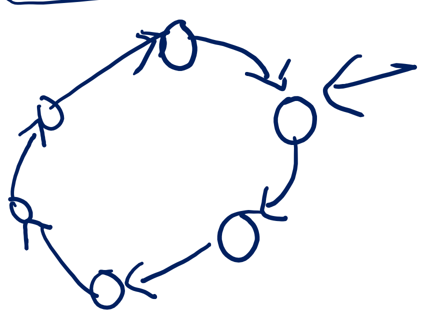
Symmetry Breaking

Parallel
distributed computing

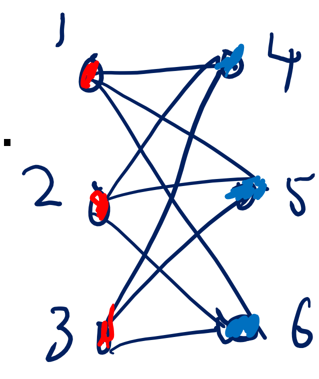
- A way to induce differences between like (symmetric) participants.

n nodes
m edges

- Useful in applications such as graph coloring
  - Generally, difficult using deterministic techniques.
    - Need randomization
- Special cases where fast, deterministic symmetry breaking can be achieved.
    - Linked lists and directed cycles are an example.



2 colors
n nodes

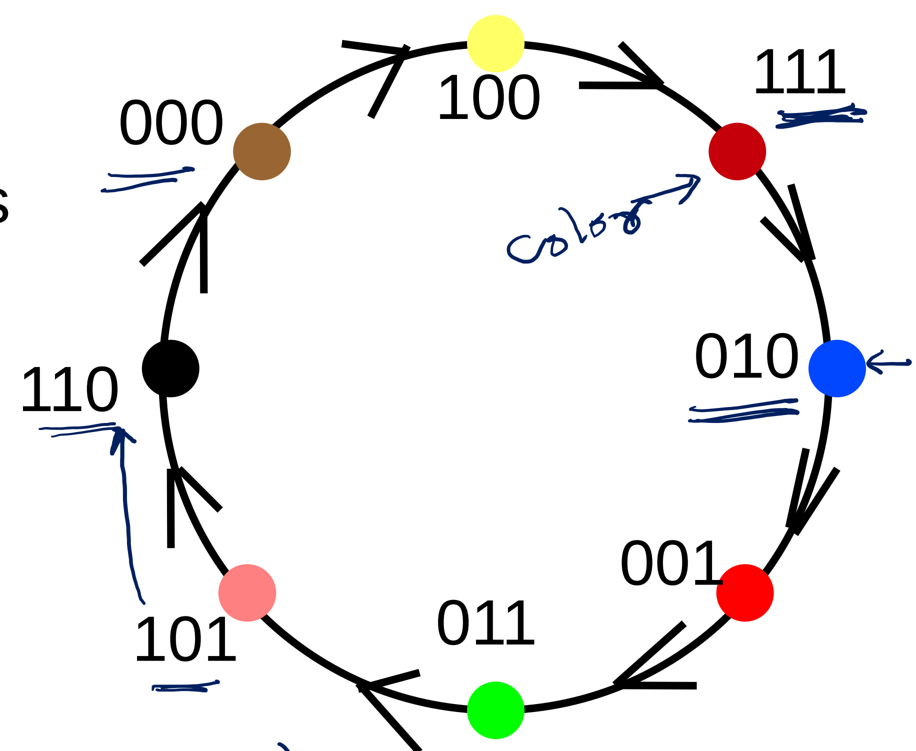


$K_{n,n}$

Coloring by Symmetry Breaking

using 3-colo

- Consider a directed cycle of n nodes numbered 1 to n.
- Treat the number of the node as its initial color.
- Can reduce colors from n to log n in one step.
  - Every node u compares its color with that of the successor, and recolors as:
  - Newcolor(u) = 2k + color(u)$_k$
  - k is the index of the first bit position that u and v differ from LSB
  - E.g., for 101 and 111, k = 1.



Color(u)

color of u

8-node directed cycle

→ colored with 8 colors

color(u): $0110\boxed{1}01$

$i : 2$ $\quad$ $color(u)_2 = 1$

u is a node

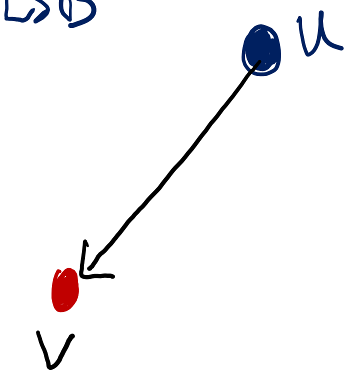
color(u) : color value in binary

$color(u)_i$: i^{th} bit from LSB in color(u)

$k \leftarrow$ LSB

color(u): $01101\boxed{0}1$

color(v): $11010\boxed{1}1$



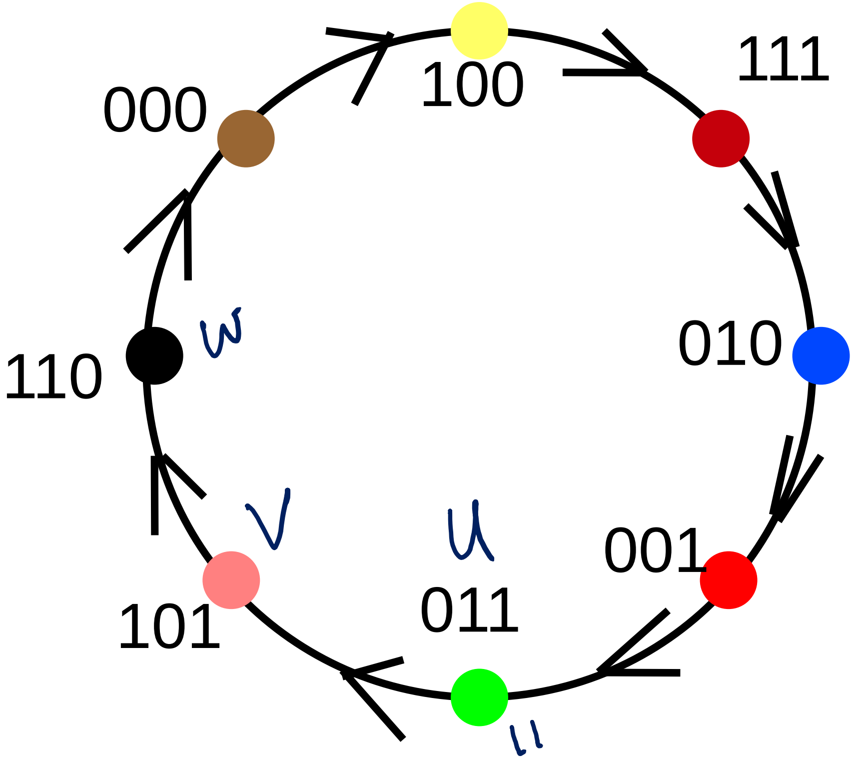
index of LSB = 0

$k = 1$

$$color(u)_k = 0$$

Coloring by Symmetry Breaking

u: 0 1 1
v: 1 0 1
k = 1



u	v	New Color(u)
110	000	11 (k = 1)
000	100	100 (k = 2) ←
100	111	00 (k = 0)
010	001	00 (k = 0)
001	011	10 (k = 1)
011	101	11 (k = 1)
111	010	01 (k = 0)
v 101	110 w	01 (k = 0)

k=0
v: 1 0 1
w: 1 1 0

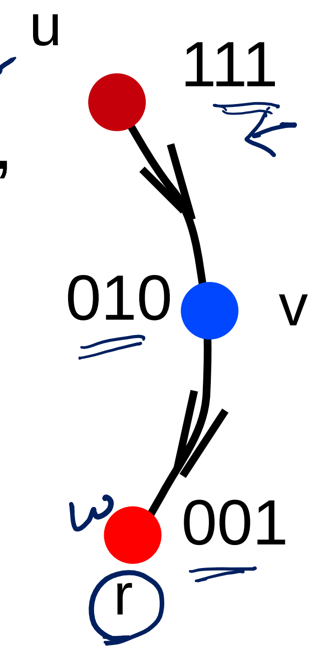
$2\times 0+1 = 1 = (01)_2$

$2\times 1+1 = 3 = (11)_2$

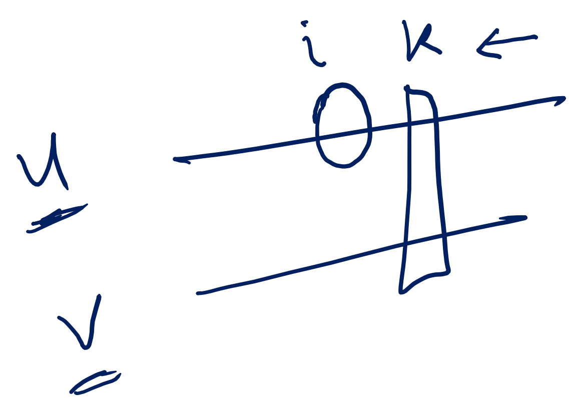
- Consider a directed cycle of n nodes numbered 1 to n.
- Treat the number of the node as its initial color.
- Can reduce colors to log n in one step.
  - Compare its color with the successor, $\text{Newcolor}(u) = 2k + \text{color}(u)_k$
  - k is the index of the first bit position that u and v differ from LSB

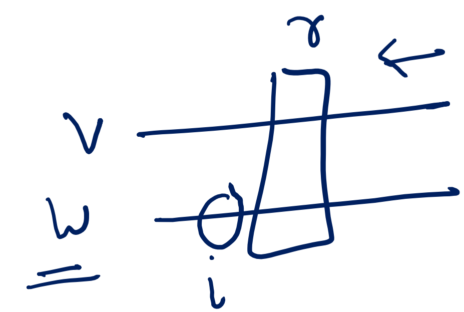
Coloring by Symmetry Breaking

- Claim: The new colors are valid. ← No two nbrs have the same color
- Proof: Suppose that for u and v such that u ≠ v, NewColor(u) = NewColor(v).
- Let NewColor(u) = 2k + color(u)$_k$, and NewColor(v) = 2r + color(v)$_r$.
- Let k = r. However, color(u)$_k$ ≠ color(v)$_k$. Why?
- Let k ≠ r. Then, color(u)$_k$ – color(v)$_r$ = 2(r – k).
  - The LHS has an absolute value of at most 1 and the RHS has an absolute value of at least 2.



$k \le (\log n - 1)$





max 2k + color(u)$_k$
= 2(log n − 1) + 1

Coloring by Symmetry Breaking

$n = 2^t$

- In one iteration, can reduce the number of colors from n to 2log n – 1 < 2 log n.
  - Initial colors are log n bits
  - New colors are only 1+ [loglog n] bits.
- Can we repeat again?
  - Yes.
  - Reduces number of bits from t to 1+[log t].
  - But, at some point t < 1 + [log t]. No advantage any further.
  - Happens at t = 3.
- So, repeat till only 8 colors are used.

n colors

log n bits

n

2 log n

2 log log n

2 log log log n colors

Coloring by Symmetry Breaking

- At that point, can still reduce the number of colors as follows:
- For i = 8 downto 3 in sequence
  - If node u is colored i, then u chooses a color among {1,2,3} that is not same as the colors of its neighbors.
- Possible to do so. Why?

8, 7, 6, 5, 4, 3

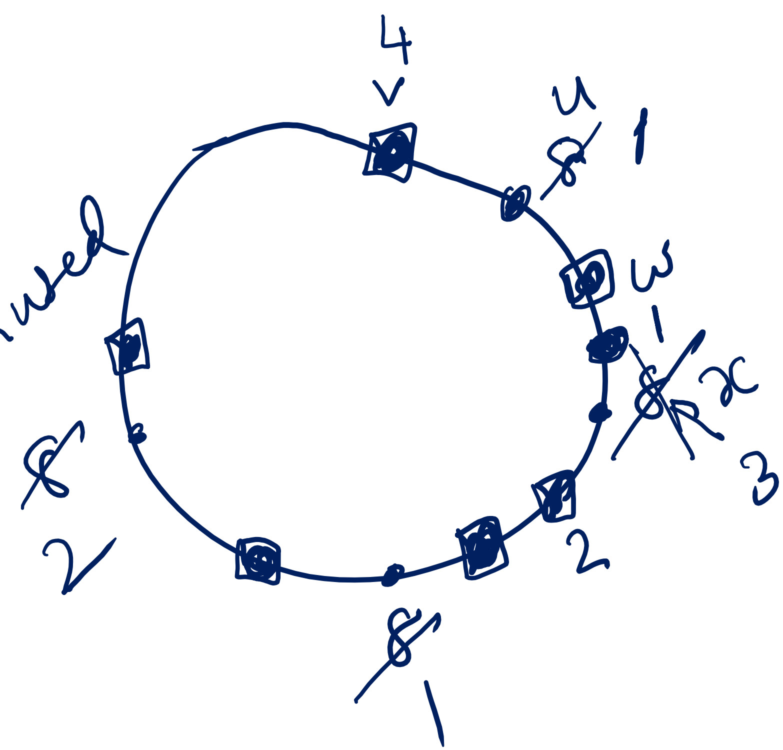
6 iterations

Color(u) = 8

color(v) ≠ 8

Color(w) ≠ 8

color(u) = Smallest unused color



Coloring by Symmetry Breaking

- At that point, can still reduce the number of colors as follows:
- For i = 8 downto 3 in sequence

not counting

  - If node u is colored i, then u chooses a color among {1,2,3} that is not same as the colors of its neighbors.
- Possible to do so. Why?
  - Each node has only two neighbors.
  - So, only some two colors amongst {1,2,3} can be used up already.

$T(n) = T(\log n) + 1$

$= ??$

Coloring by Symmetry Breaking

O

- Total time analyzed as follows:
  - Each iteration of symmetry breaking reduces number of bits from to 1+⌈log t⌉.
  - The recurrence relation is $T(n) = T(\log n) + 1$
  - Solution: $T(n) = O(\log^* n)$.
  - In the next phase, only 5 iterations.
  - So, overall time = $O(\log^* n)$
- Work however is $O(n\log^* n)$.
- $\log^* n = i$ such that

$$\underbrace{\log(\log(\ldots\ldots(\log n)))}_{i} = 1;$$

growing slow

n	$\log^* n$
16	3
2^{16}	4
$2^{2^{16}}$	5

- The algorithm extends to lists and rooted trees also.

$32 \xrightarrow{\log} 5 \xrightarrow{\log} 2+\varepsilon$

Coloring to Independent Sets $\rightarrow$

- For bounded degree graphs colored with O(1) colors, a coloring is equivalent to finding a large independent set.

$\Delta = O(1)$

- Iterate on each color and count the number of nodes with a given color.
- Pick the subset of like colored nodes of the largest size.
  - Clearly, an independent set.
  - Has a size of at least a fraction of n.

$G = (V, E)$

$U \subseteq V$

for i = 1 to c do

Pick all nodes of color i

Remove their nbrs

Ind. set

if no two elements of U are nbrs

List Ranking

- List ranking is a fundamental problem in parallel computing.
- Given a list of elements, find the distance of the elements from one end of the list.
- In sequential computation, not a serious problem.
  - Can simply traverse the list from one end.
- But this approach does not scale well for parallel architectures.

List Ranking

List

$1 \rightarrow 8 \rightarrow 5 \rightarrow 11 \rightarrow 2 \rightarrow 6 \rightarrow 10 \rightarrow 4 \rightarrow 3 \rightarrow 7 \rightarrow 12 \rightarrow 9$

Succ	8	6	7	3	11	10	12	5	--	4	2	9
Rank	1	5	9	8	3	6	10	2	12	7	4	11

- Representation via an array of successor pointers.