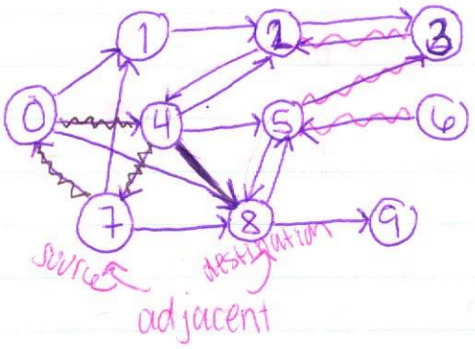
Graphs

Dec 1
COMP 2402

$G = (V, E)$



path: 6-2
cycle: 7-0-4
reachable: 5 → 6 ✗
6 → 5 ✓
in-degree(i): # of edges to i
out degree(i): # of edges from i

Adjacency Matrix

from \ to	0	1	2	3	4	5	6	7	8	9
0	0	1	0	0	1	0	0	0	1	0
1	0	0	1	0	0	0	0	0	0	0
2	0	0	0	1	1	0	0	0	0	0
3	0	0	1	0	0	0	0	0	0	0
4	0	0	1	0	0	1	0	1	1	0
5	0	0	0	1	0	0	0	0	1	0
6	0	0	0	0	0	1	0	0	0	0
7	1	1	0	0	0	0	0	0	1	0
8	0	0	0	0	0	1	0	0	0	1
9	0	0	0	0	0	0	0	0	0	0

memory requirement
↳ n^2

```
inEdges(i) // O(n)
List edges = new List();
for(j=0 to n-1){
if (a[j][i] ==1)
   edges.add(j);
}
```

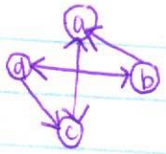
source → ← destination

```
addEdge(i,j)
    a[i][j] = 1   // O(1)

removeEdge(i,j)
    a[i][j] = 0   // O(1)

hasEdge(i,j)
    return a[i][j] == 1 // O(1)
```

```
outEdges(i) // O(n)
  List edges = new List();
  for(j=0 to n-1){
  if(a[i][j] ==1) {
     edges.add(j);
  }
}
```



A	a	b	c	d
a	0	0	1	0
b	1	0	0	1
c	1	0	0	0
d	0	1	1	0

	AM	AL
add	O(1)	O(1)
remove	O(1)	O(deg(i))
has	O(1)	O(deg(i))
outEdge	O(n)	O(1)
inEdge	O(n)	O(m+n)
space used	$O(n^2)$	O(m+n)

A^2

	a	b	c	d
a	1	0	0	0
b	0	1	2	0
c	0	0	1	1
d	2	0	0	1

Adjacency List

adj		
0	1 4 8	
1	2	
2	3 4	
3	2	
4	2 5 7 8	
5	3 8	
6	5	
7	0 1 8	
8	6 9	
9		

memory requirement
↳ O(m+n)

```
addEdge(i,j); // O(1)
adj[i].add(j);
```

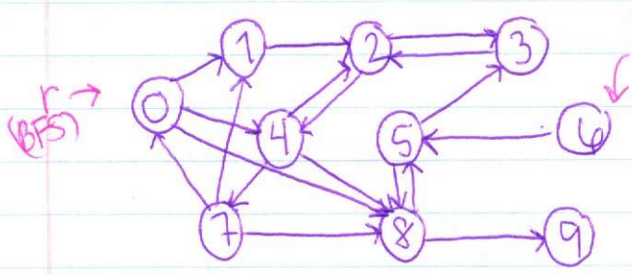
might be constant

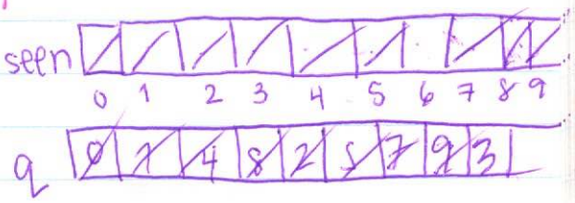
```
removeEdge(i,j)
adj[i].remove(j); // O(deg(i)) ≤ O(n)
```

```
hasEdge(i,j)
adj[i].contains(j)
```

```
outEdges(i)   // O(1)
return adj[i]
```

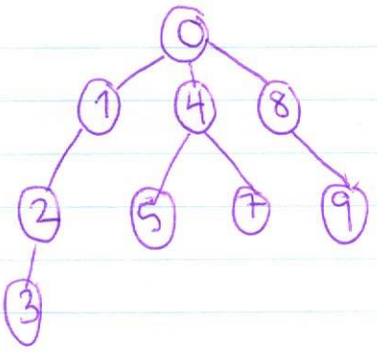
```
inEdges(i);   // O(n+m)
List edges
for(j=0 to n-1)
  if(adj[i].contains(i)) {
    edges.add(j);
  }
```





Breadth First search

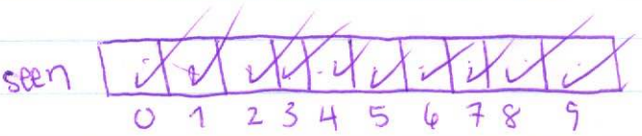
```
BFS(G,r)
    Array seen[n]
    Queue q
    q.add(r)
    seen[0] = true
    while(q is not empty)
        i = q.remove()
        for(j in outEdges(i))
            if(not seen[j])
                q.add(j)
                seen(j)
```

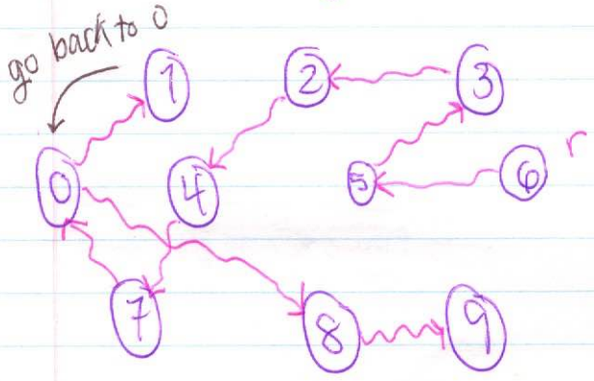


O(n+m) - adj list
$O(n^2)$ - adj matrix

depth first search

```
dFS(G,r)
    seen(i) = true
    for(j in outEdges(i))
        if(not seen[j])
            dfs(G,j);
```





~~dfs(4)~~
~~dfs(2)~~
~~dfs(1)~~
~~dfs(0)~~
~~dfs(7)~~
dfs(4)
~~dfs(2)~~
~~dfs(3)~~
~~dfs(5)~~
~~dfs(6)~~

call stack

5 the below