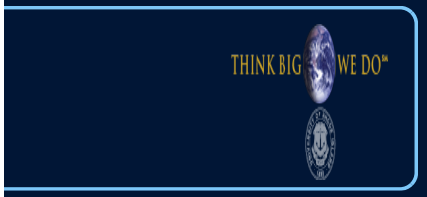


CSC 212
Data Structures & Algorithms

Fall 2022 | Jonathan Schrader

Graphs

Housekeeping

Lab 12: Graphs

Assignment 5
- Due 12/2 11:59p

Term Project
- Due 12/5 11:59p
- Presentation signup

INTRODUCTION

Graphs

Graph:

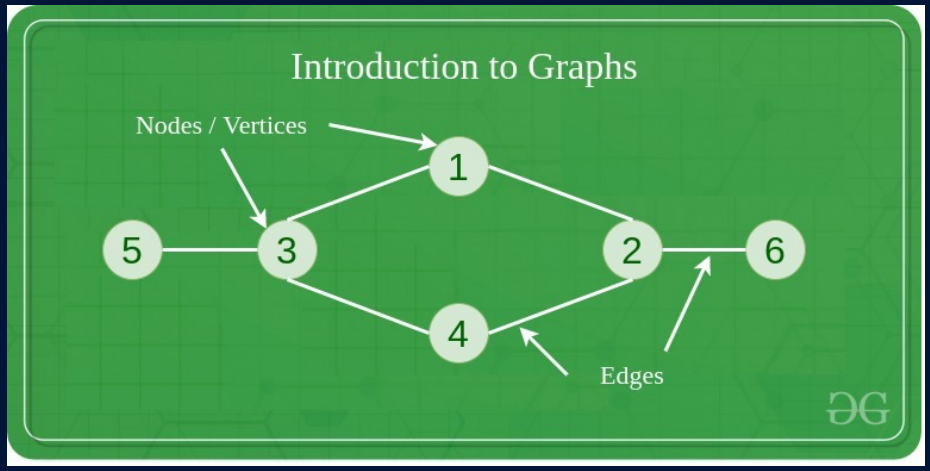
- a set of vertices connect pairwise by edges

Vertices:

- fundamental units of the graph known as vertex or nodes;
- every node/vertex can be labeled or unlabelled

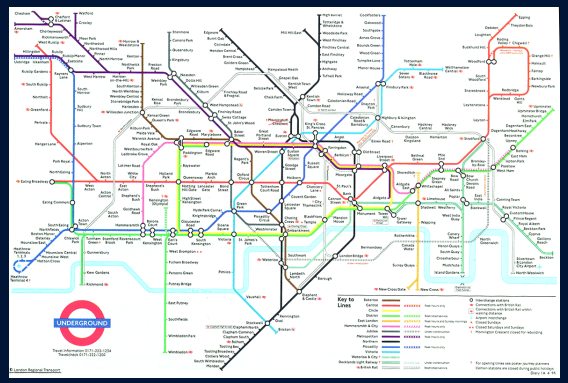
Edges:

- are drawn or used to connect two nodes of the graph
- can connect any two nodes in any possible way
- can be labeled or unlabelled



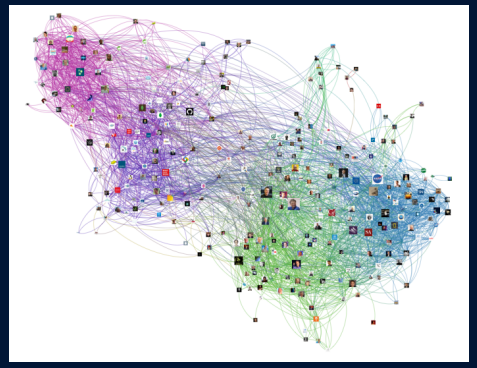
gfg: graph vertices and edges

Examples

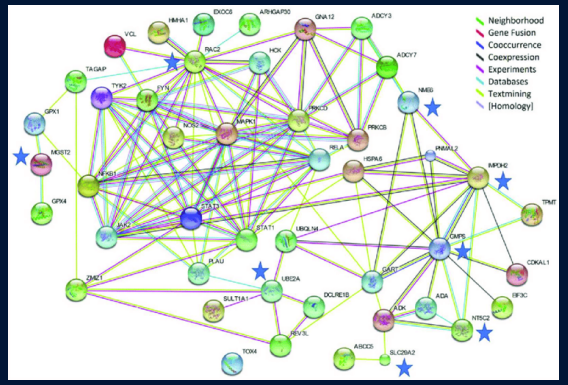


London Underground (Tube) Map
Vertex = subway stop; edge = direct route

Facebook Social Network Map
Vertex = person; edge = social relationship



Twitter followers
Vertex = Twitter account; edge = Twitter follower



Protein-protein interation network
Vertex = protein; edge = interaction

Applications…

graph	vertex	edge
cell phone	phone	placed call
infectious disease	person	infection
financial	stock, currency	transactions
transportation	intersection	street
internet	router	fiber cable
web	web page	URL link
social relationship	person	friendship

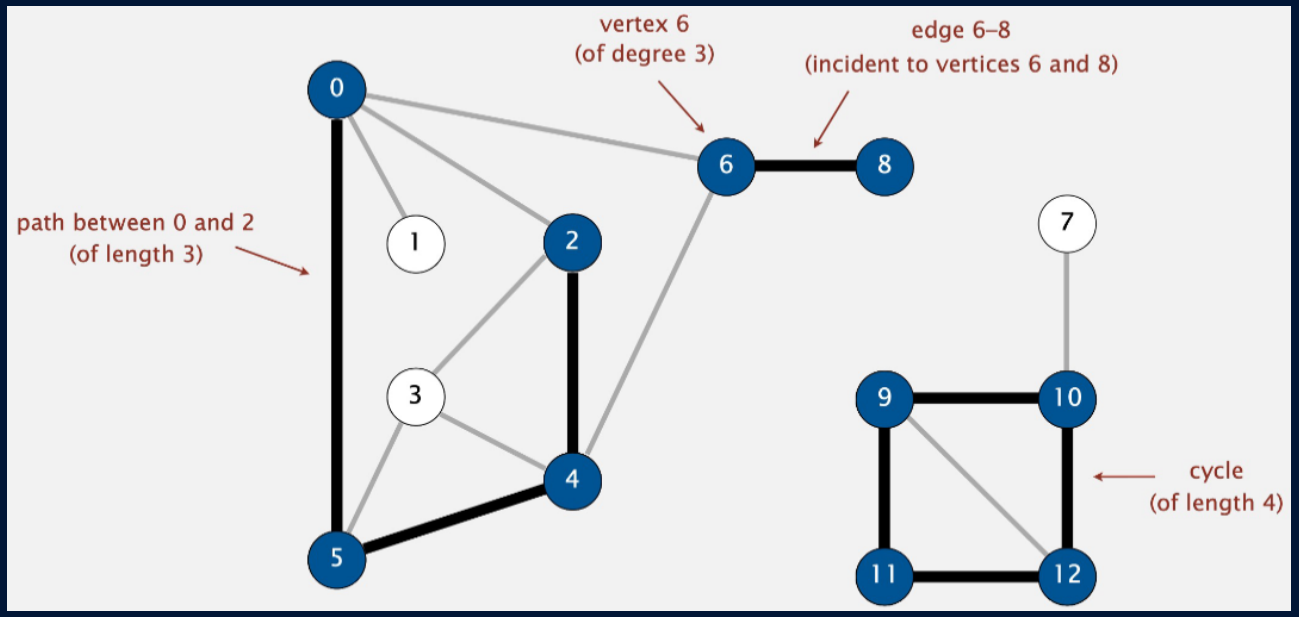
Undirected Graphs

Graph
- a set of vertices connect pairwise by edges

Path
- sequence of vertices connected by edges, with no repeated edges

Definition
- two vertices are connected if there is a path between them

Cycle
- path (with ≥ 1 edge) whose first and last vertices are the same



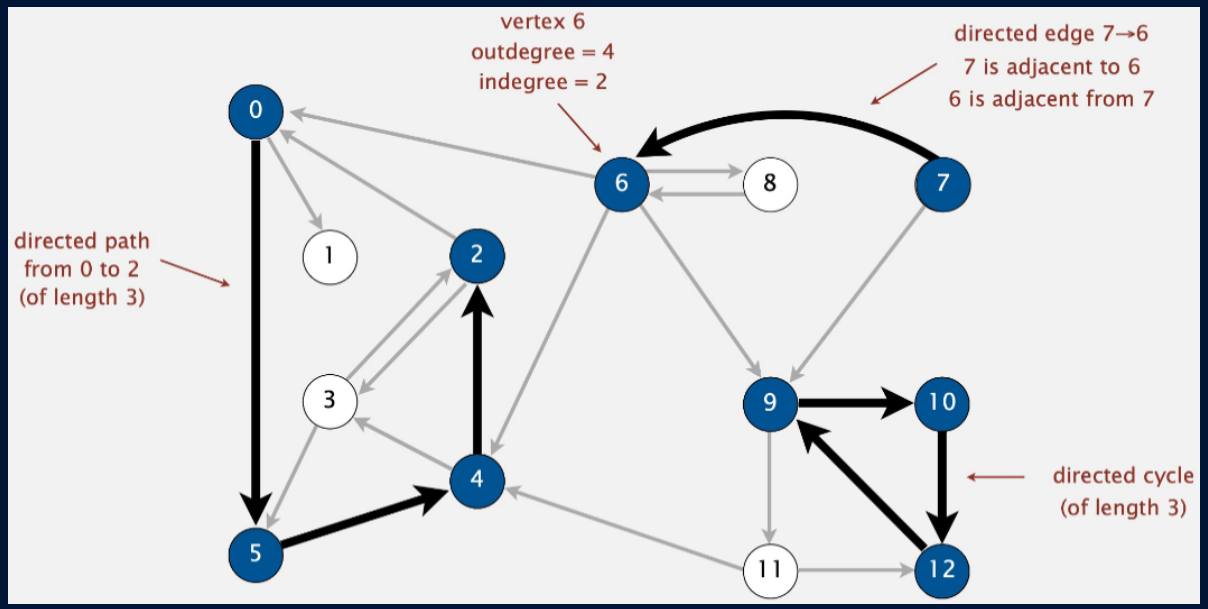
Directed graphs

Digraph
- a set of vertices connect pairwise by directed edges

Directed Path
- sequence of vertices connected by edges, with no repeated edges

Definition
- vertex w is reachable from vertex v if there is a directed path from v to w

Directed Cycle
- directed path (with ≥ 1 edge) whose first and last vertices are the same



Graph Processing Problems

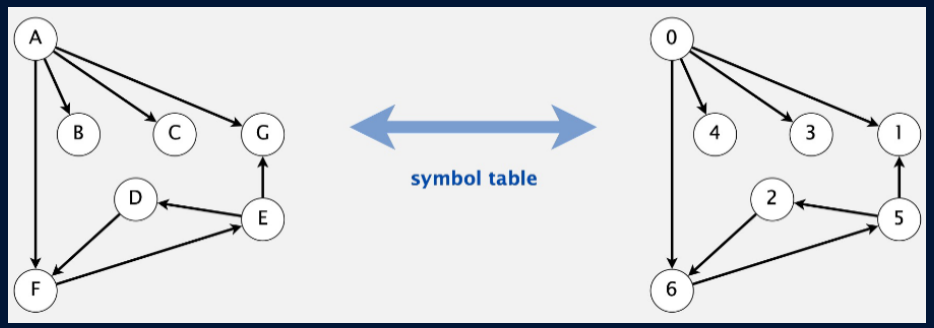
problem	description
s-t path	Find a path between s and t
shortest s-t path	Find a path with the fewest edges between s and t
cycle	Find a cycle
Euler cycle	Find a cycle that uses each edge exactly once
Hamilton cycle	Find a cycle that uses each vertex exactly once
connectivity	Is there a path between ev ery pair of vertices?
graph isomorphism	Are two graphs isomorphic?
planarity	Draw the graph in the plane with no crossing edges

REPRESENTATION

Digraph

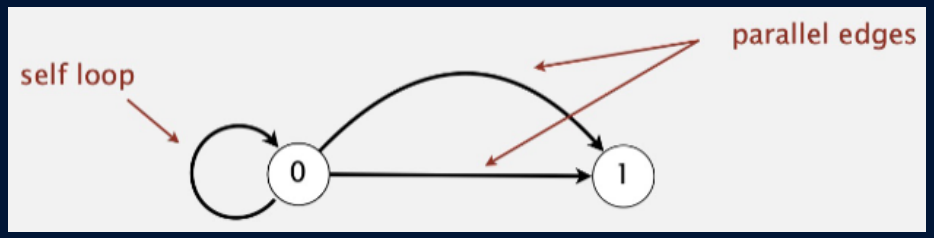
Vertex representation

- for 0 through all positive values $V - 1$
- Applications: use symbol table to convert between names and integers



Definition

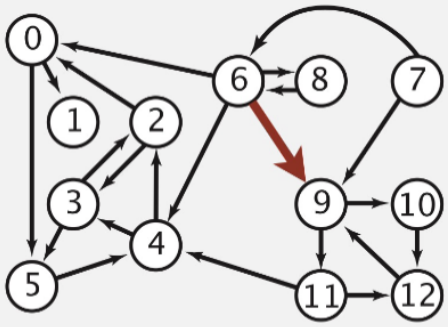
- a digraph is simple if it has no self-loops or parallel edges



Adjacency-matrix

Maintain a V-by-V boolean array; for each edge $v \rightarrow w$ in the digraph:

$$adj[v][w] \; = \; true$$

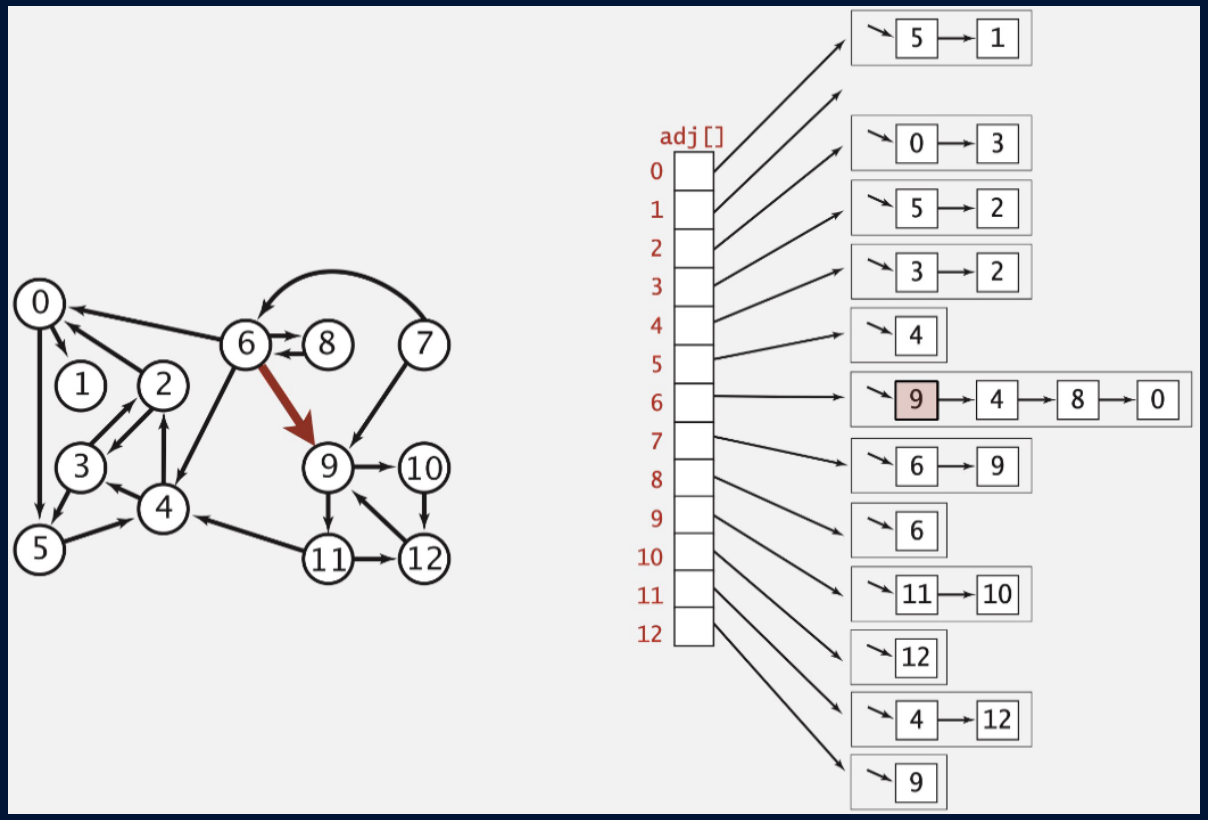


from \ to	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0	1	0	0	0	1	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0
2	1	0	0	1	0	0	0	0	0	0	0	0	0
3	0	0	1	0	0	1	0	0	0	0	0	0	0
4	0	0	1	1	0	0	0	0	0	0	0	0	0
5	0	0	0	0	1	0	0	0	0	0	0	0	0
6	0	0	0	0	1	0	0	0	1	1	0	0	0
7	0	0	0	0	0	0	1	0	0	1	0	0	0
8	0	0	0	0	0	0	1	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	1	1	0
10	0	0	0	0	0	0	0	0	0	0	0	0	1
11	0	0	0	0	1	0	0	0	0	0	0	0	1
12	0	0	0	0	0	0	0	0	0	1	0	0	0

Note: parallel edges disallowed

Adjacency-lists

Maintain vertex-indexed array of lists



In Practice

Use adjacency-lists

- Algorithms based on iterating over vertices adjacent from v
- Real-world graphs tend to be sparce ($\Theta(V)$ edges), not dense ($\Theta(V^2)$ edges)

representation	space	add edge from v to w	has edge from v to w	iterate over vertices adjacent from v
adjacent matrix	V^2	1*	1	V
adjacent lists	$E + V$	1	$outdegree(v)$	$outdegree(v)$

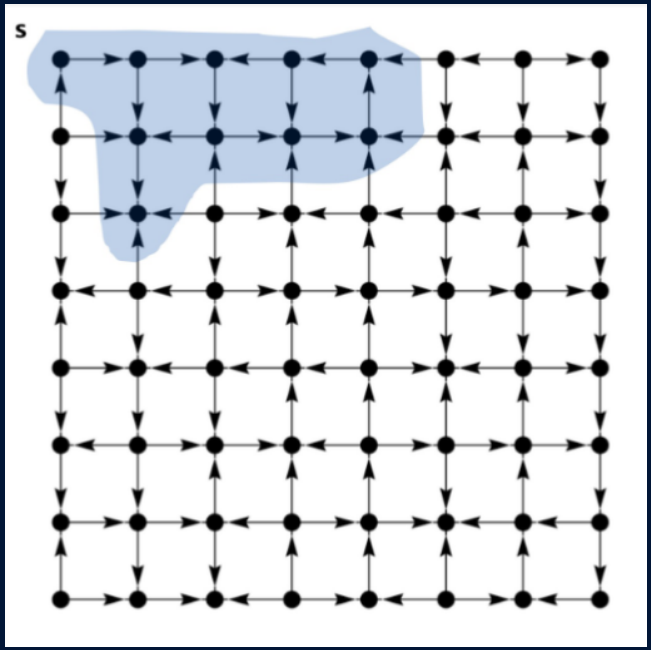
* disallows parallel edges

DEPTH-FIRST SEARCH

Reachability

Problem

- Given a digraph G and vertex s, find all the vertices reachable from s



Depth-first search

Goal

- Systemically traverse a digraph

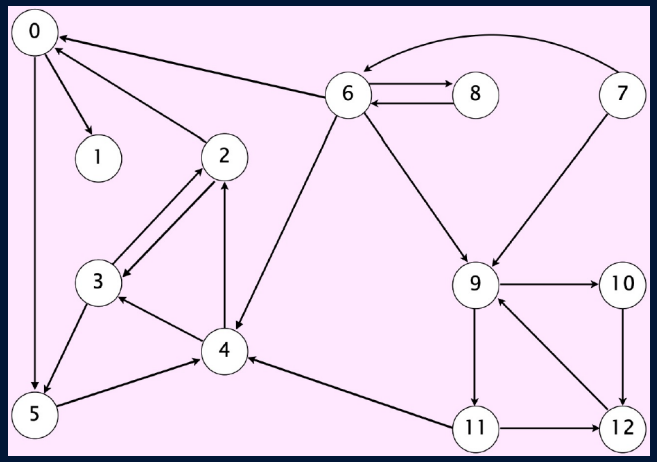
```
void Graph::dfs( Vertex v )
{
  v.visited = true;
  for each Vertex w adjacent to v
    if( !w.visited )
      dfs( w );
}
```

Typical applications

- Reachability: finding all vertices reachable from a given vertex
- Path finding: find directed path from one vertex to another

Directed depth-first search

To visit vertex v:
- Mark vertex v
- Recursively visit all unmarked vertices adjacent from v



a directed graph

4→2
2→3
3→2
6→0
0→1
2→0
11→12
12→9
9→10
9→11
8→9
10→12
11→4
4→3
3→5
6→8
8→6
5→4
0→5
6→4
6→9
7→6

directed edges

DFS: properties

Proposition

- DFS marks all vertices reachable from s in $\Theta(E + V)$ time in the worst case

Proof

- Initializing an array of length v takes $\Theta(V)$ time
- Each vertex is visited at most once
- Visiting a vertex takes time proportional to its outdegree:

$$outdegree(v_0) + outdegree(v_1) + outdegree(v_2) + \cdots = E$$

$$in\ worst\ case,\ all\ V\ vertices\ reachable\ from\ s$$

Note

- If all vertices are reachable from s, then $E \geq V - 1$, so V is a lower-order term

Reachability application: program control-flow analysis

Ever program is a digraph

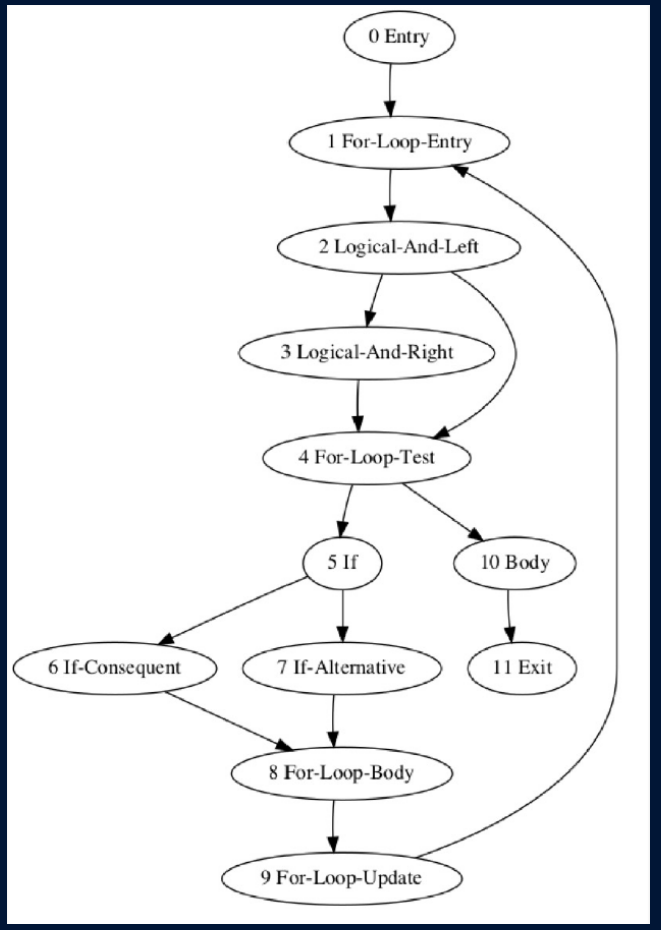
- Vertex = basic block of instructions (straight-line program)
- Edge = jump

Dead-code elimination

- Find (and remove) unreachable code

Infinite-loop detection

- Determine whether exit is unreachable



Reachability application: mark-sweep garbage collection

Ever program is a digraph

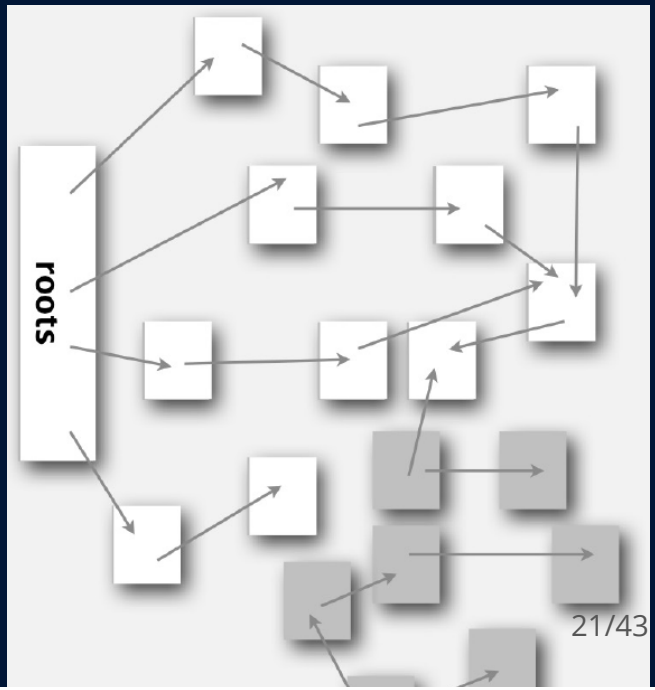
- Vertex = basic block of instructions (straight-line program)
- Edge = jump

Roots

- Objects known to be directly accessible by a program (eg stack frame)

Reachable objects

- Objects indirectly accessible by program (starting at a root and following a chain of pointers)



Mark-sweep algorithm

PATH FINDING

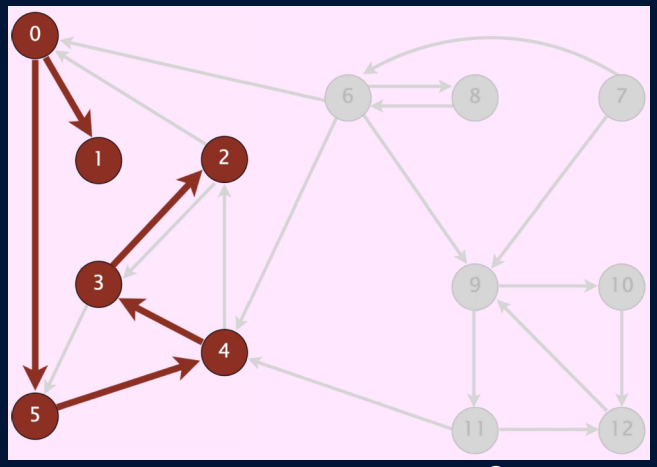
Directed paths DFS

Goal

- DFS determines which vertices are reachable from s. How to reconstruct paths?

Solution

- Use parent-link representation



reachable from 0

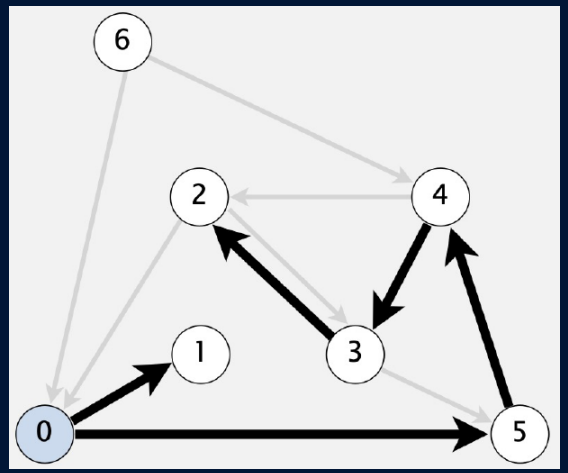
v	marked[]	edgeTo[]
0	T	–
1	T	0
2	T	3
3	T	4
4	T	5
5	T	0
6	F	–
7	F	–
8	F	–
9	F	–
10	F	–
11	F	–
12	F	–

parent-link representation of paths from vertex 0

DFS: path finding

Parent-link representation of paths from s

- Maintain an integer array $edgeTo[]$
- Interpretation: $edgeTo[v]$ is the next-to-last vertex on a path from s to v
- To reconstruct path from s to v, trace $edgeTo[]$ backward from s to v (and reverse)



v	marked[]	edgeTo[]
0	T	–
1	T	0
2	T	3
3	T	4
4	T	5
5	T	0
6	F	–

```
public Iterable<Integer> pathTo(int v) {
  if (!marked[v]) return null;
  Stack<Integer> path = new Stack<>();
  for (int x = v; x != s; x = edgeTo[x])
    path.push(x);
  path.push(s);
  return path;
}
```

UNDIRECTED GRAPHS

Flood Fill

Problem

- Implement flood fill (Photoshop magic wand)

magic wand tool

DFS: undirected graphs

Problem

- Given an undirected graph G and vertex s, find all vertices connected to s

Solution

- Treat undirected graph as a digraph, replacing each edge with two antiparallel edges

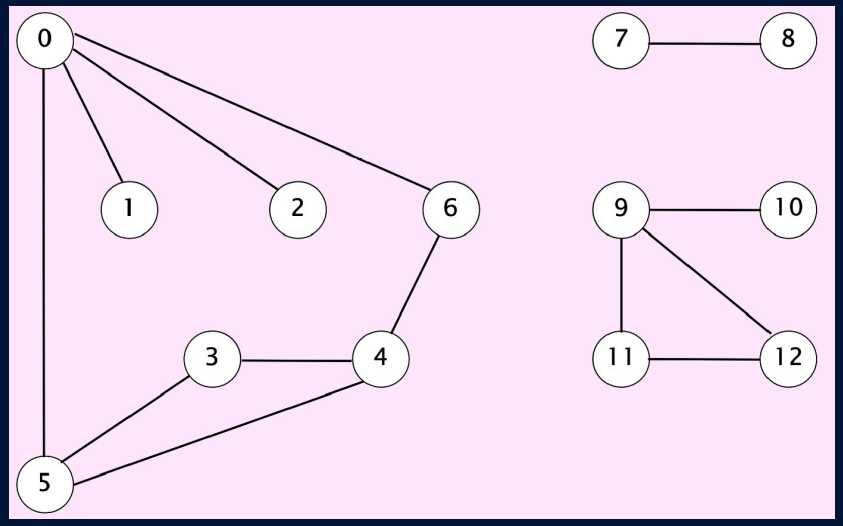
```
DFS (to visit a vertex v)
-------------------------
Mark vertex v.
Recursively visit all unmarked
    vertices w adjacent to v
-------------------------
```

Typical applications

- Find all vertices connected to a given vertex
- Find a path between two vertices

DFS: undirected search

To visit vertex v:
- Mark vertex v
- Recursively visit all unmarked vertices adjacent from v



graph g

tinyG.txt

V → 13
E → 13
0 5
4 3
0 1
9 12
6 4
5 4
0 2
11 12
9 10
0 6
7 8
9 11
5 3

connected edges

Graph search

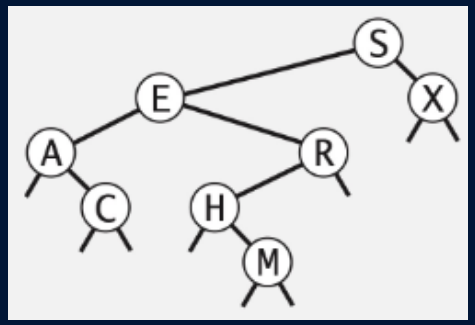
Tree traversal Many ways in a binary tree

stack/recursion

- Inorder | A C E H R S X |
- Preorder | S E A C R H M X |
- Postorder | C A M H R E X S |

queue

- Level-order | S E X A R C H M |



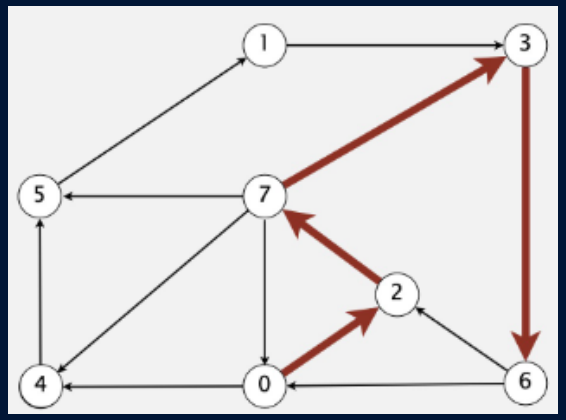
Graph search Many ways to explore a graph

- DFS preorder: vertices in order of calls to $dfs(G, v)$
- DFS postorder: vertices in order of return calls from $dfs(G, v)$

BREADTH-FIRST SEARCH (in digraphs)

Shortest paths in a digraph

Problem Find directed path from s to each other vertex that uses the fewest edges



directed paths from 0 to 6

$$0 \to 2 \to 7 \to 4 \to 5 \to 1 \to 3 \to 6$$

$$0 \to 4 \to 5 \to 1 \to 3 \to 6$$

$$0 \to 2 \to 7 \to 3 \to 6$$

$$0 \to 2 \to 7 \to 0 \to 2 \to 7 \to 3 \to 6$$

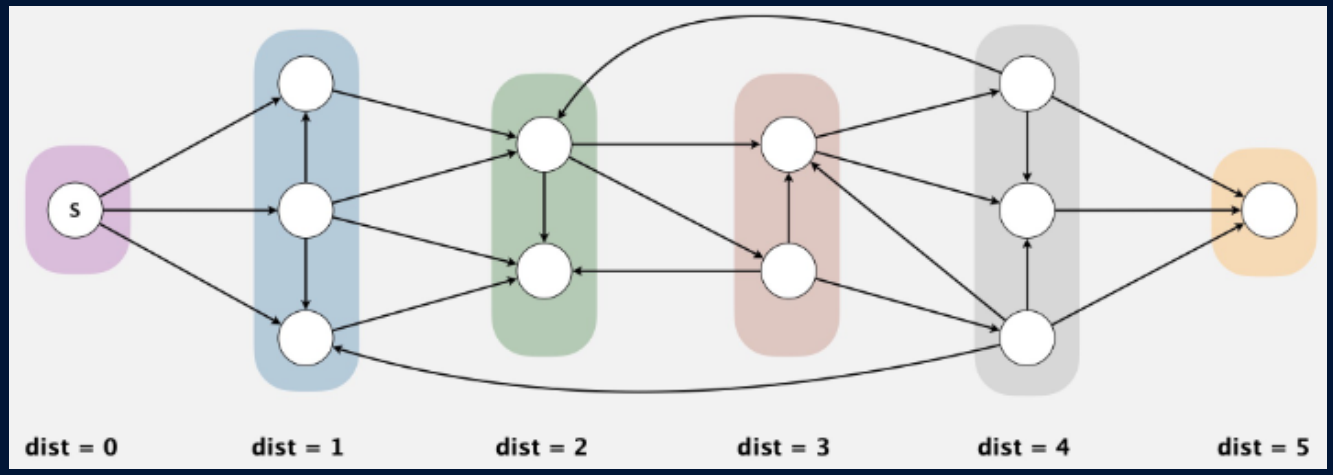
shortest path from 0 to $6 (length = 4)$

$$0 \to 2 \to 7 \to 3 \to 6$$

Shortest paths in a digraph

Problem Find directed path from s to each other vertex that uses the fewest edges

Key idea Visit vertices in increasing order of distance from s

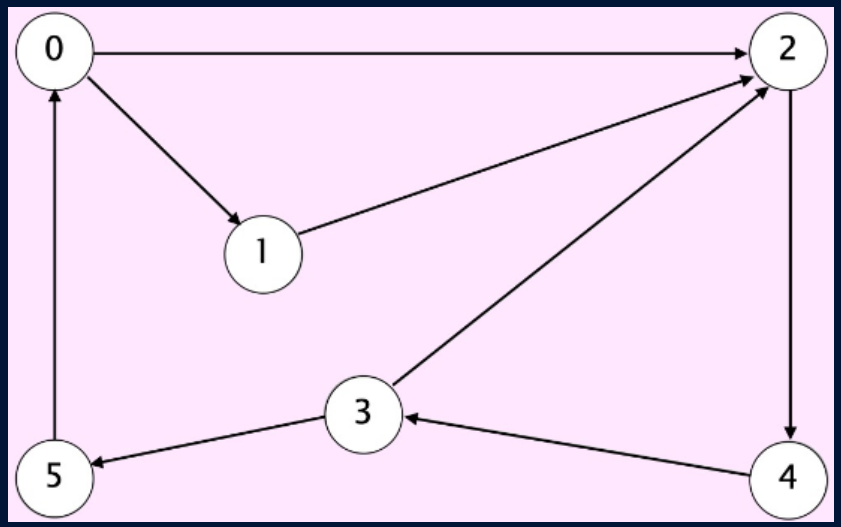


Key data structure Queue of vertices to visit

Breadth-first search

Repeat until queue is empty:
- Remove vertex v from queue
- Add to queue all unmarked vertices adjacent from v and mark them

---



graph g

tinyDG2.txt

V → 6 ← E

8

5 0

2 4

3 2

1 2

0 1

4 3

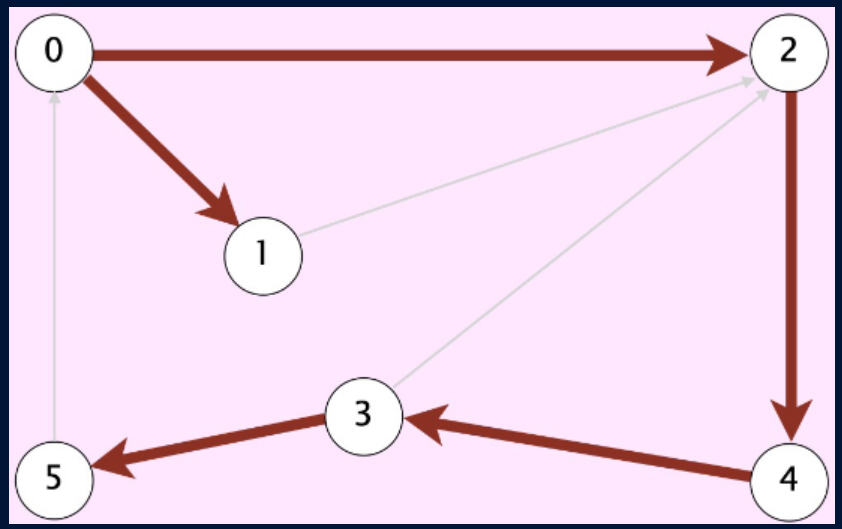
3 5

0 2

Breadth-first search, con't

Repeat until queue is empty:
- Remove vertex v from queue
- Add to queue all unmarked vertices adjacent from v and mark them

---



vertices reachable from 0
(and the shortest directed paths)

v	$edgeTo[]$	$marked[]$	$distTo[]$
0	-	T	0
1	0	T	1
2	0	T	1
4	2	T	2
3	4	T	3
5	3	T	4

Breadth-first search, con't

Repeat until queue is empty:
- Remove vertex v from queue
- Add to queue all unmarked vertices adjacent from v and mark them

---

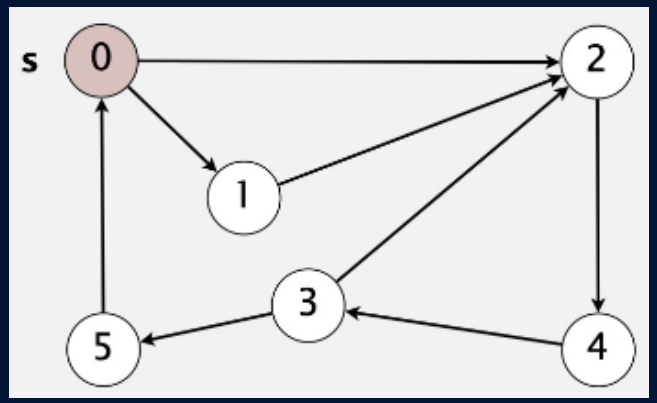
```
BFS (from source s)

------------------------------------------------

Add s to FIFO queue and mark s

Repeat until the queue is marked empty:

- remove the least recently added vertex

- for each unmarked vertex w adjacent from v:

  - add w to queue and mark w
```

Breadth-first search properties

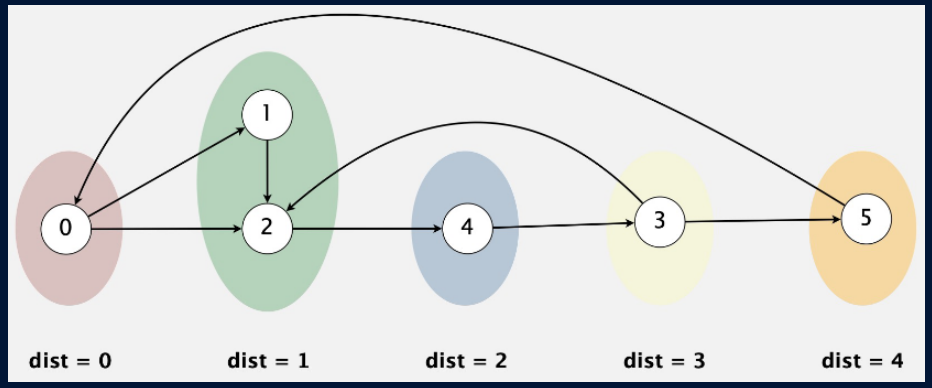
Proposition In the worst case, BFS takes $\Theta(E + V)$ time
Proof Each vertex reachable from s is visited once

Proposition BFS computes shortest paths from s
Proof BFS examines vertices in increasing distance (number of edges) from s

---



digraph G

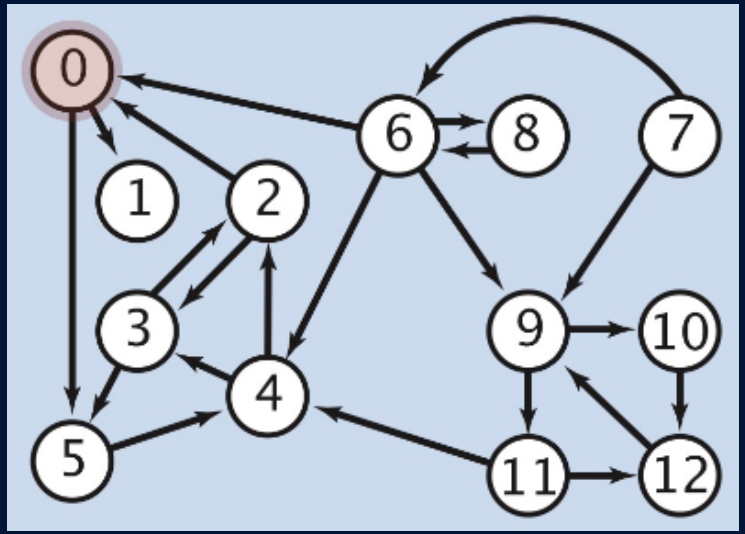


Single-sink Shortest Paths

Given a digraph and a target vertex t, find shortest path from every vertex to t.

Example $t = 0$

- Shortest path from 7
  - is $7 \to 6 \to 0$
- Shortest path from 5
  - is $5 \to 4 \to 2 \to 0$
- Shortest path from 12
  - is $12 \to 9 \to 11 \to 4 \to 2 \to 0$

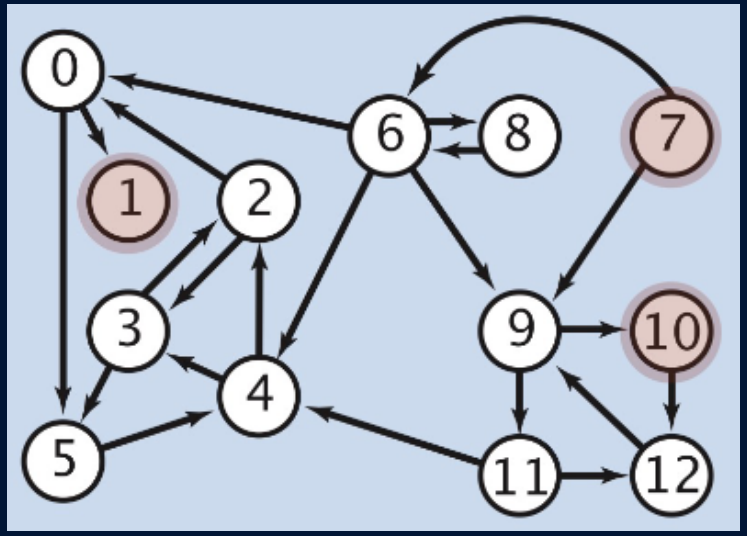


Multiple-source Shortest Paths

Given a digraph and a set of source vertices, find shortest path from any vertex in the set to every other vertex

Example $S = \{1, 7, 10\}$

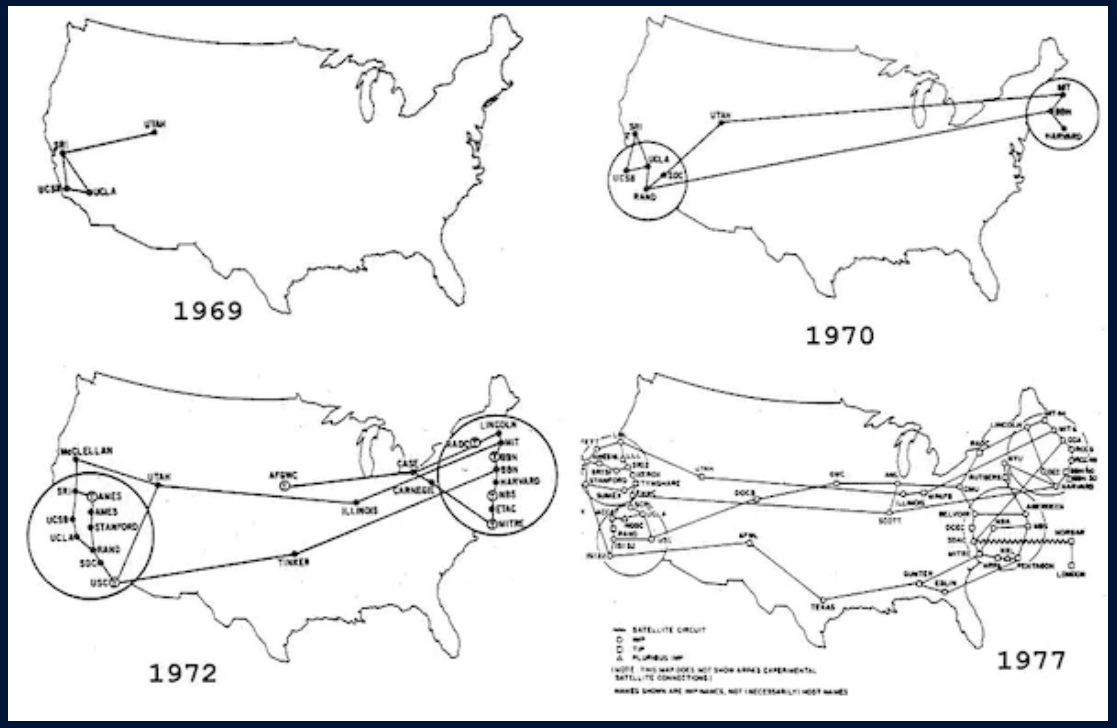
- Shortest path to 4
  - is $7 \to 6 \to 4$
- Shortest path to 5
  - is $7 \to 6 \to 0 \to 5$
- Shortest path to 12
  - is $10 \to 12$



BREADTH-FIRST SEARCH (in graphs)

Breadth-first search application: routing

Fewest number of hops in a communication network



BFS: undirected graphs

Problem Find path between s and each other vertex that uses fewest edges
Solution Treat as a digraph, replacing each undirected edge with two antiparallel edges

---

```
BFS (from source s)
-----------------------------------------------
Add s to FIFO queue and mark s
Repeat until the queue is empty:
- remove the least recently added vertex v
- for each unmarked vertex w adjacent from v:
  - addw to queue and mark w
```

SUMMARY

Graph traversal

BFS and DFS enables efficient solution to many (but not all) graph and digraph problems

graph problem	BFS	DFS	time
s-t path	✔	✔	E + V
shortest s-t path	✔		E + V
shortest directed cycle	✔		E V
Euler cycle		✔	E + V
Hamilton cycle			$2^{1.657V}$
bipartiteness	✔	✔	E + V
connected components	✔	✔	E + V
strong conmponents		✔	E + V
planarity		✔	E + V
graph morphism			$2^{c\ ln^3\ V}$