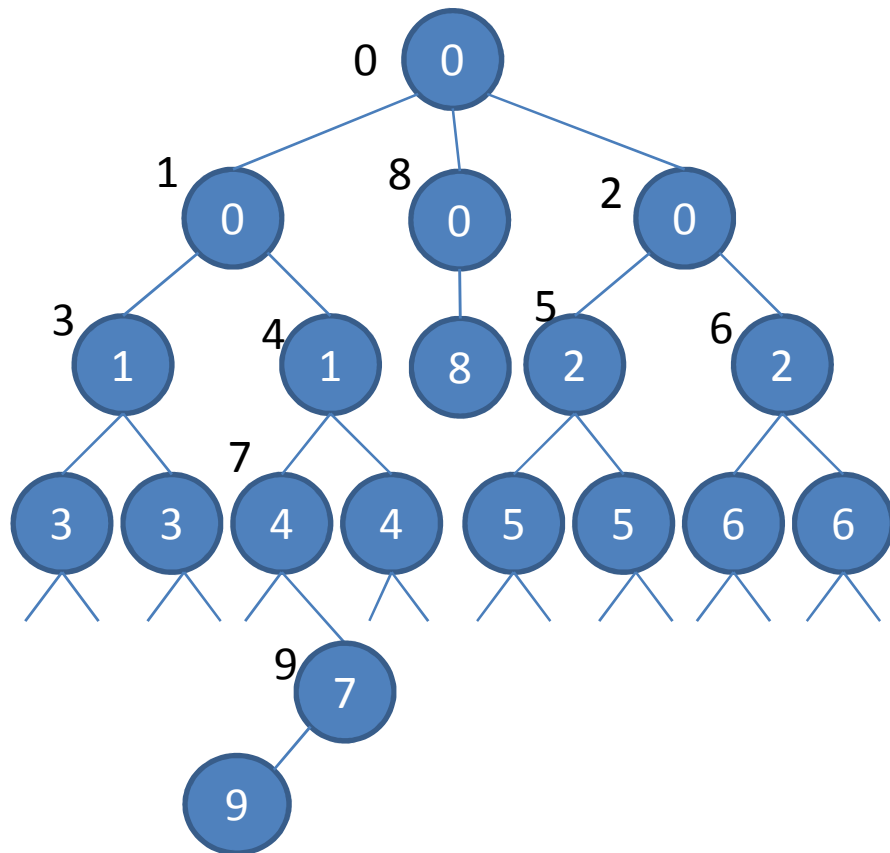


# CSL 860: Modern Parallel Computation

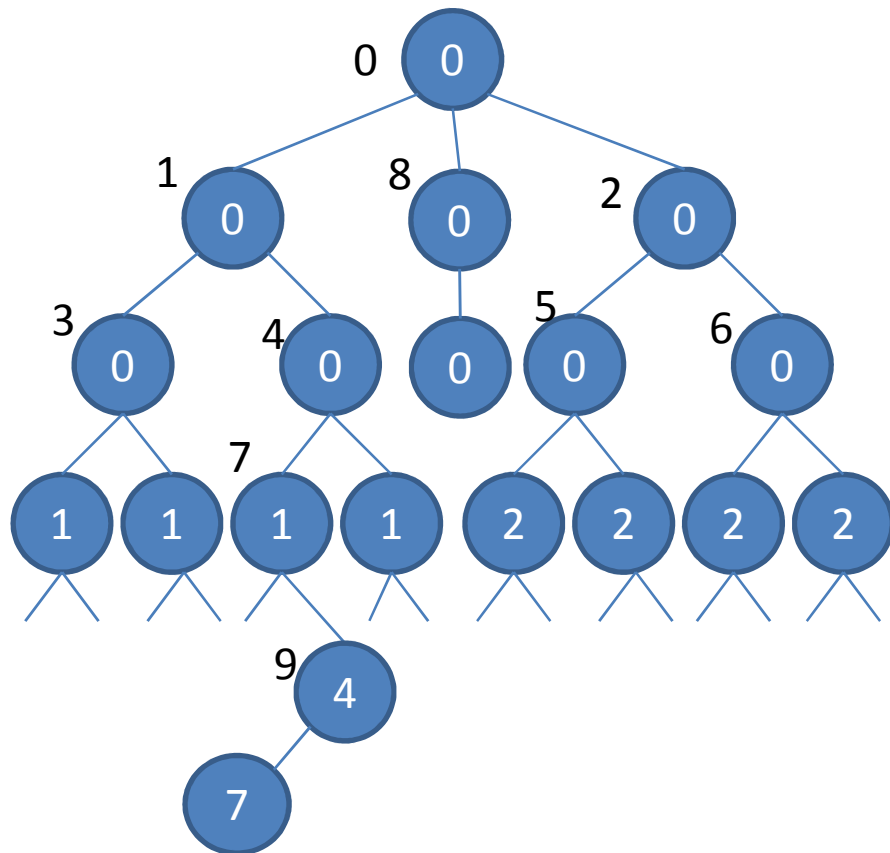
**PARALLEL ALGORITHM TECHNIQUES:  
PATH DOUBLING / POINTER JUMPING**

# Find Roots in a Forest



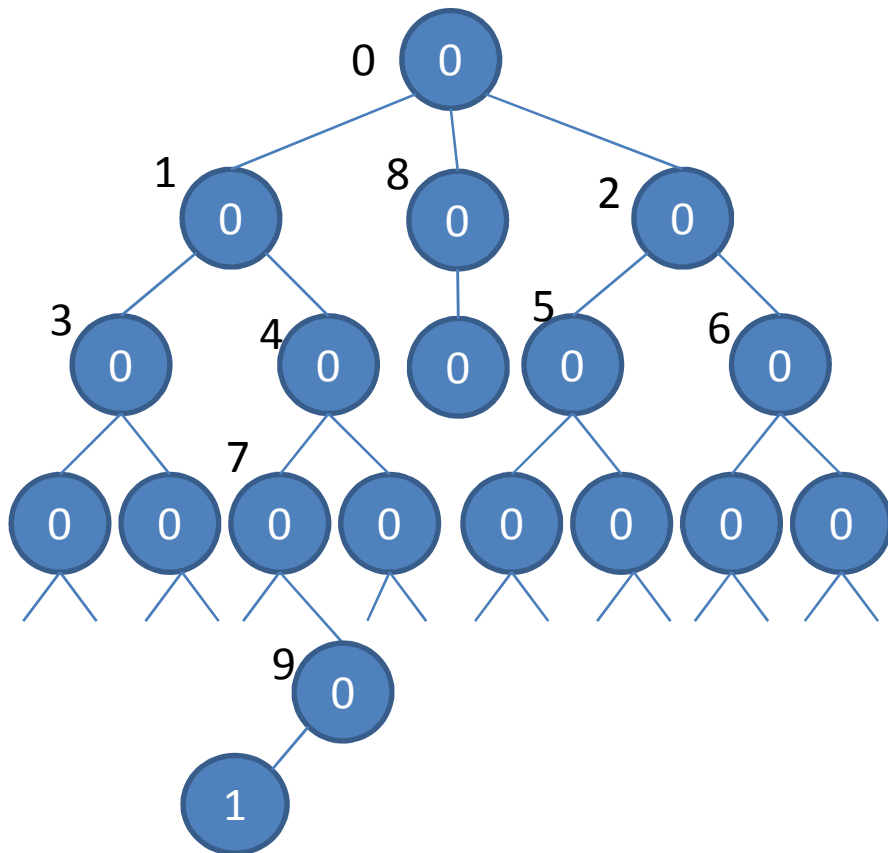
- $p(i)$ : parent of node  $i$
- Do in parallel
  - $p(i) = p(p(i))$
- Stop if  $p(i) = i$
- Time:  $\log(\text{height})$
- Work:  $n \log(\text{height})$

# Find Roots in a Forest



- $p(i)$ : parent of node  $i$
- Do in parallel
  - $p(i) = p(p(i))$
- Stop if  $p(i) = i$
- Time:  $\log(\text{height})$
- Work:  $n \log(\text{height})$

# Find Roots in a Forest



- $p(i)$ : parent of node  $i$
- Do in parallel
  - $p(i) = p(p(i))$
- Stop if  $p(i) = i$
- Time:  $\log(\text{height})$
- Work:  $n \log(\text{height})$

# Pointer Jumping

- Progressively push computation to all elements at a given distance
  - Doubling the distance at each step
  - After  $k$  steps the computation has been performed for elements within a distance of  $2^k$
- Applies to array, list, tree

# List Ranking

- Compute the number of nodes before node  $i$  in a list

Parallel for all  $i$  do:

if  $\text{next}[i] = \text{null}$  then  $d[i] \leftarrow 0$

else  $d[i] \leftarrow 1$

Parallel for all  $i$  do:

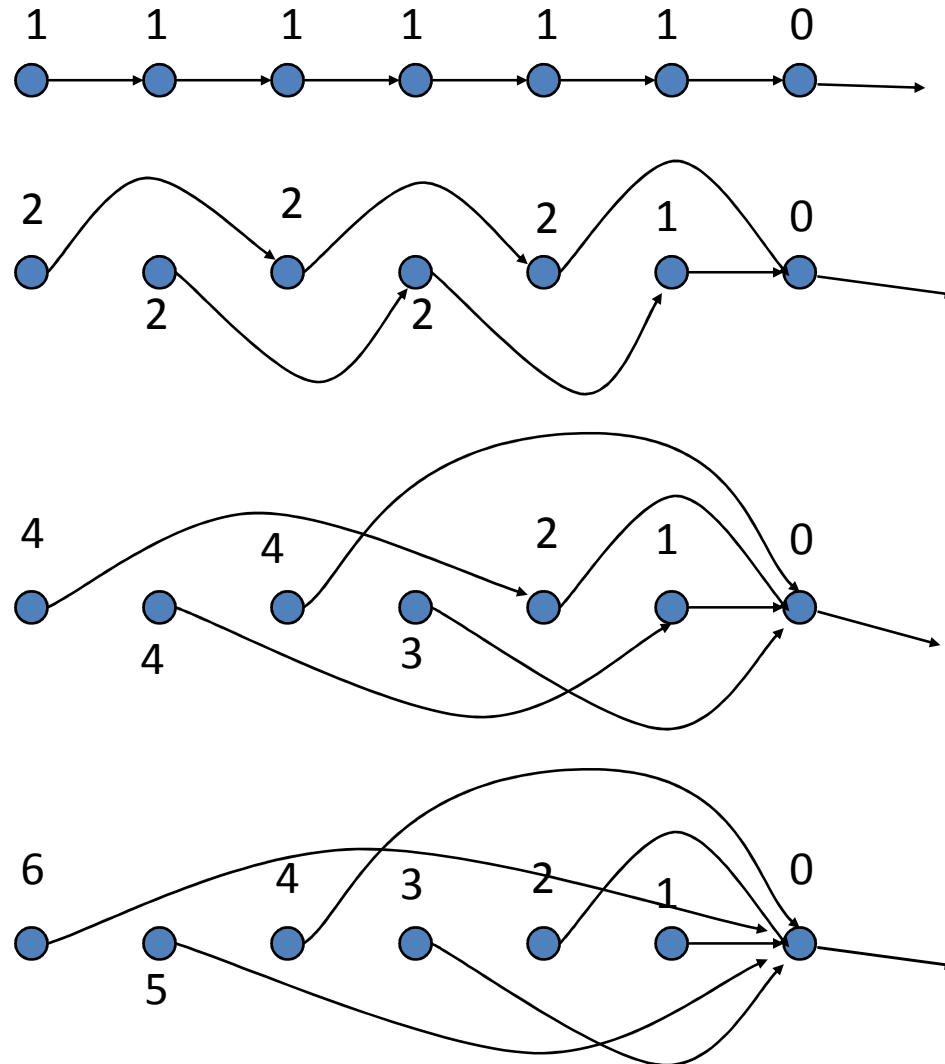
if  $\text{next}[i] \neq \text{null}$

$d[i] \leftarrow d[i] + d[\text{next}[i]]$

$\text{next}[i] \leftarrow \text{next}[\text{next}[i]]$

until  $\text{next}[i] == \text{null}$  for all  $i$

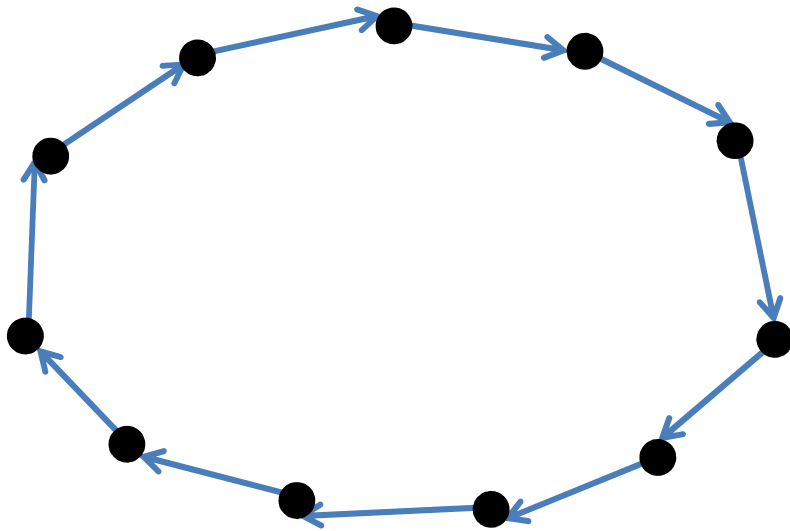
$d(i)$  = distance of  $i$  from  
the end of the list



**$O(n \log n)$  work,  
 $O(\log n)$  time**

# **PARALLEL ALGORITHM TECHNIQUES: SYMMETRY BREAKING**

# Cycle Coloring

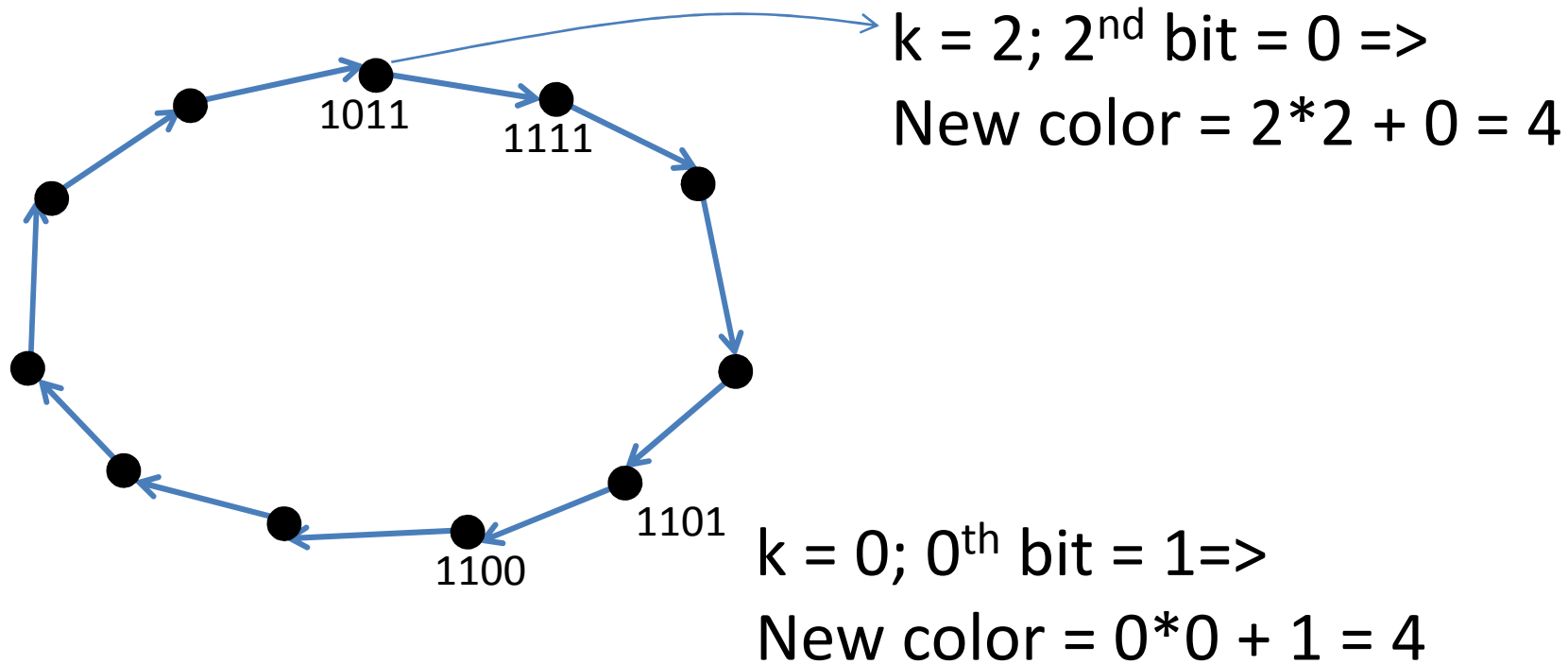


- $C[i] \neq c[j]$  if  $e[i] = j$ 
  - $e[i]$  = The next vertex from  $i$
- $O(n)$  Sequential algorithm
- Parallel:
  - Make local decision
  - But all vertices look the same locally
    - Break symmetry
    - Put vertices in identifiable classes

# 3-Coloring

- Use vertex index to break symmetry
  - But too many indices
- Algorithm:
  - Initialize  $c[i] = i$
  - Iterate, reducing #colors each time
  - parallel for all  $i$  do
    - Let  $k =$  Least significant bit in which  $c[i]$  and  $c[e[i]]$  differ
    - Set  $c[i] = 2^k + \text{bit } k \text{ of } c[i]$

# 3-Coloring



**Why does this  
produce a coloring?**

# 3-Coloring

- #new colors =  $O(\log(\text{\#old colors}))$ 
  - Time  $O(\log^* n)$
  - Work  $O(n \log^* n)$
- But can't reduce the number of colors below 6
- Three more steps of removing the extra colors
  - parallel for all  $i$ 
    - If  $(c[i] = i)$ 
      - $c[i] = \{0,1,2\}$  and NOT  $(c[[e[i]], c[\text{predecessor}[i]])$