L19 - Breadth First Search/Traversal/Visit

Thursday, May 28, 2020 8:16 AM

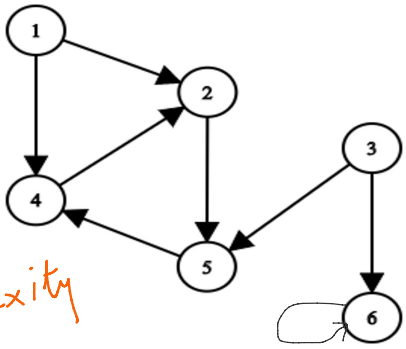
To traverse/reach each reachable vertex of a graph from a given starting vertex.

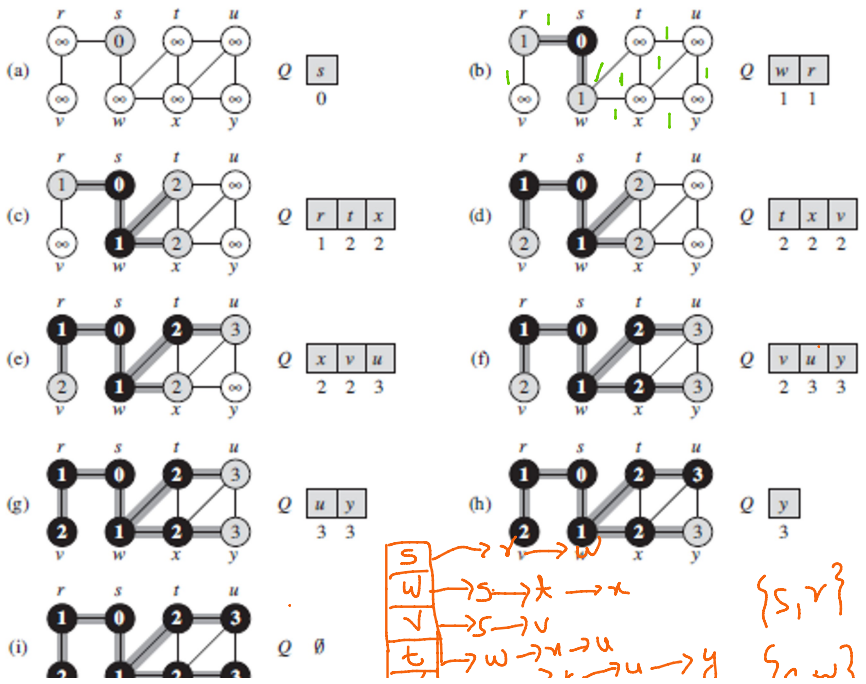
```
BFS(G,s)
 1. for each vertex u ∈ G.V - {s}
 2.     u. color =  WHITE
 3.     u. d = ∞
 4.     u. π  = NIL
 5.  s. color = GRAY
 6.  s. d = 0
 7.  s. π  = NIL
 8.  Q =  ∅
 9.  ENQUEUE(Q, s)
10.  while Q
11.      q = DEQUEUE(Q)
12.      for each v ∈ G. Adj[q]      // explore edge (q, v)
13.          if v. color == WHITE
14.              v. d = q. d + 1
15.              v. color = GRAY
16.              v. π = q
17.              ENQUEUE(Q, v)
18.      q. color = BLACK            // finished with q
```

- Vertex with **WHITE** color means that it has not been visited yet.
- Vertex with **GRAY** color means that it has been visited but has not yet been fully explored.
- Vertex with **BLACK** color means that all of it's adjacent vertices has also been visited (fully explored).

O(|V| + |E|) = O(n+ e)

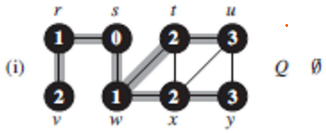
Predecessor:	Nil	1	1	2
Nodes:	~~1~~	~~2~~	~~4~~	~~5~~
Distance from 1:	0	1	1	2

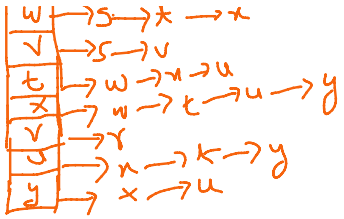





```
BFS(G,s)
 1. for each vertex u ∈ G.V - {s}
 2.     u. color =  WHITE
 3.     u. d = ∞
 4.     u. π  = NIL
 5.  s. color = GRAY
 6.  s. d = 0
 7.  s. π  = NIL
 8.  Q =  ∅
 9.  ENQUEUE(Q, s)
10.  while Q
11.      q = DEQUEUE(Q)
12.      for each v ∈ G. Adj[q]      // explore edge (q, v)
13.          if v. color == WHITE
14.              v. d = q. d + 1
15.              v. color = GRAY
16.              v. π = q
17.              ENQUEUE(Q, v)
18.      q. color = BLACK            // finished with q
```

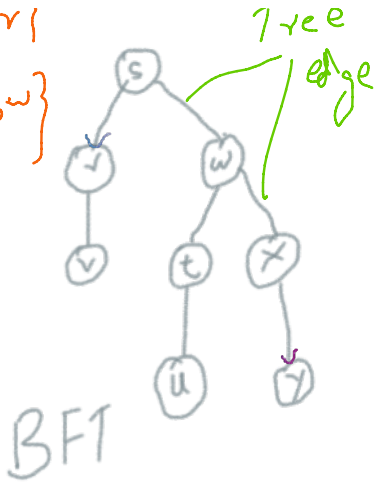
O(|V| + |E|) = O(n+ e)





{s, r}

{s, w}



1st Iter. While Loop: s
1st iter. of For loop: w
2nd iter. of For loop: r

2nd Iter. While Loop: w
1st iter. of For loop: s
2nd iter. of For loop: t
3rd iter. of For loop: x

3rd Iter. While Loop: r
1st iter. of For loop: s
2nd iter. of For loop: v

4th iter. While loop: t
1st iter. of For loop: w
2nd iter. of For loop: x
3rd iter. of For loop: u

π:	Nil	s	s					
Q:	~~s~~	w	r					
d:	0	1	1					

π	s	s	w	w				
Q:	~~w~~	r	t	x				
d:	1	1	2	2				

π:	s	w	w	r				
Q:	~~r~~	t	x	v				
d:	1	2	2	3				

π:	w	w	r	t				
Q:	~~t~~	x	v	u				
d:	2	2	3	3				

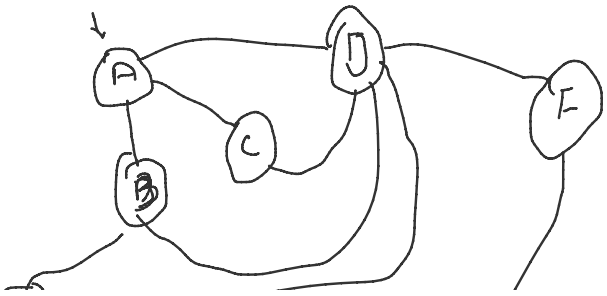
Predecessor:	Nil	s	s	w	w	r	t	x
Node:	s	w	r	t	x	v	u	y
Distance from s:	0	1	1	2	2	2	3	3

Predecessor:	Nil	s	s	w	w	r	t	x
Node:	s	w	r	t	x	v	u	y
Distance from s:	0	1	1	2	2	2	3	3

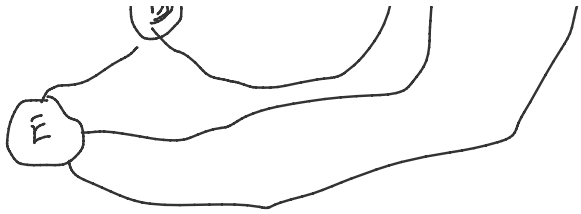
Path from s to y: s w x y

```
PRINT-PATH(G; s; v)
1.  if v == s
2.      print s
3.  else if v.π == NIL
4.      print “no path from” s “to” v “exists”
5.  else PRINT-PATH(G; s; v.π)
6.      print v
```

PRINT-PATH(G; s; y) ----> PRINT-PATH(G; s; x) ----> PRINT-PATH(G; s; w) -----> PRINT-PATH(G; s; s)
y x w s




```
BFS(G,s)
1.  for each vertex u ∈ G.V - {s}
2.      u. color = WHITE
3.      u. d = ∞
4.      u. π  = NIL
5.  s. color = GRAY
6.  s. d = 0
7.  s. π  = NIL
8.  Q = ∅
```



```
5.  s. color = GRAY
6.  s. d = 0
7.  s. π  = NIL
8.  Q =  ∅
9.  ENQUEUE(Q, s)
10. while Q
11.     q = DEQUEUE(Q)
12.     for each v ∈ G. Adj[q]      // explore edge (q, v)
13.         if v. color == WHITE
14.             v. d = q. d + 1
15.             v. color = GRAY
16.             v. π = q
17.             ENQUEUE(Q, v)
18.     q. color = BLACK            // finished with q
```

O(|V| + |E|) = O(n+ e)