A friendly introduction to Distributed Algorithms & Locality … through two examples

Jukka Suomela
Aalto University

Distributed algorithms

- You write a program for ***one computer***
  - you can use also commands like “send this message to this communication port” etc.
- Your adversary constructs ***a network of n computers***, all running your program
- Switch everything on, see what happens

Distributed algorithms

- Useful abstraction: identical computers, working in a synchronous manner
- Running time = ***number of communication rounds***
- Key challenge: what to do in the middle of a very large network?

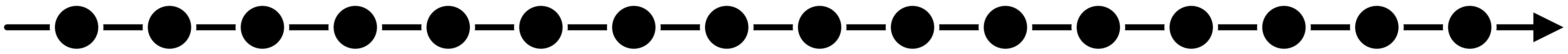
1

Theme:

Symmetry breaking

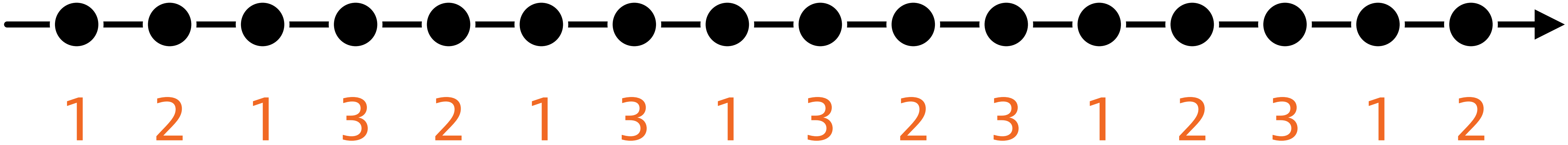
Example:

3-coloring cycles



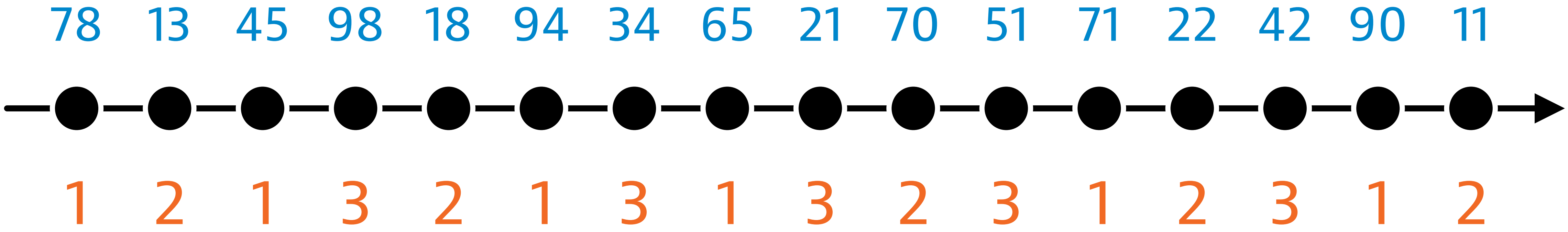
Computer network: directed cycle

All nodes have a well-defined “successor” and “predecessor”

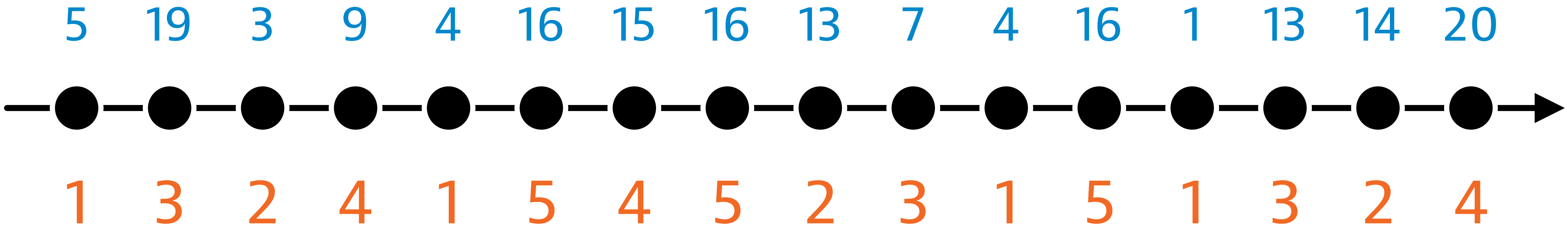


Coloring:

how to break symmetry?



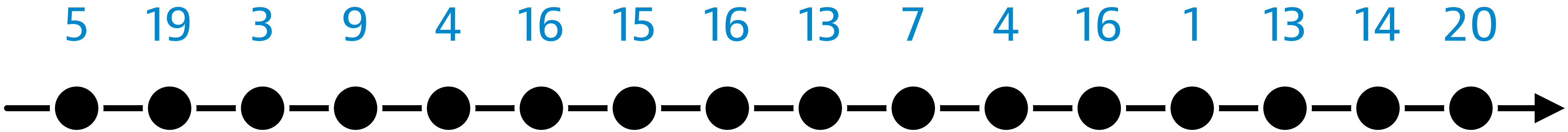
Color reduction:
unique identifiers or random strings
→ poly(n) colors → 3 colors



One color reduction step:

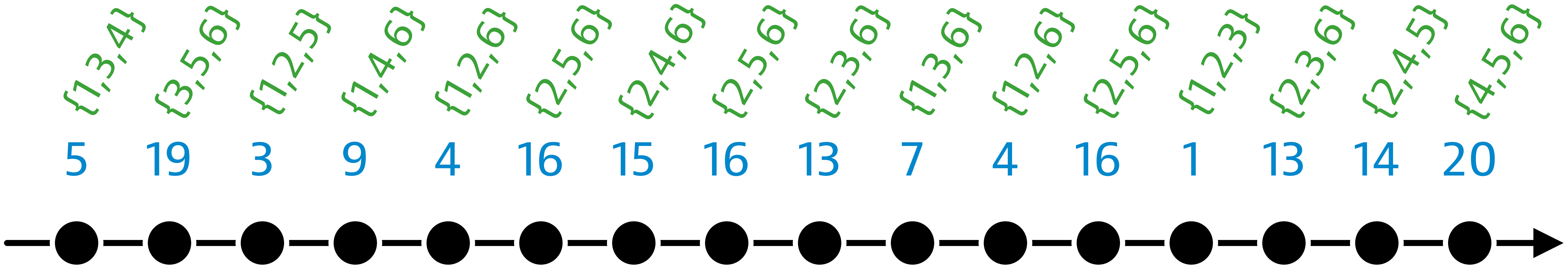
20 colors → 6 colors

???



One color reduction step:

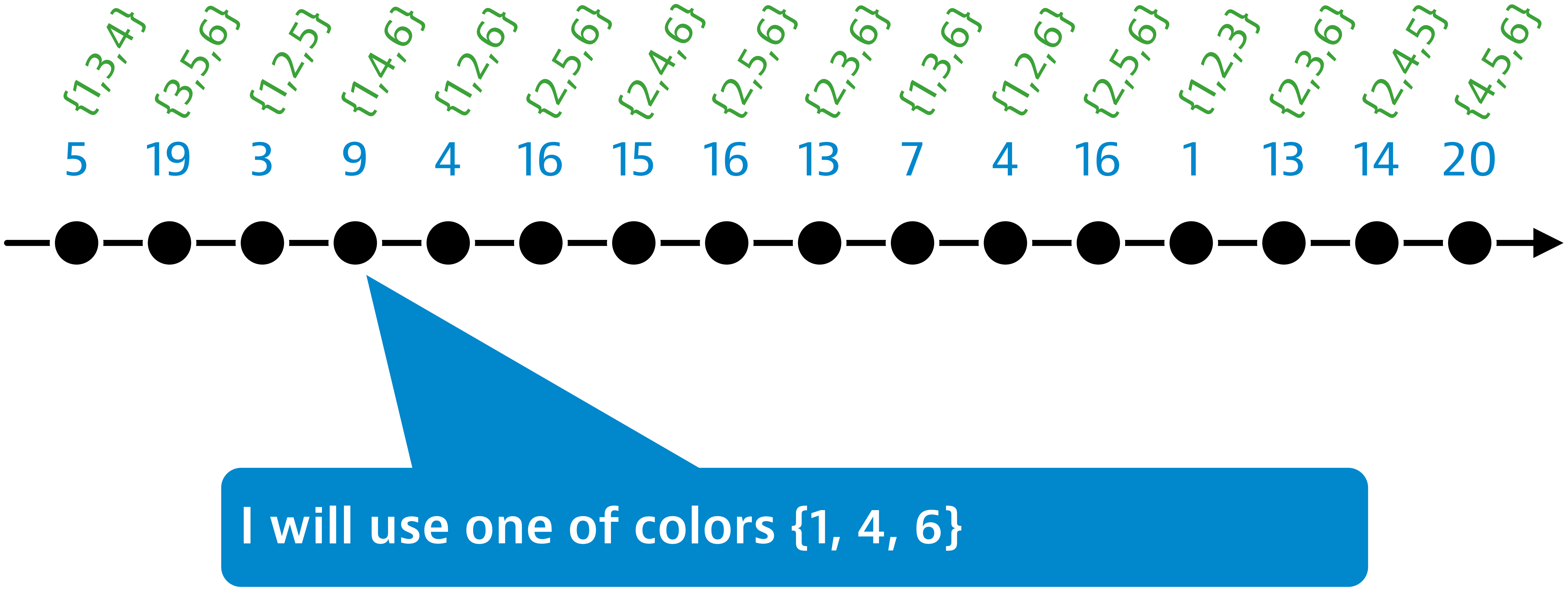
20 colors → …

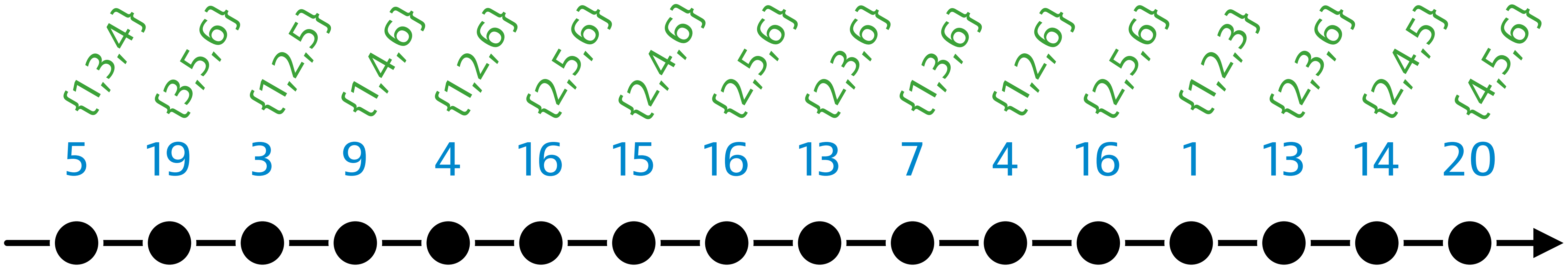


One color reduction step:

20 colors → 3-element subsets of {1, 2, ..., 6}

→ …

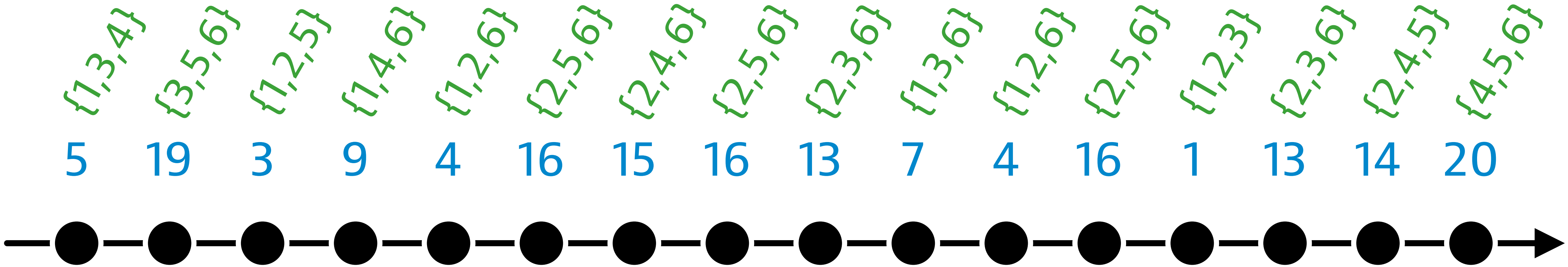

{1,3,4}
{3,5,6}
{1,2,5}
{1,4,6}
{1,2,6}
{2,5,6}
{2,4,6}
{2,5,6}
{2,3,6}
{1,3,6}
{1,2,6}
{2,5,6}
{1,2,3}
{2,3,6}
{2,4,5}
{4,5,6}
5 19 3 9 4 16 15 16 13 7 4 16 1 13 14 20
I will use one of colors {1, 4, 6}



One color reduction step:

20 colors → 3-element subsets of {1, 2, ..., 6}

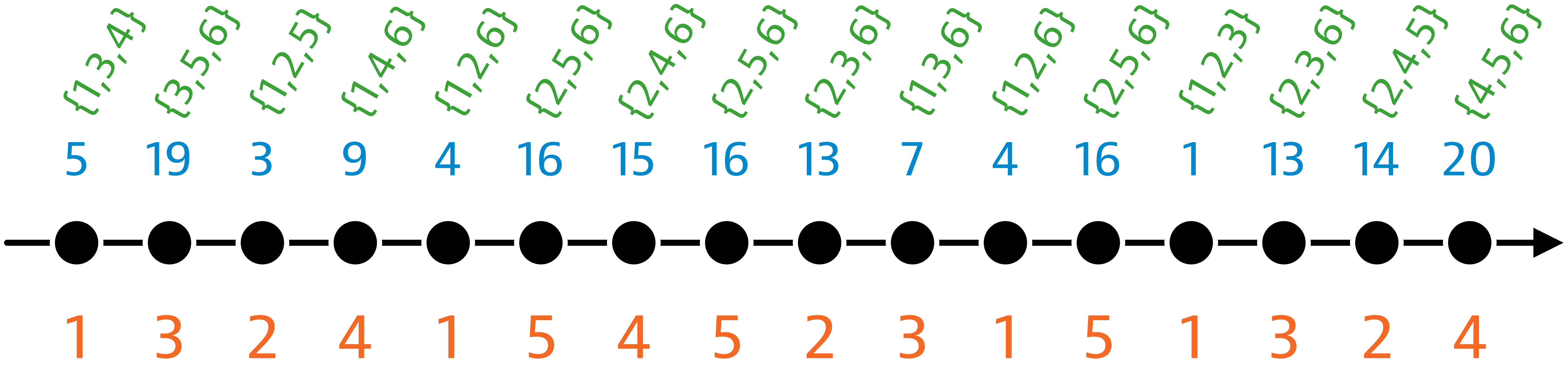
→ one communication round → ...



I promised to use one of colors {1, 4, 6}

My successor will use one of colors {1, 2, 6}

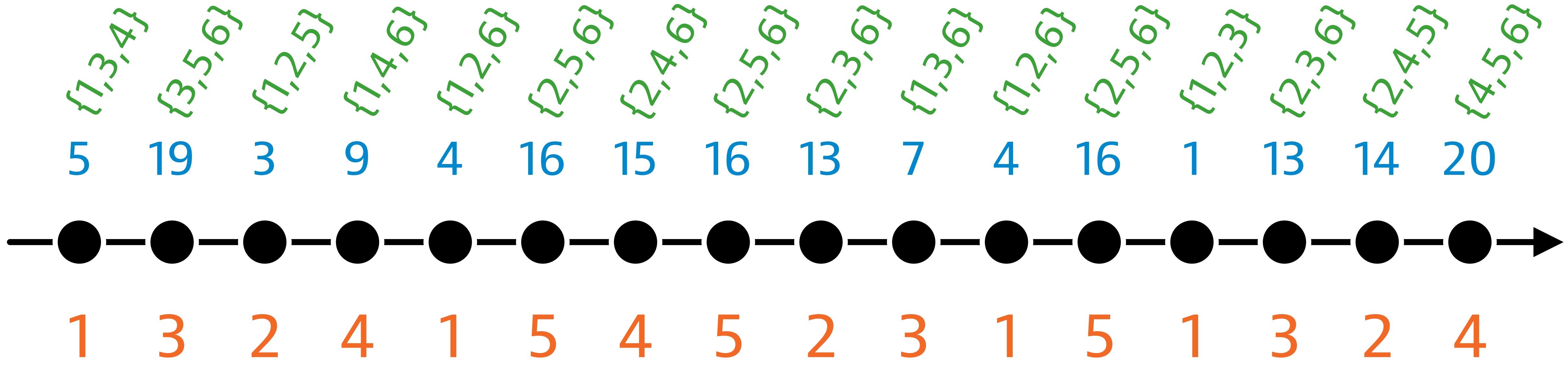
I can therefore safely output 4



One color reduction step:

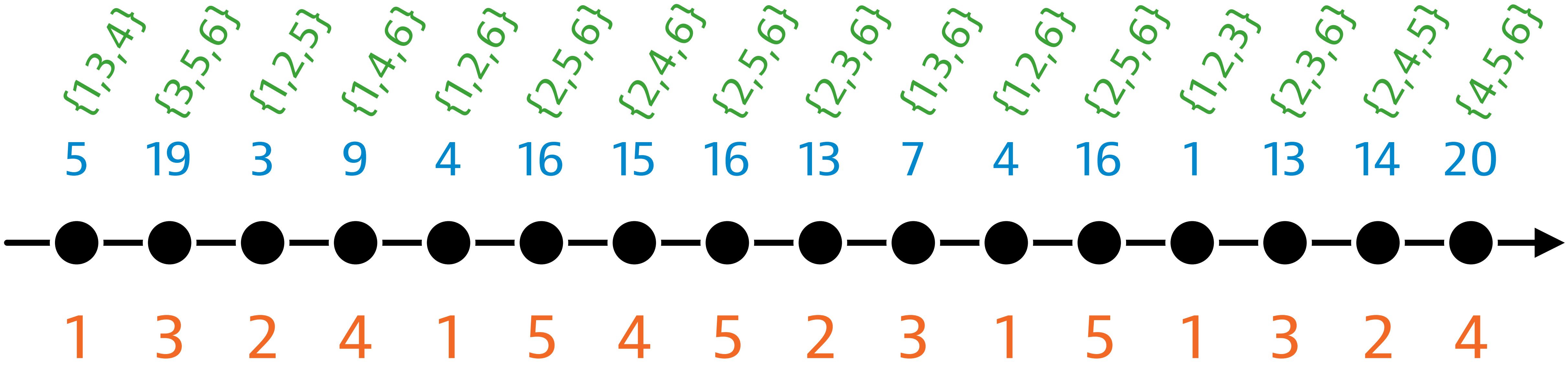
20 colors → 3-element subsets of {1, 2, ..., 6}

→ one communication round → 6 colors



One color reduction step:

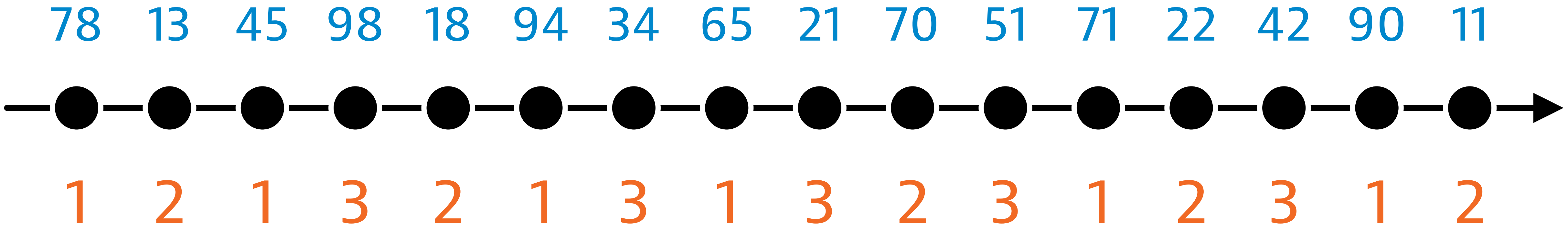
($2k$ choose k) colors → $2k$ colors



One color reduction step:

($2k$ choose k) colors → $2k$ colors

c colors → $O(\log c)$ colors



Color reduction:

unique identifiers or random strings

→ poly(n) colors

→ $O(\log^* n)$ color reduction steps

→ 3 colors

- **Algorithm:** $O(\log^* n)$ rounds
  - ***color reduction:*** c colors in T rounds
    → $O(\log c)$ colors in $T + 1$ rounds
  - poly(n) colors in 0 rounds
    → 3 colors in $O(\log^* n)$ rounds
- **Lower bound:** this is optimal!
  - ***round elimination:*** c colors in T rounds
    → 2^c colors in $T - 1$ rounds
  - 3 colors in $o(\log^* n)$ rounds
    → ≪ poly(n) colors in 0 rounds → impossible

Locality

Each node knows its final output
after ***T* communication rounds**

$\Updownarrow$

Each node can compute its final output
if it knows its ***T*-radius neighborhood**

2

Theme:
Local coordination

Example:
Maximal matching

Maximal matching

- **Setting:**
  - 2-colored graph — “black” and “white” nodes
  - maximum degree Δ
- **Goal:**
  - maximal matching: if you are unmatched, all your neighbors must be matched

Proposal algorithm

- **Black nodes:**
  - send a proposal to 1st neighbor
  - if rejected, send a proposal to 2nd neighbor …
- **White nodes:**
  - wait for proposals
  - accept the first proposal
  - reject all other proposals

- **Algorithm:** $O(\Delta)$ rounds
  - ***proposal algorithm:*** after failed attempts you run out of neighbors to propose and can safely output "unmatched"
- **Lower bound:** this is optimal!
  - ***round elimination:*** maximal matching in T rounds $\rightarrow$ a sloppy matching in $T - 1$ rounds
  - maximal matching in $o(\Delta)$ rounds $\rightarrow$ something nontrivial in 0 rounds $\rightarrow$ contradiction

Summary

3-coloring cycles:

- trivial sequential greedy algorithm
- $O(\log^* n)$-round distributed algorithm
- tight, by round elimination

Maximal matching:

- trivial sequential greedy algorithm
- $O(\Delta)$-round distributed algorithm
- tight, by round elimination