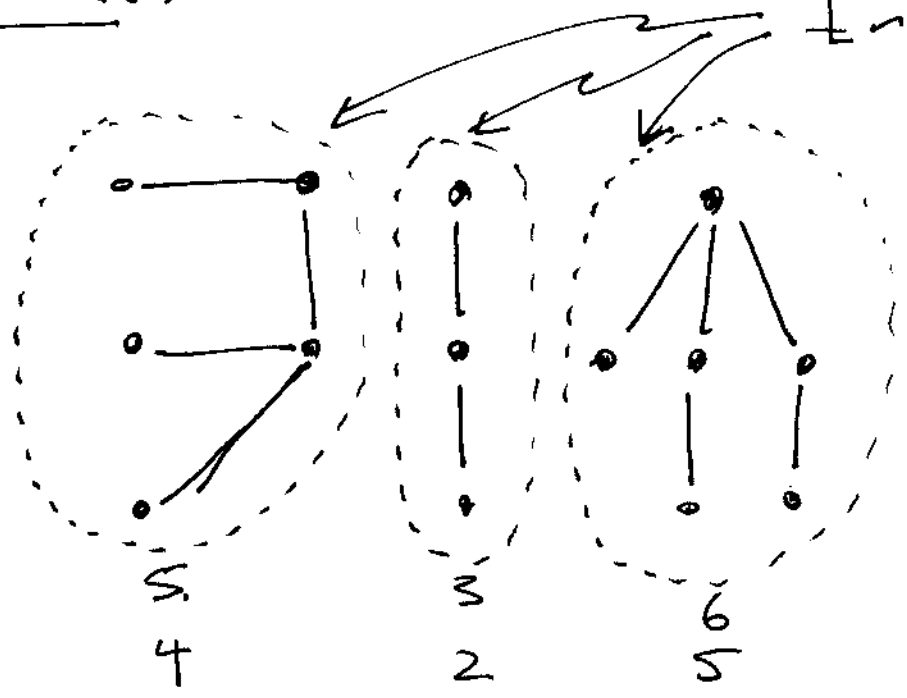
cse 101 4-14-20

Defn

a graph G is called acyclic iff it contains no cycles. also called a forest.

Ex



Defn

A tree is a graph that is both acyclic & connected.

Theorem

If T is a tree with n vertices and m edges, then $m = n - 1$.

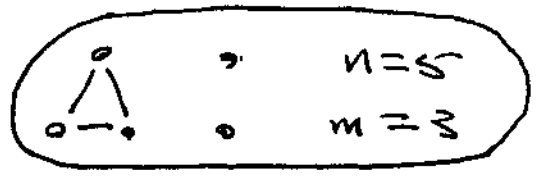
In fact given 3 properties:

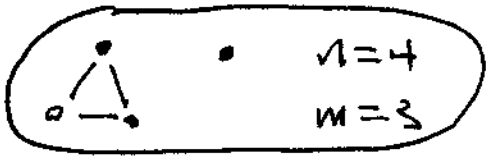
(i) connected
(ii) acyclic
(iii) $m = n - 1$

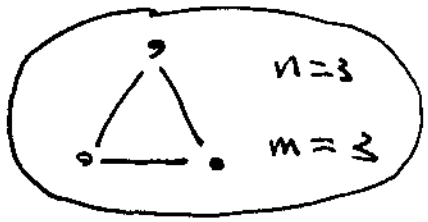
Any graph with 2 of these properties must have all 3.

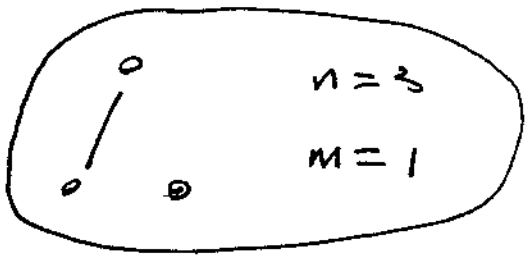
Examples

(i)	(ii)	(iii)	
F	F	F	
F	F	T	
F	T	F	
F	T	T	← impossible : lemma 6
T	F	F	
T	F	T	← impossible : lemma 5
T	T	F	← impossible : lemma 1
T	T	T	









any tree

Representations of graphs

- Incidence matrix

requires ordered V & E

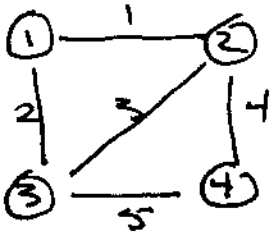
$$V = \{x_1, x_2, \ldots, x_n\}$$
$$E = \{e_1, \ldots e_m\}$$

$I = I(G)$ is an $n \times m$ matrix

$$I_{ij} = \begin{cases} 1 & \text{if } x_i \text{ is } \underline{\text{incident}} \text{ w/ } e_j \\ 0 & \text{otherwise} \end{cases}$$

Ex

G :



$$I(G) = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

- Adjacency Matrix

  requires ordered V (not E)

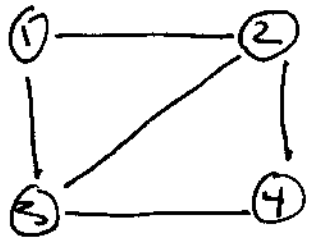
  $V = \{x_1, x_2, \ldots, x_n\}$

  $A = A(G)$ is an $n \times n$ matrix.

$$A_{ij} = \begin{cases} 1 & \text{if } x_i \text{ adjacent to } x_j \\ 0 & \text{otherwise} \end{cases}$$

Ex.

G:



$$A = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix}$$

Thm for all $k \geq 0$,

$[A^k]_{ij}$ = # of walks of length k from x_i to x_j

Exercise: Verify this in examples!

- Adjacency List

requires V ordered

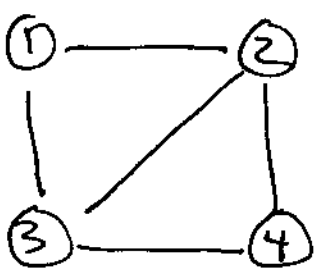
$V = \{x_1, x_2, \ldots x_n\}$

Adj(G) is an array of n lists

Adj[i] = list of neighbors of x_i $(1 \leq i \leq n)$

Ex

G



adjL

1 : 2, 3

2 : 1, 3, 4

3 : 1, 2, 4

4 : 2, 3

Graph Algorithms handout

SSSP Problem

define

$$\delta(x,y) = \begin{cases} \text{length of a min. length } x\text{-}y \text{ path, if } y \text{ reachable fr. } x \\ \infty \quad \text{otherwise} \end{cases}$$

Problem:
Given G and $s \in V(G)$, determine $\delta(s,x)$ for all $x \in V(G)$. and, for all x reachable from s, find a s-x Path of length $\delta(s,x)$.

Breadth First Search (BFS)

- each vertex $x \in V(G)$ must have 3 attributes
  - color[x] : (w, g, b)
    - w: undiscovered
    - g: discovered, not discovered all neighbors
    - b: finished
  - d[x] : distance from s to x, $\delta(s,x)$
  - P[x] : Parent (or Predecessor) of x in shortest s-x Path.

- maintain a FIFO queue Q
- use adjacency list representation of G.

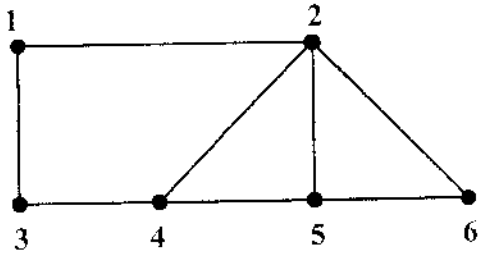
BFS(G, s)

```
1.  for x ∈ V(G) − {s}
2.      color[x] = white
3.      d[x] = ∞
4.      p[x] = nil
5.  color[s] = gray     // discover the source s
6.  d[s] = 0
7.  p[s] = nil
8.  Q = ∅               // construct a new empty queue
9.  Enqueue(Q, s)
10. while Q ≠ ∅
11.     x = Dequeue(Q)
12.     for y ∈ adj[x]
13.         if color[y] == white    // y is undiscovered
14.             color[y] = gray     // discover y
15.             d[y] = d[x] + 1
16.             p[y] = x
17.             Enqueue(Q, y)
18.     color[x] = black            // finish x
```

Lines 1-7 initialize all vertex attributes, while lines 8-9 initialize the FIFO queue Q. Then while loop (lines 10-18) systematically explores all vertices reachable from the source s.

Example 1

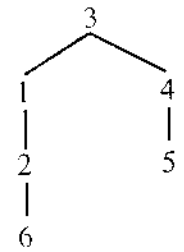
Source $s = 3$



		color[ ]	d[ ]	p[ ]
1:	2 3	~~w~~ g b	~~∞~~ 1	~~nil~~ 3
2:	1 4 5 6	~~w~~ g b	~~∞~~ 2	~~nil~~ 1
3:	1 4	g b	0	nil
4:	2 3 5	~~w~~ g b	~~∞~~ 1	~~nil~~ 3
5:	2 4 6	~~w~~ g b	~~∞~~ 2	~~nil~~ 4
6:	2 5	~~w~~ g b	~~∞~~ 3	~~nil~~ 2

Q: 3 1 4 2 5 6

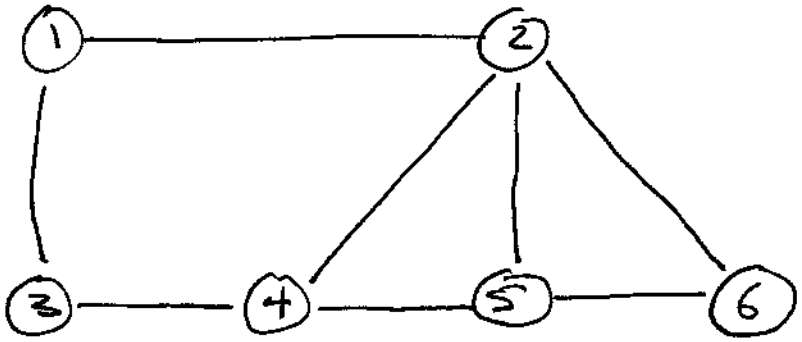
BFS Tree:



Ex

s = 3

G

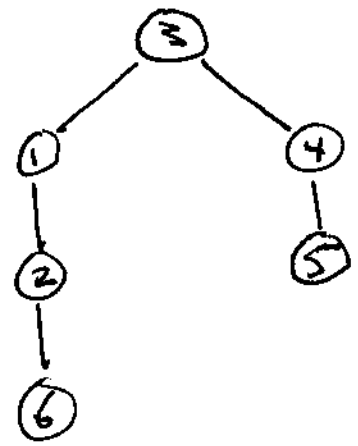


	adj	color	d	p
1	2 3	w g b	∞ 1	nil 3
2	1 4 5 6	w g b	∞ 2	nil 1
s → 3	1 4	g b	0	nil
4	2 3 5	w g b	∞ 1	nil 3
5	2 4 6	w g b	∞ 2	nil 4
6	2 5	w g b	∞ 3	nil 2

Q: 3 1 4 2 5 6

Shortest Paths tree

BFS Tree :



depth

0

1

2

3

Extension:

Paz 1 more day

Fri. 10 PM