W reduces to $O(n)$

As we switch to fast algo now, we need n' processors — where $n' = \frac{n}{\log \log n}$

and $W(n') = O(n' \log \log n')$

$= O\left(\frac{n}{\log \log n} \underbrace{\log \log \frac{n}{\log \log n}}\right)$

$= O(n)$

$\log (\log n - \log \log \log n)$

$< \frac{1}{2} \log \log n$

Time : $\left[\cancel{O(\log \log n')} \approx \cancel{O(\log \log n)} \right]$

$\hookrightarrow \underbrace{O(\log \log \log n)}_{\text{BT algo 1}} + \underbrace{O(\log \log n)}_{\text{DL algo 2}}$

So, Time $= O(\log \log n)$

Lecture 18 : 16 March 2021

Symmetry Breaking < Parallel ~~Symmetry Breaking~~ distributed computing

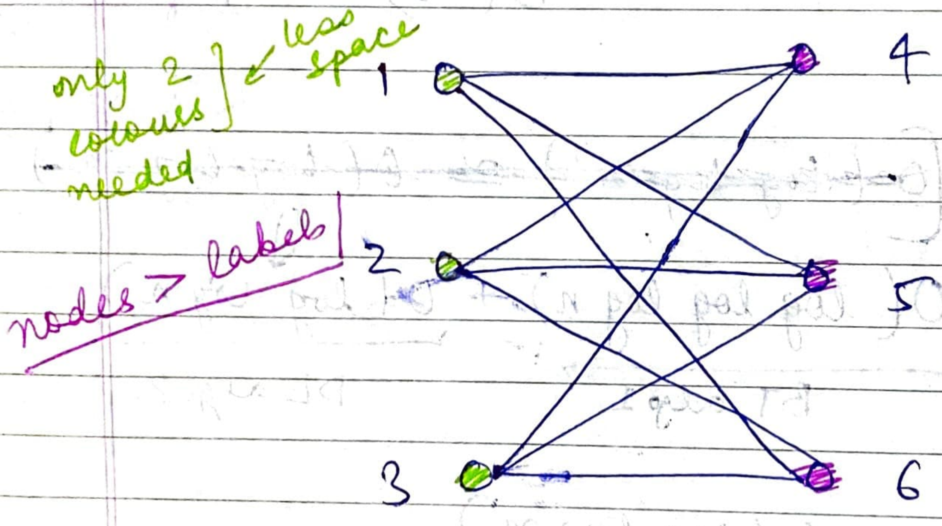
└ a way to induce differences b/w like (symmetric participants).

→ used in graph coloring

→ symmetric participants like nodes or edges in a graph.

the one way to segregate edges/nodes are identifiers / labels / numbers for each node / edge but it takes up a lot of space.

eg. Complete Barpelite graph
graph colouring



There are special cases where fast deterministic sym. breaking can be achieved.
└ directed cycle / CLL

Colouring by SB

8-node directed cycle

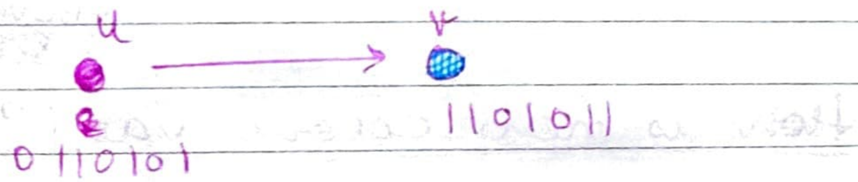
n nodes → n colours
↓ reduce to
x colours

1. Directed C of nodes
2. Treat no. of node as its colours
3. To reduce colours from n to $\log n$
Every node u compares its colour & its successor and recolours as:

Imagine "u" is a node
colour (u) = colour value in binary
colour$(u)_i$ = i^{th} bit from LSB in colour(u).

eg. colour (u) = 0 1 1 0 $\boxed{1}$ 0 1
colour $(u)_2$ = 1

Successor of u is v



color(u) = 0 1 1 0 1 $\boxed{0}$ 1
color(v) = 1 1 0 1 0 $\boxed{1}$ 1

pick index where they differ first from LSB that index is (k)

k = 1

so, $\text{colour}(u)_k = 0$

Date
Page

$$\boxed{\text{new colour}(u) = 2k + \text{colour}(u)_k}$$

eg:

$u = 011$
$v = 101$
$k = 1$
$\text{newcol}(u) = 2 \times 1 + 1 = 2+1 = 3$
$= 011$ (base 2)

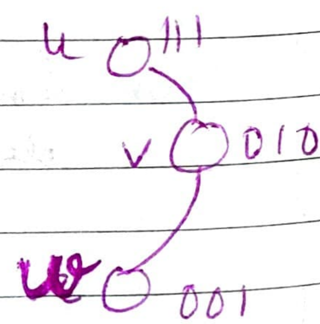
$v = 101$
$w = 110$
$k = 0$
$\text{new}_v = 2 \times 0 + 1 = 1 = 01$ (base 2)

$\boxed{\text{Why is this correct?}}$

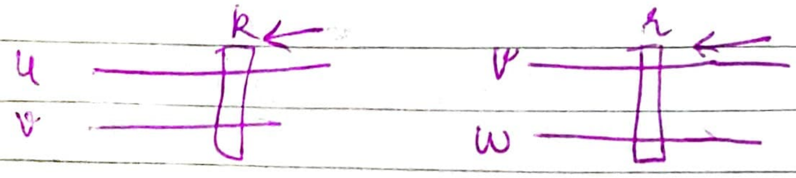
Claim = New colours are valid
no two neighbours have same colour

How is new colour valid?

Proof = Suppose $u \neq v$ k
$\text{newColour}(u) = \text{NewC}(v)$



$\Rightarrow nc(u) = 2k + c(u)_k$
$nc(v) = 2k + c(v)_k$



we have no control of colours of u, v, w

at some index k where $u_k = v_k$,
w_k might be u_k
so, its possible $\Rightarrow$

Case 1: $k = r$

But, $colour(u)_k \neq color(v)_k$

$\Rightarrow new(u) \neq new(v)$ ← contradict^n

~~$new(u) = 2k + c(u)_k$~~
~~$new(v) = 2k + c(v)_k$~~
~~$= 2k + c(v)_k$~~

~~so for $new(v) \neq new(u)$~~

$c(u)_k \neq c(v)_k$

Case 2: if $k \neq r$

assumpt^n: $n(u) = n(v)$

$2k + c(u)_k = 2r + c(u)_r$

$2(r-k) = c(u)_k - c(u)_r$

RHS: at most 1
LHS: at least 2

$\Rightarrow$ contradiction

Date
Page

→ What is the largest value of new colour for any node? if there are n nodes.

$\cancel{nc(u) = 2k + c(u)_k}$

$\cancel{2831}$

$\cancel{= 264}$

if n nodes

no. of bits req. to represent n nodes $= \log n$

So largest value of k = $\log n - 1$

→ $\boxed{k \leq (\log n - 1)}$

So, largest value of nc

$= 2(k) + 1$

$= 2 \log n - 2 + 1$

$\boxed{nc \leq 2 \log n - 1}$

So, in one iteration,

we reduce # colours from

n to $2 \log n$

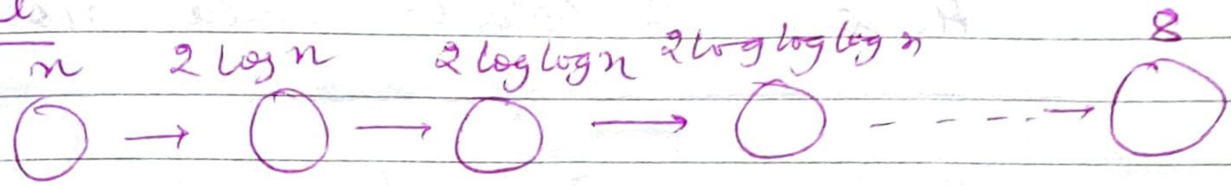
So, no. of bits

colours	# bits
n	$\log n$
$2 \log n$	$1 + \lceil \log \log n \rceil$

Date ______
Page ______

Phase 1

→ Repeat



\# bits decrease from t to $1+\lceil \log t \rceil$

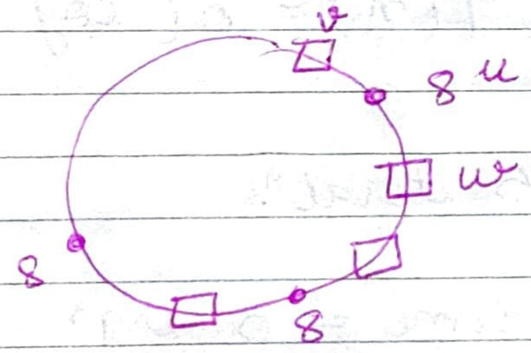
At some point $t < 1+\lceil \log t \rceil$. No advantage any further. This happens at $t=3$.

So, repeat until 8 colours are used.

Phase 2 =

→ At this point we can still reduce # colours

Run another loop → for $i=8$ to ~~4~~ (5 iterat^n^)



colour(u) = 8
" V ≠ 8
" w ≠ 8

↓

so colour(u) = smallest (unused colour by v & w)

So,

for(i=8 to 3) {
if node u is coloured ?
then u chooses colour among {1,2,3} i.e. not same as the colour of its neighbours.
}

Hence, 3 colours used now

→ # iterations in phase 1

colours : $n \to 2\log n \to 2\log\log n \to \cdots \to 8$

bits : $\log n \to 1 + \lceil \log\log n \rceil \to \cdots \to 3$

colours : $x \to 2(\log x)$

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

Soln : $T(n) = O(\log^* n)$

Phase 2 : 5 iterat^ns.

So, Time = $O(\log^* n)$

[$\log^* n = i$ s.t. $\underbrace{\log\log \ldots}_{i \text{ times}} \log(n) = 1$;]

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

$O(\log^* n)$ ↓ grows slowly

In all $\log^* n$ iterations all n nodes are working to recolour themselves to less no. of colours

WORK : $O(n \log^* n)$

The same algo can be extend to rooted trees / linked list.

COLOURING TO INDEPENDENT SETS

Independent Set : $G = (V, E)$

$U \subseteq V$

U is independent if no two elements of U are neighbours of each other

For bounded degree graphs coloured w O(1) colours, a colouring is equivalent to finding a large independent set.

We iterate on each color

for i=1 to C do
    pick all nodes of color i
    remove their neighbors

Size of independent set is at least N/C

Colouring n ~~dots~~ nodes w C colours so $\underbrace{n/c \text{ have common colour}}_{\text{these are independent of each other}}$ ← pigeon hole