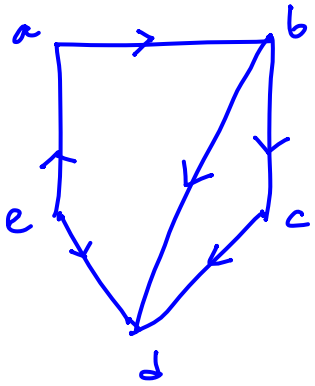
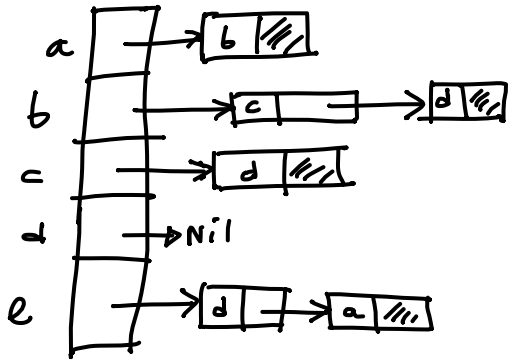
Breadth First Search (BFS)



Graph



Adjacency List representation

	a	b	c	d	e
a	0	1	0	0	0
b	0	0	1	1	0
c	0	0	0	1	0
d	0	0	0	0	0
e	1	0	0	1	0

Adjacency Matrix

Search in a graph means to explore the whole graph

Start vertex s (or source vertex)

From current vertex v visit some other vertex w which is adjacent to v. w now becomes current vertex.

This way we would like to discover all vertices which are reachable from s.

Search procedures differ in the way they select the next vertex to be visited.

In BFS all vertices at distance d from 's' are visited before any vertex at distance >d from 's' is visited.

In BFS, we start at s.

Then visit all v s.t. $\delta(s, v) = 1$

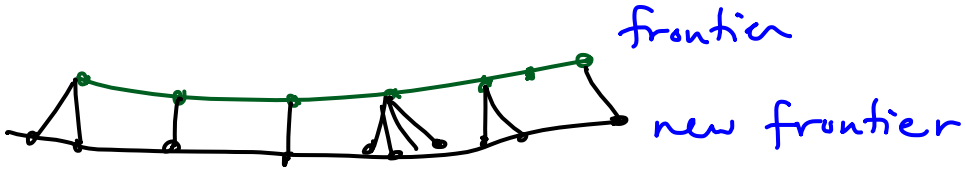
" $\delta(s, v) = 2$

" $\delta(s, v) = 3$

⋮

$\delta(s, v) = |V| - 1$

In BFS, at any stage there is a frontier of vertices



We define BFS by means of an algorithm.

Algorithm uses three additional vertex attributes:

1. $v.col$ — White → Grey → Black
   - White: undiscovered
   - Grey: discovered and in current frontier
   - Black: discovered and not in current frontier (left the frontier)
2. $v.d$ — $\delta(s, v)$, [distance of v from s, or length of the shortest path from s to v]
3. $v.\pi$ — BFS creates a tree, called BFS tree.

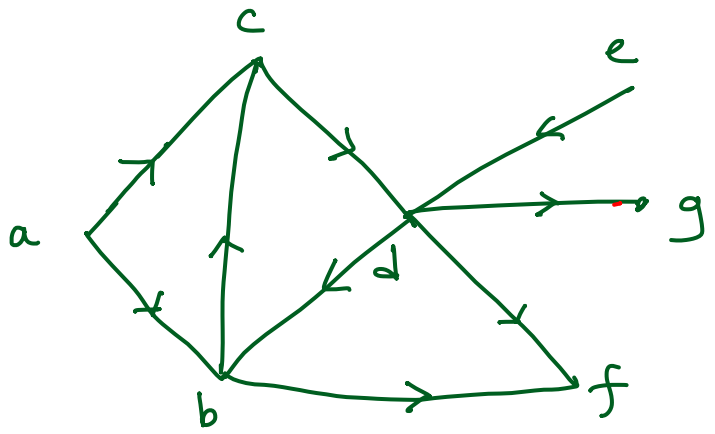
Whenever a vertex v is discovered by visiting from a vertex u. Then an edge (v, u) is created.

u is the parent of v in the BFS tree.

$v.\pi = u$

s is the root of BFS tree, the unique path from s to v in this tree is a shortest path from s to v.

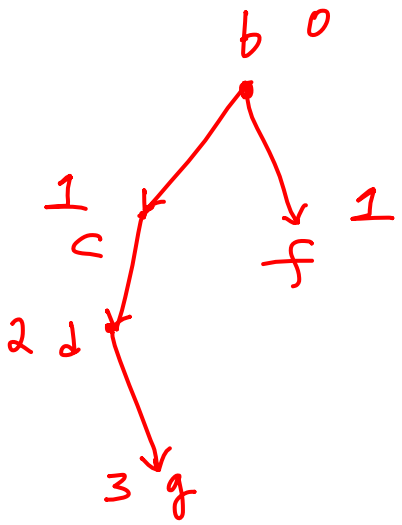
```
BFS(G, s)      // G = (V, E), s ∈ V
    for all v ∈ V do
        v.col = white
        v.d   = ∞
        v.π   = nil
    s.col = Grey
    s.d   = 0
    s.π   = nil
    Enqueue(Q, s)
    While notempty(Q) do
        x = dequeue(Q)
        for all y in 'Adjacency List of x' do
            if (y.col == white)
                y.col = Grey
                y.d   = x.d + 1
                y.π   = x
                Enqueue(Q, y)
        x.col = Black
```



$S \equiv b$

Iterations of while loop	Q (front to rear)	Grey vertices	Black
0	b	b	—
1	cf	cf	b
2	fd	fd	b, c
3	d	d	f, b, c
4	g	g	f, b, c, d
5	—	—	f, b, c, d, g

Procedure terminates.



Two vertices: a, e have not been visited because they are not reachable from b.

$\delta(b, a) = \delta(b, e) = \infty$

Complexity Analysis of BFS(G, s)

Aggregate Analysis.

1. Each vertex can be put in the queue at most once.
   (So it can be dequeued also at most once)

   Total no. of queue operations (over all iterations of while loop)
   is $O(|V|)$.

2. For loop corresponding to a vertex (x) will be executed at most once.
   No. of iteration in this for loop is $d_{out}(x)$

   Total no. of for loop iteration (summed over all while loop iteration)
   is $\leq \sum_{x \in V} d_{out}(x) = |E|$

   Each for loop iteration take $O(1)$ step.

No. of instructions is bounded by

$O(|V|)$ // initialization

$+ \quad O(|V|)$ // queue operations

$+ \quad O(|E|)$ // 'for loop iterations'

$\Longrightarrow \quad O(|V| + |E|)$

Runtime complexity of DFS(G, s) $[G = (V, E)]$

is $O(|V| + |E|)$.

Correctness still needs to be proved

We need to show.

1. All reachable vertices will be visited by BFS.

2. At the end of $BFS(G, s)$, for all $v \in |V|$,

$$v.d = \delta(s, v)$$

3. Edges $v.\pi$ induce a tree structure on vertices reachable from s.

Unique path from s to any v in this tree is a shortest length path from s to v.