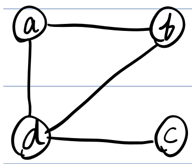
Recap:

Graphs $G = (V, E)$

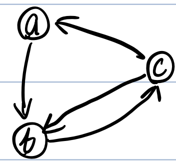
vertices edges



$V = \{a, b, c, d\}$

$E = \{\{a,b\}, \{a,d\}, \{b,d\}, \{c,d\}\}$

undirected



$V = \{a, b, c\}$

$E = \{(a,b), (b,c), (c,b), (c,a)\}$

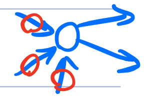
directed

1. Graphs (cont'd)

Teminology/Notations:

$n = |V|$, $m = |E|$

Degree of a vertex u : # of edges connected to u
($\deg(u)$)

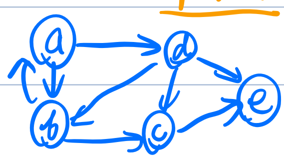


In-degree of a vertex u: # edges into u

$|\{(v,u) : (v,u) \in E\}|$

Out-degree of a vertex u: # edges out of u

Path: sequence of nodes/edges from one node to another

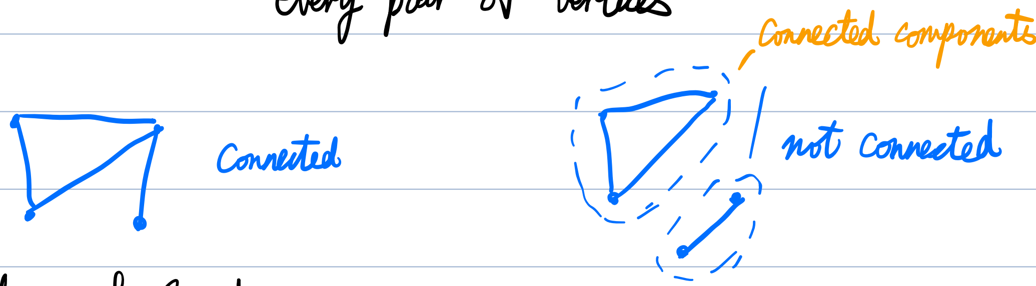


e.g. (a, d, b, c, e) $((a,d), (d,b), (b,c), (c,e))$

(Simple path: a path w/o repeating vertices
e.g. (a,d,b,c,e) is simple path, (a,b,a,d) is not)

Cycle: A path that starts and ends at same vertex, and does not have repeated edges and vertices
e.g. (a,d,b,a) (apart from first & last vertex)

G is **connected** if there is a path b/w every pair of vertices

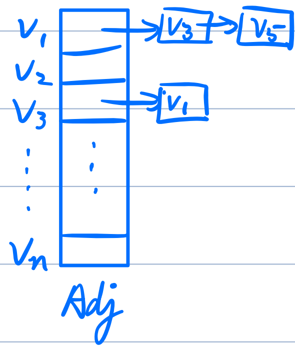


Applications of Graphs

- Google Page Rank
- Facebook social graph
- Many More ...
- Maps
- Internet / traffic routing

Graph Representations:

① Adjacency List



Array Adj of $|V|$ linked lists, one linked list for each vertex

In each linked list, Adj [u] stores u's neighbors.

$$Adj[u] = \{ v \in V \mid (u,v) \in E \}$$

Default Representation

Space : $\Theta(|V| + |E|)$

② Adjacency Matrix

(v_1, v_3)

$|V| \times |V|$ bit-matrix

$A[u,v] = 1 \Leftrightarrow (u,v) \in E$

$$\begin{array}{c|ccccc} & v_1 & v_2 & v_3 & \cdots & v_n \\ \hline v_1 & & & 1 & & \\ v_2 & & & & & \\ v_3 & 0 & & & & \\ \vdots & & & & & \\ v_n & & & & & \end{array}$$

Space: $\Theta(|V|^2)$

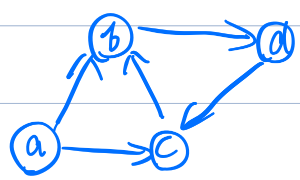
Good if $|E| \approx |V|^2$, very fast for look up if (u,v) exists!

2. Breath-First Search (BFS)

Single Source Shortest Paths (SSSP):

Input: Graph $G = (V, E)$ & source $s \in V$

Output: shortest path from s to any $v \in V$ (if it exists)



$s = a$

$b: (a,b)$

$c: (a,c)$

$d: (a,b)(b,d)$

Breadth-First Search (BFS) (a graph exploration problem)

Idea: - look at vertices reachable in 0 steps,
1 steps,
2 steps
....

- Careful not repeatedly visit same vertices
- Work for both directed/undirected

Queue Q: $\boxed{a} \leftarrow \boxed{b} \leftarrow \boxed{c}$, First-In-First-Out (FIFO)

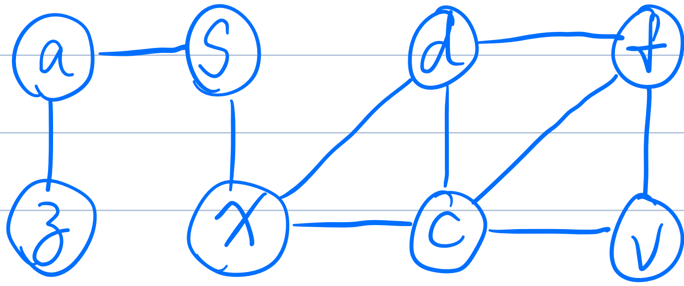
Q.push(d) : $\boxed{a} \leftarrow \boxed{b} \leftarrow \boxed{c} \leftarrow \boxed{d}$ $\quad O(1)$

Q.pop( ) : $\boxed{a} \quad \boxed{b} \leftarrow \boxed{c} \leftarrow \boxed{d}$ $\quad O(1)$

```
BFS(G,s):
    For all v ∈ V:
        dist(v) ← ∞
        parent(v) ← ⊥                      O(|V|)
        color(v) ← WHITE
    dist(s) = 0
    Q.push(s)                              O(1)
    Color(s) = YELLOW
    While Q is not empty:
        u ← Q.pop()
            For all v ∈ Adj[u]
                If color(v) = WHITE:
                    dist(v) = dist(u)+1            O(|E|)
                    parent(v) = u          O(1)
                    color(v) = YELLOW
                    Q.push(v)
            color(u) = GREEN
```

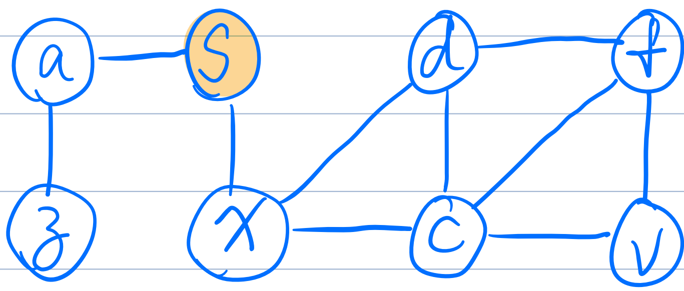
dist(v) gives the shortest distance from s to v

The path? Reverse (v, parent(v), parent(parent(v)), ...)



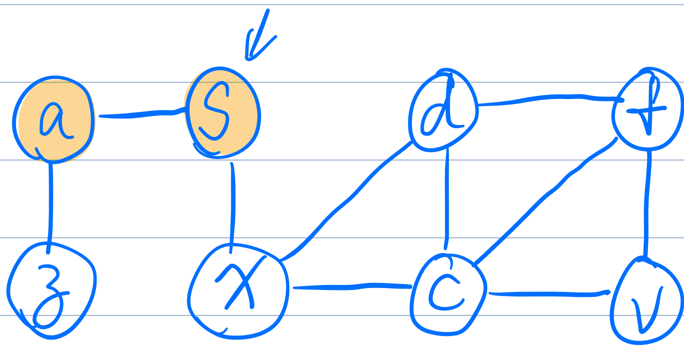
	dist	parent
a	∞	$\perp$
c	∞	$\perp$
d	∞	$\perp$
f	∞	$\perp$
s	∞	$\perp$
v	∞	$\perp$
x	∞	$\perp$
z	∞	$\perp$

Q:



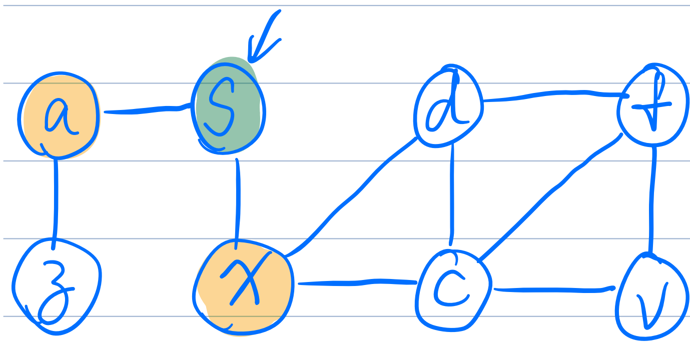
	dist	parent
a	∞	$\perp$
c	∞	$\perp$
d	∞	$\perp$
f	∞	$\perp$
s	0	$\perp$
v	∞	$\perp$
x	∞	$\perp$
z	∞	$\perp$

Q: s



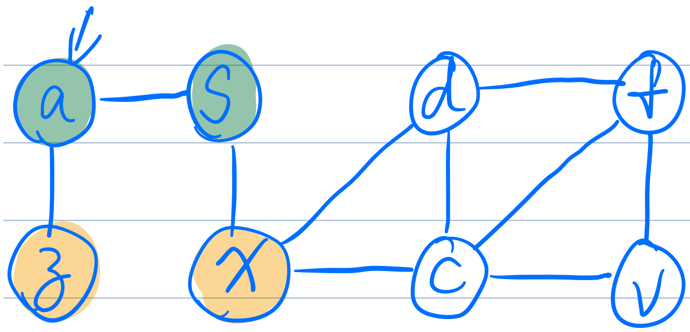
	dist	parent
a	1	S
c	∞	$\perp$
d	∞	$\perp$
f	∞	$\perp$
S	0	$\perp$
v	∞	$\perp$
x	∞	$\perp$
z	∞	$\perp$

Q: a



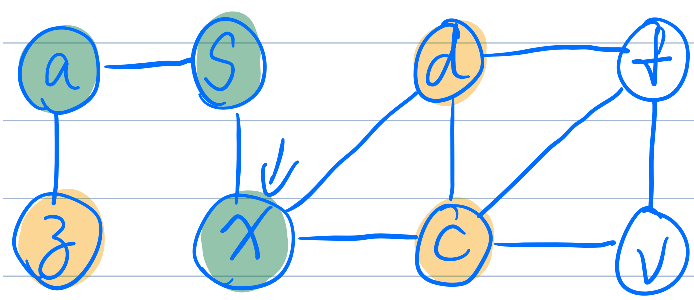
	dist	parent
a	1	S
c	∞	$\perp$
d	∞	$\perp$
f	∞	$\perp$
S	0	$\perp$
v	∞	$\perp$
x	1	S
z	∞	$\perp$

Q: a, x



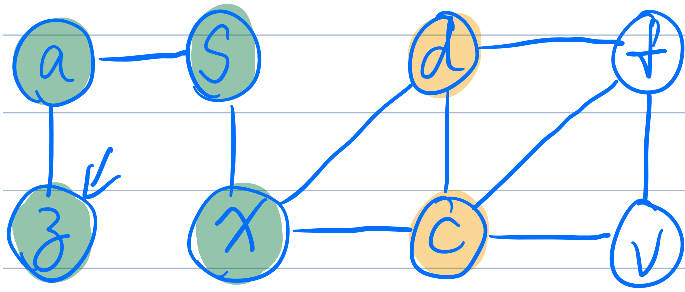
Q: x, z

	dist	parent
a	1	s
c	∞	$\perp$
d	∞	$\perp$
f	∞	$\perp$
s	0	$\perp$
v	∞	$\perp$
x	1	s
z	2	a



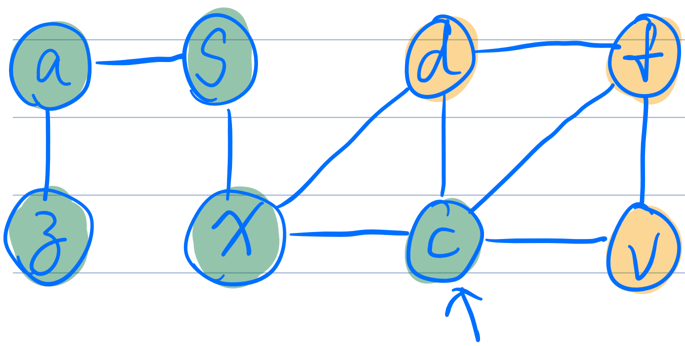
Q: z, c, d

	dist	parent
a	1	s
c	2	x
d	2	x
f	∞	$\perp$
s	0	$\perp$
v	∞	$\perp$
x	1	s
z	2	a



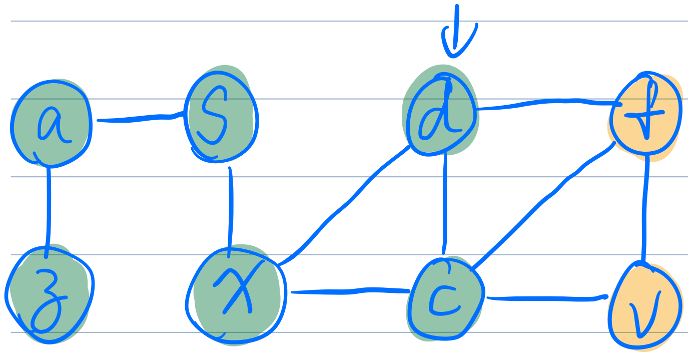
	dist	parent
a	1	s
c	2	x
d	2	x
f	∞	⊥
s	0	⊥
v	∞	⊥
x	1	s
z	2	a

Q: c, d



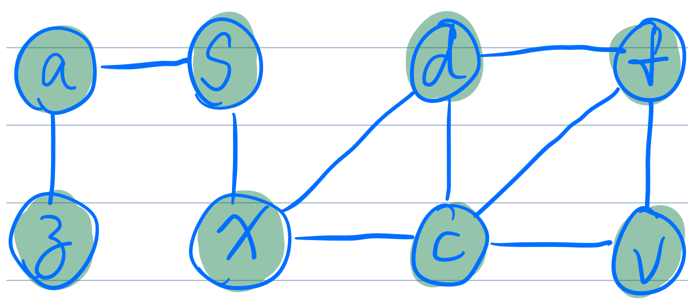
	dist	parent
a	1	s
c	2	x
d	2	x
f	3	c
s	0	⊥
v	3	c
x	1	s
z	2	a

Q: d, f, v



	dist	parent
a	1	s
c	2	x
d	2	x
f	3	c
s	0	$\perp$
v	3	c
x	1	s
z	2	a

Q: f, v



	dist	parent
a	1	s
c	2	x
d	2	x
f	3	c
s	0	$\perp$
v	3	c
x	1	s
z	2	a

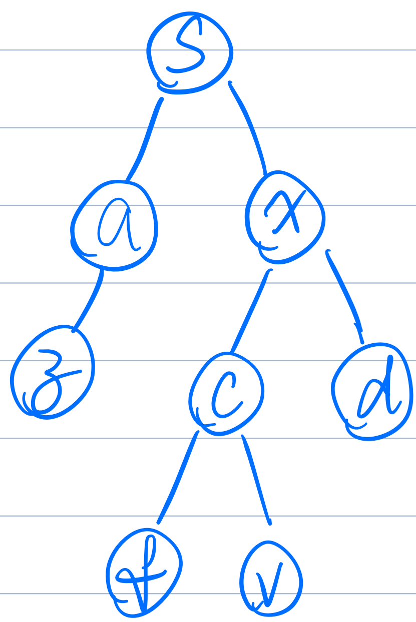
Q:

○ : vertices not added to Q yet; "undiscovered"
○ : vertices currently in Q ; "unexplored"
○ : vertices out of Q already; "finished"

Coloring kinda overkill here, could use a bool "visited".
But will be useful in the future...

* Sometimes also colored as ○, ●, ●

This gives us a <u>BFS Tree</u> with root s.



<u>Runtime</u>: $\Theta(|V|+|E|)$

Correctness:

Key Lemma: For any u, v, if u added to Q before v, $dist(u) \leq dist(v)$.

Proof: Loop Invariant:

① For any two vertices v_1, v_2 currently in Q,

$$dist(v_1) \leq dist(v_2) + 1$$

② For any "discovered" (O / O) v_1, v_2,
if v_1 pushed into Q before v_2,
($\Leftrightarrow$ v_1 popped out before v_2.
$\Leftrightarrow$ v_1 "finished" before v_2),
then $dist(v_1) \leq dist(v_2)$

Init: ✓

Maintenance:

① By invariant ②, all vertices in Q are in increasing order of their dist $\Rightarrow$ their dist is at least $dist(u)$, but at most $dist(u)+1$. The new v's we add into Q have dist of $dist(u)+1$. So everything in Q has $dist(u)$ or $dist(u)+1$.

② If not, that means when we're discovering v (adding it to Q), there is a vertex v' already discovered with $dist(v') > dist(v) = dist(u)+1$.

i) if v' added to Q before u,

$dist(v') \leqslant dist(u)$, Contradiction!

ii) if v' added after u,

that means v' still in Q! By ①,

$dist(u) \leq dist(v') \leq dist(u)+1$, contradiction!

Termination: For any u, v, if u added to Q before v, $dist(u) \leqslant dist(v)$.

For full proof of correctness, see CLRS §20.2. (optional)