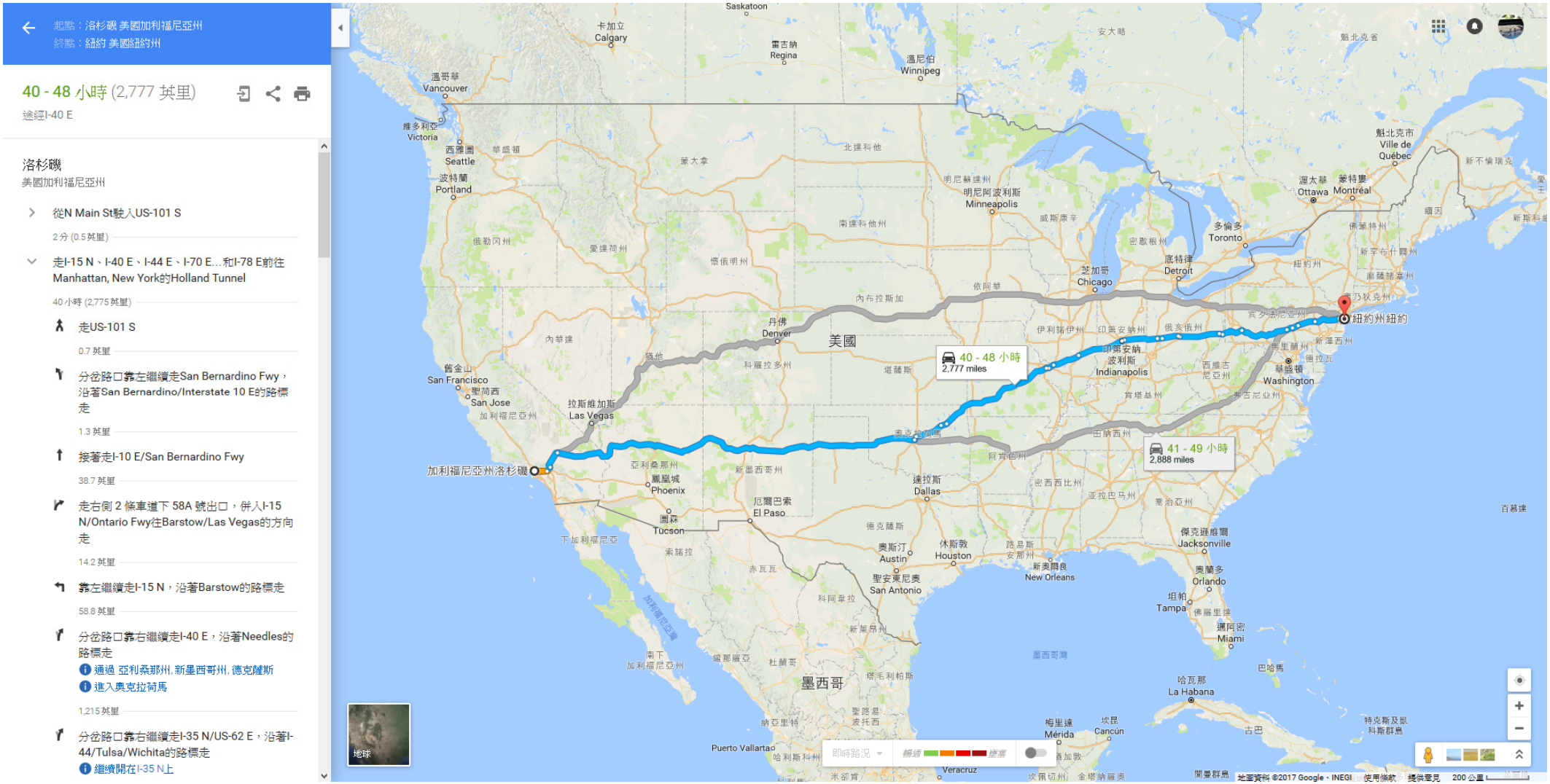
Shortest Paths

- Path with shortest length from node i to node j
  - also known as BFS path, or geodesic path





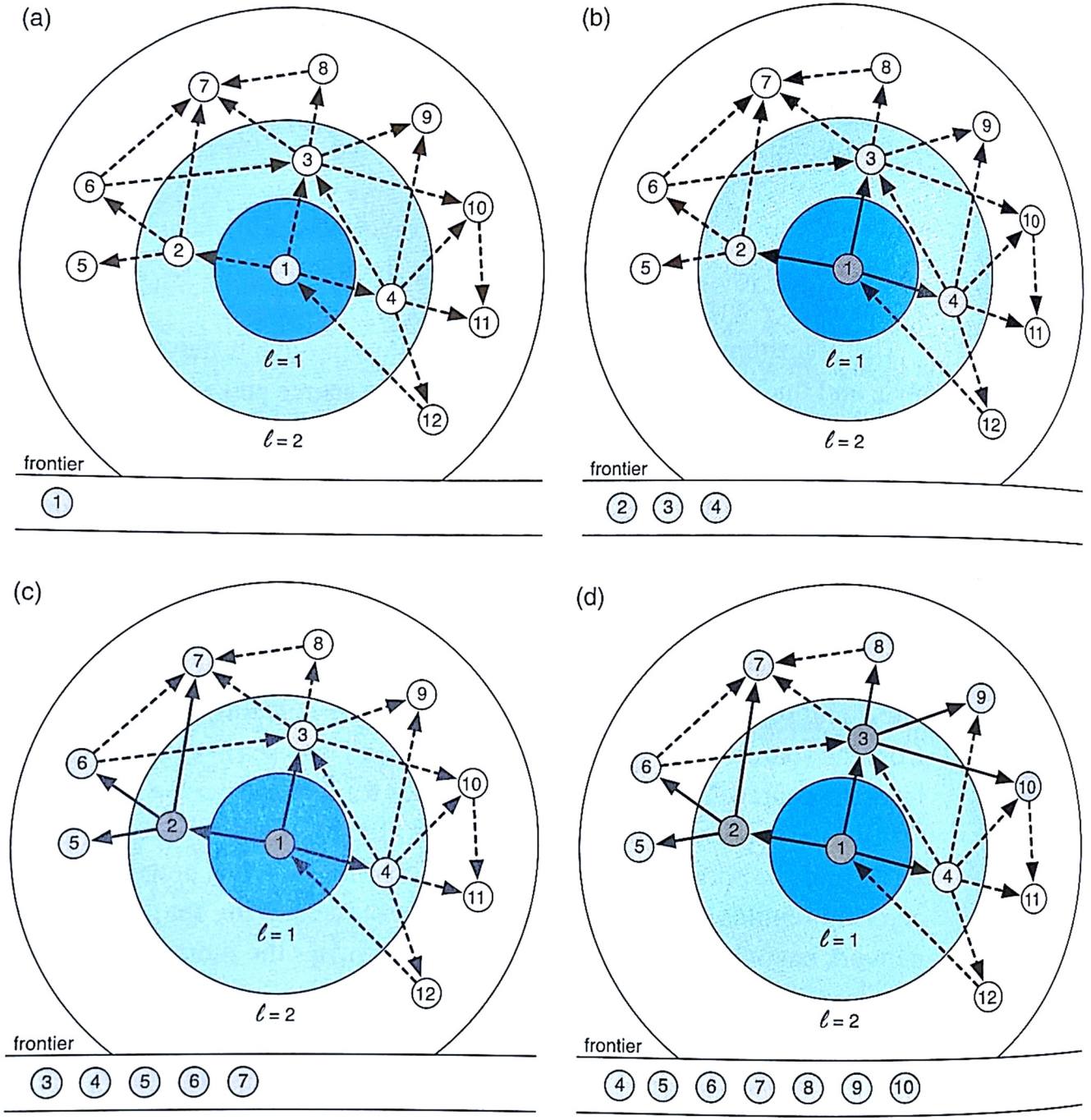
The Shortest Path in Real World



Shortest Paths on Weighted Graphs

- In many scenarios, we want the shortest path/distance between two nodes in a graph
  - E.g. disseminate info, travel planning, acquaintance estimation
- Shortest paths on weighted graphs are harder to be identified
  - There are several well known algorithms for finding single-source, or all-pairs shortest paths
- Dijkstra's Algorithm
  - Designed for weighted graphs with non-negative edges
  - Find shortest paths starting from a given node s to all other nodes
  - Find both shortest paths and their respective lengths

Finding Shortest Paths by BFS



Short Summary

- Uninformed Graph Search
  - Breadth-First Search (BFS) [Queue]
  - Depth-First Search (DFS) [Stack]
  - Dijkstra's Algorithm [Priority Queue]
- Informed Graph Search
  - Greedy Best-First Search
  - A* Algorithm