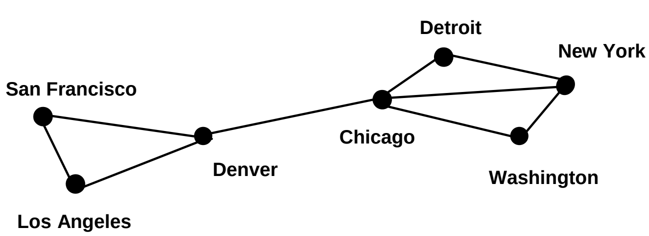
Analysis of Algorithms

Graph Algorithms

Simple Graph

Definition 1. A **simple graph** *G = (V, E)* consists of *V*, a nonempty set of **vertices**, and *E*, a set of unordered pairs of distinct elements of *V* called **edges**.

A simple graph



How many vertices? How many edges?

A simple graph

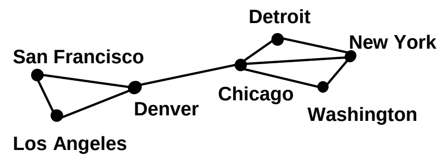
SET OF VERTICES

V = { Chicago, Denver, Detroit, Los Angeles, New York, San Francisco, Washington }

SET OF EDGES

E = { {San Francisco, Los Angeles}, {San Francisco, Denver}, {Los Angeles, Denver}, {Denver, Chicago}, {Chicago, Detroit}, {Detroit, New York}, {New York, Washington}, {Chicago, Washington}, {Chicago, New York} }

A simple graph



The network is made up of computers and telephone lines between computers. There is at most 1 telephone line between 2 computers in the network. Each line operates in both directions. No computer has a telephone line to itself.

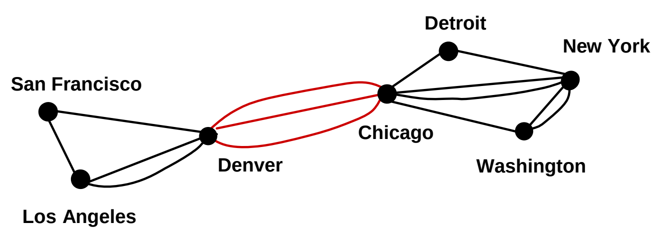
These are undirected edges,
each of which connects two distinct vertices, and
no two edges connect the same pair of vertices.

A Non-Simple Graph

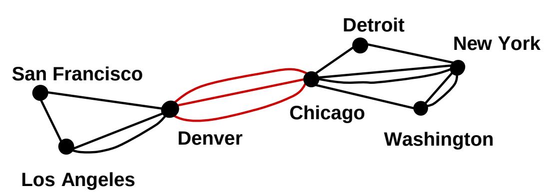
Definition 2. In a **multigraph** *G = (V, E)* two or more edges may connect the same pair of vertices.

A Multigraph

THERE CAN BE MULTIPLE TELEPHONE LINES BETWEEN TWO COMPUTERS IN THE NETWORK.



Multiple Edges



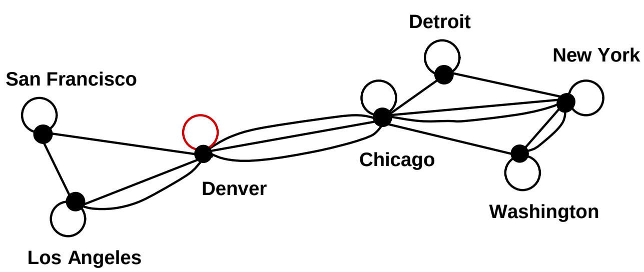
Two edges are called *multiple or parallel edges* if they connect the same two distinct vertices.

Another Non-Simple Graph

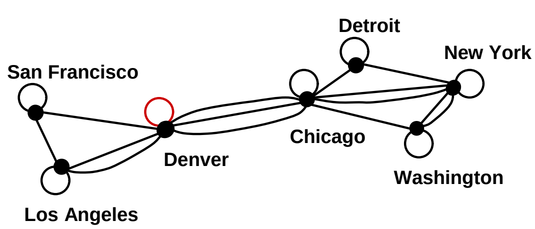
Definition 3. In a **pseudograph** *G = (V, E)* two or more edges may connect the same pair of vertices, and in addition, an edge may connect a vertex to itself.

A Pseudograph

THERE CAN BE TELEPHONE LINES IN THE NETWORK
FROM A COMPUTER TO ITSELF (for diagnostic use).

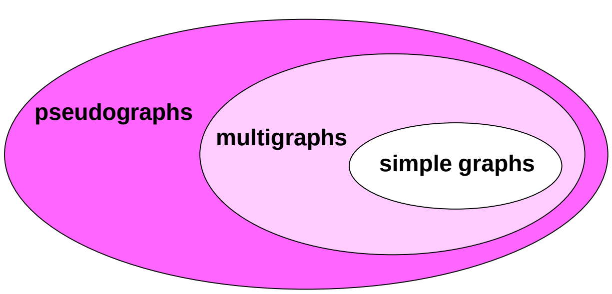


Loops



An edge is called a *loop*
if it connects a vertex to itself.

Undirected Graphs

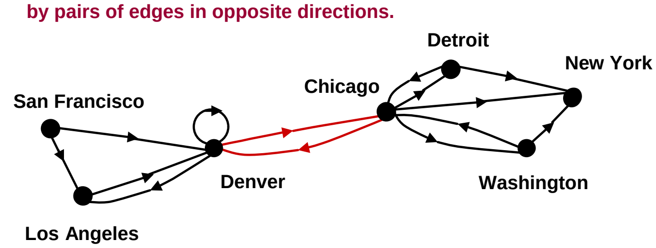


A Directed Graph

Definition 4. In a **directed graph** *G = (V, E)* the edges are ordered pairs of (not necessarily distinct) vertices.

A Directed Graph

SOME TELEPHONE LINES IN THE NETWORK
MAY OPERATE IN ONLY ONE DIRECTION .
Those that operate in two directions are represented
by pairs of edges in opposite directions.

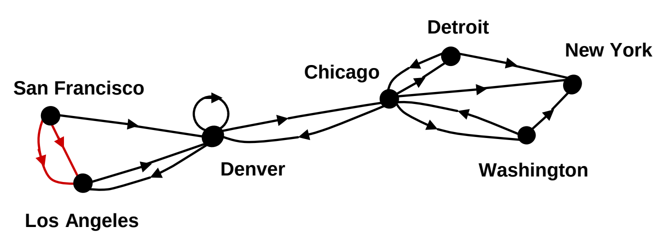


A Directed Multigraph

Definition 5. In a **directed multigraph** *G = (V, E)* the edges are ordered pairs of (not necessarily distinct) vertices, and in addition there may be multiple edges.

A Directed Multigraph

THERE MAY BE SEVERAL ONE-WAY LINES
IN THE SAME DIRECTION FROM ONE COMPUTER
TO ANOTHER IN THE NETWORK.



Types of Graphs

TYPE	EDGES	MULTIPLE EDGES ALLOWED?	LOOPS ALLOWED?
Simple graph	Undirected	NO	NO
Multigraph	Undirected	YES	NO
Pseudograph	Undirected	YES	YES
Directed graph	Directed	NO	YES
Directed multigraph	Directed	YES	YES

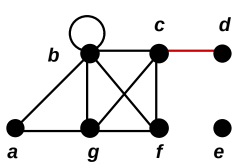
Adjacent Vertices (Neighbors)

Definition 1. Two vertices, *u* and *v* in an undirected graph G are called adjacent (or neighbors) in G, if {*u*, *v*} is an edge of G.

An edge *e* connecting *u* and *v* is called incident with vertices *u* and *v*, or is said to connect *u* and *v*. The vertices *u* and *v* are called endpoints of edge {*u*, *v*}.

Degree of a vertex

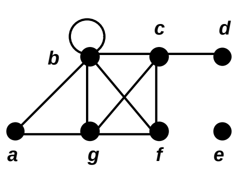
Definition 1. The degree of a vertex in an undirected graph is the number of edges incident with it, except that a loop at a vertex contributes twice to the degree of that vertex.



deg(*d*) = 1

Degree of a vertex

Definition 1. The degree of a vertex in an undirected graph is the number of edges incident with it, except that a loop at a vertex contributes twice to the degree of that vertex.

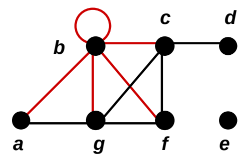


deg(*e*) = 0

Degree of a vertex

Definition 1. The degree of a vertex in an undirected graph is the number of edges incident with it, except that a loop at a vertex contributes twice to the degree of that vertex.

deg(*b*) = 6

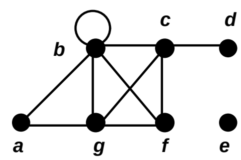


Degree of a vertex

Find the degree of all the other vertices.

deg(*a*) deg(c) deg(f) deg(*g*)

deg(*b*) = 6



deg(*d*) = 1

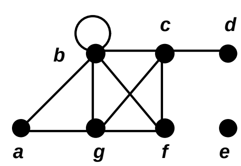
deg(*e*) = 0

Degree of a vertex

Find the degree of all the other vertices.

deg(*a*) = 2 deg(c) = 4 deg(f) = 3 deg(*g*) = 4

deg(*b*) = 6



deg(*d*) = 1

deg(*e*) = 0

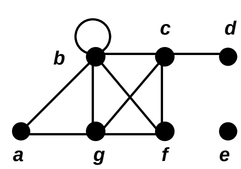
Degree of a vertex

Find the degree of all the other vertices.

deg(*a*) = 2 deg(c) = 4 deg(f) = 3 deg(*g*) = 4

TOTAL of degrees = 2 + 4 + 3 + 4 + 6 + 1 + 0 = 20

deg(*b*) = 6



deg(*d*) = 1

deg(*e*) = 0

Degree of a vertex

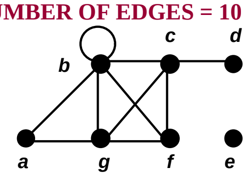
Find the degree of all the other vertices.

deg(a) = 2   deg(c) = 4   deg(f) = 3   deg(g) = 4

TOTAL of degrees = 2 + 4 + 3 + 4 + 6 + 1 + 0 = 20

TOTAL NUMBER OF EDGES = 10

deg(b) = 6



deg(d) = 1

deg(e) = 0

Handshaking Theorem

Theorem 1. **Let $G = (V, E)$ be an undirected graph G with e edges. Then**

$$\sum_{v \in V} \deg(v) = 2e$$

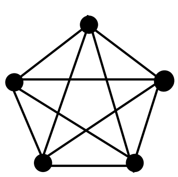
"The sum of the degrees over all the vertices equals twice the number of edges."

NOTE: This applies even if multiple edges and loops are present.

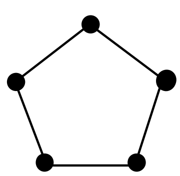
Subgraph

Definition 6. **A subgraph of a graph $G = (V, E)$ is a graph $H = (W, F)$ where $W \subseteq V$ and $F \subseteq E$.**

C_5 is a subgraph of K_5



K_5

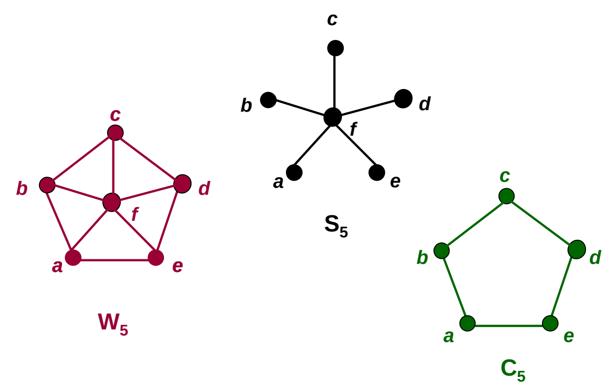


C_5

Union

Definition 7. **The union of 2 simple graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ is the simple graph with vertex set $V = V_1 \cup V_2$ and edge set $E = E_1 \cup E_2$. The union is denoted by $G_1 \cup G_2$.**

W_5 is the union of S_5 and C_5

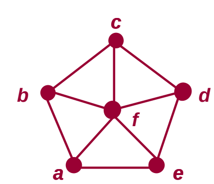


Adjacency Matrix

A simple graph $G = (V, E)$ with n vertices can be represented by its adjacency matrix, A, where entry a_{ij} in row i and column j is

$$a_{ij} = \begin{cases} 1 & \text{if } \{v_i, v_j\} \text{ is an edge in G,} \\ 0 & \text{otherwise.} \end{cases}$$

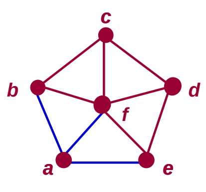
Finding the adjacency matrix



W_5

This graph has 6 vertices a, b, c, d, e, f. We can arrange them in that order.

Finding the adjacency matrix

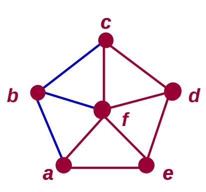


W_5

FROM \ TO	a	b	c	d	e	f
a	0	1	0	0	1	1
b						
c						
d						
e						
f						

There are edges from a to b, from a to e, and from a to f

Finding the adjacency matrix

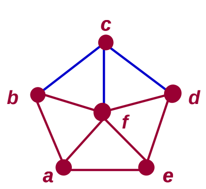


W_5

FROM \ TO	a	b	c	d	e	f
a	0	1	0	0	1	1
b	1	0	1	0	0	1
c						
d						
e						
f						

There are edges from b to a, from b to c, and from b to f

Finding the adjacency matrix

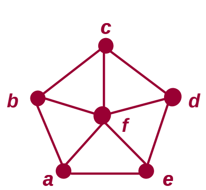


W_5

FROM \ TO	a	b	c	d	e	f
a	0	1	0	0	1	1
b	1	0	1	0	0	1
c	0	1	0	1	0	1
d						
e						
f						

There are edges from c to b, from c to d, and from c to f

Finding the adjacency matrix

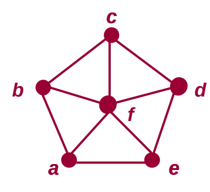


W_5

FROM \ TO	a	b	c	d	e	f
a	0	1	0	0	1	1
b	1	0	1	0	0	1
c	0	1	0	1	0	1
d						
e						
f						

COMPLETE THE ADJACENCY MATRIX . . .

Finding the adjacency matrix



W_5

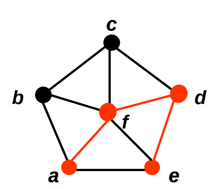
FROM \ TO	a	b	c	d	e	f
a	0	1	0	0	1	1
b	1	0	1	0	0	1
c	0	1	0	1	0	1
d	0	0	1	0	1	1
e	1	0	0	1	0	1
f	1	1	1	1	1	0

Notice that this matrix is symmetric. That is $a_{ij} = a_{ji}$ Why?

Path of Length n

Definition 1. A path of length n from u to v in an undirected graph is a sequence of edges $e_1, e_2, \ldots, e_n$ of the graph such that
edge e_1 has endpoints x_0 and x_1,
edge e_2 has endpoints x_1 and x_2,
. . .
and edge e_n has endpoints x_{n-1} and x_n,
where $x_0 = u$ and $x_n = v$.

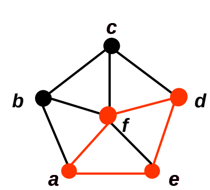
One path from *a* to *e*



W_5

This path passes through vertices f and d in that order.

One path from *a* to *a*



W_5

This path passes through vertices f, d, e, in that order. It has length 4.

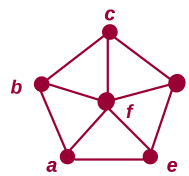
It is a circuit because it begins and ends at the same vertex.

It is called simple because it does not contain the same edge more than once.

Path of Length n

Definition 3. An undirected graph is called connected if there is a path between every pair of distinct vertices of the graph.

IS THIS GRAPH CONNECTED?



W_5

Representing Graphs

- Assume V = {1, 2, …, *n*}
- An *adjacency matrix* represents the graph as a *n* x *n* matrix A:
  - A[*i*, *j*] = 1 if edge (*i*, *j*) ∈ E (or weight of edge)
    = 0 if edge (*i*, *j*) ∉ E
  - Storage requirements: $O(V^2)$
    - A dense representation
  - But, can be very efficient for small graphs
    - Especially if store just one bit/edge
    - Undirected graph: only need one diagonal of matrix

Finding Shortest Paths

- Given an undirected graph and source vertex *s*, the length of a path in a graph (without edge weights) is the number of edges on the path. Find the shortest path from *s* to each other vertex in the graph.
- Brute-Force: enumerate all simple paths starting from *s* and keep track of the shortest path arriving at each vertex. There may be *n*! simple paths in a graph…

Breadth-First-Search (BFS)

- Given:
  - *G* = (*V, E*)
  - A distinguished *source* vertex
- Systematically explores the edges of *G* to discover every vertex that is reachable from *s*
  - Computes (shortest) distance from *s* to all reachable vertices
  - Produces a *breadth-first-tree* with root *s* that contains all reachable vertices

Breadth-First-Search (BFS)

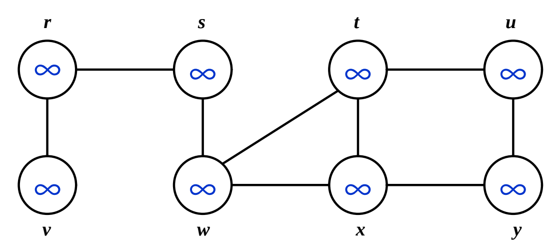
- BFS colors each vertex:
  - white -- undiscovered
  - gray -- discovered but "not done yet"
  - black -- all adjacent vertices have been discovered


```
BFS(G,s)
1.  for each vertex u in (V[G] \ {s})
        do color[u] ¬ white
           d[u] ¬ µ
           p[u] ¬ nil
    color[s] ¬ gray
    d[s] ¬ 0
    p[s] ¬ nil
    Q ¬ F
    enqueue(Q,s)
    while Q ¹ F
        do u ¬ dequeue(Q)
           for each v in Adj[u]
               do if color[v] = white
                      then color[v] ¬ gray
                           d[v] ¬ d[u] + 1
                           p[v] ¬ u
                           enqueue(Q,v)
           color[u] ¬ black
```

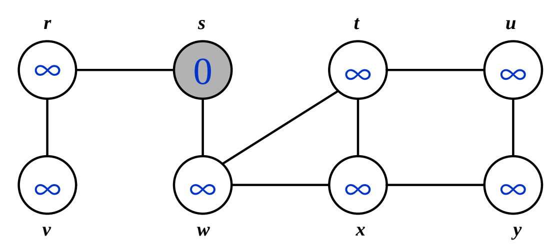
white: undiscovered
gray: discovered
black: finished

Q: a queue of discovered vertices
color[v]: color of v
d[v]: distance from s to v
π[u]: predecessor of v

Breadth-First Search: Example

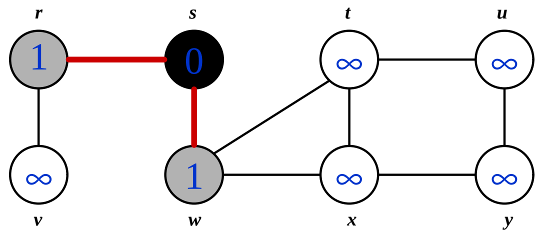


Breadth-First Search: Example



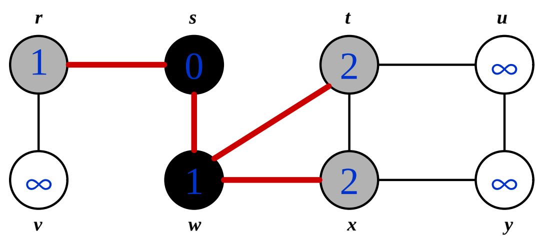
Q: *s*

Breadth-First Search: Example



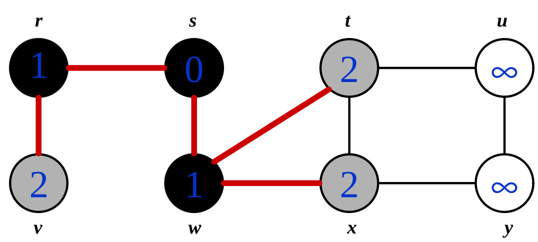
Q: | *w* | *r* |

Breadth-First Search: Example



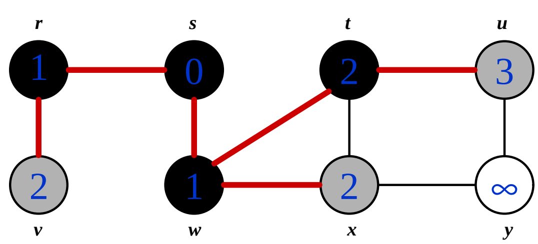
Q: | *r* | *t* | *x* |

Breadth-First Search: Example



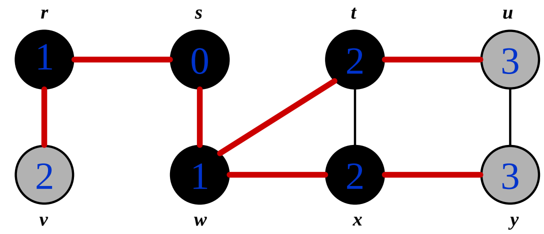
Q: | *t* | *x* | *v* |

Breadth-First Search: Example



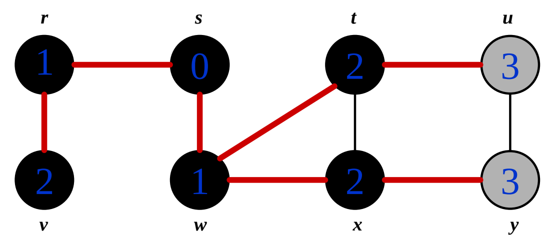
Q: | *x* | *v* | *u* |

Breadth-First Search: Example



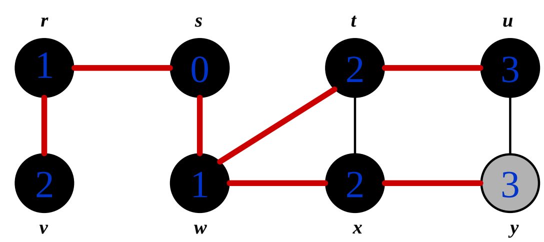
Q: | *v* | *u* | *y* |

Breadth-First Search: Example



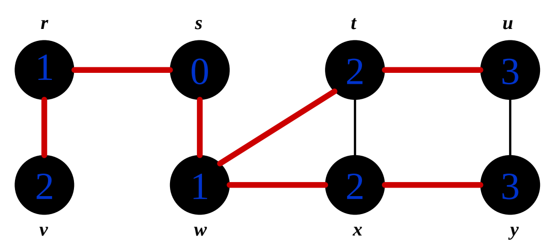
Q: | *u* | *y* |

Breadth-First Search: Example



Q: | *y* |

Breadth-First Search: Example



Q: Ø

Breadth-First Search: Properties

- BFS builds ***breadth-first tree***, in which paths to root represent shortest paths in G
  - Thus can use BFS to calculate shortest path from one vertex to another in O(V+E) time