4.5 Inserting into a $(2, 3)$-tree

- Step 3, works in phases; one phase for every level of the tree
- Step 4, works in rounds; in each round a different set of elements is inserted

Observation
We can start with phase i of round r as long as phase i of round $r-1$ and (of course), phase $i-1$ of round r has finished.

This is called Pipelining. Using this technique we can perform all rounds in Step 4 in just $\mathcal{O}(\log k + \log n)$ many parallel steps.

4.6 Symmetry Breaking

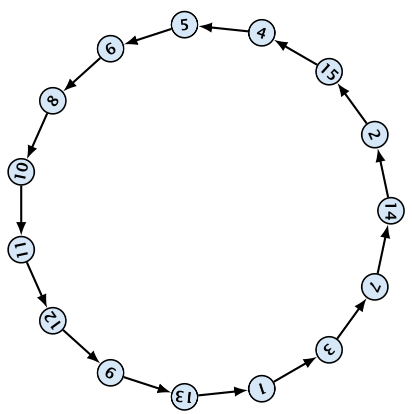
The following algorithm colors an n-node cycle with $\lceil \log n \rceil$ colors.

Algorithm 9 BasicColoring

```
for 1 ≤ i ≤ n pardo
    col(i) ← i
    k_i ← smallest bitpos where col(i) and col(S(i)) differ
    col'(i) ← 2k_i + col(i)_{k_i}
```

(bit positions are numbered starting with 0)

4.6 Symmetry Breaking



v	col	k	col′
1	0001	1	2
3	0011	2	4
7	0111	0	1
14	1110	2	5
2	0010	0	0
15	1111	0	1
4	0100	0	0
5	0101	0	1
6	0110	1	3
8	1000	1	2
10	1010	0	0
11	1011	0	1
12	1100	0	0
9	1001	2	4
13	1101	2	5

4.6 Symmetry Breaking

Applying the algorithm to a coloring with bit-length t generates a coloring with largest color at most

$$2(t-1)+1$$

and bit-length at most

$$\lceil \log_2(2(t-1)+1) \rceil \le \lceil \log_2(2t) \rceil = \lceil \log_2(t) \rceil + 1$$

Applying the algorithm repeatedly generates a constant number of colors after $\mathcal{O}(\log^* n)$ operations.

Note that the first inequality holds because $2(t-1)-1$ is odd.

4.6 Symmetry Breaking

As long as the bit-length $t \geq 4$ the bit-length decreases.

Applying the algorithm with bit-length 3 gives a coloring with colors in the range $0, \ldots, 5 = 2t - 1$.

We can improve to a 3-coloring by successively re-coloring nodes from a color-class:

Algorithm 10 ReColor

```
for ℓ ← 5 to 3
    for 1 ≤ i ≤ n pardo
        if col(i) = ℓ then
            col(i) ← min{{0,1,2} \ {col(P[i]), col(S[i])}}
```

This requires time $\mathcal{O}(1)$ and work $\mathcal{O}(n)$.

4.6 Symmetry Breaking

Lemma 7

We can color vertices in a ring with three colors in $\mathcal{O}(\log^* n)$ time and with $\mathcal{O}(n \log^* n)$ work.

not work optimal

4.6 Symmetry Breaking

Lemma 8

Given n integers in the range $0, \ldots, \mathcal{O}(\log n)$, there is an algorithm that sorts these numbers in $\mathcal{O}(\log n)$ time using a linear number of operations.

Proof: Exercise!

4.6 Symmetry Breaking

Algorithm 11 OptColor

```
1: for 1 ≤ i ≤ n pardo
2:     col(i) ← i
3: apply BasicColoring once
4: sort vertices by colors
5: for ℓ = 2⌈log n⌉ to 3 do
6:     for all vertices i of color ℓ pardo
7:         col(i) ← min{{0,1,2} \ {col(P[i]), col(S[i])}}
```

We can perform Lines 6 and 7 in time $\mathcal{O}(n_\ell)$ only because we sorted before. In general a statement like "**for** constraint **pardo**" should only contain a contraint on the id's of the processors but not something complicated (like the color) which has to be checked and, hence, induces work. Because of the sorting we can transform this complicated constraint into a constraint on just the processor id's.

Lemma 9

A ring can be colored with 3 colors in time $\mathcal{O}(\log n)$ and with work $\mathcal{O}(n)$.

work optimal but not too fast