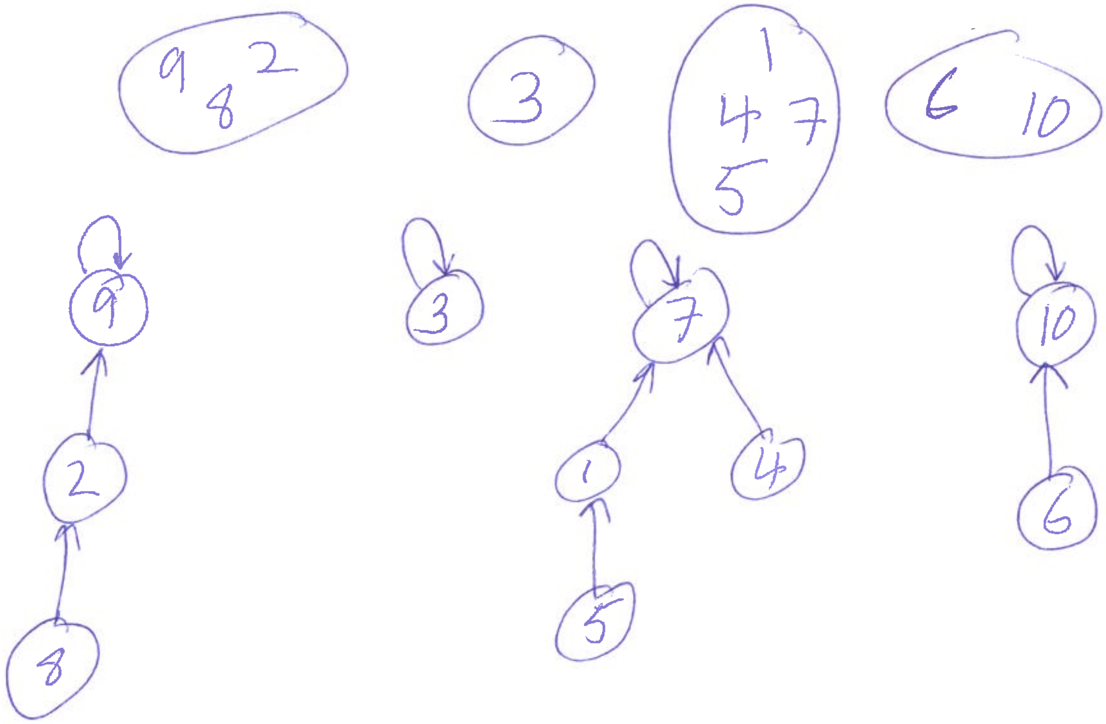
CSCE 750
11/9/2023

Disjoint Set Systems (cont.)
Graph Algos

Forest of trees to rep a disj. set system:



MakeSet(x)   x.parent := x

Find(x)   ~~while~~ if x.parent == x then ~~return~~ x
    else Find(x.parent)

Union(x, y)
    s := Find(x)
    t := Find(y)
    s.parent := t

Want to keep depth small: Include a rank attribute with each node. Currently x.rank is the depth of the (sub)tree rooted at x.

(Revised) MakeSet(x):

```
    x.parent := x
    x.rank := 0
```

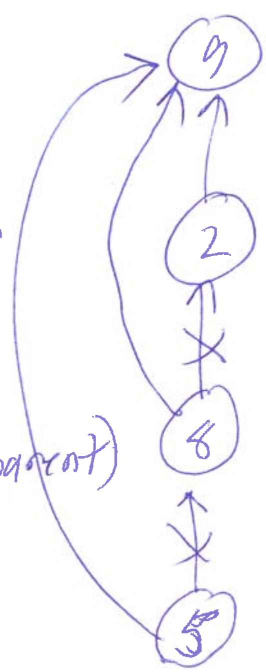
Find(x) [unchanged]

```
Union(x, y)
    s := Find(x)
    t := Find(y)
    if s.rank < t.rank then
        s.parent := t
    if t.rank < s.rank then
        t.parent := s
    else    // ranks are equal
        s.parent := t
        t.rank ++
```

"Union by rank"

Path compression

```
Find(x)   (revised)
  if x.parent == x then
      return x
  else
      x.parent := Find(x.parent)
      return x.parent
```



Find(5)

rank may be ~~less~~ more than the depth of the tree
because of path compression, but
~~Un~~ Union by rank still runs provably fast without change.

Analysis is nontrivial.

Cost of a sequence of n operations is

$$O(n\,\alpha(m)) \quad \text{where } m = \#\text{ of MakeSet ops}$$

and α is the inverse Ackerman function.

$\lim_{m\to\infty} \alpha(m) = \infty$ but α grows so slowly

that for $m \le 16^{512}$, $\alpha(m) \le 4$.

"practically constant"

Graph Algos

Graph representations & terminology
BFS & DFS

A directed graph G, also called a digraph, has 2 attributes:

$G.V$ — finite set of vertices $\{v_1, \ldots, v_n\}$

$G.E$ — set of edges (ordered pairs) ~~from~~

$$G.E \subseteq G.V \times G.V$$

$u, v \in G.V$

$u \xrightarrow{e} v$ means $e = (u, v) \in G.E$

(Undirected) graph: edges connect pairs of distinct vertices;

no self-loops

Representing digraphs

$G.V = \{v_1, \ldots, v_n\}$

1) Adjacency matrix A ($n \times n$) 0-1 matrix
(a_{ij})

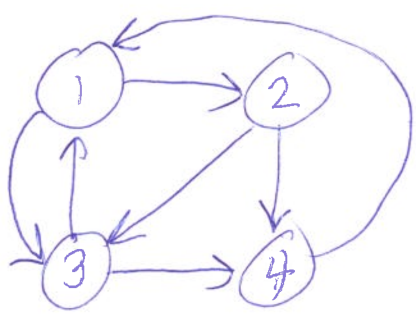
so that

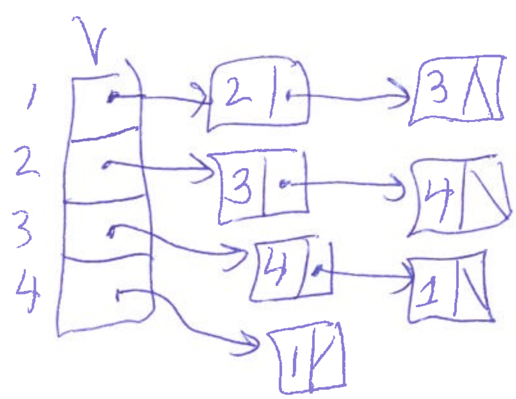
$$a_{ij} = \begin{cases} 1 & \text{if } (v_i, v_j) \in G.E \\ 0 & \text{otherwise} \end{cases}$$

size $\Theta(n^2)$

2) Adjacency lists rep: $V[1..n]$ of pointers.

$V[i]$ points to a linked list of the edges leaving v_i





$\Theta(n+m)$ where $m := |G.E| \leq n^2$

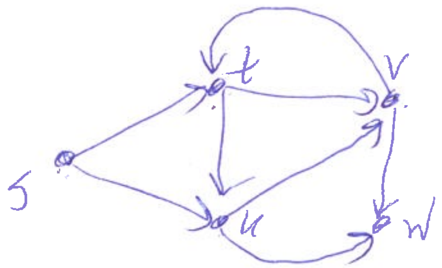
By default, we assume G is represented by adjacency lists. Take the size of G to be $n+m$

[an algo running in time $\Theta(n+m)$ is linear time]

Represent an undirected graph by a digraph as follows:

BFS(v) [v ∈ G.V]

Systematically visits all vertices reachable from v in order of increasing shortest path length.

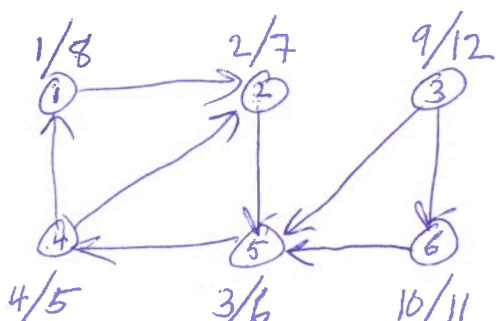


BFS(s) visits (in order)

s, t, u, v, w
0 1 1 2 2

BFS(v) runs in linear time ~~time $O(r)$ where r is the number of vertices reachable from v.~~

DFS runs a subroutine DFSfrom(v) ~~until unv~~ until all vertices are "finished".



DFS:
 DFSfrom(1);
 DFSfrom

[start/finish]

[start, finish] intervals

1: [1, 8]
2: [2, 7]
3: [9, 12]
4: [4, 5]
5: [3, 6]
6: [10, 11]

Can't have

[ ]
 [ ]

linear time.