

- **Weeks 1–2: informal introduction**

- network = path



- **Week 3: graph theory**

- **Weeks 4–7: models of computing**

- what can be computed (efficiently)?

- **Weeks 8–11: lower bounds**

- what cannot be computed (efficiently)?

- **Week 12: recap**

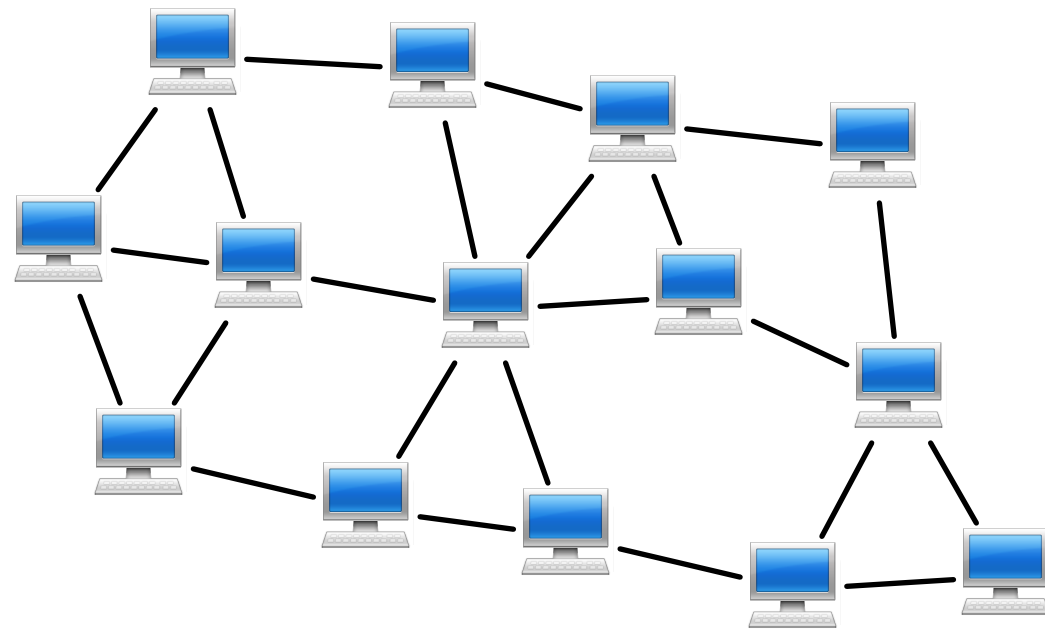
Week 12

- Conclusions

Recap:

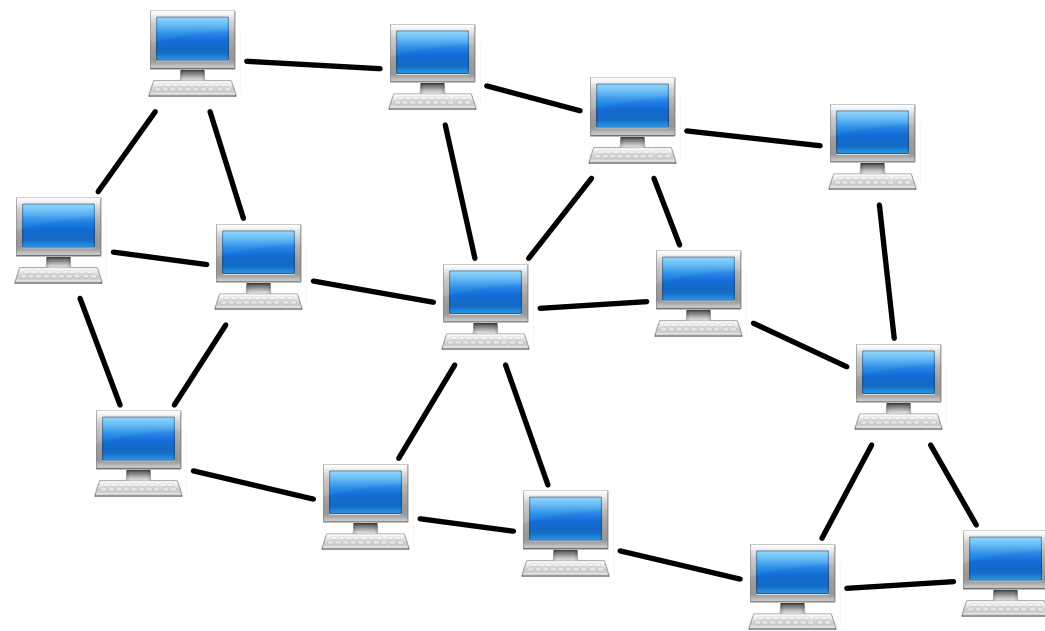
Distributed algorithms

Algorithms for computer networks



Recap: **Distributed algorithms**

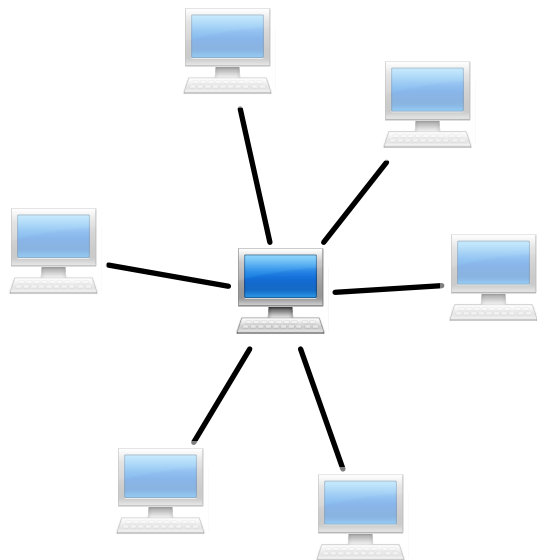
Identical computers in an **unknown network,
all running the **same algorithm****



Recap:

Distributed algorithms

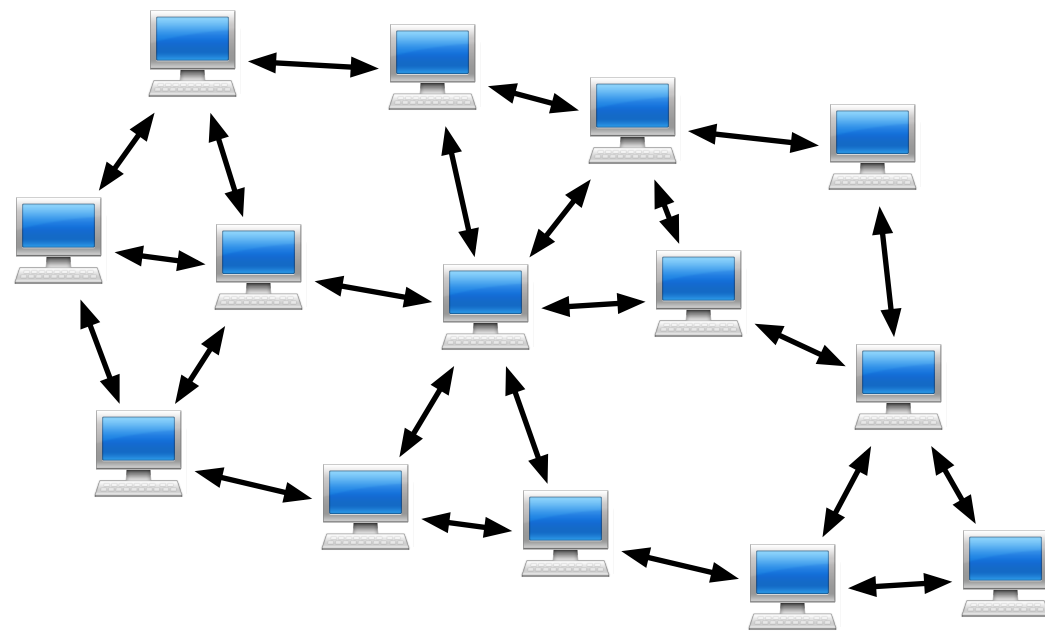
Initially each computer only aware of its immediate neighbourhood



Recap:

Distributed algorithms

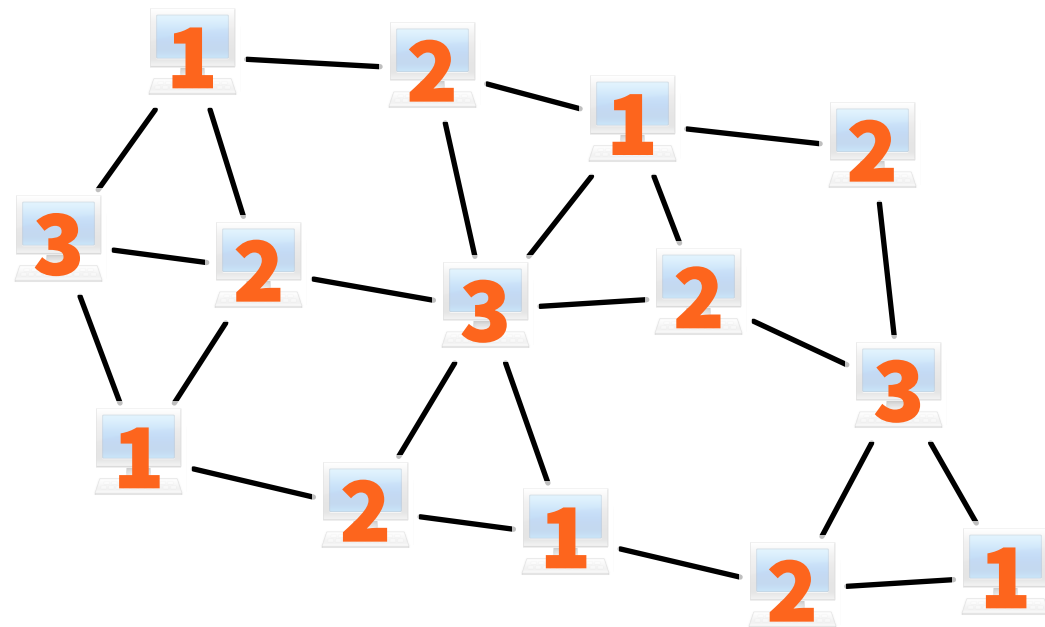
**Nodes can exchange messages
with their neighbours to learn more...**



Recap:

Distributed algorithms

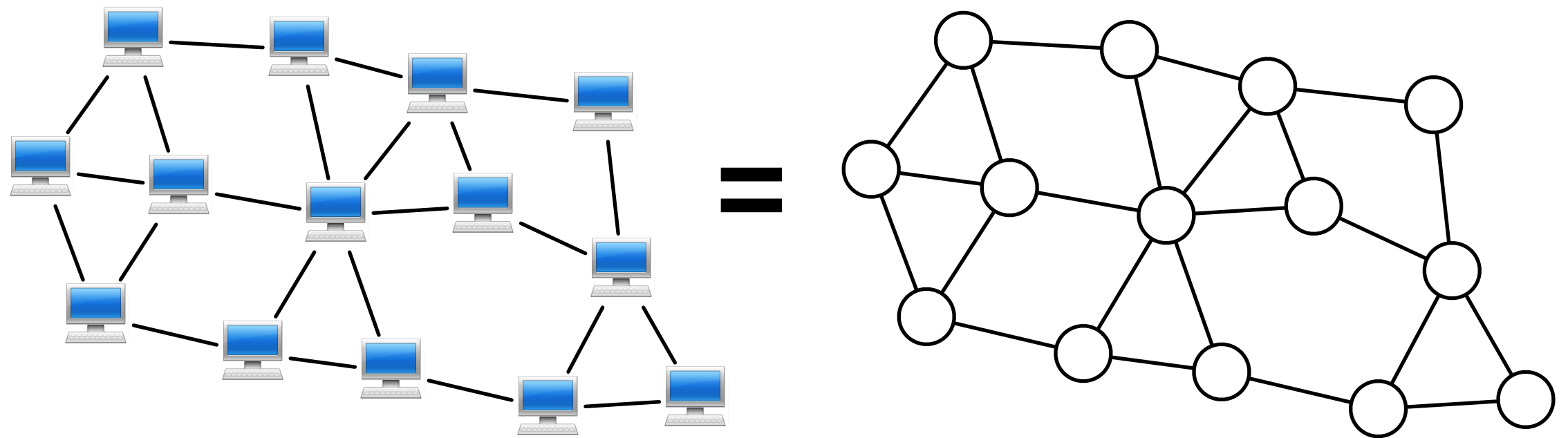
Finally, each computer has to stop and produce its own **local output**



Recap:

Distributed algorithms

Focus on graph problems:
network topology = input graph

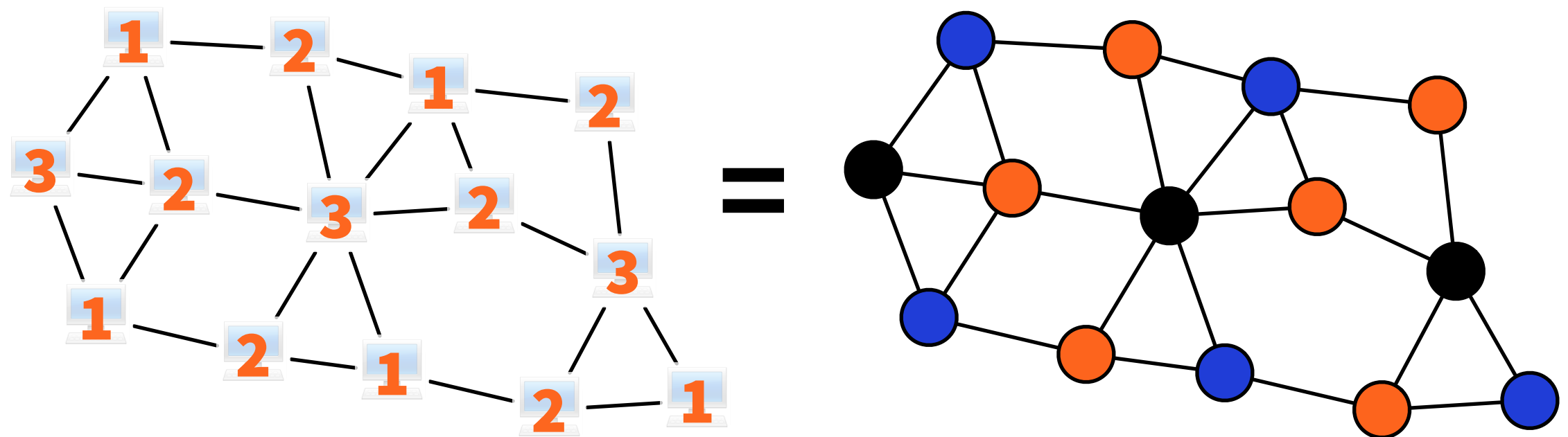


Recap:

Distributed algorithms

Focus on graph problems:

local outputs = solution (here: graph colouring)



Recap:

Distributed algorithms

Typical research question:

“How fast can we solve graph problem X ?”

Time = number of communication rounds

What have we learned?

- Dealing with *unknown systems*
- Dealing with *partial information*
- Dealing with *parallelism*
- Applications beyond distributed computing:
fault tolerance, online, streaming, multicore...

Learning objectives

- **Models**
- **Algorithms**
- **Lower bounds**
- **Graph theory**

Objective 1:

Models of computing

- **Precisely** what is a “*distributed algorithm*”
- **In each of these models:**
 - PN, LOCAL, CONGEST
 - deterministic, randomised

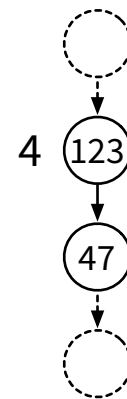
Objective 2:

Algorithms

- **Colouring paths:** LOCAL, $O(\log^* n)$
- **Colouring graphs:** LOCAL, $O(\log n)$ w.h.p.
- **Gather everything:** LOCAL, $O(\text{diam}(G))$
- **Bipartite maximal matching:** PN, $O(\Delta)$
- **All-pairs shortest paths:** CONGEST, $O(n)$

Algorithm P3CBit: Fast colour reduction

$c_0 = 123 = 01111011_2$ (my colour)
 $c_1 = 47 = 00101111_2$ (successor's colour)
 $i = 2$ (bits numbered 0, 1, 2, ... from right)
 $b = 0$ (in my colour bit number i was 0)
 $c = 2 \cdot 2 + 0 = 4$ (my new colour)



$k = 8$, reducing from $2^8 = 256$ to $2 \cdot 8 = 16$ colours

Algorithm P3CBit.

Fast

$c_0 = 12$

$c_1 = 4$

$i = 2$ (

$b = 0$ (

$c = 2 \cdot 2$

k

Algorithm idea 3

- Colour palette: $\{1, 2, \dots, \Delta + 1\}$
- Active with probability $1/2$
- If *active*, pick a random *free* colour
 - not used by any neighbour that has stopped
- Try again if conflicts...

Algorithm P3CBit.

Fast

$c_0 = 12$

$c_1 = 4$

$i = 2$

$b = 0$

$c = 2 \cdot 2$

Algo

• Color

• Active

• If *act*

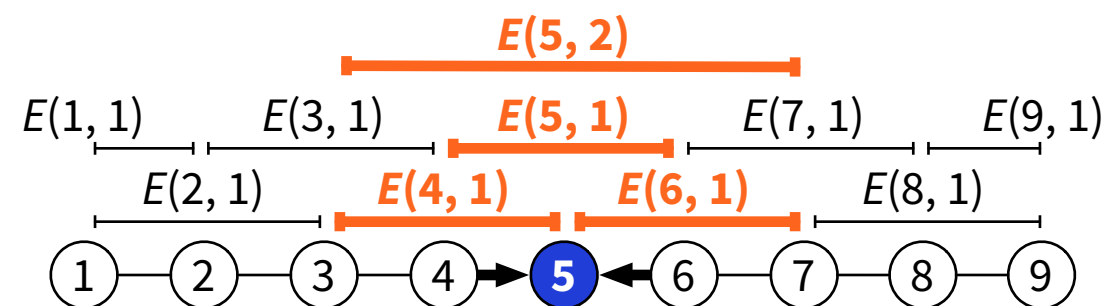
• no

• Try a

Gathering everything

• Given $E(v, r)$, we can learn $E(v, r + 1)$ in 1 round

• send $E(v, r)$ to all neighbours, take union



Algorithm P3CBit.

Fast

$c_0 = 12$

$c_1 = 4$

$i = 2$ (

$b = 0$ (

$c = 2 \cdot 2$

k

Algo

• Color

• Activ

• If *act*

• no

• Try a

Gat

• Given

• se

$E(1, 1)$

①

Algorithm BMM: **Maximal matching**

- **Blue nodes** send proposals to their orange neighbours one by one
 - using port numbers
- **Orange nodes** accept the first proposal that they get
 - using port numbers to break ties

Algorithm D3CBit.

Fast

$$c_0 = 12$$

$$c_1 = 4$$

$$i = 2$$

$$b = 0$$

$$c = 2 \cdot 2$$

Algo

• Color

• Activ

• If *act*

• no

• Try a

Gate

• Given

• se

$$E(1, 1)$$

1

Algorithm BMM.

Max

• Blue

orange

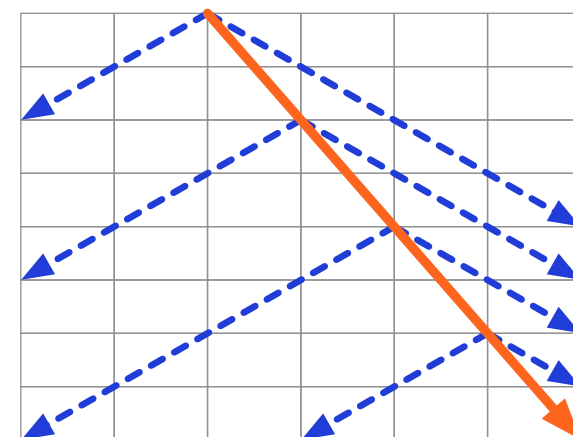
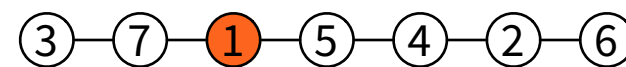
• us

• Orange

the f

• us

Algorithm APSP



wave

token

Objective 2:

Algorithms

- **Reductions!**
- **Graph colouring is a very useful subroutine**

Objective 3:

Lower bounds

- **Covering maps:**
what cannot be solved at all in PN model
- **Local neighbourhoods:**
what cannot be solved fast in any model
- **Ramsey's theorem:**
what cannot be solved in $O(1)$ time

Objective 4:

Graph theory

- **Basic definitions**
- **Connections between graph problems**
 - e.g. maximal matching \rightarrow small vertex covers
- **Ramsey's theorem**
 - at least for $c = 2, k = 2$

What else is studied in distributed computing?

- Fault-tolerance
- Asynchrony
- Shared memory
- Physical models
- Robot navigation
- Nondeterminism
- Complexity measures
- High-performance computing
- Practical aspects of networking ...

What next?

- **CS-E4000 Seminar in Computer Science: Algorithms**
 - 3rd–4th period, 5 credits
 - reading seminar
 - algorithms and theory of computation

What next?

- **CS-E4580 Programming Parallel Computers**
 - 5th period, 5 credits, intensive course
 - programming modern parallel computers: multicore, GPU, memory hierarchies ...
 - hands-on programming exercises
 - main goal: make it as fast as you can!

What next?

- **Just ask if you want to do more!**
 - master's thesis topics?
 - summer internships?
 - doctoral studies?

- **Weeks 1–2: informal introduction**

- network = path



- **Week 3: graph theory**

- **Weeks 4–7: models of computing**

- what can be computed (efficiently)?

- **Weeks 8–11: lower bounds**

- what cannot be computed (efficiently)?

- **Week 12: recap**