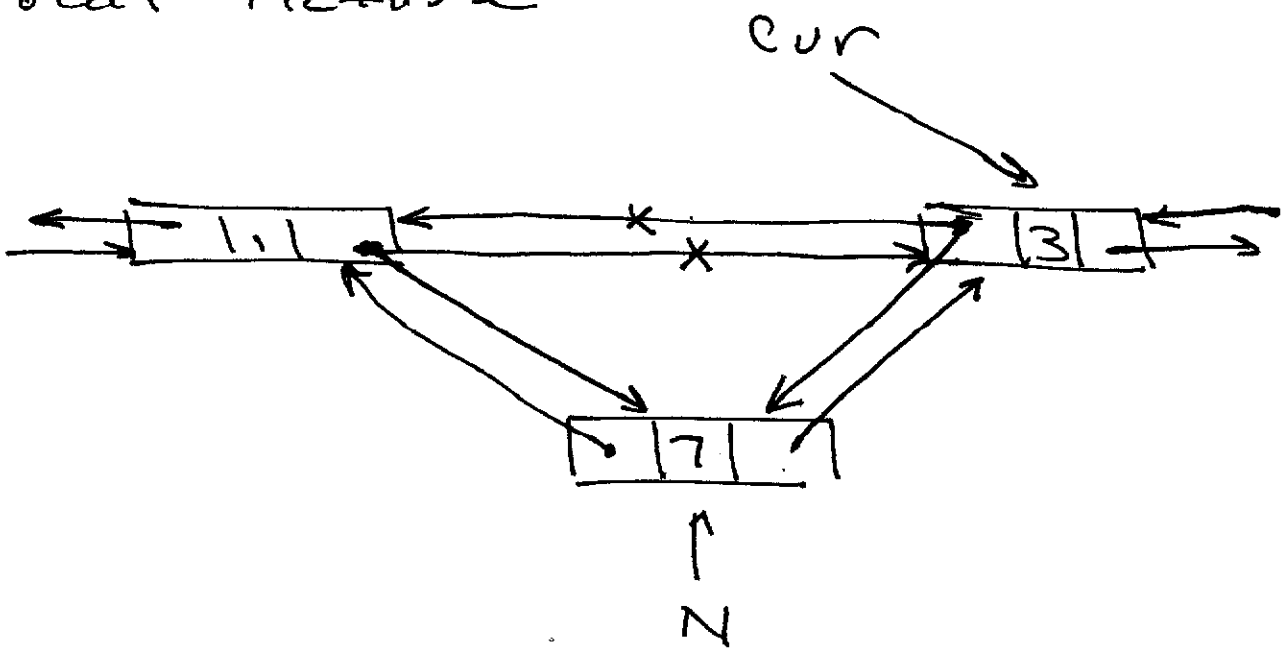
CSE 101 1-13-22

Pa1: ext. 1 last day

Quiz1: Post solns soon.

Recall List ADT example:

local Picture




```
L->cur->Prev->next = N;
N->Prev = L->cur->Prev;
L->cur->Prev = N;
N->next = L->cur;
```

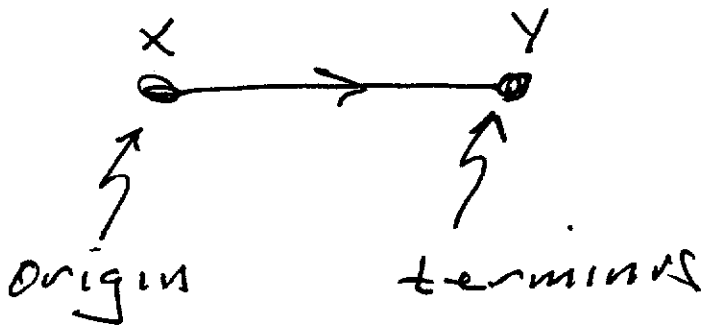
Defn

A Directed Graph (Dig-aph) is a Pair $G = (V, E)$ of sets

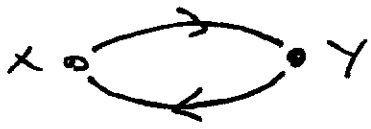
- $V \neq \phi$ (vertices)
- $E \subseteq V \times V$ = {ordered pairs of vertices}

if $(x, y) \in E$

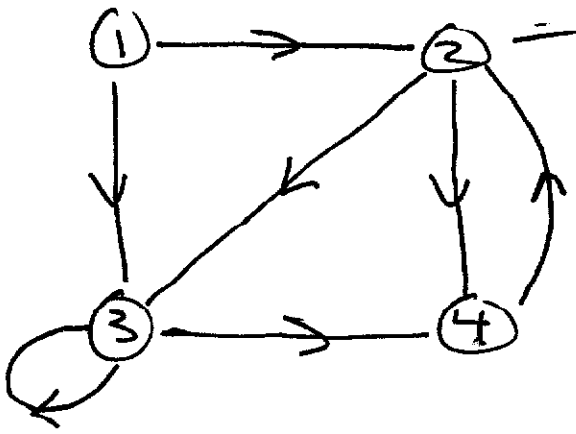
Picture



allowed  (x, x)

"  $(x, y) \neq (y, x)$

Ex.



$id(2) = 2$
$od(2) = 2$
$deg(2) = 4$

$V = \{1, 2, 3, 4\}$

$E = \{(1,2), (1,3), (2,3), (2,4), (4,2), (3,3), (3,4)\}$

Directed 1 – 4 Path:

1, 2, 3, 4 length = 3

Adjacency List representation :

The adjacency list of G (notation : adj(G)) consists an array of lists of neighbors of each vertex

undirected case $V = \{x_1, x_2, \ldots, x_n\}$.

adj[1] : ... neighbors of x_1
adj[2] : ... " " x_2
:
adj[n] : ... " " x_n

Directed case : $V = \{x_1, \ldots, x_n\}$

$adj[1]$: list of termini of edges with origin x_1

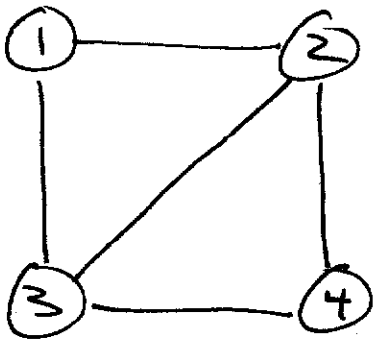
$adj[2]$: x_2

$\vdots$

$adj[n]$: x_n

Ex. undirected

G :



$adj(G)$

1 : 2 3

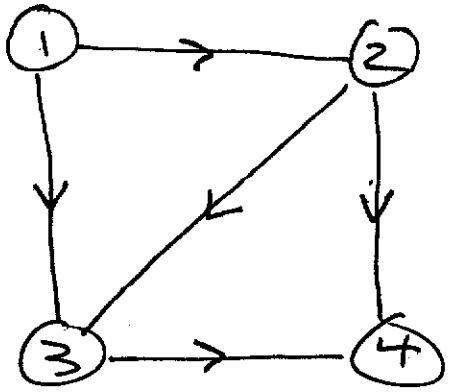
2 : 1 3 4

3 : 1 2 4

4 : 2 3

Ex. Directed

G



adj(G) {
1 : 2 3
2 : 3 4
3 : 4
4 :

Single source shortest Path problem

(SSSP)

Let G be a graph or digraph.

Let $x, y \in V(G)$. The distance $\delta(x,y)$ from x to y is

$$\delta(x,y) = \begin{cases} \text{minimum length of an } x\text{-}y \text{ path, if one exists.} \\ \infty \text{ if no } x\text{-}y \text{ path exists} \end{cases}$$

SSSP:

Given G (directed or undirected) and a vertex $s \in V(G)$ (called source),

(1) determine $\delta(s,x)$ for all $x \in V(G)$.

(2) for each $x \in V(G)$ with $\delta(s,x) < \infty$, determine a shortest s-x path in G.

Breadth-First Search (BFS)

solves SSSP in both graphs and digraphs. uses the adj-list representation.

Vertex attributes: for $x \in V(G)$

- Color[x] = { white : undiscovered; gray : discovered (neighbors may not); black : finished }
- d[x] = distance estimate from s (when done: $d[x] = \delta(s, x)$)
- P[x] = Parent or predecessor of x in a shortest s-x path.

note: $V(G) = \{1, 2, 3, \ldots, n\}$

we implement color, d, p as arrays on indices $1, 2, \ldots n$.

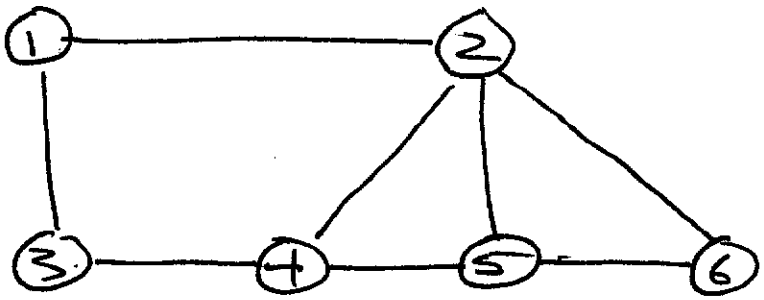
BFS starts at source s, then systematically discovers every vertex reachable from s.

we discover all vertices at distance k from s before we discover any vertices at dist. $k+1$,

Uses a FIFO queue Q.

Q contains all gray vertices.

Ex. G , s = 3

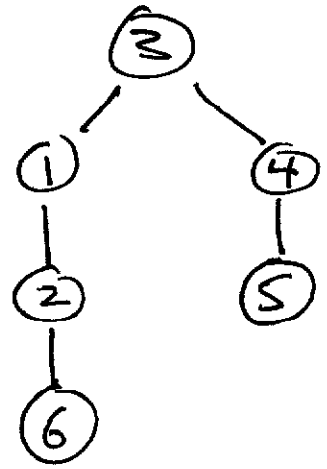


x	adj[x]	color[x]	d[x]	π[x]
1	2 3	~~w~~ ~~g~~ b	~~∞~~ 1	~~NIL~~ 3
2	1 4 5 6	~~w~~ ~~g~~ b	~~∞~~ 2	~~NIL~~ 1
→ 3	1 4	~~g~~ b	0	NIL
4	2 3 5	~~w~~ ~~g~~ b	~~∞~~ 1	~~NIL~~ 3
5	2 4 6	~~w~~ ~~g~~ b	~~∞~~ 2	~~NIL~~ 4
6	2 5	~~w~~ ~~g~~ b	~~∞~~ 3	~~NIL~~ 2

Q : ~~3~~ ~~1~~ ~~4~~ ~~2~~ ~~5~~ ~~6~~

dist

BFS Tree:



0

1

2

3