Australian National University

COMP3600/6466 Algorithms

Lecture 22

S2 2016

Dr. Hassan Hijazi

Prof. Weifa Liang

Graph Algorithms

FUNDAMENTAL!

Users Interaction = Graph

Routing Map = Graph

Telecom Network = Graph

Supply Chain = Graph

Assets Correlation = Graph

Power System = Graph

Air traffic = Graph

...

Graph Representation

What's a Graph?

$$G = (V, E)$$

V is the set of vertices (nodes)

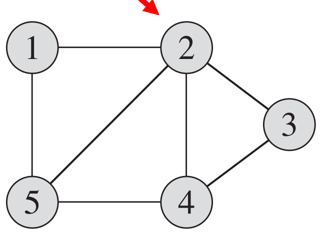
E is the set of edges (links, lines, arcs)

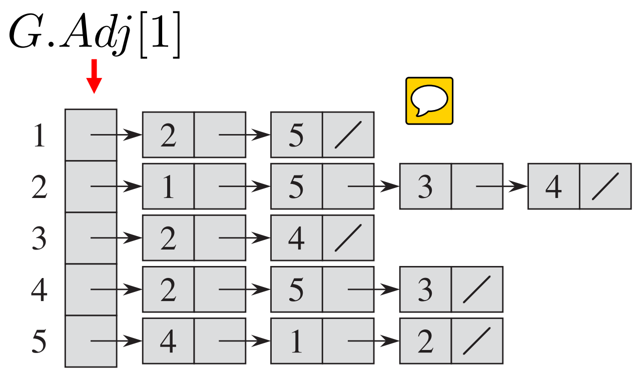
$$E = V \times V$$

Graph Representation

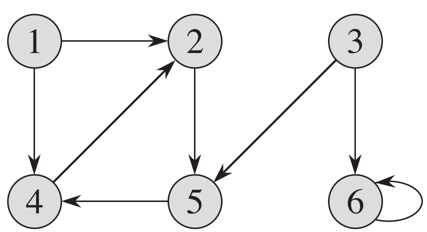
1. Adjacency-Lists

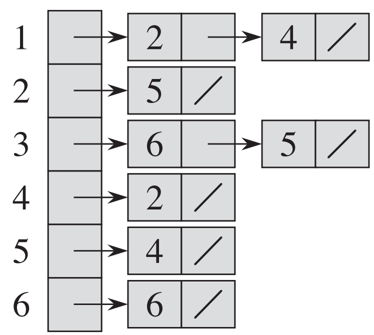
Degree(2) = 4





Undirected

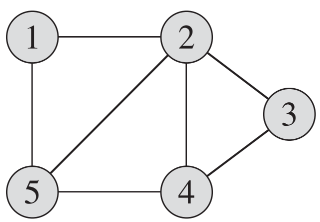




Directed

Graph Representation

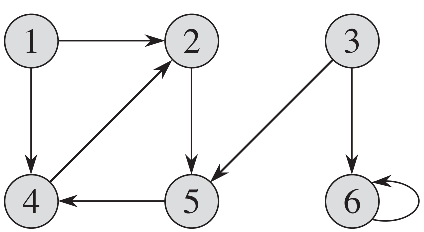
2. Adjacency-Matrix



	1	2	3	4	5
1	0	1	0	0	1
2	1	0	1	1	1
3	0	1	0	1	0
4	0	1	1	0	1
5	1	1	0	1	0

Undirected

Few non-zeros = waste of space!



	1	2	3	4	5	6
1	0	1	0	1	0	0
2	0	0	0	0	1	0
3	0	0	0	0	1	1
4	0	1	0	0	0	0
5	0	0	0	1	0	0
6	0	0	0	0	0	1

Directed

Breadth-First Search (BFS)

INPUT: $G = (V, E)$, source node $s \in V$.

OUTPUT: The shortest distance (in number of edges) from s to all its reachable nodes.

Breadth-First?

The **BFS** algorithm explores all nodes at distance **k** from **s** before discovering any nodes at distance **k+1**

Breadth-First Search (BFS)

v

$v.colour \in \{\text{white, grey, black}\}$

$v.\pi$, previous node on the path to v

$v.d$, distance from the source s

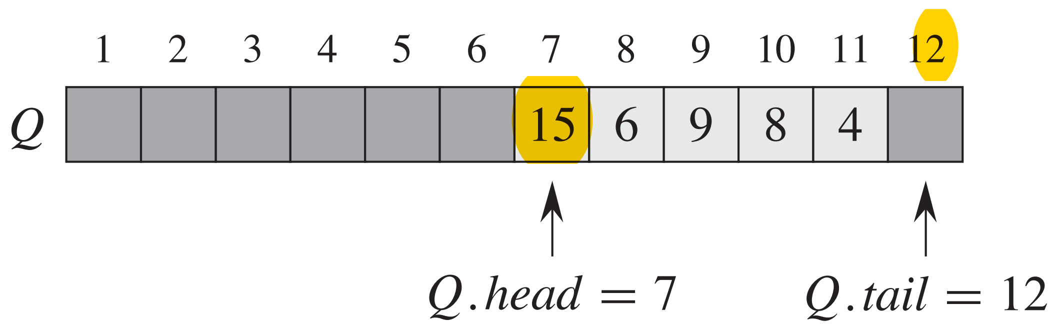
Let Q be a First-In First-Out (FIFO) queue

	1	2	3	4	5	6	7	8	9	10	11	12
Q							15	6	9	8	4	

$Q.head = 7$ $\quad$ $Q.tail = 12$

Breadth-First Search (BFS)

Let Q be a First-In First-Out (FIFO) queue

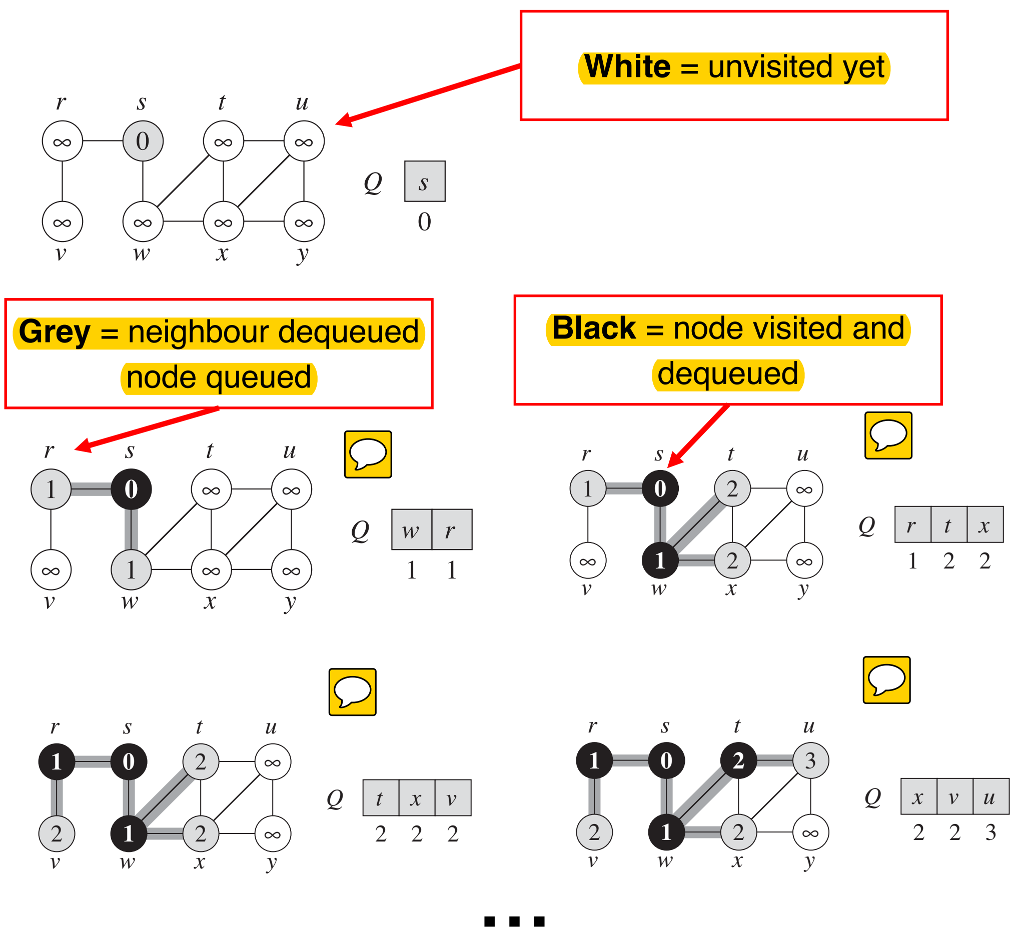


$\textsc{Enqueue}(Q, v)$: insert vertex v at the tail of Q

$\textsc{Dequeue}(Q)$: remove the vertex at the head of Q

Breadth-First Search (BFS)

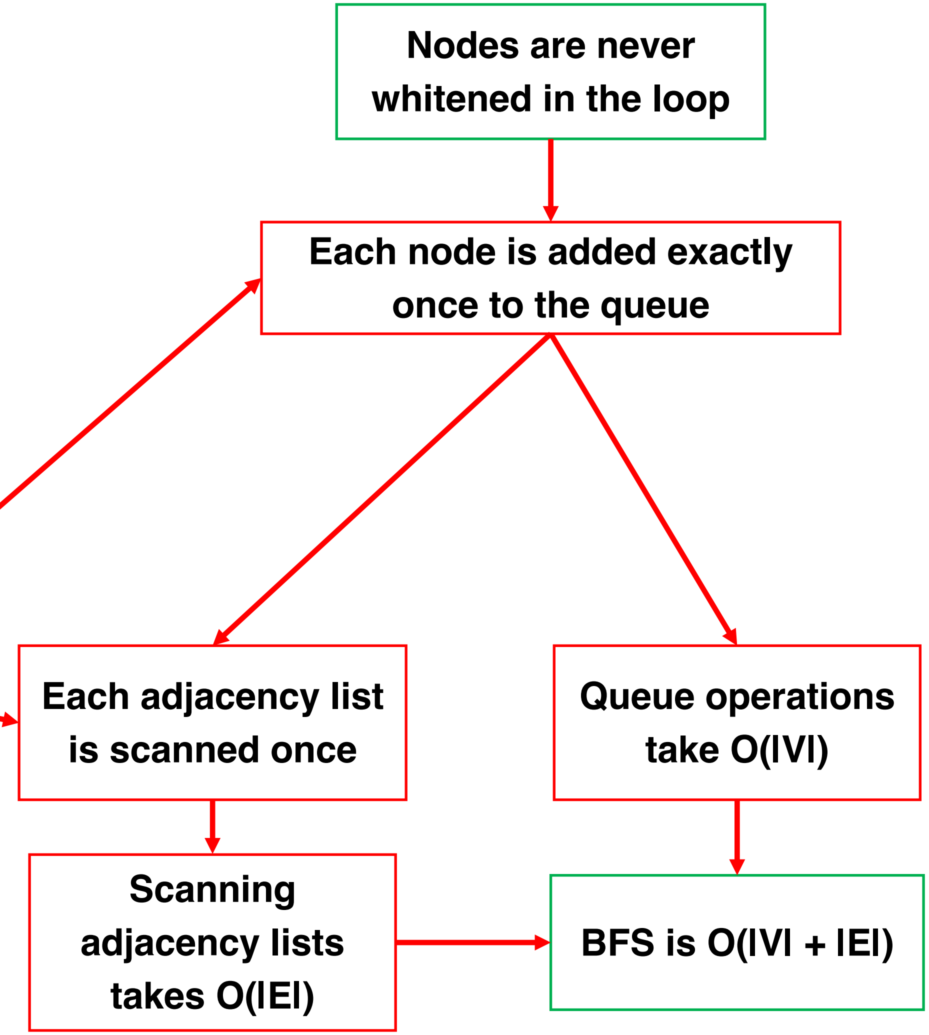
```
BFS(G, s)
for each vertex u ∈ G.V − {s}
    u.color = WHITE
    u.d = ∞
    u.π = NIL
s.color = GRAY
s.d = 0
s.π = NIL
Q = ∅
ENQUEUE(Q, s)
while Q ≠ ∅
    u = DEQUEUE(Q)
    for each v ∈ G.Adj[u]
        if v.color == WHITE
            v.color = GRAY
            v.d = u.d + 1
            v.π = u
            ENQUEUE(Q, v)
    u.color = BLACK
```



...

BFS Analysis

```
BFS(G, s)
for each vertex u ∈ G.V − {s}
    u.color = WHITE
    u.d = ∞
    u.π = NIL
s.color = GRAY
s.d = 0
s.π = NIL
Q = ∅
ENQUEUE(Q, s)
while Q ≠ ∅
    u = DEQUEUE(Q)
    for each v ∈ G.Adj[u]
        if v.color == WHITE
            v.color = GRAY
            v.d = u.d + 1
            v.π = u
            ENQUEUE(Q, v)
    u.color = BLACK
```



Shortest Paths

$\delta(s, v)$

shortest-path distance

Let $\delta(s, v)$ denote the minimum number of edges in any path from s to v. If these vertices are not connected, then $\delta(s, v) = \infty$.

We call a path of length $\delta(s, v)$ from s to v a **shortest path**.

Claim. BFS computes the shortest path from s to any reachable node in V.

$\Updownarrow$ Equivalent

Claim. Starting from s, any reachable node v in V will satisfy $v.d = \delta(s, v)$.

Shortest Paths

Lemma 22.2

Let $G = (V, E)$ be a directed or undirected graph, and suppose that BFS is run on G from a given source vertex $s \in V$. Then upon termination, for each vertex $v \in V$, the value $v.d$ computed by BFS satisfies $v.d \geq \delta(s, v)$.

Corollary 22.4

Suppose that vertices v_i and v_j are enqueued during the execution of BFS, and that v_i is enqueued before v_j. Then $v_i.d \leq v_j.d$ at the time that v_j is enqueued.

Proof by induction.
(Textbook p.598-599)

Claim. Starting from s, any reachable node v in V will satisfy $v.d = \delta(s, v)$.

Shortest Paths

Claim. Starting from s, any reachable node v in V will satisfy $v.d = \delta(s, v)$.

Proof Outline. (By Contradiction)
Let v be the vertex with minimum $\delta(s, v)$ that receives an incorrect d value, that is $v.d > \delta(s, v)$; clearly $v \neq s$.

Let u be the vertex immediately preceding v on a shortest path from s to v, so that $\delta(s, v) = \delta(s, u) + 1$.

The way u is chosen, we have $u.d = \delta(s, u)$ implying $v.d > u.d + 1$.

When BFS chooses to dequeue u from Q, we have three cases:

Case 1. v is white
Line 15 in BFS sets $v.d = u.d + 1 \rightarrow$ CONTRADICTION.

Shortest Paths

Claim. Starting from s, any reachable node v in V will satisfy $v.d = \delta(s, v)$.

The way u is chosen, we have $u.d = \delta(s, u)$ implying $v.d > u.d + 1$.

Case 2. v is black
v has been dequeued before $u \implies v.d \leq u.d$ (Corollary 22.4)
$\rightarrow$ CONTRADICTION.

Case 3. v is grey
v was painted grey by a predecessor w dequeued before u
$\implies w.d \leq u.d$ (Corollary 22.4) and $v.d = w.d + 1$ (line 15).
This leads to $v.d = w.d + 1 \leq u.d + 1 \rightarrow$ CONTRADICTION.

Shortest Paths

PRINT-PATH(G, s, v)

```
if v == s
    print s
elseif v.π == NIL
    print "no path from" s "to" v "exists"
else PRINT-PATH(G, s, v.π)
    print v
```

If **v** and **s** are connected will go all the way back to **s**

Exercise 22.1

Show that using a single bit to store each vertex color suffices for BFS.

```
BFS(G, s)
for each vertex u ∈ G.V − {s}
    u.color = WHITE
    u.d = ∞
    u.π = NIL
s.color = GRAY
s.d = 0
s.π = NIL
Q = ∅
ENQUEUE(Q, s)
while Q ≠ ∅
    u = DEQUEUE(Q)
    for each v ∈ G.Adj[u]
        if v.color == WHITE
            v.color = GRAY
            v.d = u.d + 1
            v.π = u
            ENQUEUE(Q, v)
    u.color = BLACK
```

Answer: First, note that we never use the fact that a node is coloured grey, we only check if it's white, therefore, replacing grey by black will not change the algorithm execution.

Second, note that a node coloured is a node that has a finite number stored in $v.d$, therefore, we can check if a node is white or not by checking wether $v.d = \infty$, which implies that we do not need the colour attribute at all.

Exercise 22.2

Argue that in a breadth-first search, the value $u.d$ assigned to a vertex u is independent of the order in which the vertices appear in each adjacency list.

The correctness proof for the BFS algorithm shows that $u.d = \delta(s, u)$, and the algorithm doesn't assume that the adjacency lists are in any particular order.

Exercise 22.3

Starting from vertex 3, apply the BFS algorithm on the following graph.