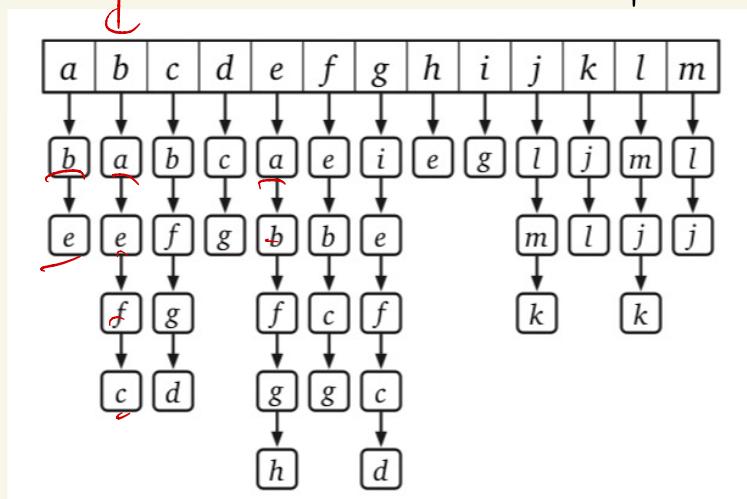



Graph Searching

How can we tell if 2 vertices are connected?

Remember, the computer only has:



Is g
connected
to m?

Bigger question: Can we tell
if all the vertices are
in a single connected
component?

Possibly you saw depth first search (DFS) and breadth first search (BFS) in data structures:

WHATEVERFIRSTSEARCH(s):

put s into the bag
while the bag is not empty
take v from the bag
if v is unmarked
mark v
for each edge vw
put w into the bag



These are essentially just search strategies:

How can we decide if u + v are connected?

Q: What "bag"?

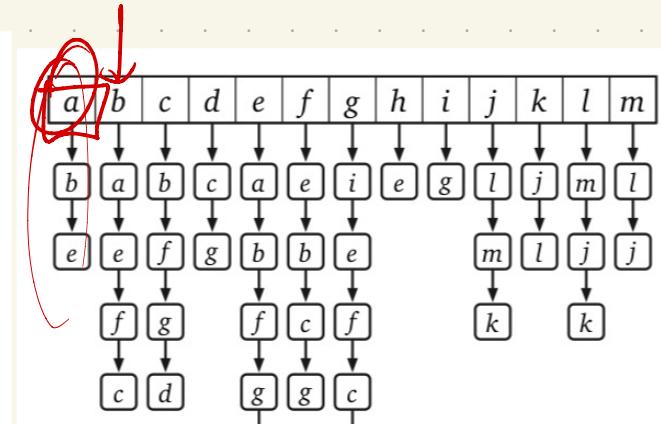
lots of data structures!

Can use this to build a
Spanning tree

WHATEVERFIRSTSEARCH(s):

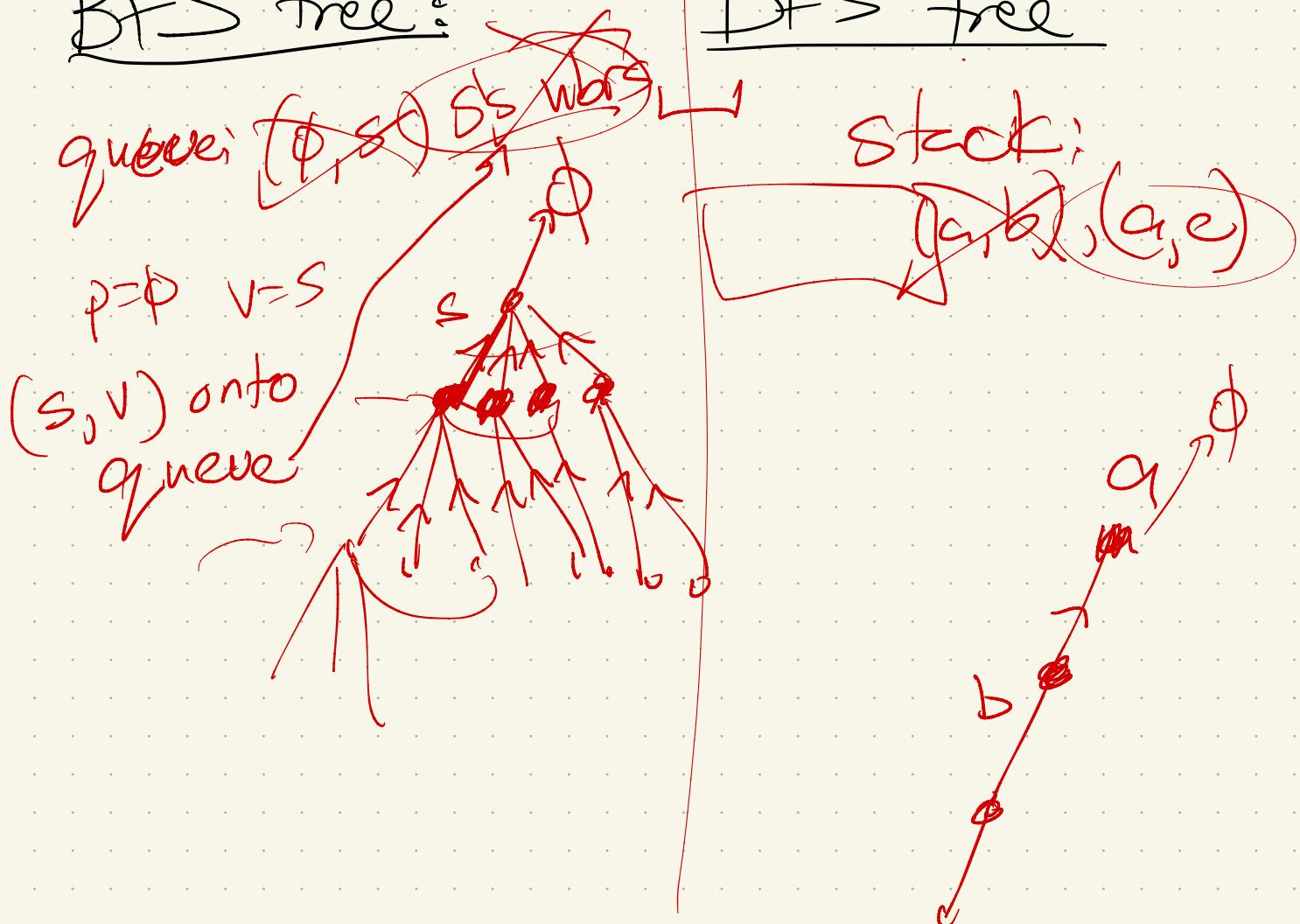
```

put  $(\emptyset, s)$  in bag
while the bag is not empty
    take  $(p, v)$  from the bag      (★)
    if  $v$  is unmarked
        mark  $v$ 
         $\text{parent}(v) \leftarrow p$ 
        for each edge  $vw$           (†)
            put  $(v, w)$  into the bag (★★)
    
```

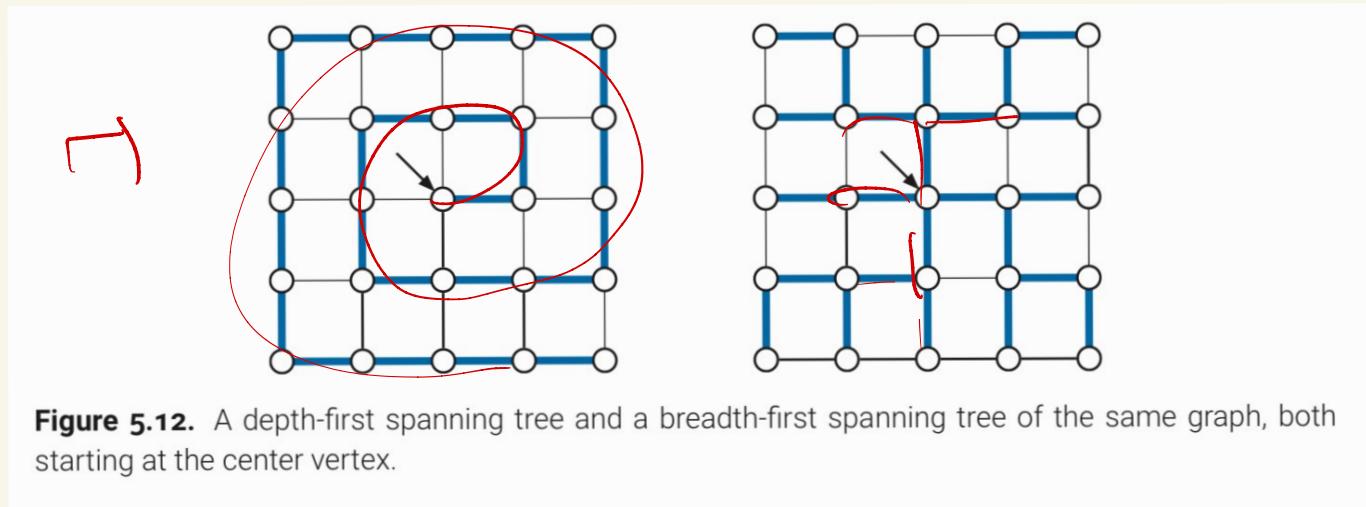


BFS tree

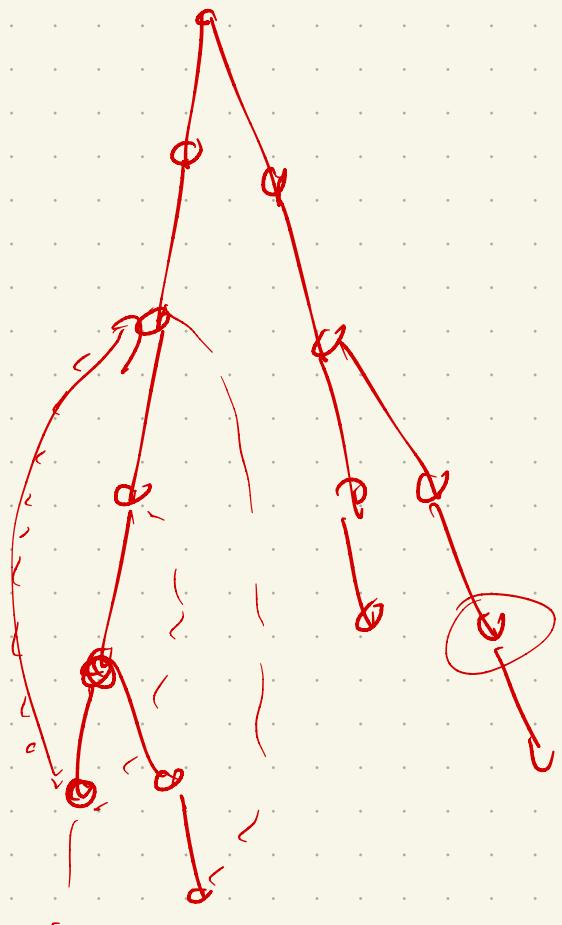
DFS tree



Just remember: different!



DFS:



BFS:

"bushy":
all of s's
nbrs are in
level 1,
level i has all
nodes w/ path of
length i to s

Runtime:

WHATEVERFIRSTSEARCH(s):

put s into the bag
while the bag is not empty
 take v from the bag
 if v is unmarked
 mark v
 for each edge vw
 put w into the bag

Correctness:

Claim: WFS will mark all reachable vertices.

Pf: induction on distance to the source:

$d=0$:

$d > 0$: Consider v at distance d , so $S \rightarrow v_1 \rightarrow v_2 \dots \rightarrow v_d \rightarrow v$
in G. 

By IH:

That means

V_{d-1} WCS

worked!

WHATEVERFIRSTSEARCH(s):

put s into the bag
while the bag is not empty
 take v from the bag
 if v is unmarked
 mark v
 for each edge vw
 put w into the bag

Claim: marked v's + parents
form a spanning tree.
(See demo's...)

Proof:

WHATEVERFIRSTSEARCH(s):

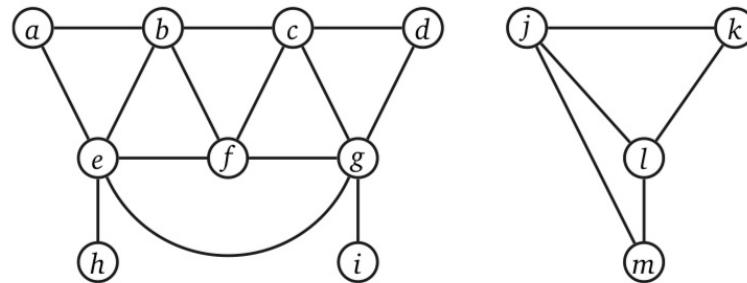
put (\emptyset, s) in bag
while the bag is not empty
 take (p, v) from the bag (\star)
 if v is unmarked
 mark v
 $\text{parent}(v) \leftarrow p$
 for each edge vw $(†)$
 put (v, w) into the bag $(\star\star)$

For each marked vertex:

In a disconnected graph:

Often want to count or
label the components
of the graph!

(WFS(v) will only visit
the piece that v
belongs to.)



Solution: Call it more
than one time!
unmark all vertices
for all vertices v :

Modification: Might want to
Count the # of connected
components?

COUNTCOMPONENTS(G):

count $\leftarrow 0$

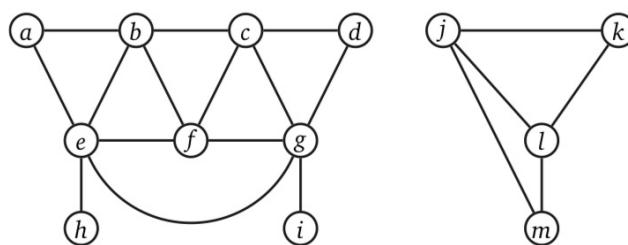
for all vertices v
 unmark v

for all vertices v
 if v is unmarked

count \leftarrow **count** + 1

 WHATEVERFIRSTSEARCH(v)

return count



Finally, can even record which component each vertex belongs to:

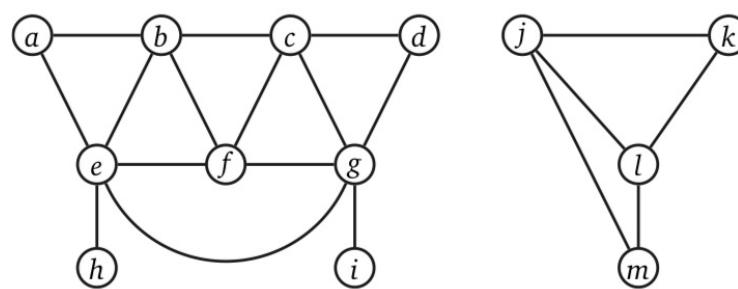
COUNTANDLABEL(G):

```
count  $\leftarrow 0$ 
for all vertices  $v$ 
    unmark  $v$ 
for all vertices  $v$ 
    if  $v$  is unmarked
        count  $\leftarrow$  count + 1
        LABELONE( $v, count$ )
return count
```

$\langle\langle$ Label one component $\rangle\rangle$

LABELONE($v, count$):

```
while the bag is not empty
    take  $v$  from the bag
    if  $v$  is unmarked
        mark  $v$ 
        comp( $v$ )  $\leftarrow$  count
        for each edge  $vw$ 
            put  $w$  into the bag
```



Dfn : Reduction

A reduction is a method of solving a problem by transforming it to another problem.

Note: you've seen/done this
in other classes!

We'll see a ton of these!

(Especially common in graphs...)

Key:

First example:

Given a pixel map, the flood-fill operation lets you select a pixel & change the color of it & all the pixels in its region.

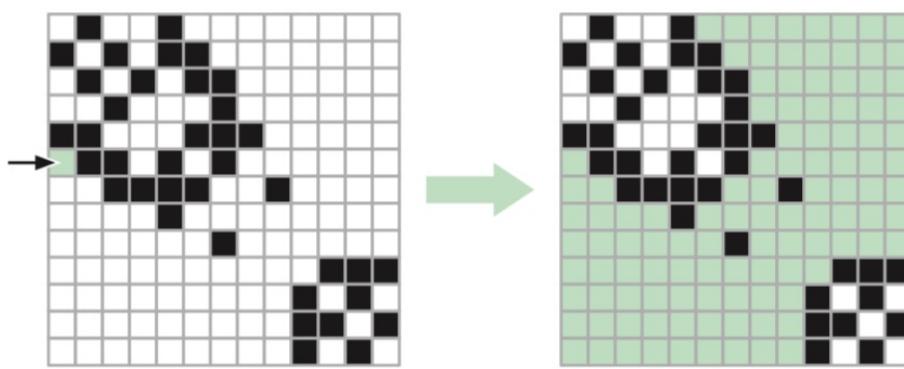
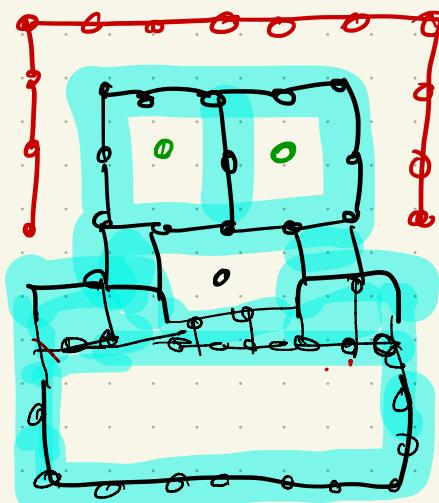
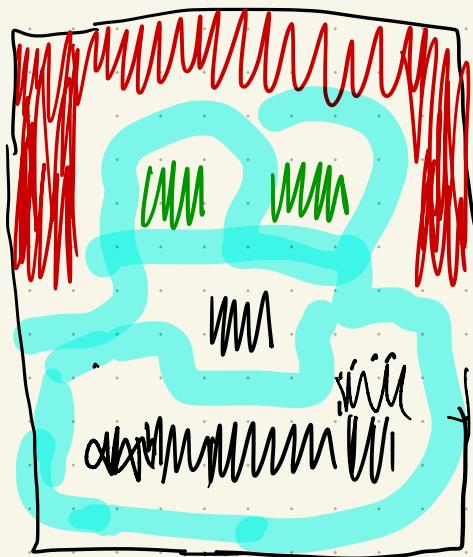


Figure 5.13. An example of flood fill

How?

So: Build a graph from pixels:



Algorithm

If pixel p is selected:

Arguably, these reductions
are the most important
thing in graphs!

Like data structures - you
won't usually have to
re-code everything.

Instead:-

- Set up graph
- Call some algorithm

So runtime / correctness:

Next chapter:

All about directed graphs!

First, though, some things to recall: graph traversals.

- Pre-order
- Post-order:
- In-order: