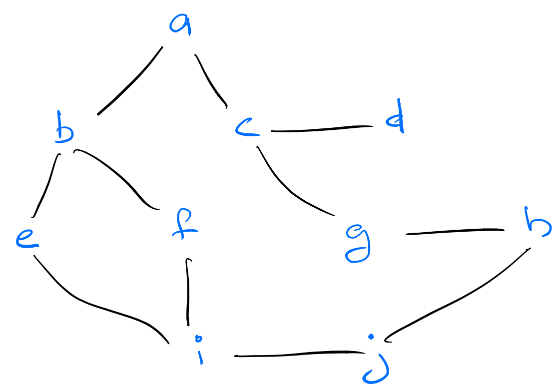
TODAY
Graph traversal (DFS, BFS, etc.)
DFS, pre-order, post-order
topo sort [& d.p.]

Exam 2 review session
- in Lab tomorrow → bring q's
- 2-4 pm Sunday → take up some practice problems, take q's
- in class Tue → bring q's

Final conflict: fill form by FRIDAY

Graph $G = (V, E)$



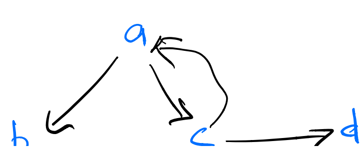
Representation:
adjacency lists

a : b, c
b : a, e, f
c : a, d
d : c
e : b, i
f : b, i
g : c, h
h : g, j
i : e, f, j
j : h, i

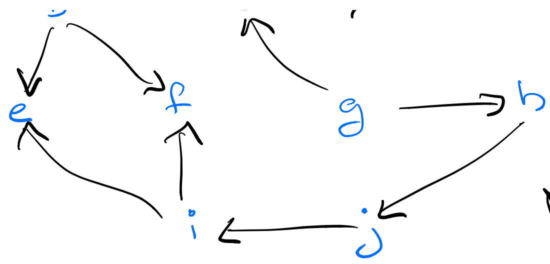
adj matrix

	a	b	c	d	e	f	g	h	i	j
a	0	1	1	0	0	0	0	0	0	0
b	1	0	0	0	1	1	0	0	0	0
c	1	0	0	1	0	0	0	0	0	0
d	0	0	1	0	0	0	0	0	0	0
e	0	1	0	0	0	0	0	0	1	0
f	0	1	0	0	0	0	0	0	1	0
g	0	0	1	0	0	0	0	1	0	0
h	0	0	0	0	0	0	1	0	0	1
i	0	0	0	0	1	1	0	0	0	0
j	0	0	0	0	0	0	0	1	1	0

- enumerate edges from node v in $O(\deg(v))$ time
- enumerate all edges in $O(E)$ time



a: b, c
b: e, f
c: a, d
d:



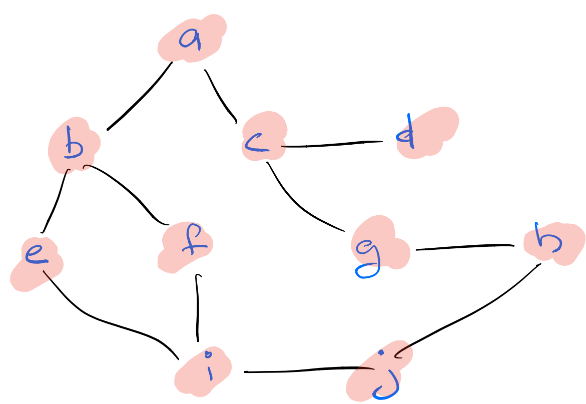
$N(v)$ = neighbors of v
aka v's adj list

Graph traversal

$DFS(G=(V,E))$
pick $v \in V$
$RDFS(G, v)$

$RDFS(G, v)$
  if v is marked
    return
  mark v

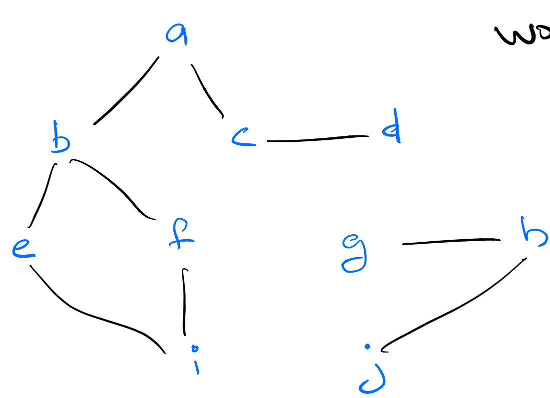
  for each $u \in N(v)$:
    $RDFS(G, u)$



RDFS(a)
  RDFS(b)
    RDFS(a)
    RDFS(e)
      RDFS(b)
      RDFS(i)

works for connected undir. graphs

works for strongly connected dir. graphs



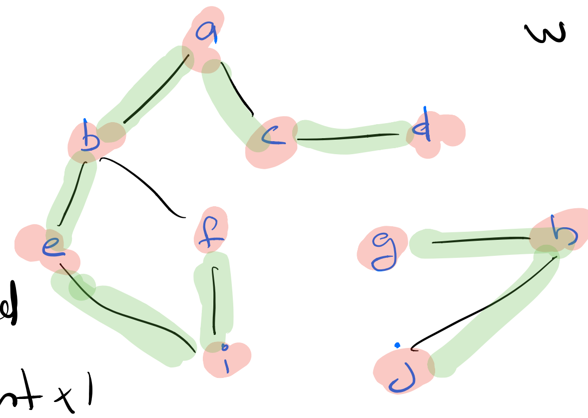
connected (undir) graph
→ for any u, v $u \neq v$
there is an undir path $u \to v$

strongly connected
for any u, v $u \neq v$
there is directed path $u \to v$

$DFS(G=(V,E))$
for $v \in V$

```
if  v is not marked
    RDFS(G, v)
```

w



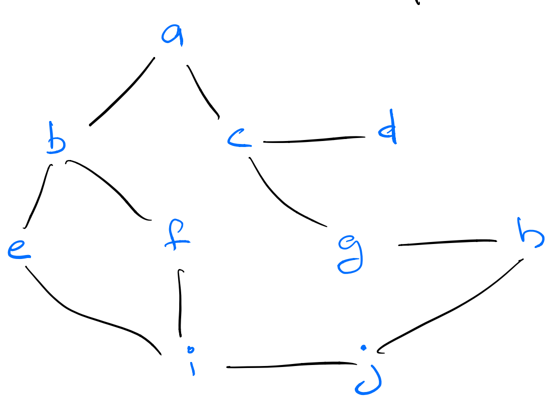
```
CountComps( G = (V,E))

  count = 0
  for  v ∈ V
    if  v is not marked
      RDFS(G, v)
      count = count + 1
  return  count.
```

```
RDFS(G, v)
  if  v is  marked
     return
  mark  v

  for each  u ∈ N(v)
     if  u not marked
       add (v,u) to ST
       RDFS(G, u)
```

```
IDFS ( G = (V,E))
  for  v ∈ V
    if  v is  not  marked
      S = stack()
      add  v  to  S
    while  S is not empty:
       pop  u  from  S
       if  u is not marked:
         mark  u
         for each  t ∈ N(u)
            push  t  onto  S
```



push a

pop a , mark a

push b, c

pop c , mark c

push a, d, g

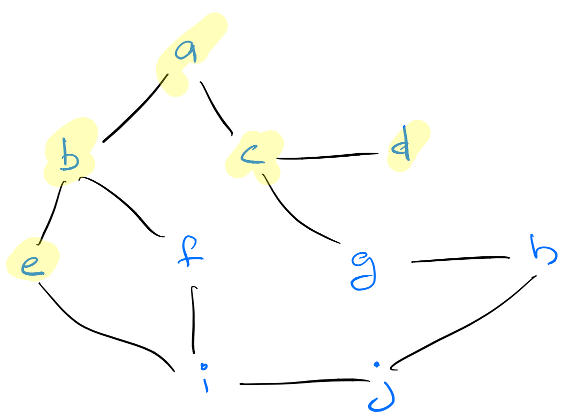
pop a ×

pop d , mark d

push c

IBFS (G= (V,E))
  for v ∈ V
    if v is not marked
      Q = Queue
      add v to Q
      while S is not empty:
        get u from Q
        if u is not marked:
          mark u
          for each t ∈ N(u)
            add t to Q

breadth-first search



~~a~~ ~~b~~ ~~c~~ ~~d~~ ~~e~~ ~~d~~ g

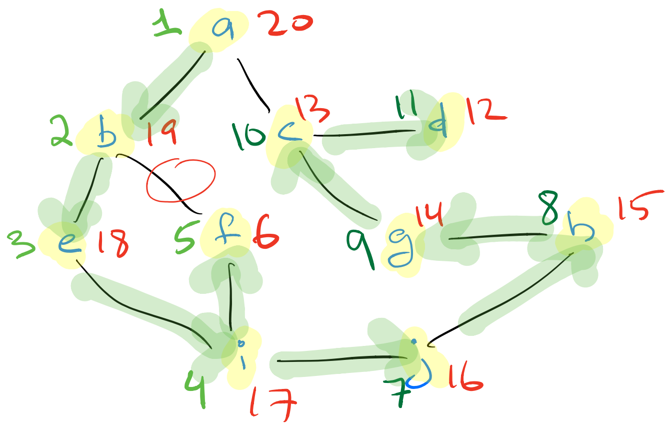
DFS (G= (V,E))
PreProcess (G)
  for v ∈ V
    if unmarked v
      RDFS (G, v)

RDFS (G, v)
  if v is unmarked
    mark v
    pre visit v)
    for u ∈ N(v)
      RDFS(G, u)
    post visit (v)

Preprocess (G)
  clock = 0 ← global variable

Previsit (v)
  clock = clock + 1
  v.pre = clock

Post Visit (v)
  clock = clock + 1
  v.post = clock



pre = [ ]

v.pre → pre[v]

order vertices by v.pre → pre-ordering

order vertices by v.post → post-ordering of ST

At a time clock=t, state of vertex v

- new if $t < v.pre$
- active if $v.pre \le t < v.post$
- finished if $t \ge v.post$

u → v in G

---

if v is new when we visit u

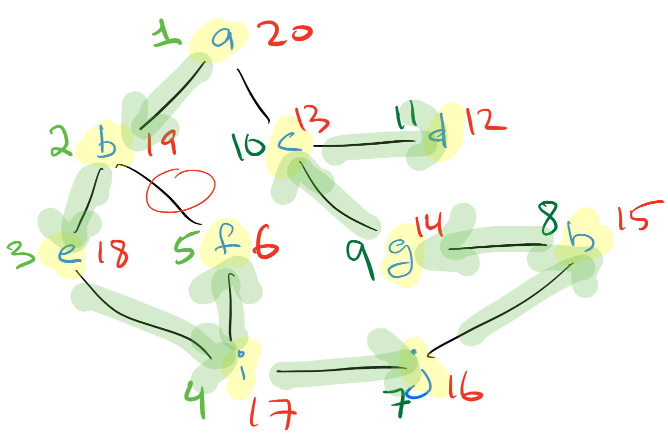
$u.pre < v.pre$

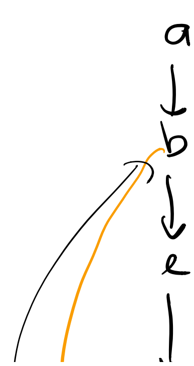
u → v is a: tree edge if its in DFST

forward edge o/w

RDFS (G, v)

```
if v is unmarked
  mark v
  pre visit v)
  for each edge (v,u)
    RDFS(G,u)
  post visit (v)
```





2. if v is active when DFS(u) is called

u → v is a backward edge $f \to b$