Distributed Computing PWr, WIT

Prof. Mirosław Kutyłowski

Warm up topic: colouring

General information

- **Schedule**: the first half of the semester , 7-8 lectures, each 2x45 minutes
- **Textbook:** Principles of Distributed Computing, Roger Wattenhofer, ETH Zurich

- **Recording:** the lecture will be recorded
- **Blackboard**: its image of the blackboard will be available as a pdf file (separate file for each topic)

- **Exercises:** with dr Gebala, grade based on result from exercises

Focus

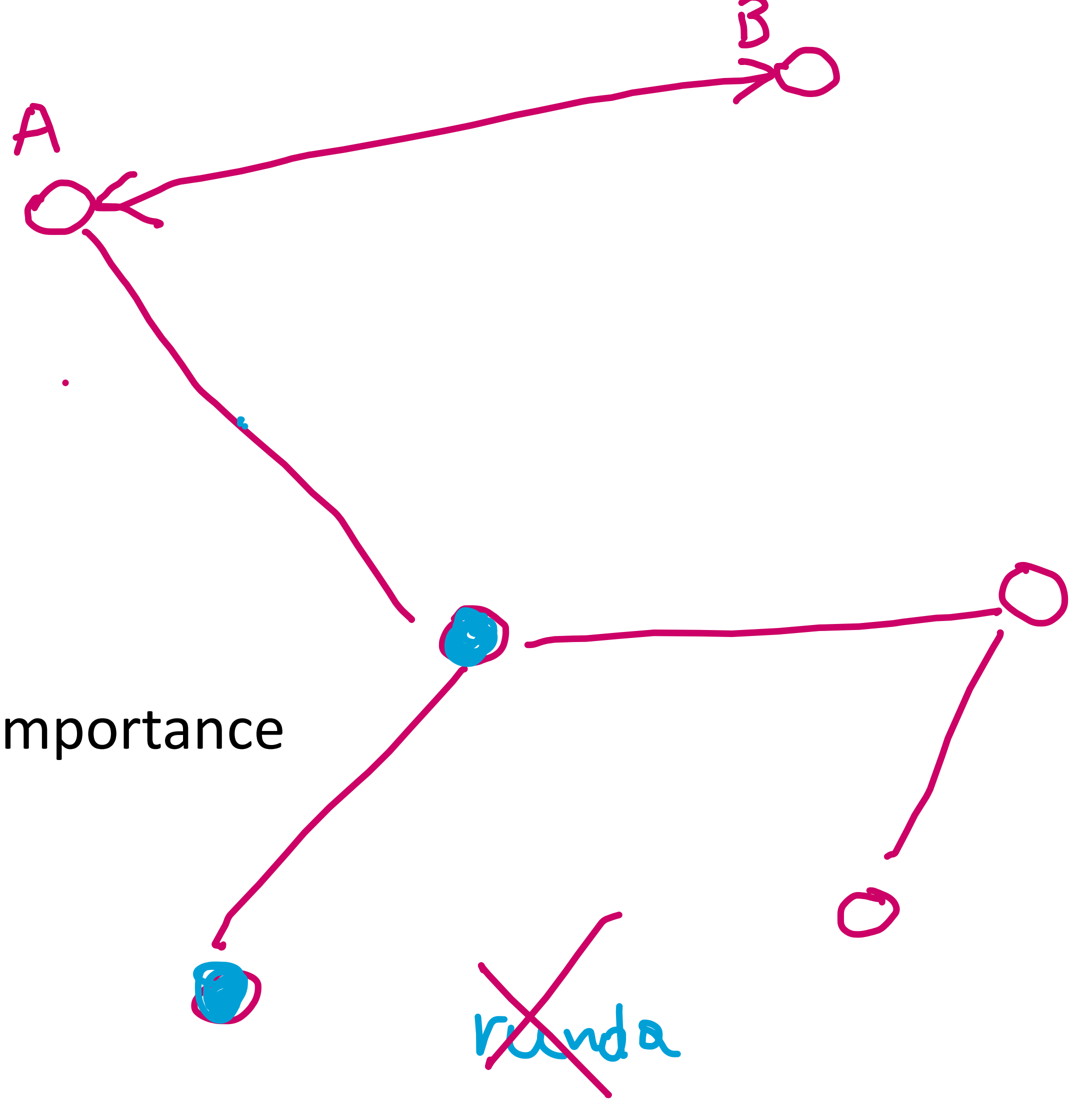
- **Main issue:**
  - creating distributed systems is completely different than designing traditional sequential programs
  - you have to “restart” your brain as a computer engineer

- **Our Focus:** security related issues
- **Topics:**
  - new topics emerge, many more will come in the future,
  - … but once you learn to deal with distributed systems you can handle new challenges

Model

Nodes:

- independent computing units
- the immediate neighbors known
- … but no global view
- local communication cost of minor importance

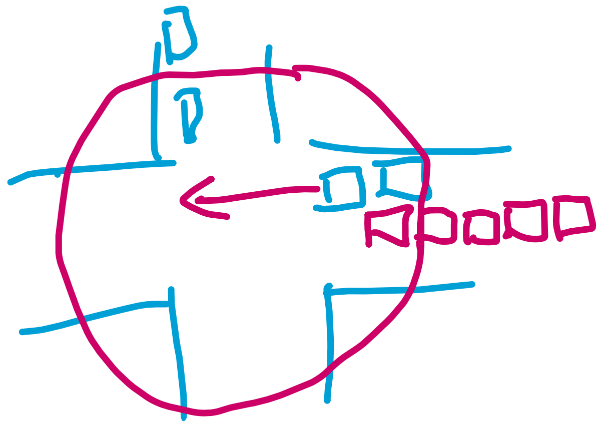


Communication:

- substantial latency
- unpredictable delays
- asynchrony
- communication failures

Model

virtual street lamps



Dynamics:

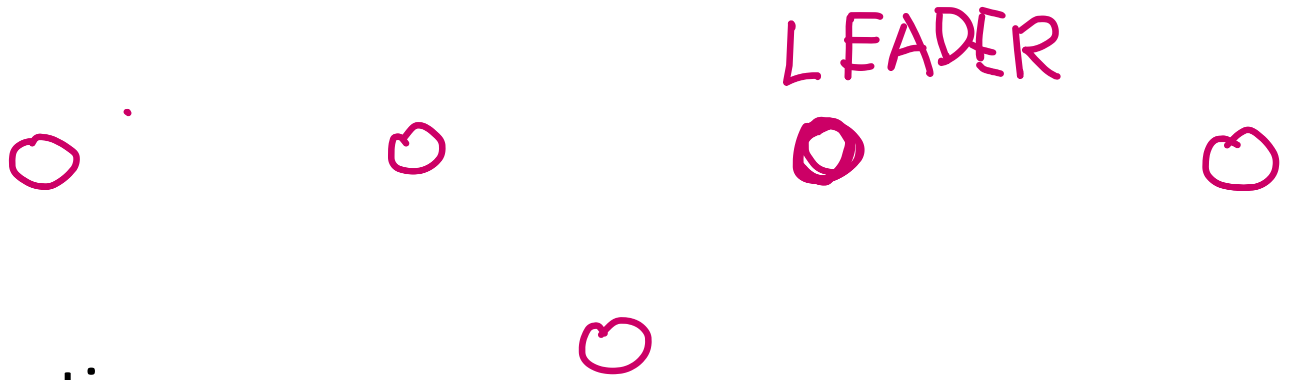
- the nodes join and leave the network
- Node mobility: the neighbors may change

Adversary:

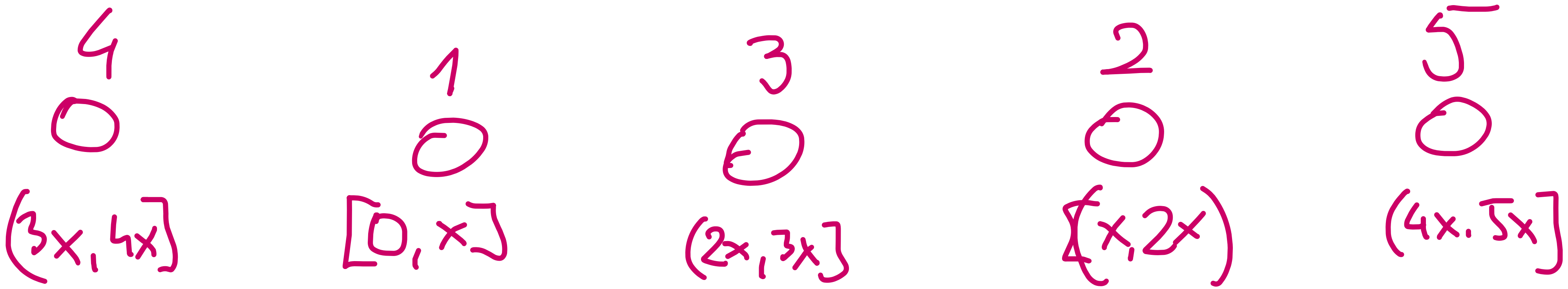
- may corrupt some nodes
- location of subverted may change
- Denial-of-service or gaining control over the network

Some fundamental problems

Leader election

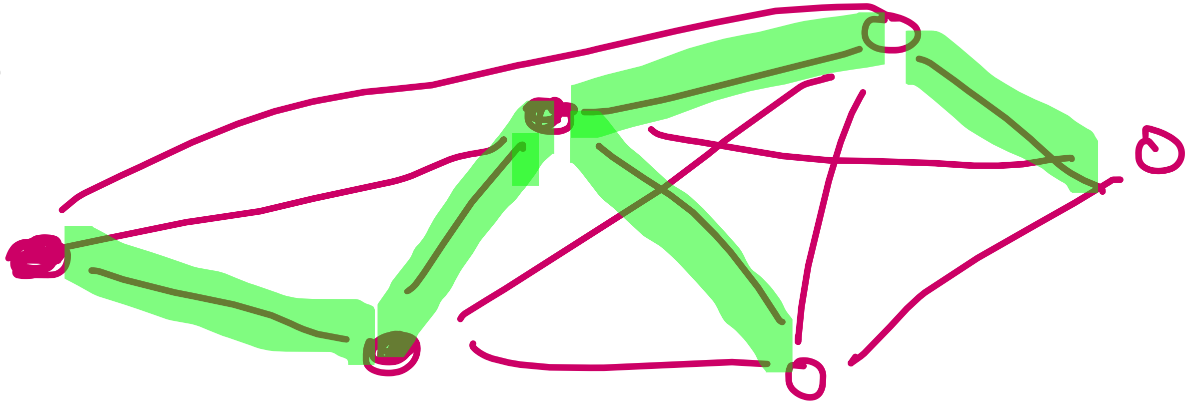


Initialization

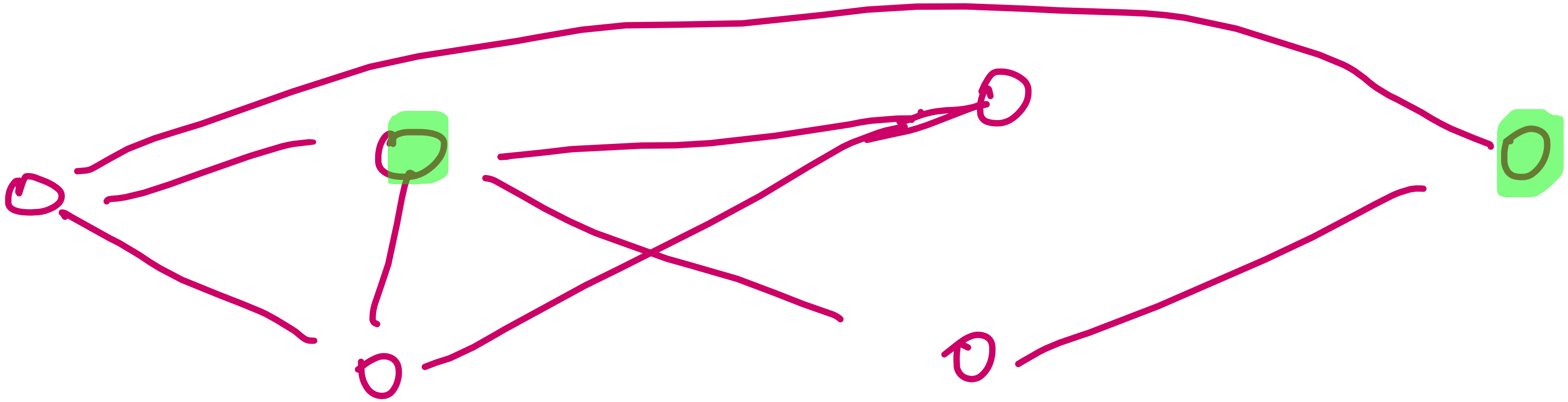


Some fundamental problems

Spanning tree

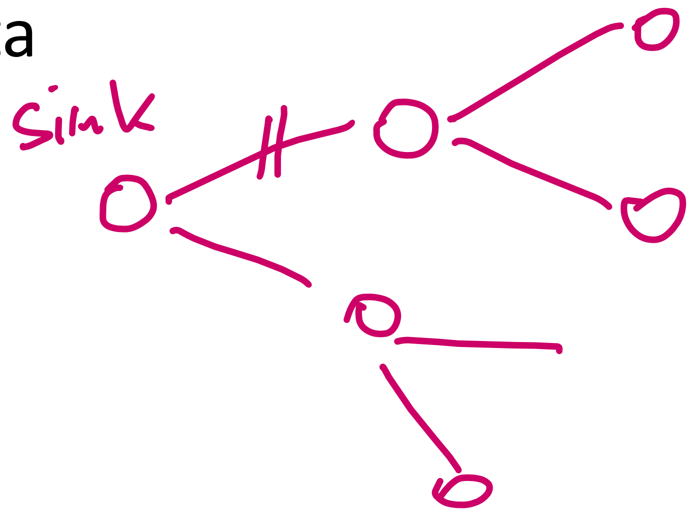


Maximal Independent set



Some fundamental problems

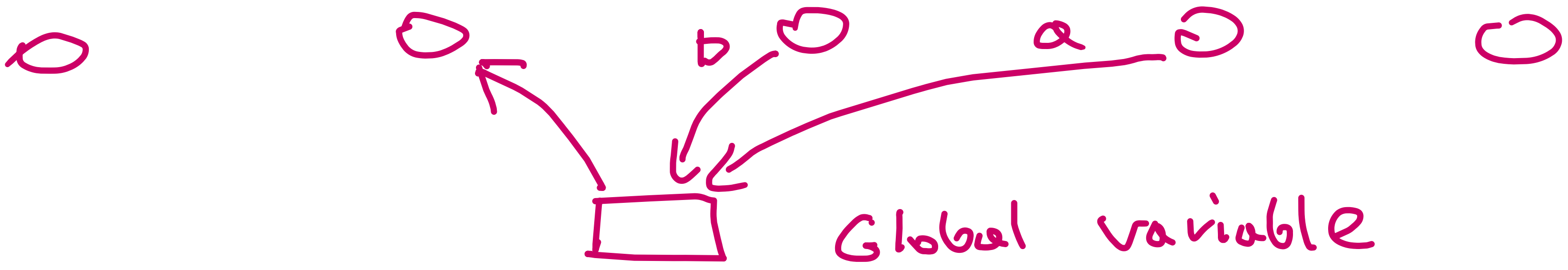
Collecting data



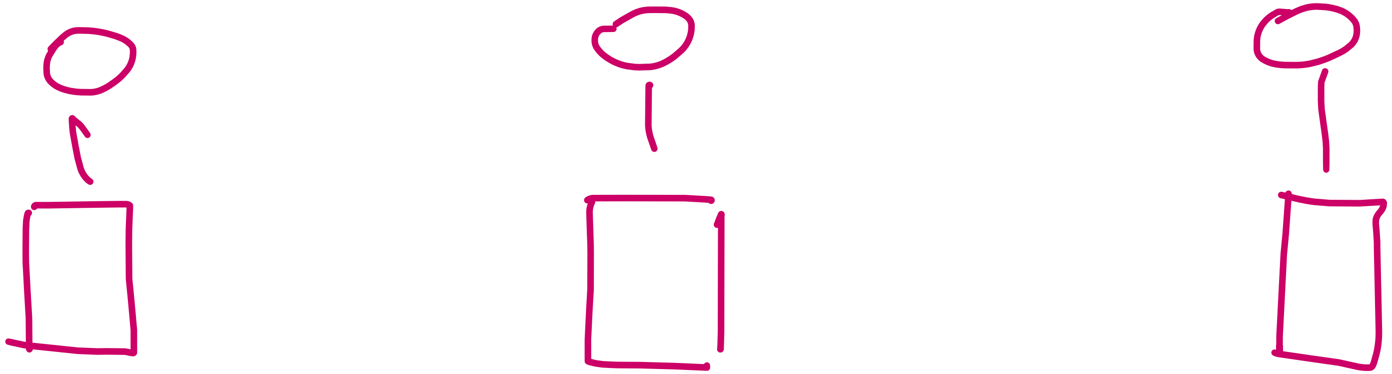
Gossiping

Some fundamental problems

Shared memory



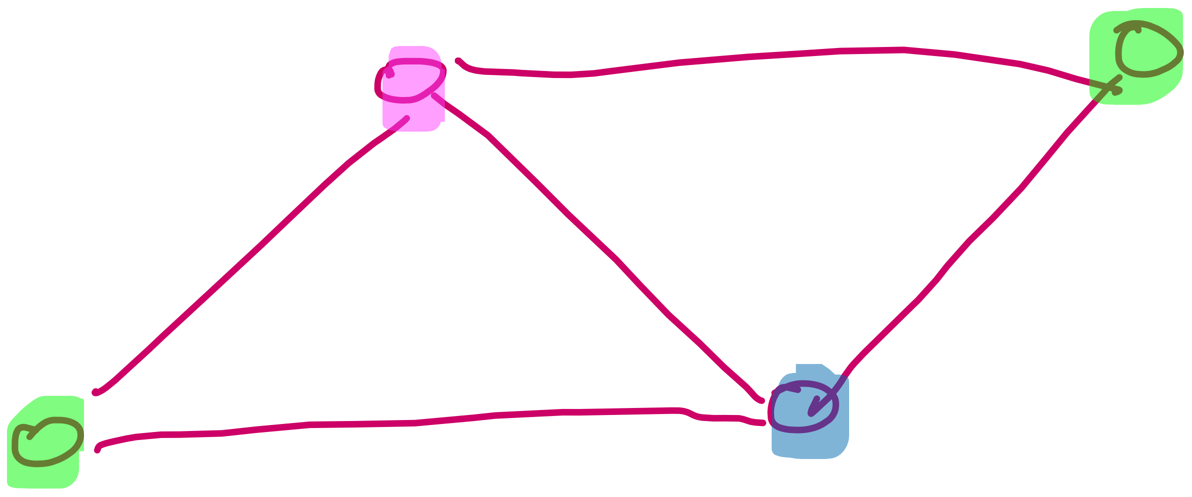
Distributed memory



Vertex Coloring Problem

Assign colors to nodes so that:

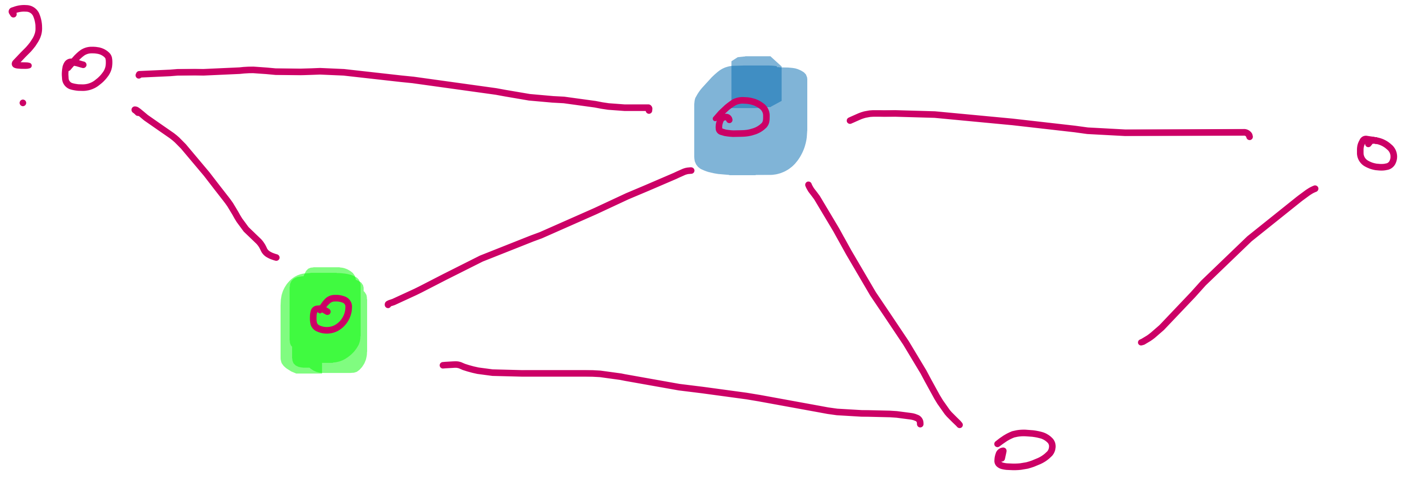
- the neighbors must not the same color
- the overall number of colors used in as small as possible



Coloring problem -motivation

Assigning frequencies for broadcasting:

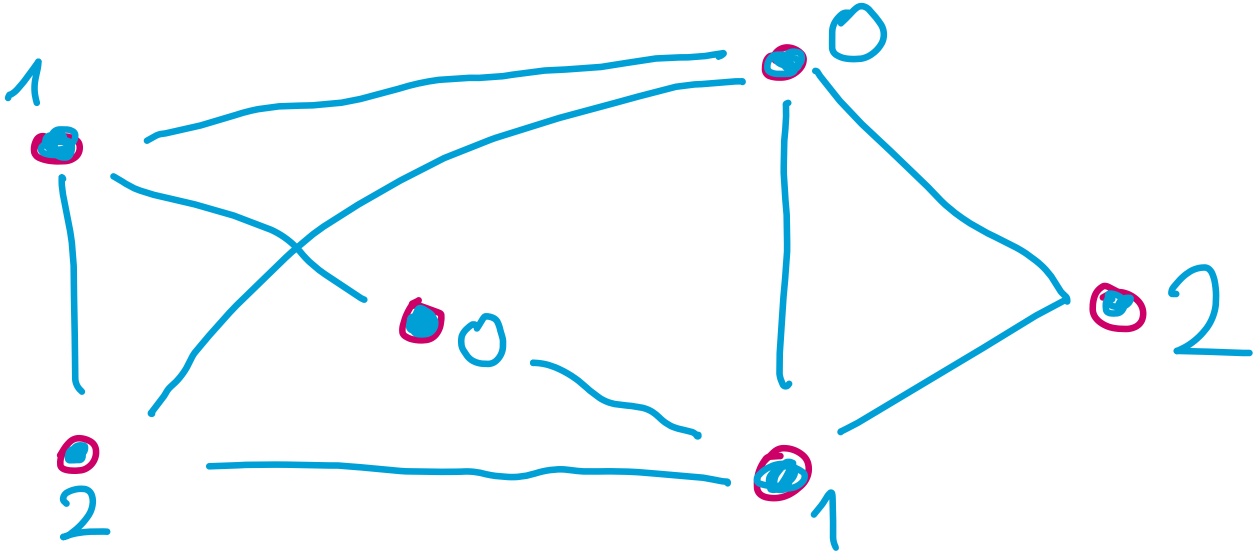
- an edge represents a possible interference
- overall number of frequencies used must be as low as possible



Greedy sequential approach

Algorithm 1.5 Greedy Sequential

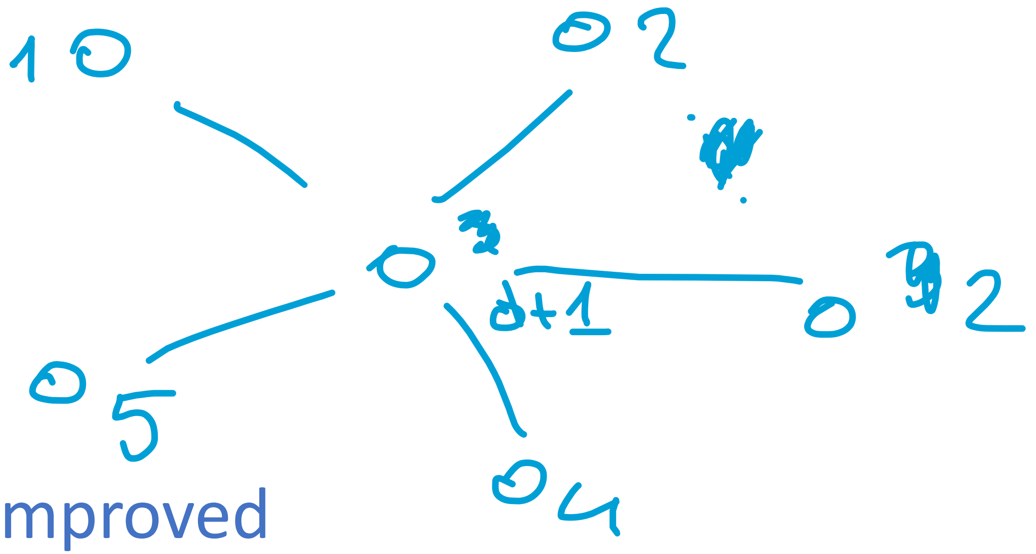
while there is an uncolored vertex v **do**
    color v with the minimal color (number) that does not conflict with the already colored neighbors
end while



Greedy sequential approach

Number of colors:

If degree of the graph is d, then the highest color used is d+1



It cannot be improved

Reduce algorithm

Algorithm 1.9 Reduce

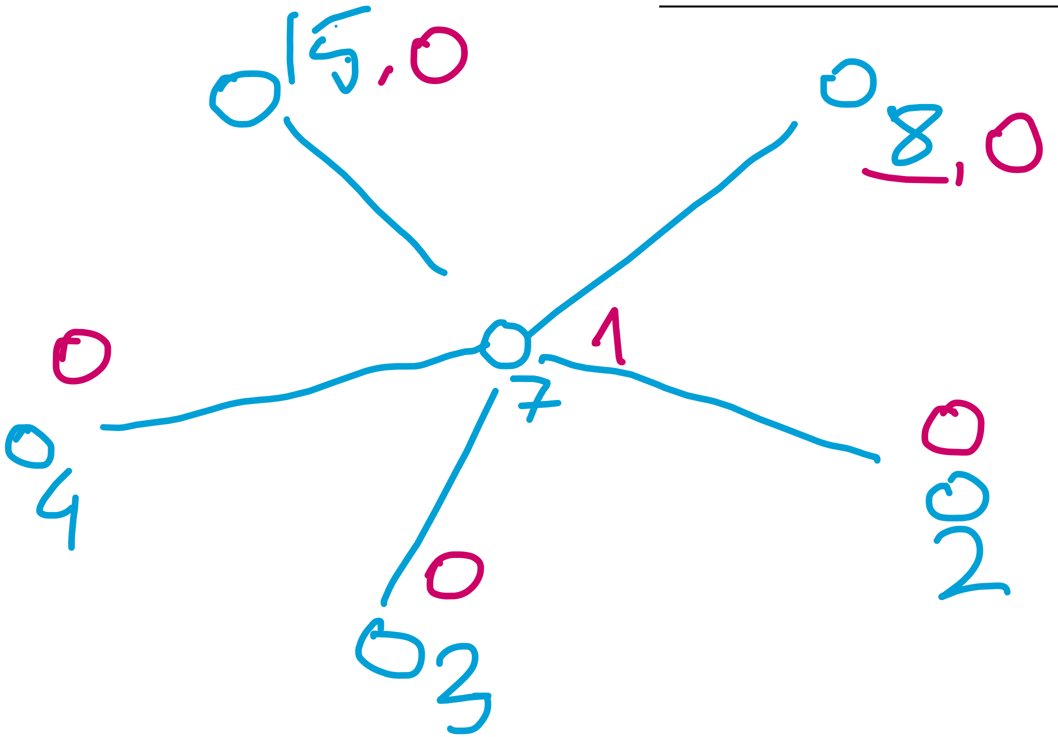
```
Assume that initially all nodes have IDs
Each node v executes the following code:
node v sends its ID to all neighbors
node v receives IDs of neighbors
while node v has an uncolored neighbor with higher ID do
    node v sends "undecided" to all neighbors
    node v receives new decisions from neighbors
end while
node v chooses the smallest admissible free color
node v informs all its neighbors about its choice
```

Reduce

•

Algorithm 1.9 Reduce

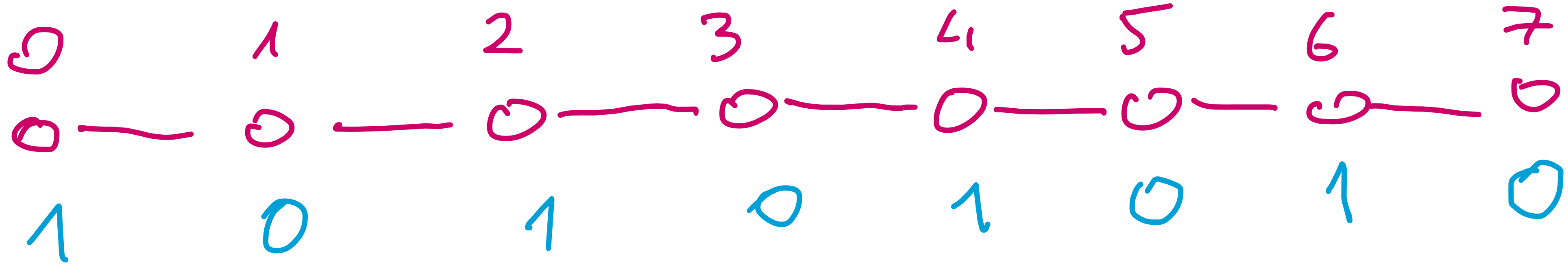
Assume that initially all nodes have IDs
Each node v executes the following code:
node v sends its ID to all neighbors
node v receives IDs of neighbors
while node v has an uncolored neighbor with higher ID **do**
    node v sends "undecided" to all neighbors
    node v receives new decisions from neighbors
end while
node v chooses the smallest admissible free color
node v informs all its neighbors about its choice



Reduce – time complexity

Number of colors is degree+1 (optimal)

Runtime is frequently ok, but not always. Think about an array:



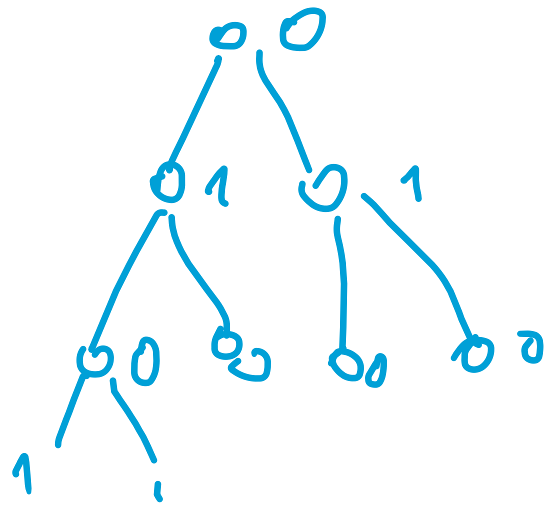
Vertex coloring of trees

2 colors suffice: color=distance to the root mod 2

Finding the distance to the root: time=height of the tree

BAD!

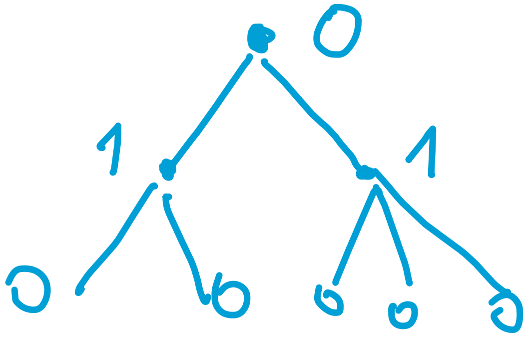
And it looks impossible to improve!



Slow algorithm

Algorithm 1.14 Slow Tree Coloring

```
Color the root 0, root sends 0 to its children
Each node v concurrently executes the following code:
if node v receives a message c_p (from parent) then
    node v chooses color c_v = 1 − c_p
    node v sends c_v to its children (all neighbors except parent)
end if
```



Log* function

Definition 1.16 (Log-Star).

$\forall x \leq 2 : \ \log^* x := 1 \qquad \forall x > 2 : \ \underbrace{\log^* x := 1 + \log^*(\log x)}$

Practically constant function:

log*(2)=1
log*(4)=2
Log*(16)=3
Log*(65536)=4
Log*(2^{65536})=5 but 2^{65536} is practically infinity

6-color algorithm in log* time

Algorithm 1.17 "6-Color"

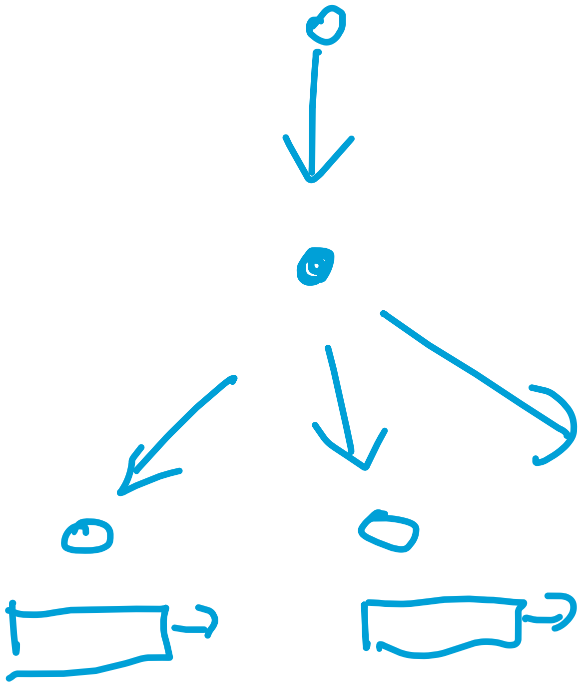
Assume that initially the nodes have IDs of size $\log n$ bits
The root assigns itself the label 0
Each other **node** v executes the following code
send own color c_v to all children
repeat
    receive color c_p from parent
    interpret c_v and c_p as bit-strings
    let i be the index of the smallest bit where c_v and c_p differ
    the new label is i (as bitstring) followed by the i^{th} bit of c_v
    send c_v to all children
until $c_w \in \{0, \dots, 5\}$ for all nodes w

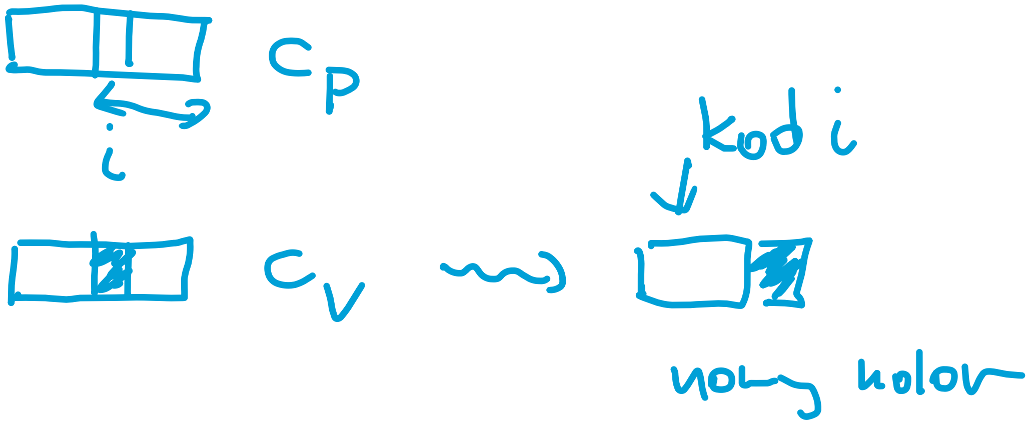
6-color algorithm

Reduction of the size of the color:
from length to log(length)+1

Algorithm 1.17 "6-Color"

Assume that initially the nodes have IDs of size $\log n$ bits
The root assigns itself the label 0
Each other **node** v executes the following code
send own color c_v to all children
repeat
  receive color c_p from parent
  interpret c_v and c_p as bit-strings
  let i be the index of the smallest bit where c_v and c_p differ
  the new label is i (as bitstring) followed by the i^{th} bit of c_v
  send c_v to all children
until $c_w \in \{0, \dots, 5\}$ for all nodes w



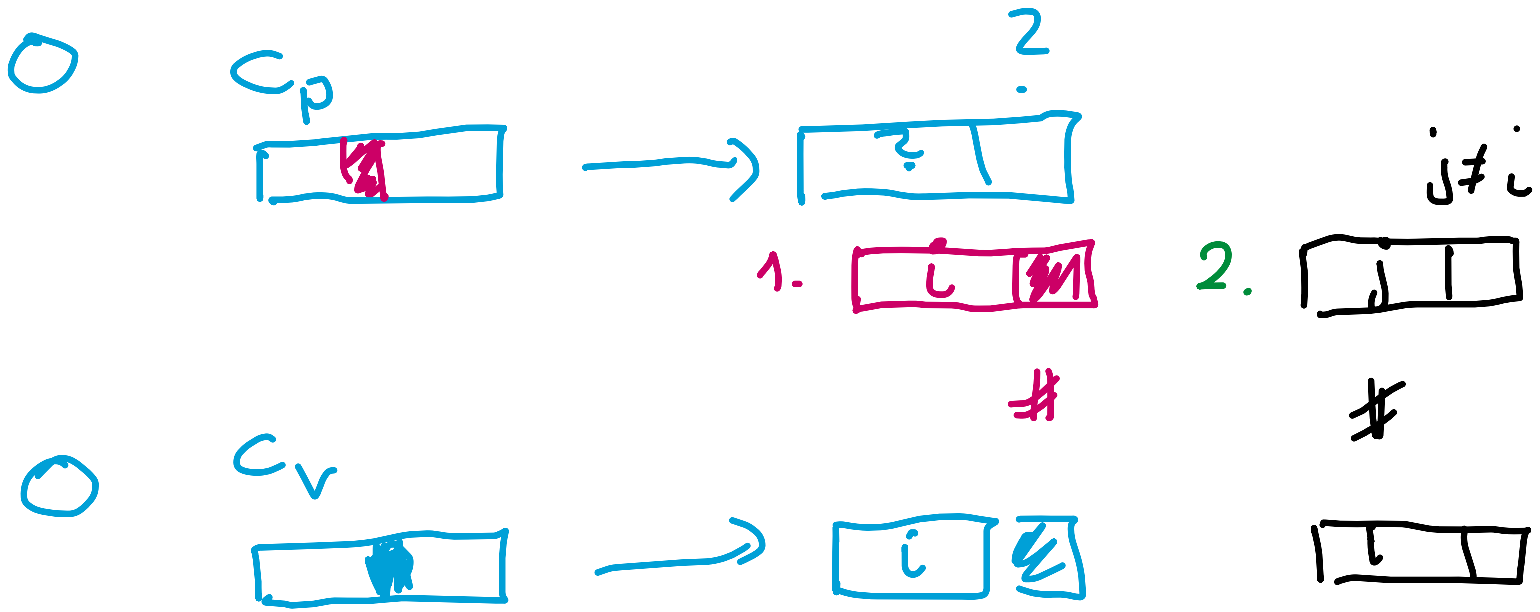


6-color algorithm

Algorithm 1.17 “6-Color”

Assume that initially the nodes have IDs of size $\log n$ bits
The root assigns itself the label 0
Each other **node** v executes the following code
send own color c_v to all children
repeat
  receive color c_p from parent
  interpret c_v and c_p as bit-strings
  let i be the index of the smallest bit where c_v and c_p differ
  the new label is i (as bitstring) followed by the i^{th} bit of c_v
  send c_v to all children
until $c_w \in \{0, \dots, 5\}$ for all nodes w

Correctness of colors:



6-color algorithm

No progress when 6 colors:

bits to denote the position:

00

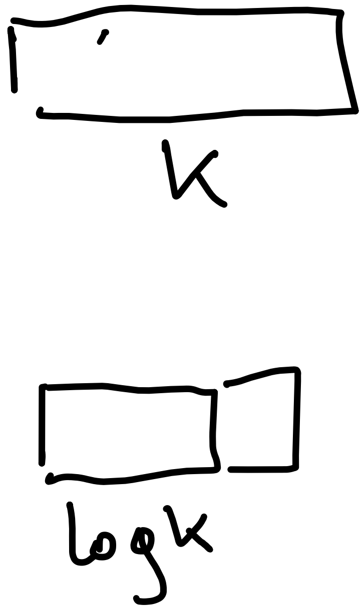
01

10

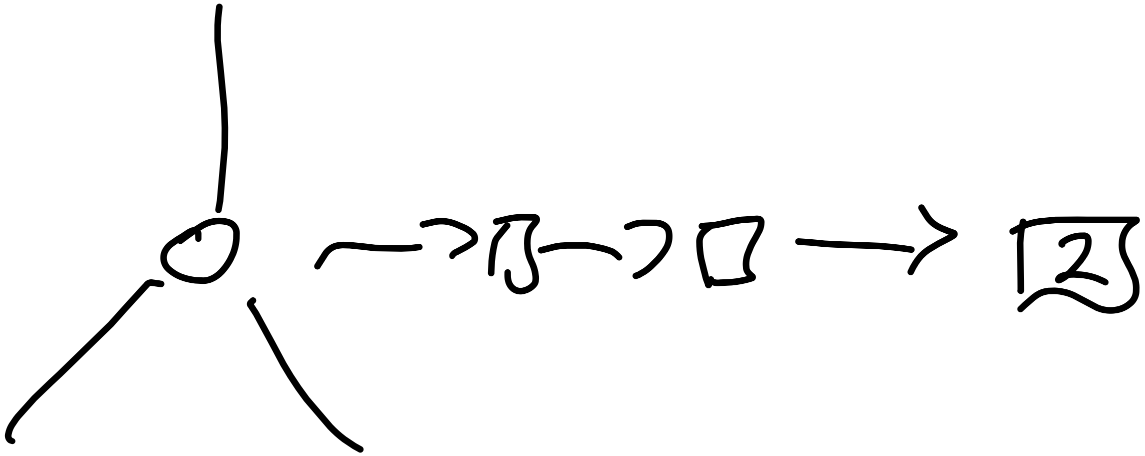
The highest color: 101 =5

Algorithm 1.17 "6-Color"

Assume that initially the nodes have IDs of size $\log n$ bits
The root assigns itself the label 0
Each other **node** v executes the following code
send own color c_v to all children
repeat
    receive color c_p from parent
    interpret c_v and c_p as bit-strings
    let i be the index of the smallest bit where c_v and c_p differ
    the new label is i (as bitstring) followed by the i^{th} bit of c_v
    send c_v to all children
until $c_w \in \{0, \ldots, 5\}$ for all nodes w



$k \to \log k + 1$


2

Improving – shift down

Sieblings will get the same color:

Algorithm 1.19 Shift Down

1: **Each** other **node** v concurrently executes the following code:
2: Recolor v with the color of parent
3: Root chooses a new (different) color from $\{0, 1, 2\}$

6-to-3 algorithm

Algorithm 1.21 Six-2-Three

Each node v concurrently executes the following code:
for $x = 5, 4, 3$ **do**
    Perform subroutine Shift down (Algorithm 1.19)
    if $c_v = x$ **then**
        choose the smallest admissible new color $c_v \in \{0, 1, 2\}$
    end if
end for

6-to-3 algorithm

Algorithm 1.21 Six-2-Three

Each node v concurrently executes the following code:
for $x = 5, 4, 3$ **do**
  Perform subroutine Shift down (Algorithm 1.19)
  if $c_v = x$ **then**
    choose the smallest admissible new color $c_v \in \{0, 1, 2\}$
  end if
end for