

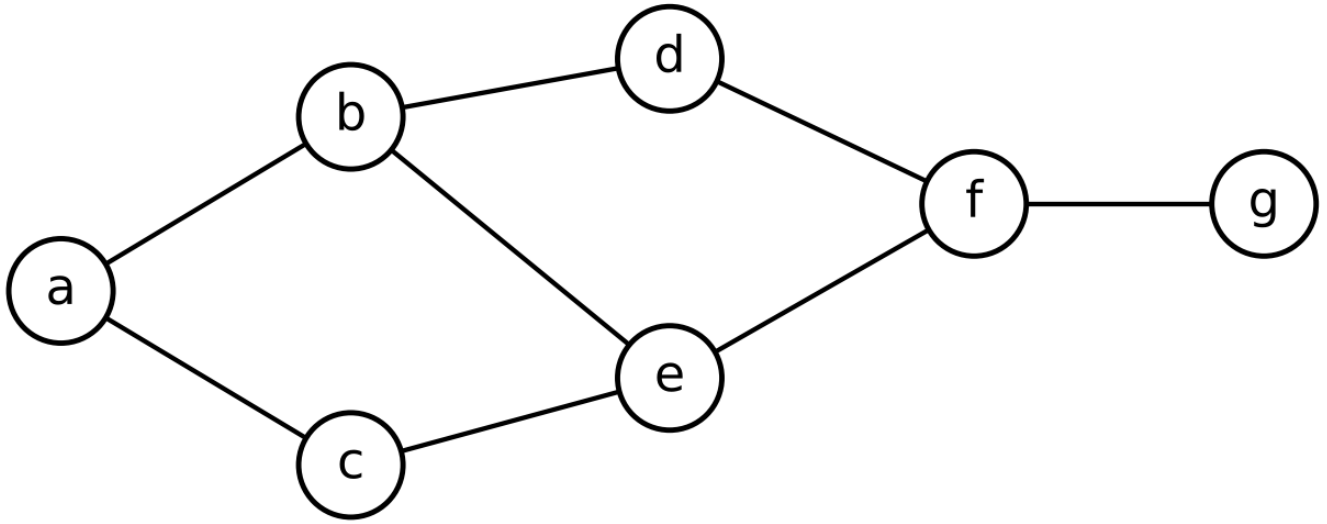
Course Algorithms and Data Structures · Lecture 13

Graphs: Representation and Breadth-First Search

Textbook: A. Levitin, Sections 1.4, 3.5

Graphs are everywhere

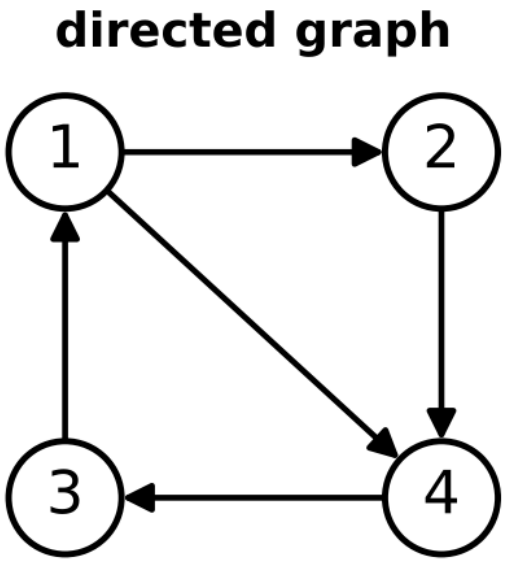
Road maps, social networks, the internet, course prerequisites. A graph G = ⟨V, E⟩ is a set of **vertices** V connected by **edges** E.

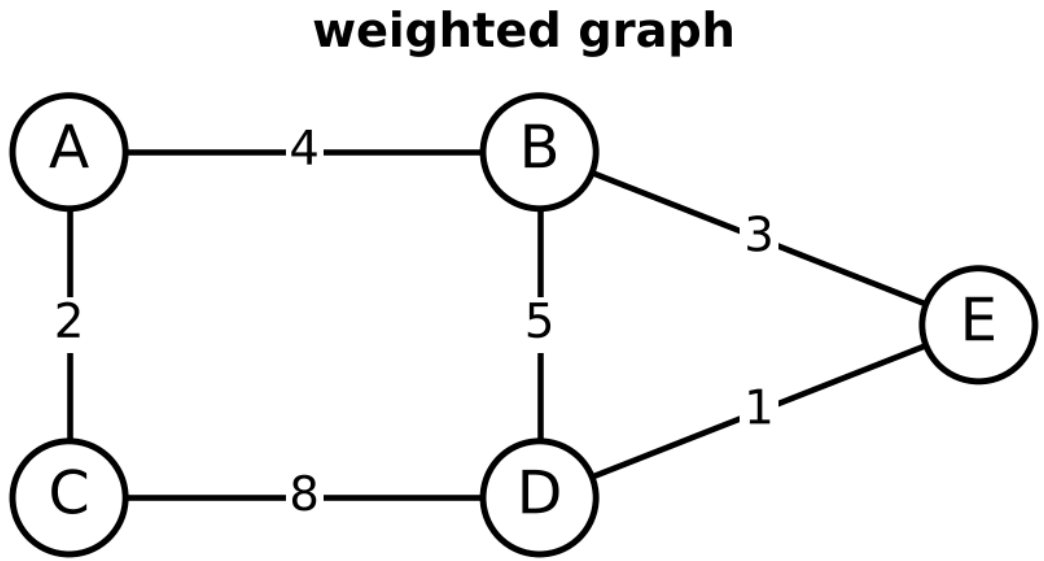


The vocabulary

Term	Meaning	In our graph
Adjacent	connected by an edge	a and b
Degree	number of edges at a vertex	deg(e) = 3
Path	a chain of adjacent vertices	a, b, e, f, g
Cycle	a path back to its start	b, d, f, e, b
Connected	a path between any two vertices	yes

Directed and weighted graphs





- **Directed**: edges have a direction (one-way streets, following someone)
- **Weighted**: each edge has a number (distance, price, time)

Adjacency matrix

	a	b	c	d	e	f	g
a	0	1	1	0	0	0	0
b	1	0	0	1	1	0	0
c	1	0	0	0	1	0	0
d	0	1	0	0	0	1	0
e	0	1	1	0	0	1	0
f	0	0	0	1	1	0	1
g	0	0	0	0	0	1	0

- A[i, j] = 1 if there is an edge
- Undirected: symmetric
- Weighted: store the weight
- Memory: $O(V^2)$

Adjacency lists

Vertex	Neighbors
a	b, c
b	a, d, e
c	a, e
d	b, f
e	b, c, f
f	d, e, g
g	f

- For every vertex, a linked list of its neighbors
- Each edge appears twice: 2E entries
- Memory: O(V + E)

Matrix or lists?

	Adjacency matrix	Adjacency lists
Memory	$O(V^2)$	$O(V + E)$
Is there an edge u–v?	$O(1)$	O(degree of u)
All neighbors of v	$O(V)$	O(degree of v)
Good for	dense graphs	sparse graphs

Real graphs are sparse: billions of users, a few hundred friends each. We use lists. (V, E = numbers of vertices and edges.)

Breadth-first search (BFS)

Like waves from a stone in water: first the start vertex, then all its neighbors, then their neighbors... layer by layer.

A queue

vertices found first are processed first: FIFO

Marking

a graph can have cycles: mark every vertex you have seen

BFS in pseudocode

```
ALGORITHM BFS(G, s)
    mark every vertex as not visited
    mark s as visited;  dist[s] ← 0
    create a queue containing s
    while the queue is not empty do
        v ← dequeue()
        for each vertex w adjacent to v do
            if w is not visited
                mark w as visited
                dist[w] ← dist[v] + 1
                enqueue(w)
```

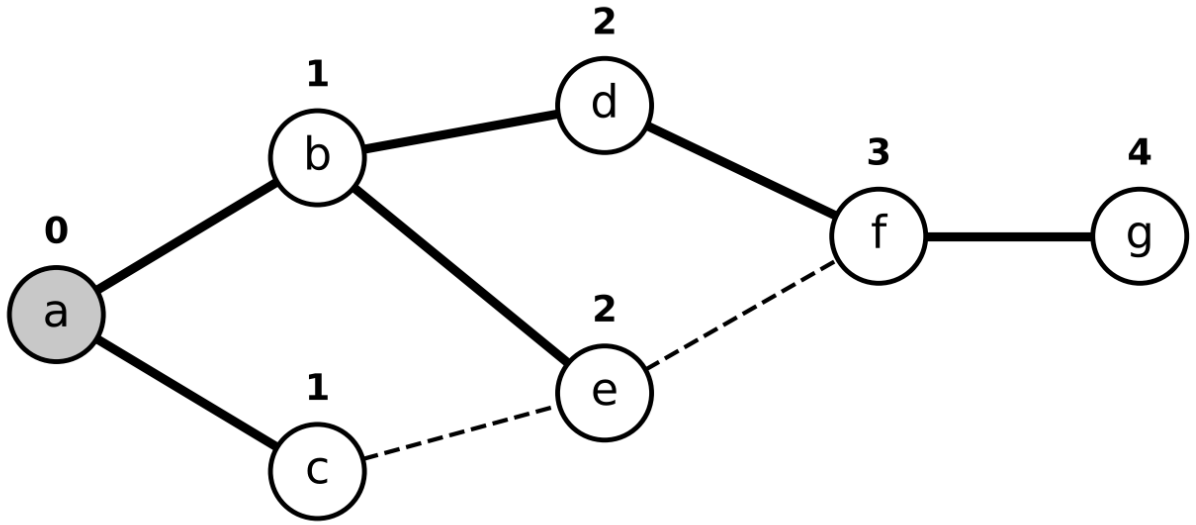
Mark a vertex when it enters the queue, so it never enters twice.

Tracing BFS from a

Step	Dequeued	Enqueued	Queue after
start		a	a
1	a	b, c	b, c
2	b	d, e	c, d, e
3	c	none	d, e
4	d	f	e, f
5	e	none	f
6	f	g	g
7	g	none	empty

Visited in the order a, b, c, d, e, f, g.

The BFS tree and shortest paths



- Thick: tree edges; dashed: cross edges
- Numbers: distance from a
- dist[g] = 4: a, b, d, f, g is a shortest path

BFS finds shortest paths by **number of edges**. For weights you need other algorithms (Dijkstra).

How fast is BFS?

O(V + E) with adjacency lists

- Every vertex enters and leaves the queue once: O(V)
- Every adjacency list is read once: O(E)
- With a matrix: $O(V^2)$
- Not connected? Start BFS again from an unvisited vertex

Summary

- A graph is a set of vertices connected by edges: directed or not, weighted or not.
- Adjacency matrix: $O(V^2)$ memory; adjacency lists: $O(V + E)$, better for sparse graphs.
- BFS visits vertices layer by layer with a queue, marking each when it is discovered.
- BFS finds shortest paths by number of edges in $O(V + E)$.

Review questions

1. What is the difference between a directed and an undirected graph? Give a real-life example of each.
2. Write the adjacency matrix and the adjacency lists of the directed graph from the lecture.
3. Run BFS on the example graph starting from g. In what order are the vertices visited, and what is the distance from g to a?

Reading: Levitin, Sections 1.4, 3.5; Bhargava, Chapter 6.