

Local Algorithms: Past, Present, Future

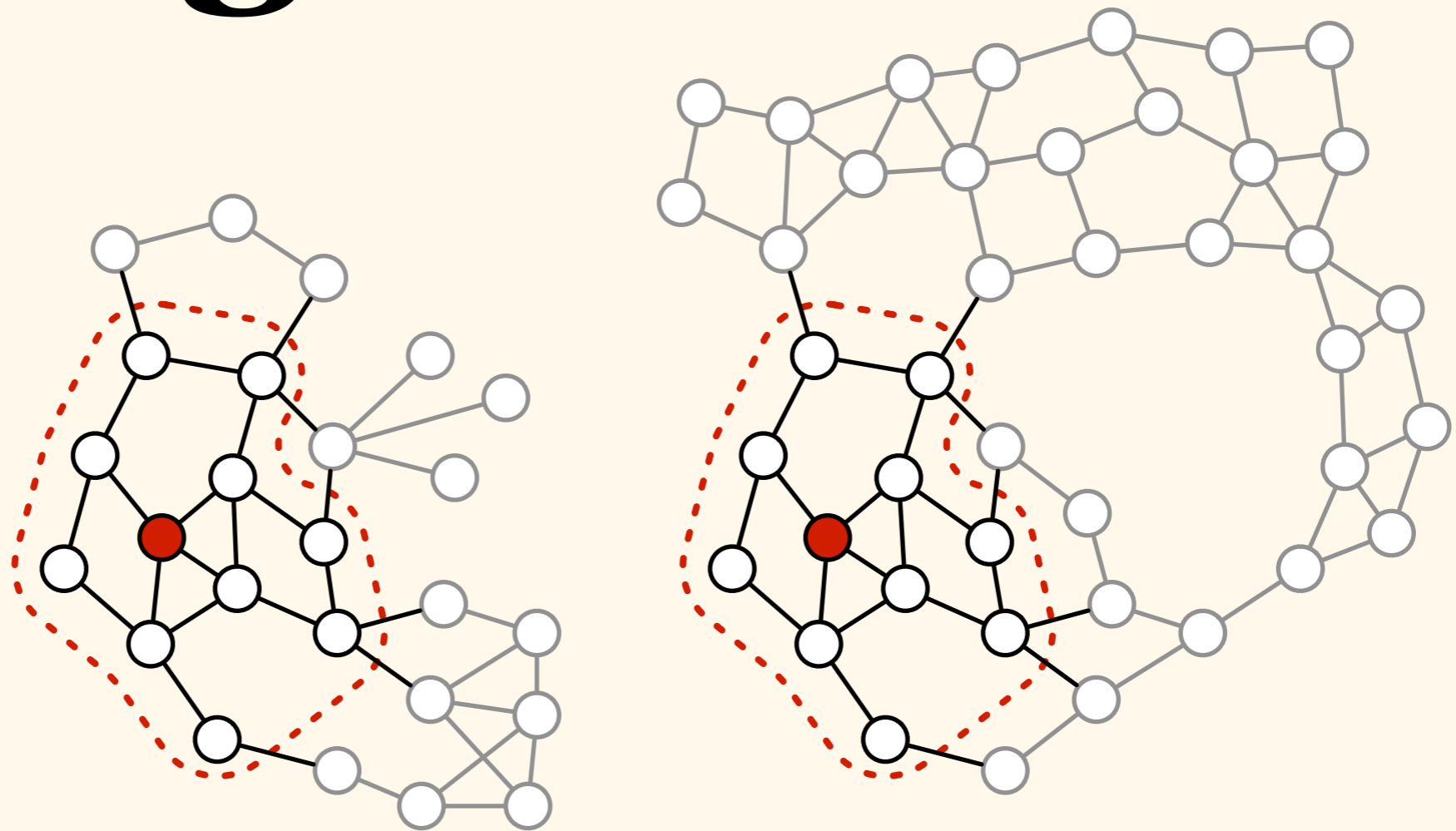
Jukka Suomela

Helsinki Institute for Information Technology HIIT
University of Helsinki

www.hiit.fi/jukka.suomela/

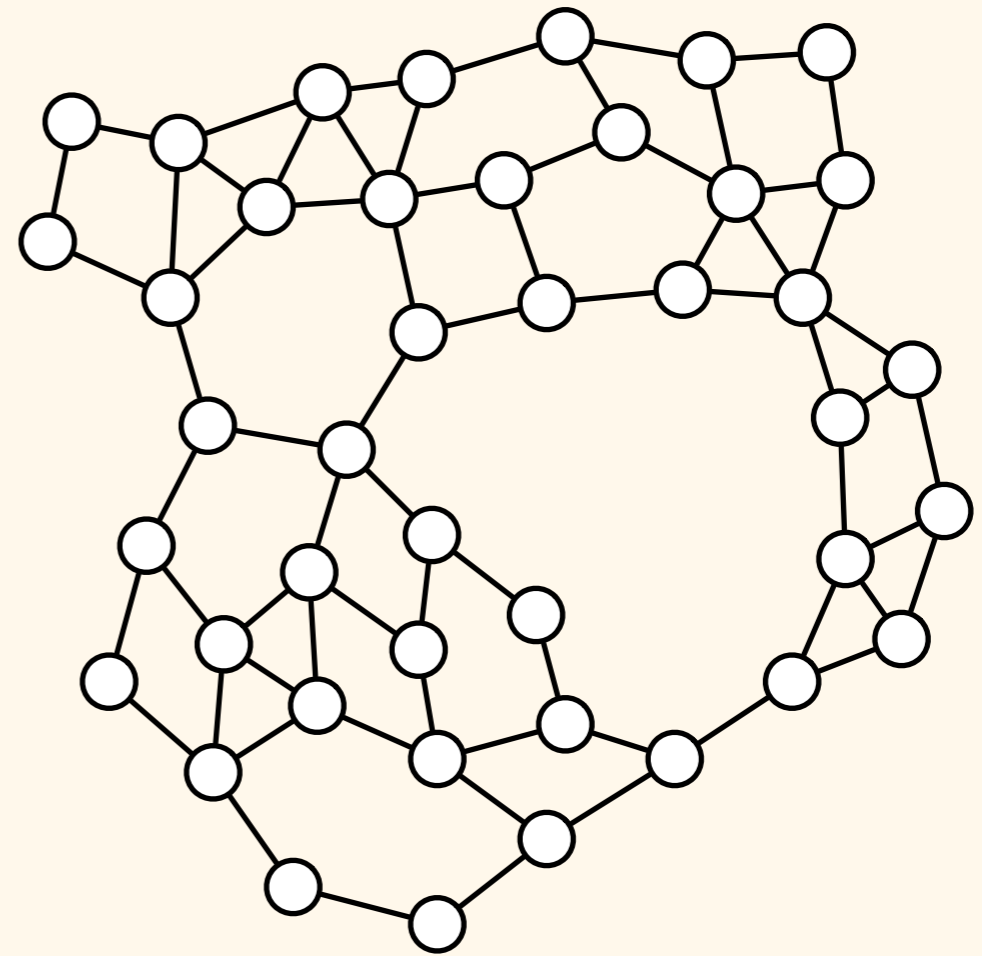
Hebrew University of Jerusalem, 23 November 2011

Background



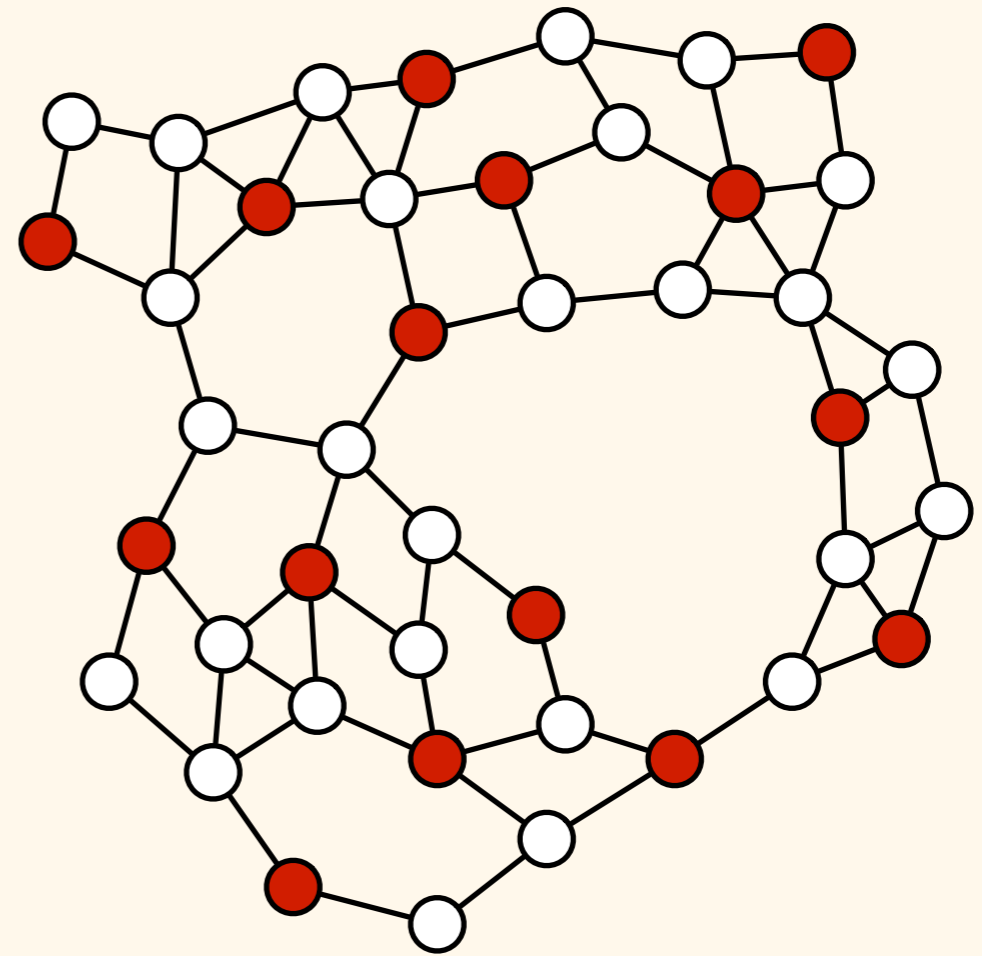
Setting

- Graphs



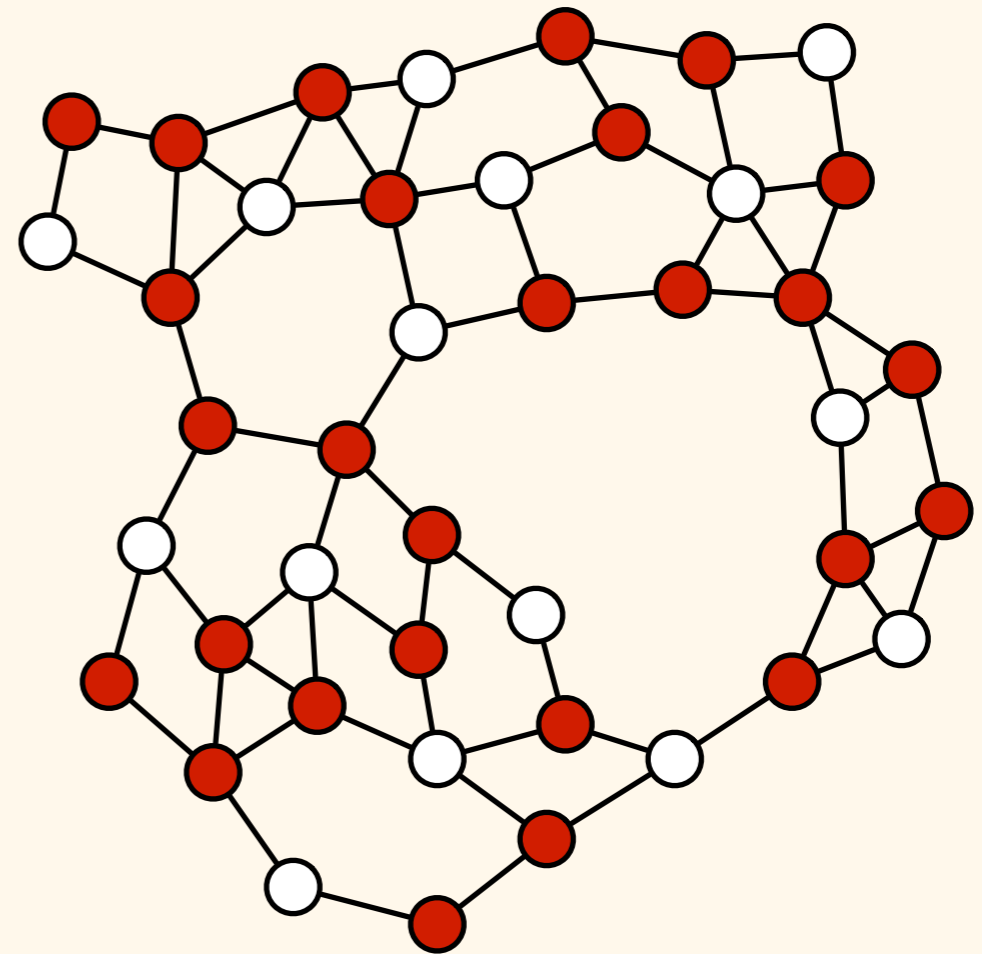
Setting

- Graphs
- Algorithms for graph problems
 - *Independent sets*



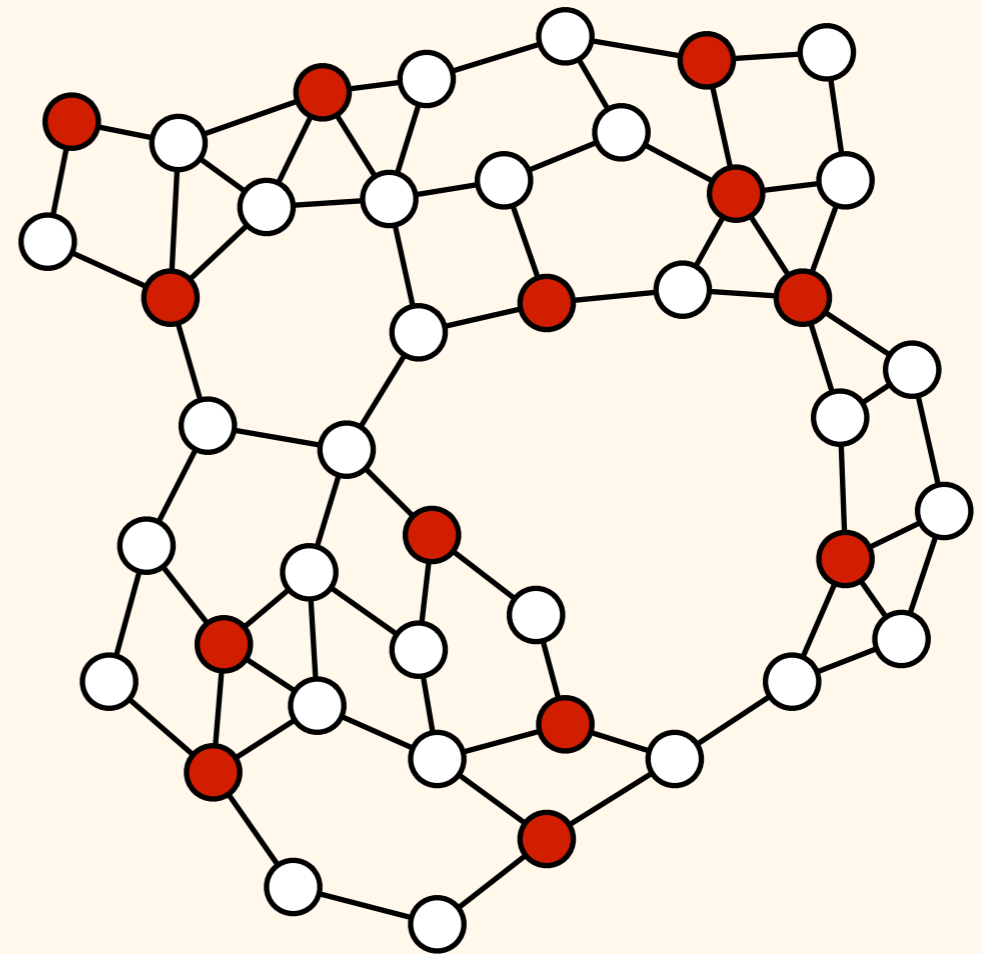
Setting

- Graphs
- Algorithms for graph problems
 - Independent sets, matchings, *vertex covers*



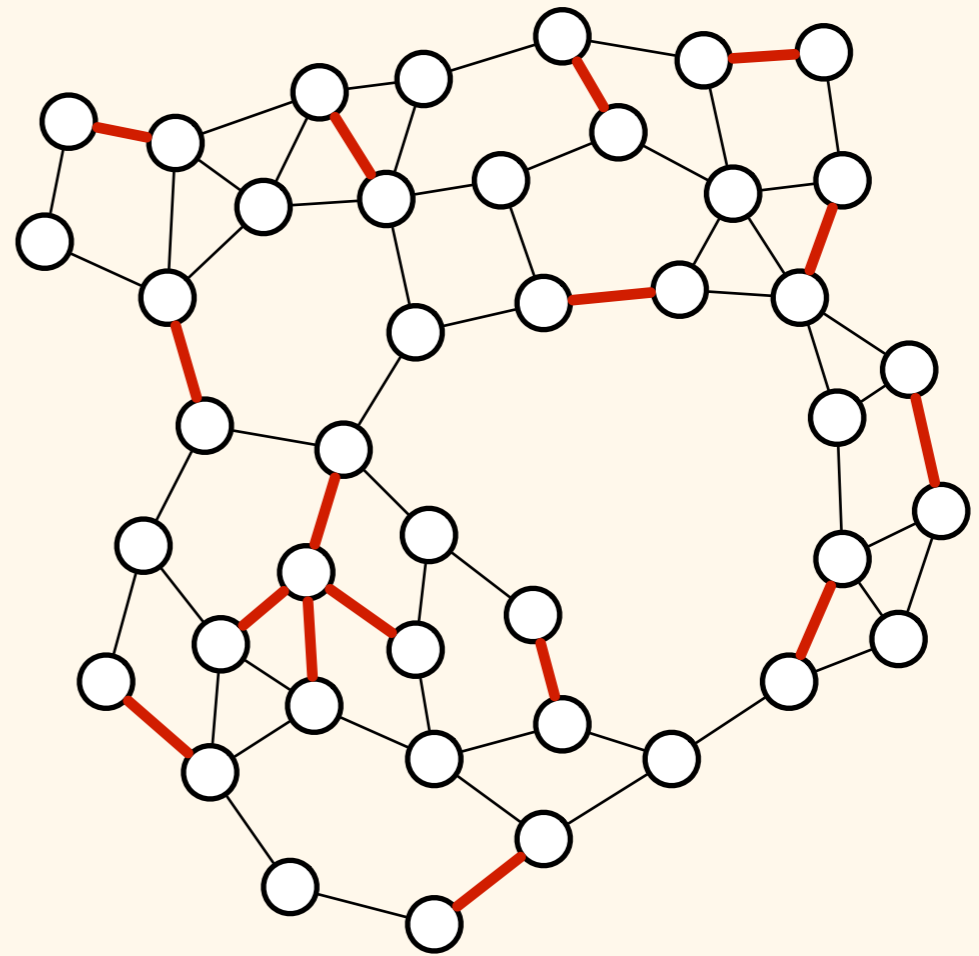
Setting

- Graphs
- Algorithms for graph problems
 - Independent sets, matchings, vertex covers, *dominating sets*



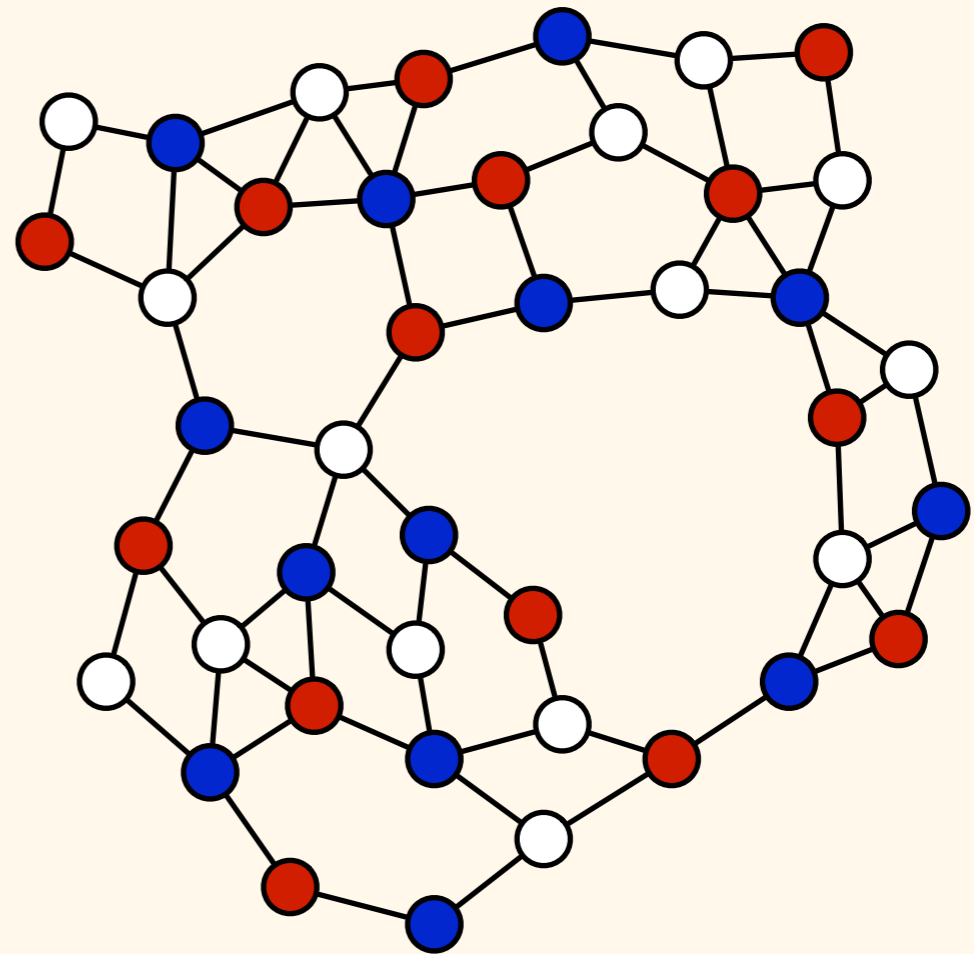
Setting

- Graphs
- Algorithms for graph problems
 - Independent sets, matchings, vertex covers, dominating sets, *edge dominating sets*



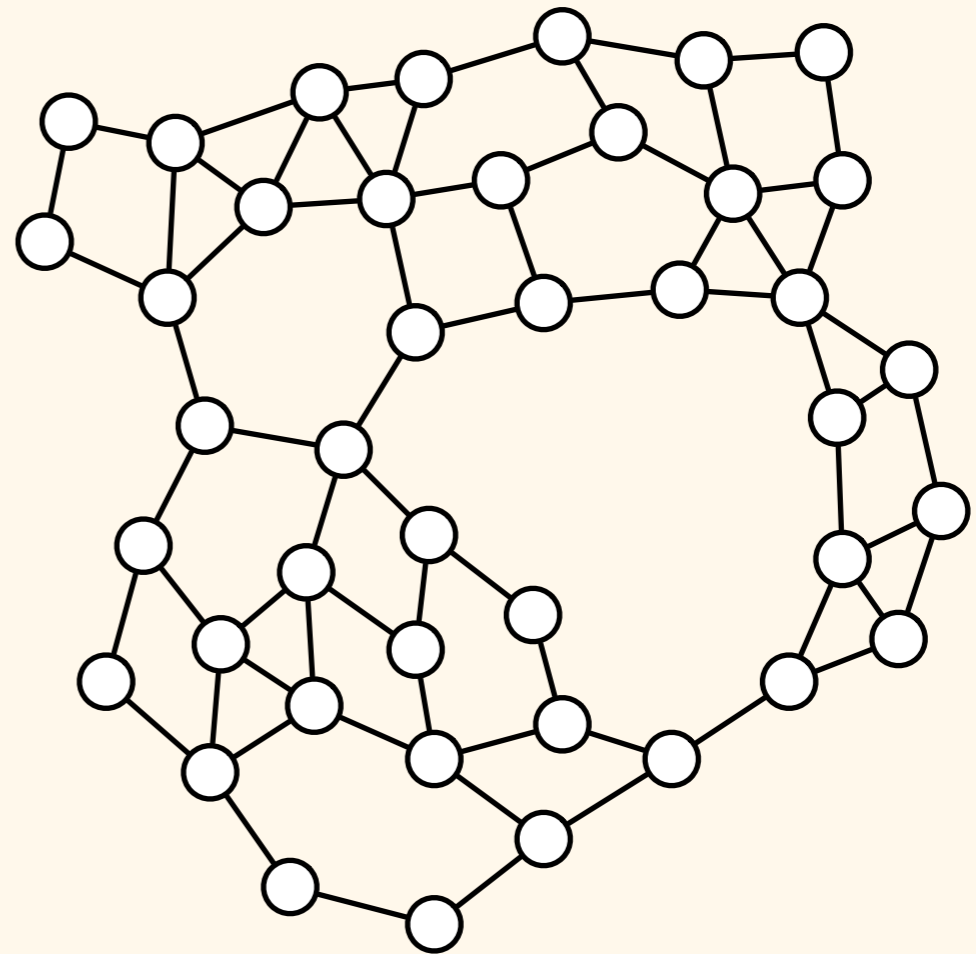
Setting

- Graphs
- Algorithms for graph problems
 - Independent sets, matchings, vertex covers, dominating sets, edge dominating sets, *graph colourings*



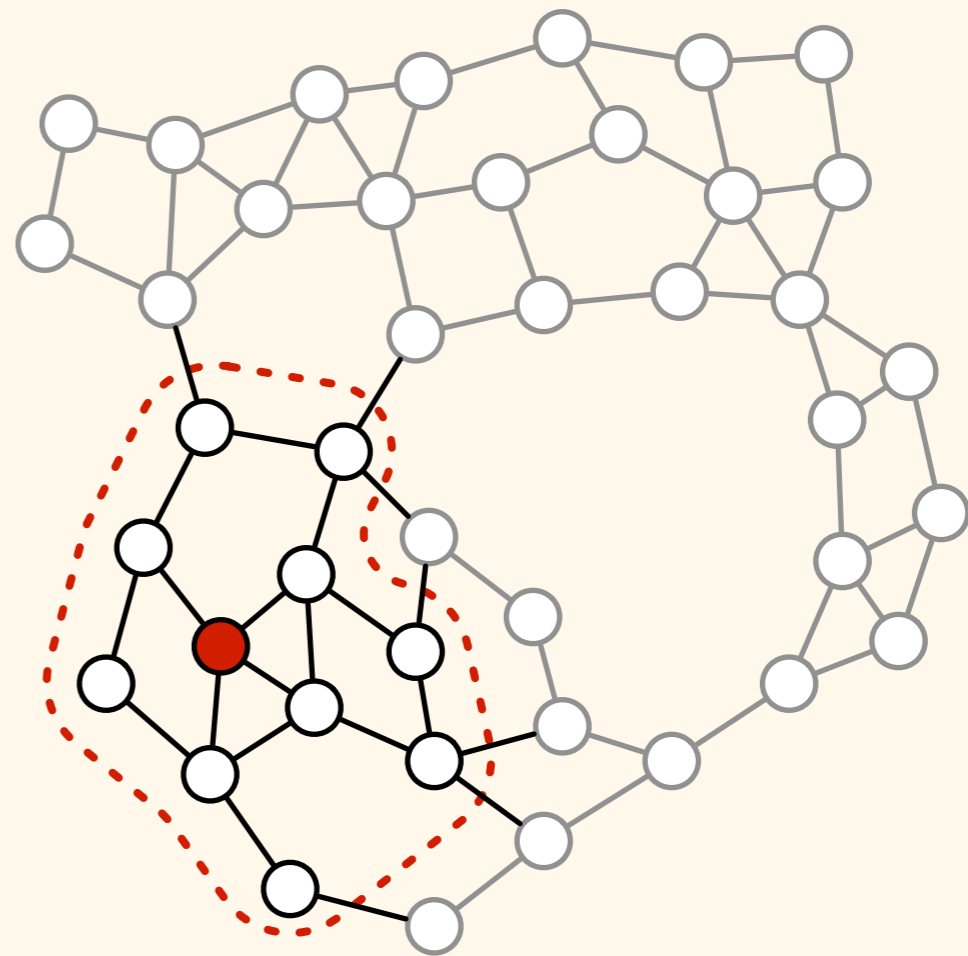
Setting

- Graphs
- Algorithms for graph problems
 - Independent sets, matchings, vertex covers, dominating sets, edge dominating sets, graph colourings, ...



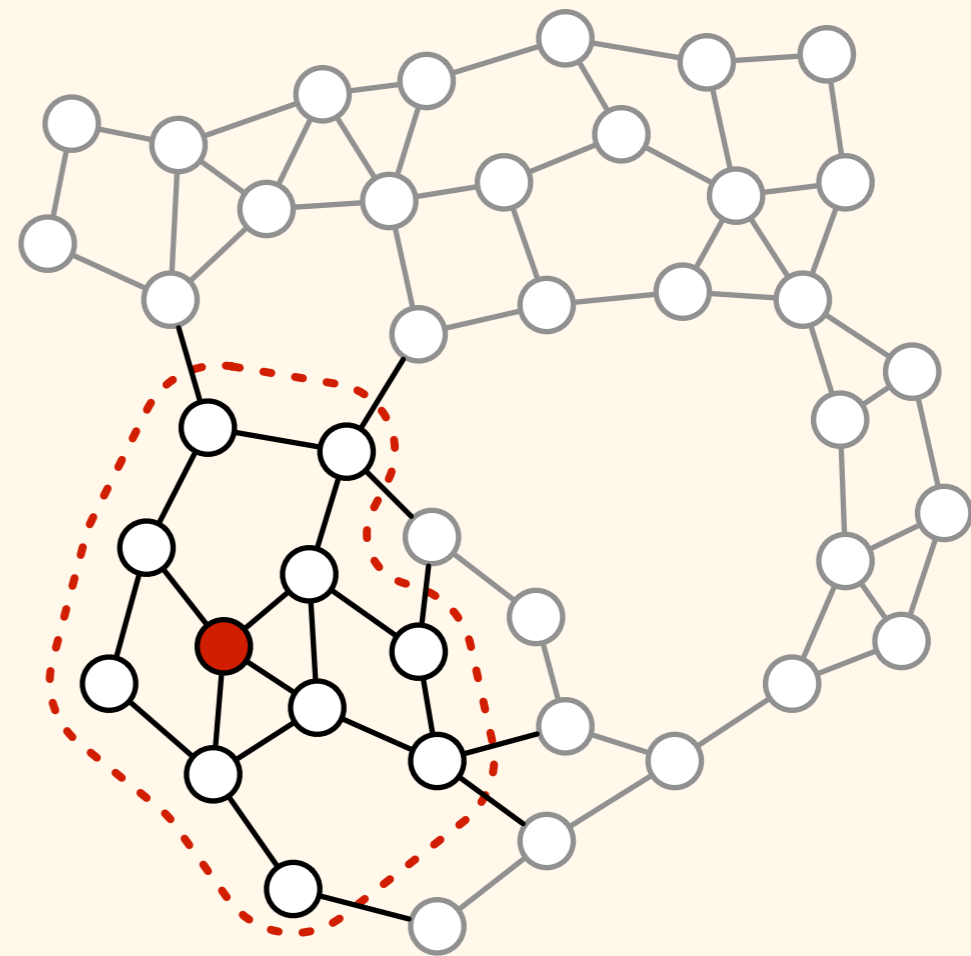
Local Algorithms

- *Local neighbourhood:*
nodes at distance r
 - Here $r = O(1)$,
independent of
number of nodes
 - Shortest-path distance,
number of edges



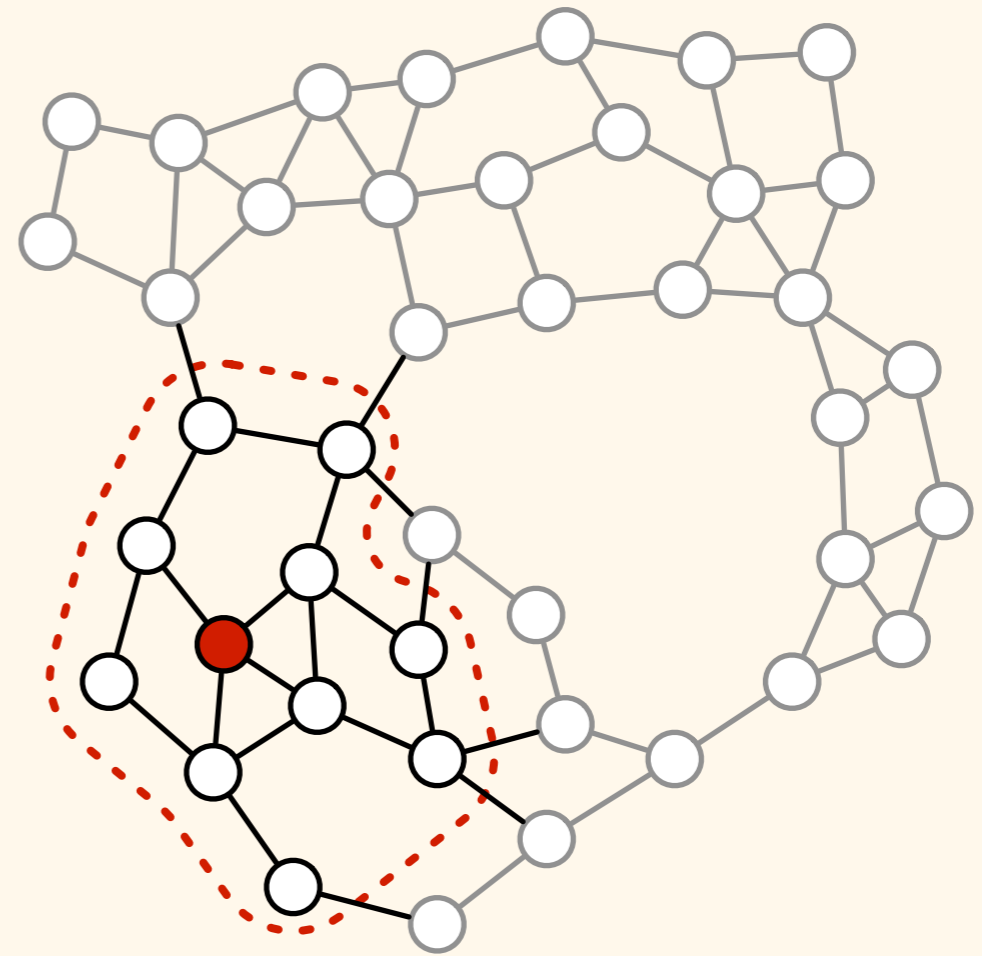
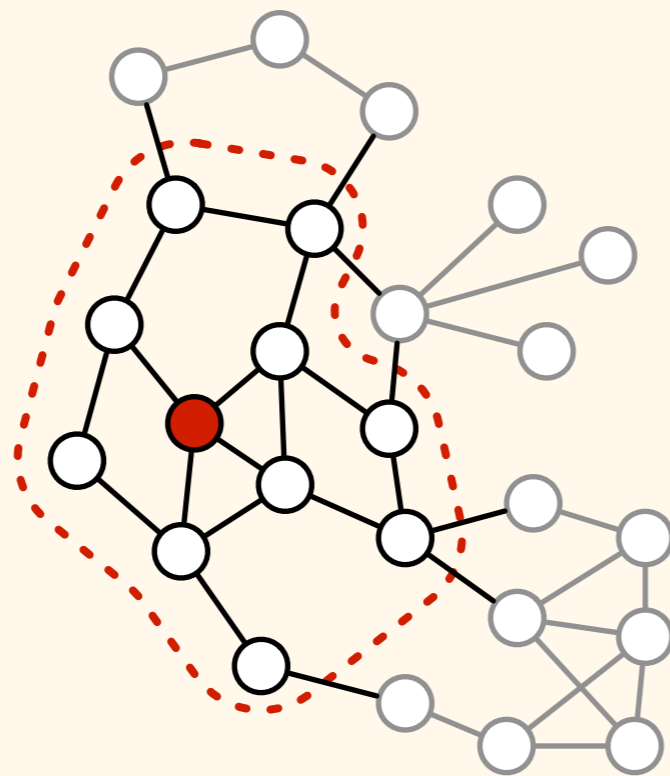
Local Algorithms

- *Local algorithm:*
each node operates based on its local neighbourhood only
 - Output is a function of local neighbourhood



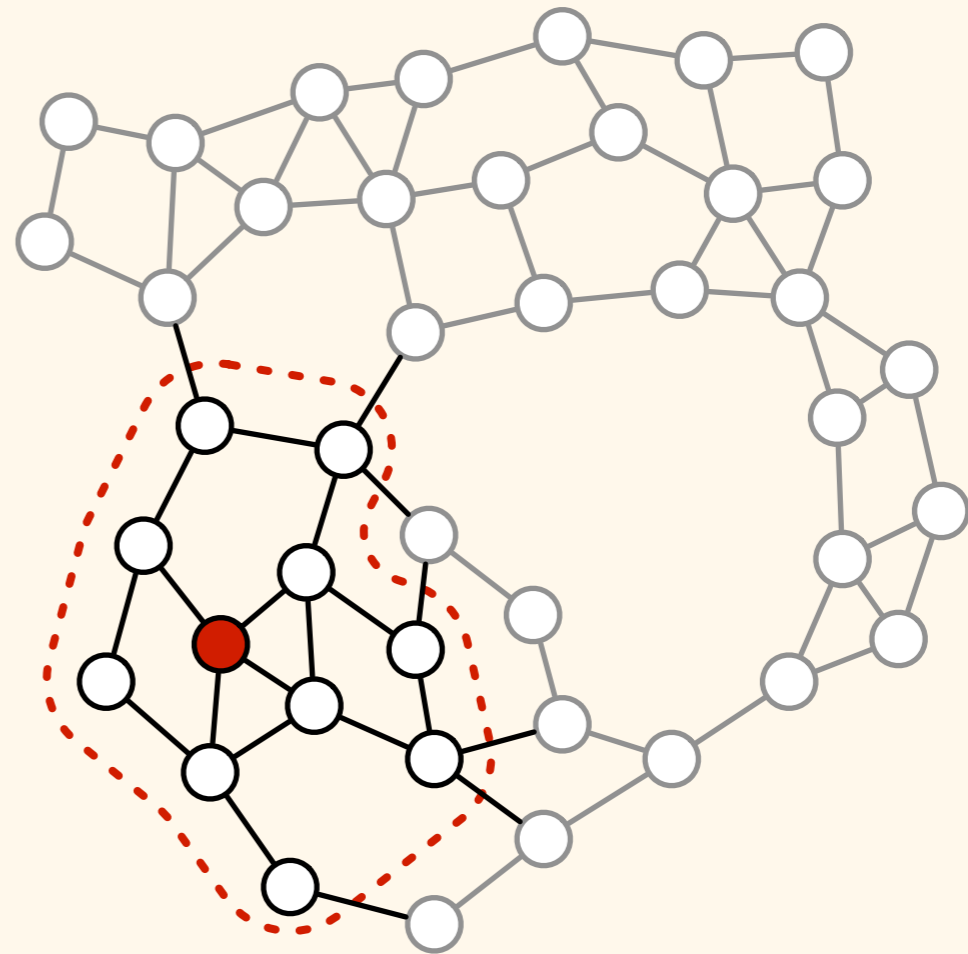
Local Algorithms

- Same neighbourhood, same output



Local Algorithms

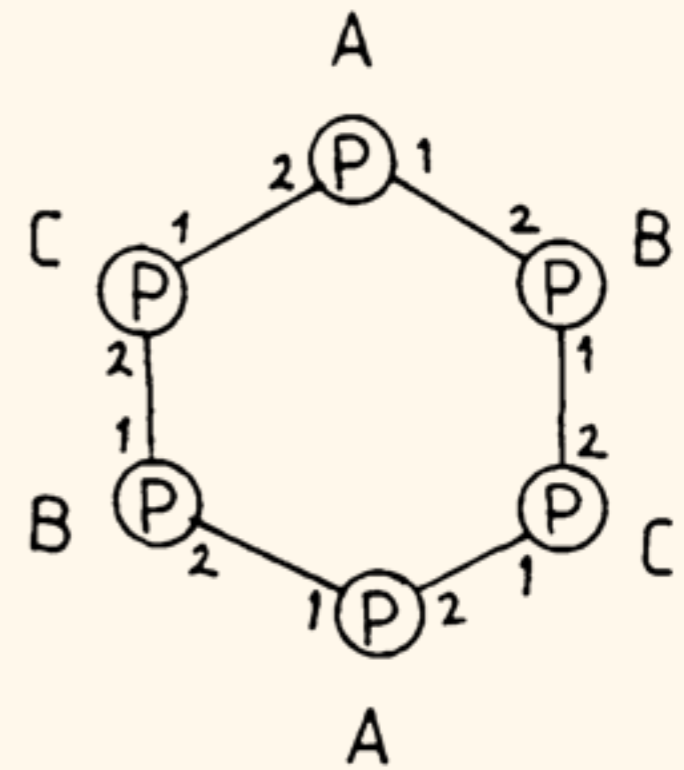
- Equivalently:
 - *Constant-time distributed algorithm*
 - Time = number of synchronous communication rounds



Advantages

- Fast and scalable distributed algorithm
 - By definition...
- Fault-tolerant and robust
 - Changes in input (or network structure): only *local changes in output*
 - We can quickly *recover from any failures*
- But do these exist?

Past

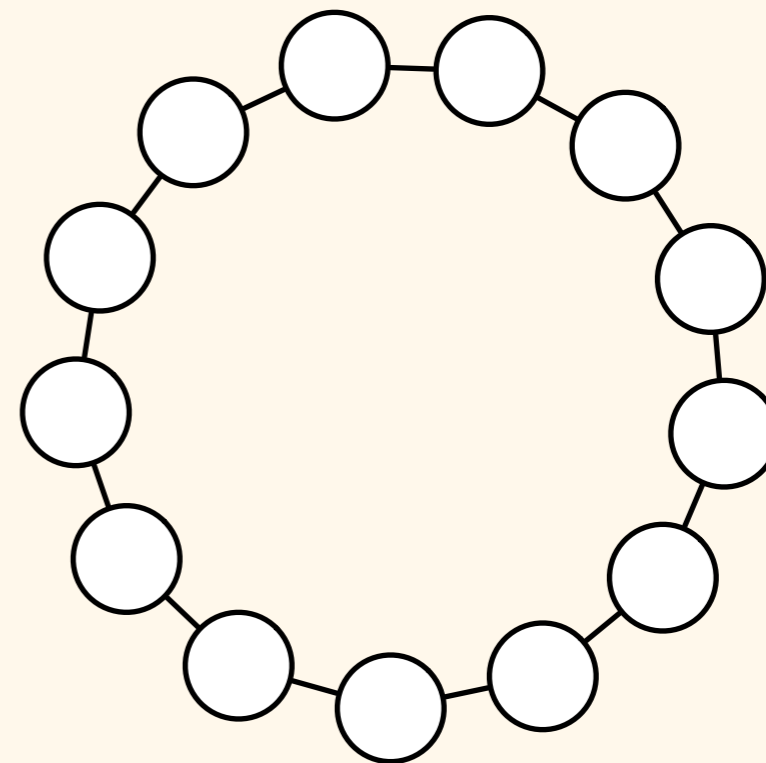


Bad News

- Long history of very strong negative results
 - *Linial* (1992)
 - *Naor & Stockmeyer* (1995)
 - *Czygrinow, Hańkowiak & Wawrzyniak* (2008)
 - *Lenzen & Wattenhofer* (2008)
 - using, e.g., results that date back to *Ramsey* (1930)

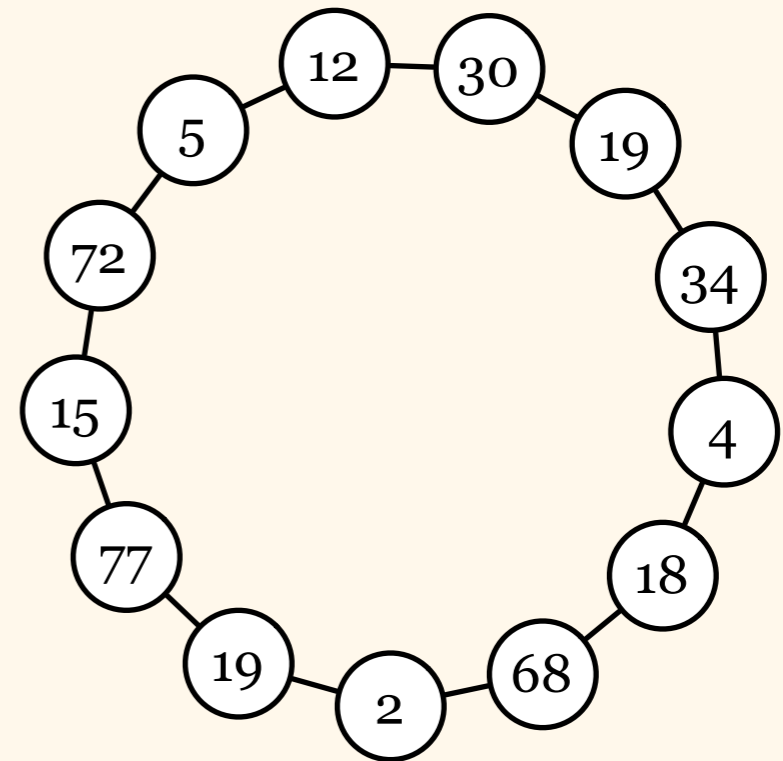
Bad News

- Even if your graph is a *cycle*...



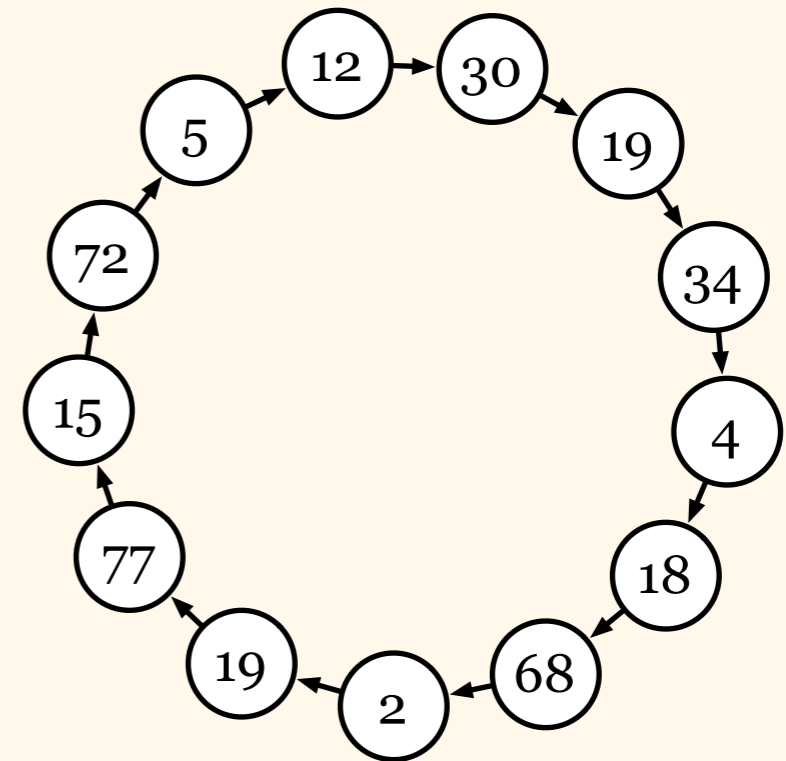
Bad News

- Even if your graph is a cycle...
- And even if you have *unique node identifiers*...



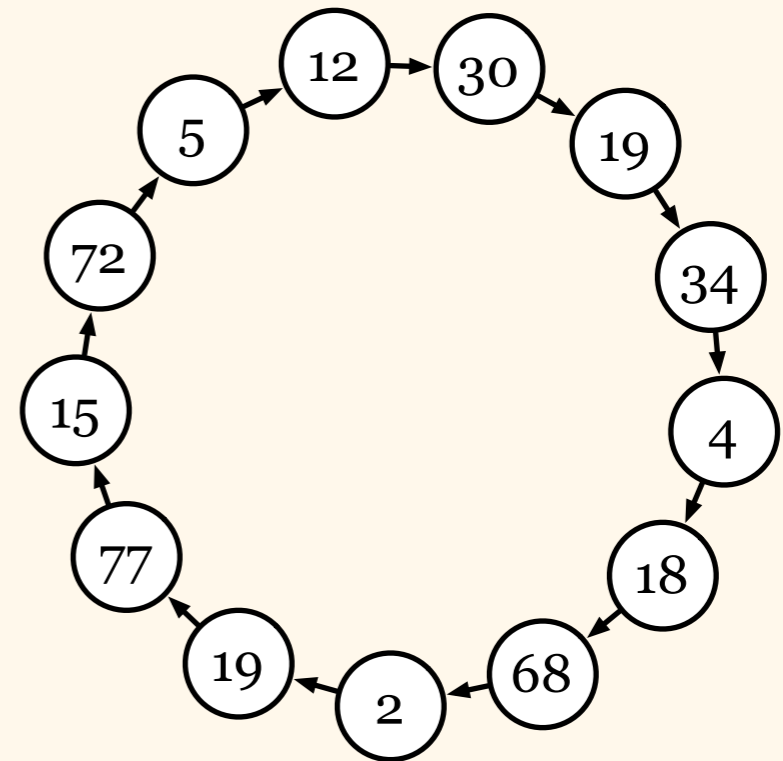
Bad News

- Even if your graph is a cycle...
- And even if you have unique node identifiers...
- And *orientation*...



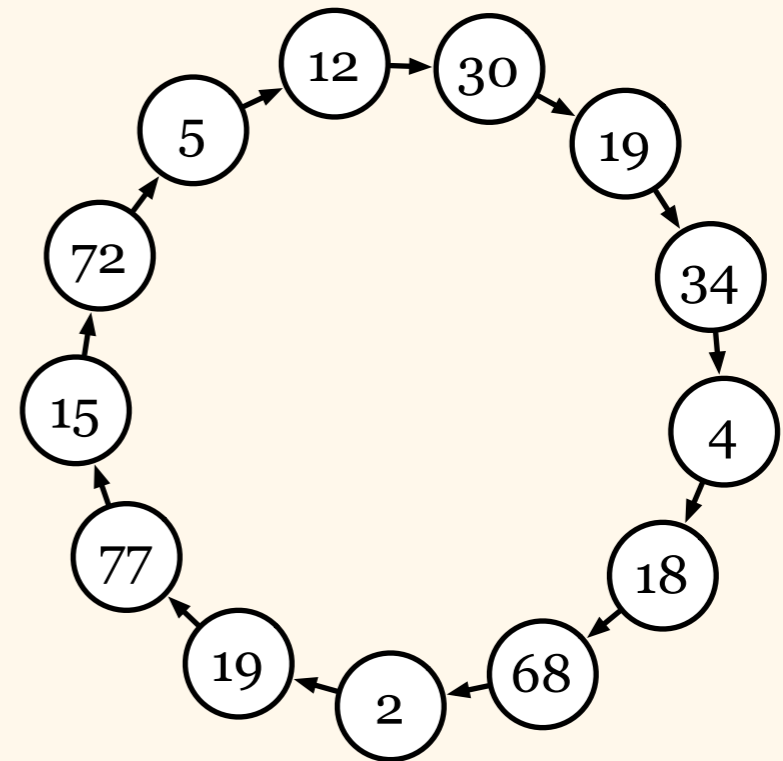
Bad News

- Even if your graph is a cycle...
- And even if you have unique node identifiers...
- And orientation...
- Then no matter which local algorithm you use, there is a *“bad input”*



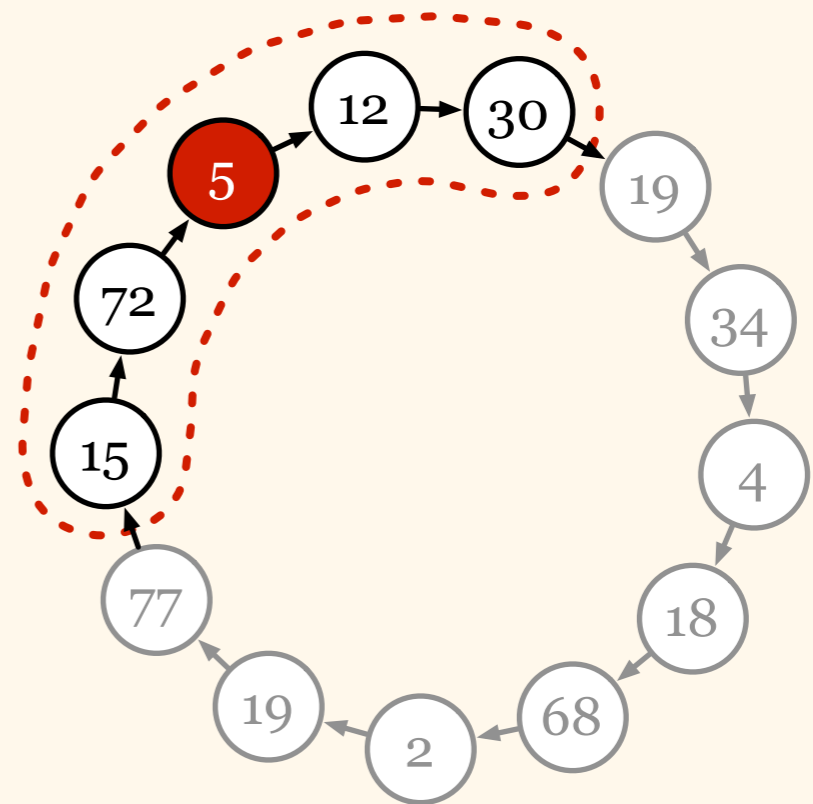
Bad News

- “Bad input”:
 - *Almost all nodes will produce the same output*
 - Graph colouring not possible
 - You can find only trivial independent sets, matchings, vertex covers, dominating sets, ...



Bad News

- Example: A is a local algorithm with $r = 2$, outputs from $\{1, 2, \dots, k\}$
 - Focus on oriented cycles
 - A maps 5-tuples of identifiers to local outputs
 - $A(15, 72, 5, 12, 30) = \dots$



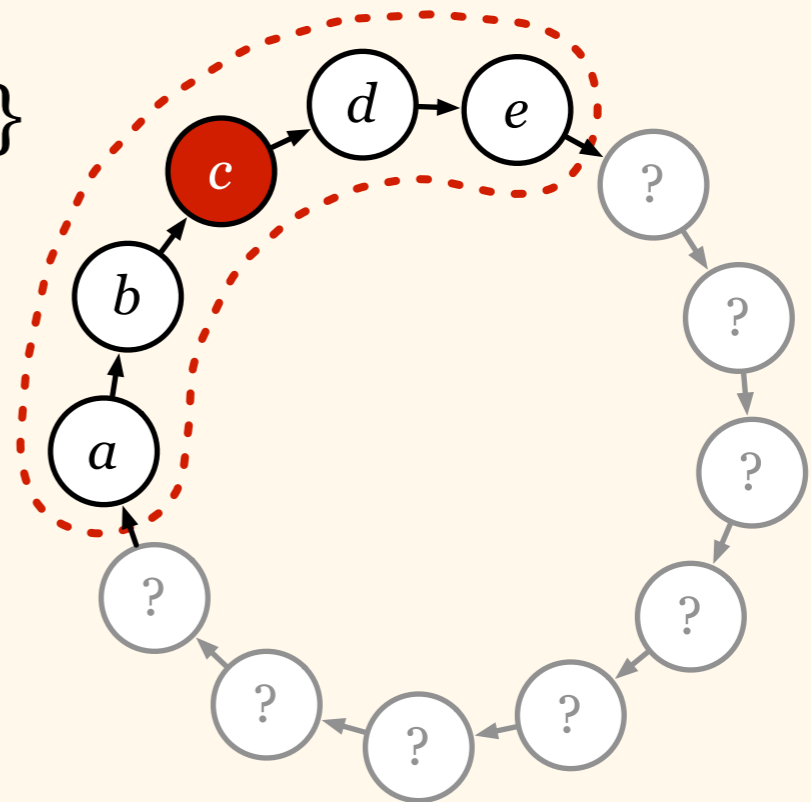
Bad News

- Example: A is a local algorithm with $r = 2$, outputs from $\{1, 2, \dots, k\}$

- Set of identifiers: $I = \{1, 2, \dots, N\}$

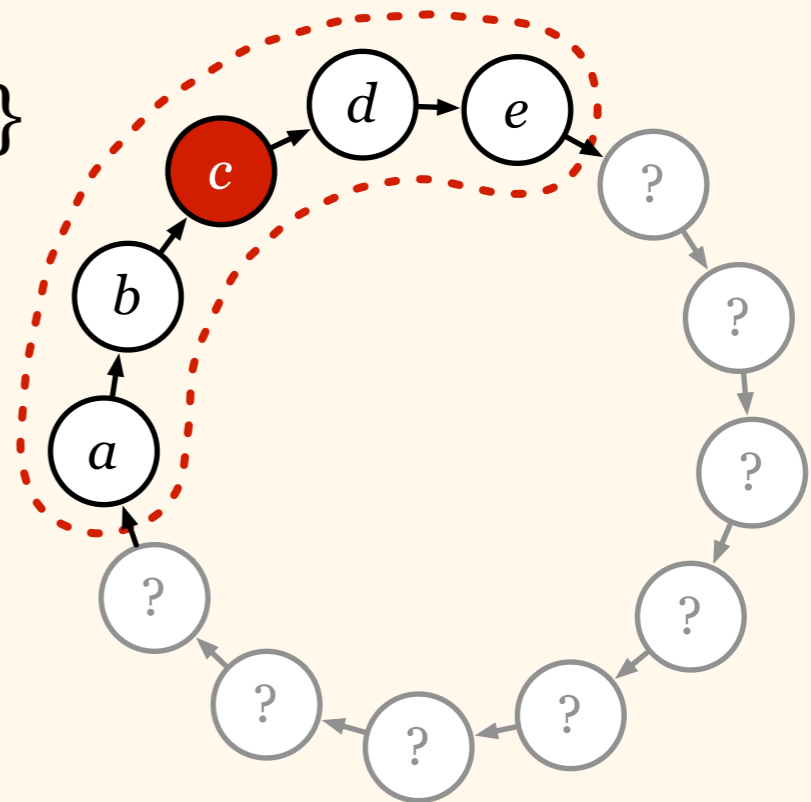
- Let $X = \{a, b, c, d, e\} \subseteq I$,
 $a < b < c < d < e$

- Define the *colour* $C(X)$ of X :
 $C(X) = A(a, b, c, d, e)$



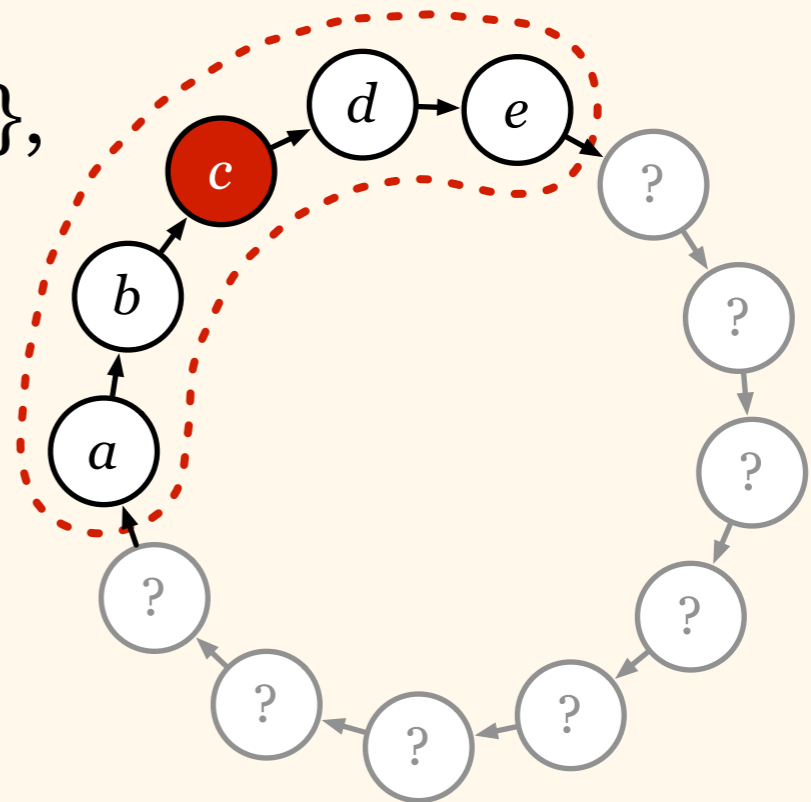
Bad News

- Example: A is a local algorithm with $r = 2$, outputs from $\{1, 2, \dots, k\}$
 - Set of identifiers: $I = \{1, 2, \dots, N\}$
 - Let $X = \{a, b, c, d, e\} \subseteq I$,
 $a < b < c < d < e$
 - Define the colour $C(X)$ of X :
 $C(X) = A(a, b, c, d, e)$
 - We will colour *all 5-subsets of I*



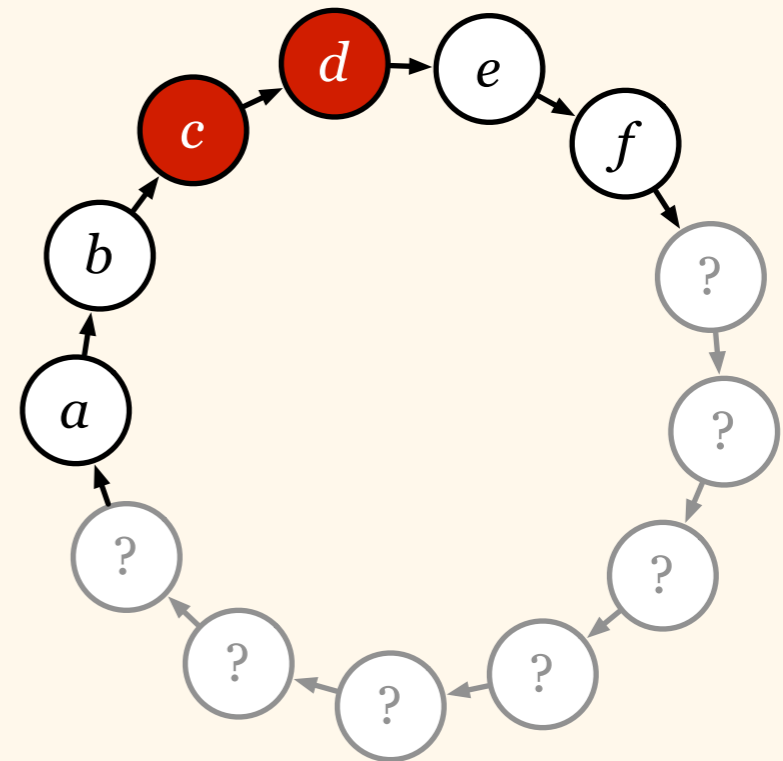
Bad News

- Example: A is a local algorithm with $r = 2$, outputs from $\{1, 2, \dots, k\}$
 - Set of identifiers: $I = \{1, 2, \dots, N\}$, colouring $C(X)$ of 5-subsets
 - *Ramsey*: if N is large enough, there exists a large *monochromatic* subset $M \subseteq I$
 - All 5-subsets $X \subseteq M$ have the same colour $C(X)$



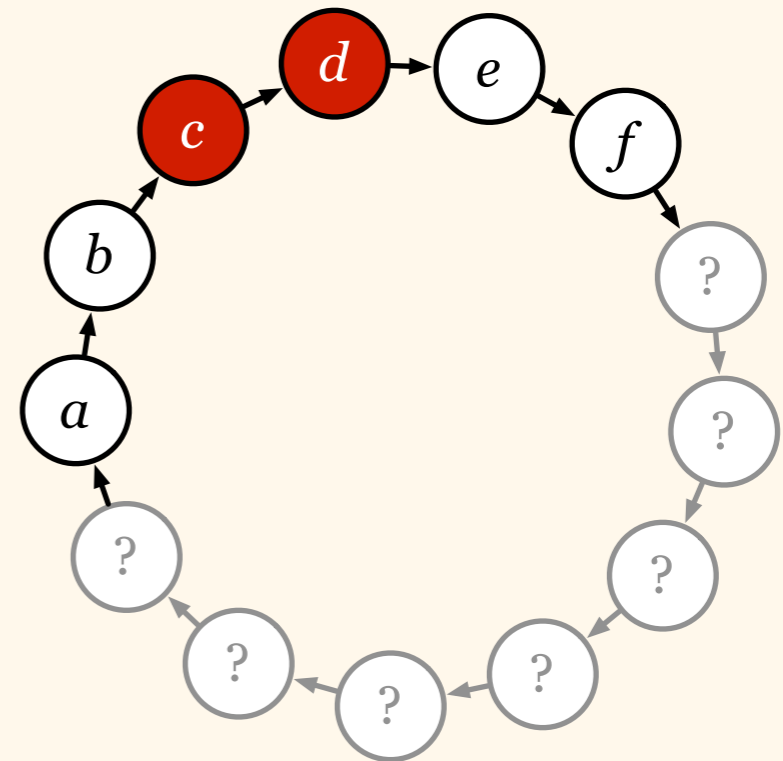
Bad News

- Example: A is a local algorithm with $r = 2$, outputs from $\{1, 2, \dots, k\}$
 - Assume that $M = \{a, b, c, d, e, f\}$ is a monochromatic subset, $a < b < c < d < e < f$
 - $C(\{a, b, c, d, e\}) = C(\{b, c, d, e, f\})$
 - $A(a, b, c, d, e) = A(b, c, d, e, f)$



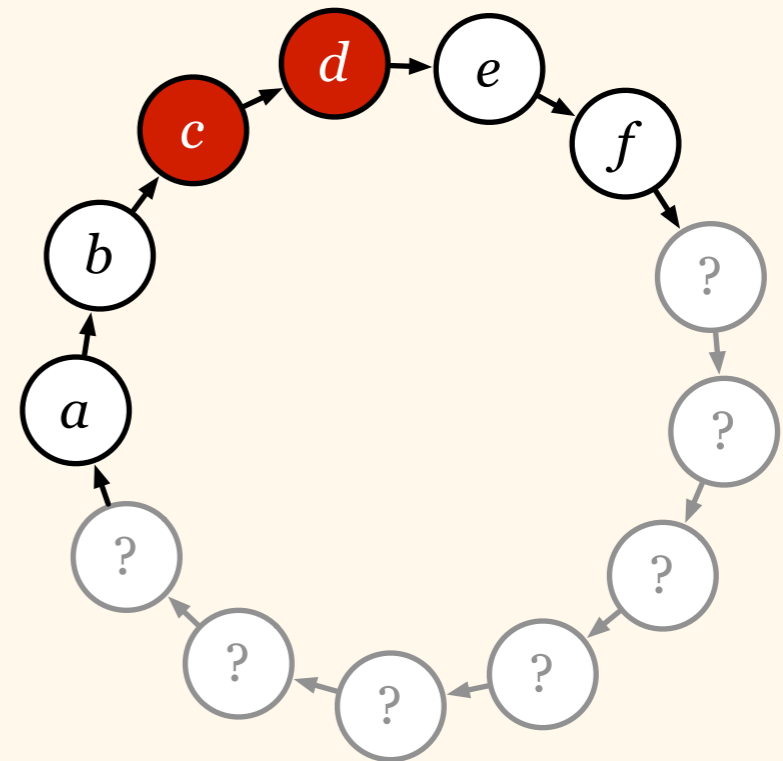
Bad News

- Example: A is a local algorithm with $r = 2$, outputs from $\{1, 2, \dots, k\}$
 - We have found a “bad input”: nodes with identifiers c and d are adjacent and they produce the same output
 - We already proved that A cannot produce a valid graph colouring!



Bad News

- Example: A is a local algorithm with $r = 2$, outputs from $\{1, 2, \dots, k\}$
 - We can apply the same idea for any value of r
 - And we can “boost” the argument and show that *almost all nodes* will produce the same output



Bad News

- For
 - any local algorithm A that finds an independent set,
 - any constant $\varepsilon > 0$, and
 - sufficiently large n ,

we can choose unique identifiers in an n -cycle so that A outputs an independent set with only εn nodes

Bad News

- For
 - any local algorithm A that finds a vertex cover,
 - any constant $\varepsilon > 0$, and
 - sufficiently large n ,

we can choose unique identifiers in an n -cycle so that A outputs a vertex cover with at least $(1 - \varepsilon)n$ nodes

Dealing with Bad News

- Three traditional escapes:
 - Randomised algorithms
 - Geometric information
 - “Almost local” algorithms

Dealing with Bad News

- Three traditional escapes:
 - *Randomised algorithms*
 - Geometric information
 - “Almost local” algorithms

Randomised Algorithms

- Nodes can *roll dice* or *toss coins*

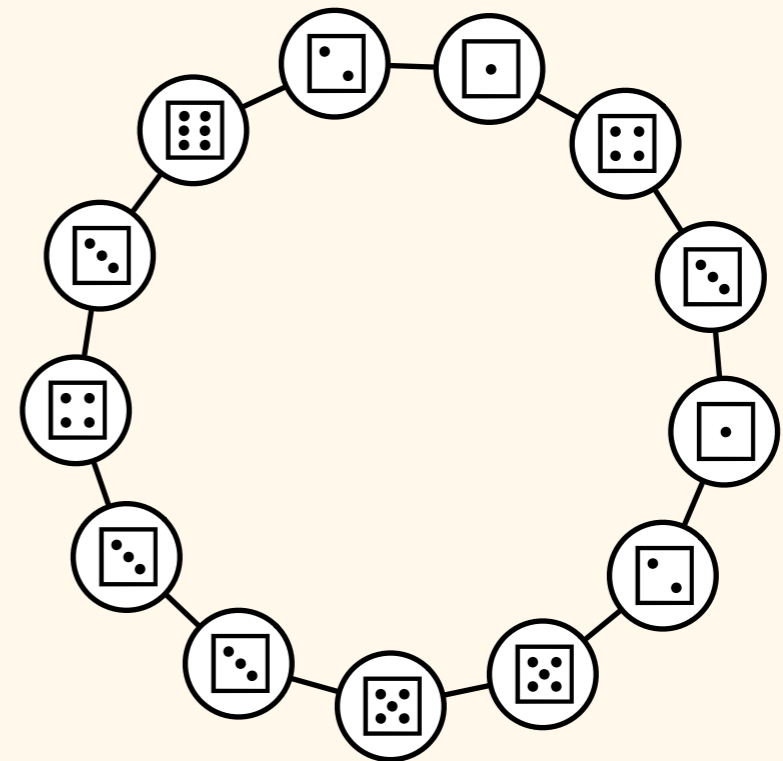


Randomised Algorithms

- Nodes can *roll dice* or *toss coins*
- We cannot guarantee that we find a good solution
 - Worst case: all coin tosses equal, no new information
- But we can find a good solution *with high probability* or *in expectation*

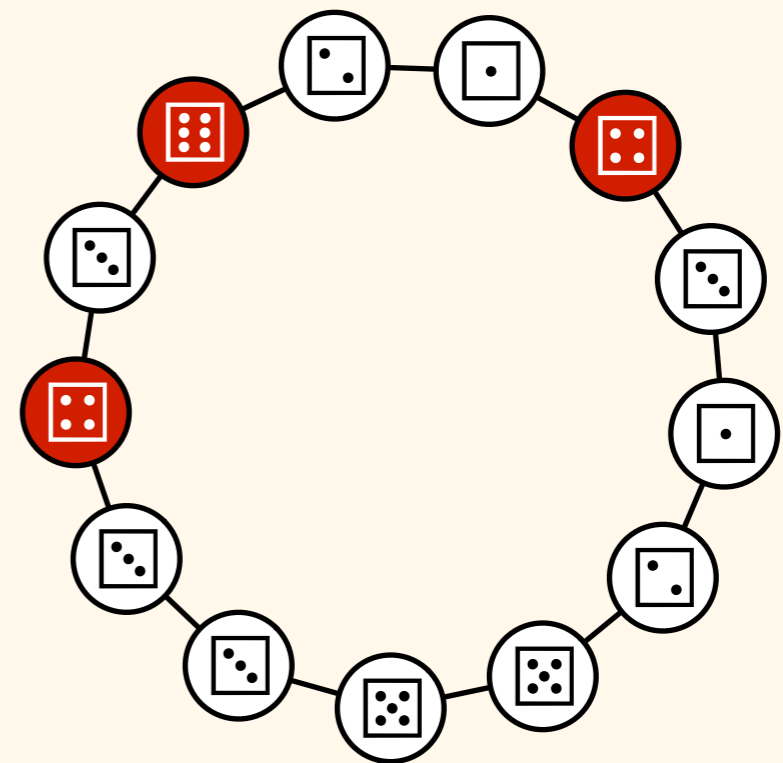
Randomised Algorithms

- *Example:* finding an independent set I
 - Each node v picks uniformly at random
 $X(v) = \square, \square, \square, \square, \square, \square$



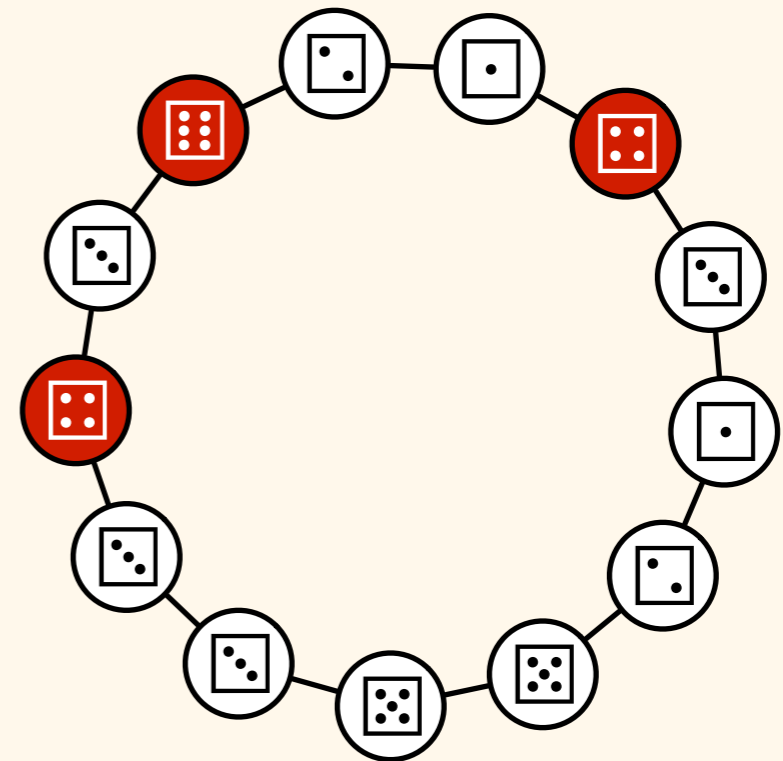
Randomised Algorithms

- *Example:* finding an independent set I
 - Each node v picks uniformly at random $X(v) = \square, \square, \square, \square, \square, \square$
 - Node v joins I if $X(v)$ is (strict) *local maximum*



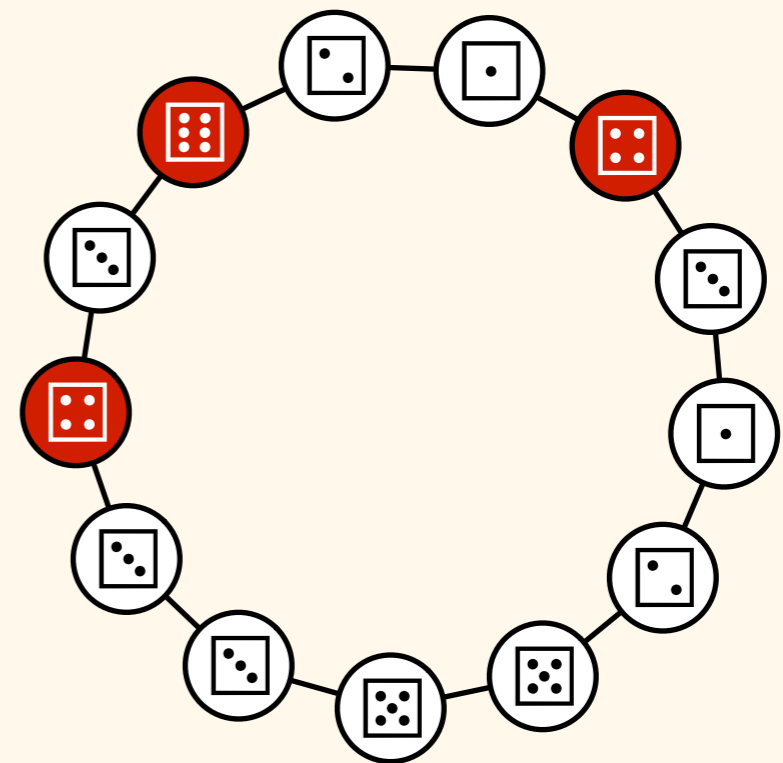
Randomised Algorithms

- *Example:* finding an independent set I
 - Each node v picks uniformly at random $X(v) = \square, \square, \square, \square, \square, \square$
 - Node v joins I if $X(v)$ is (strict) local maximum
- By construction, I is an *independent set*



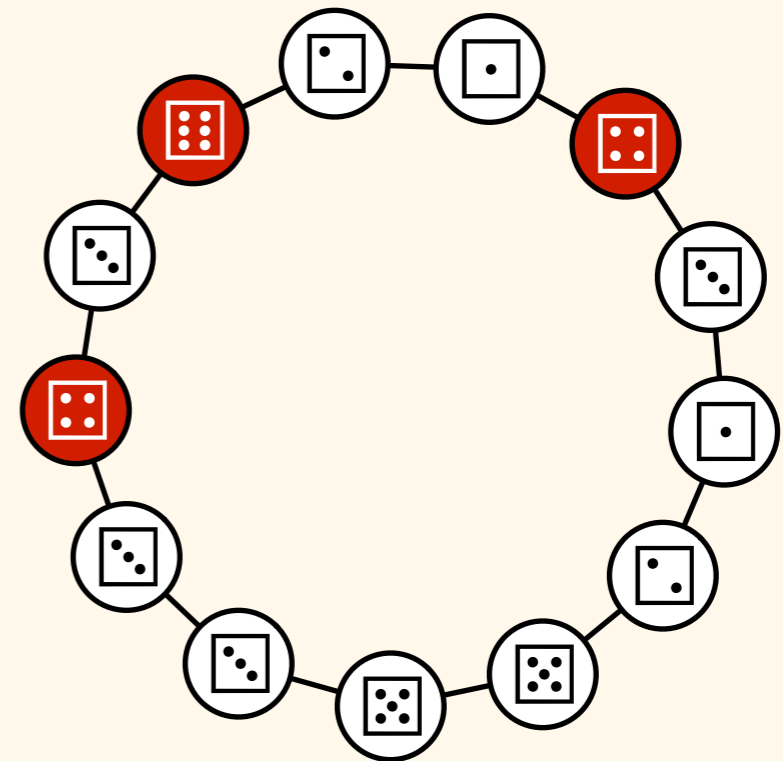
Randomised Algorithms

- *Example:* finding an independent set I
 - Each node v picks uniformly at random $X(v) = \square, \square, \square, \square, \square, \square$
 - Node v joins I if $X(v)$ is (strict) local maximum
- Expected size of I is *reasonably large*



Randomised Algorithms

- *Example:* finding an independent set I
 - A local randomised algorithm can find a large independent set
 - Approximation algorithm (in expectation)
 - However, we cannot find *maximum* independent set or *maximal* independent set

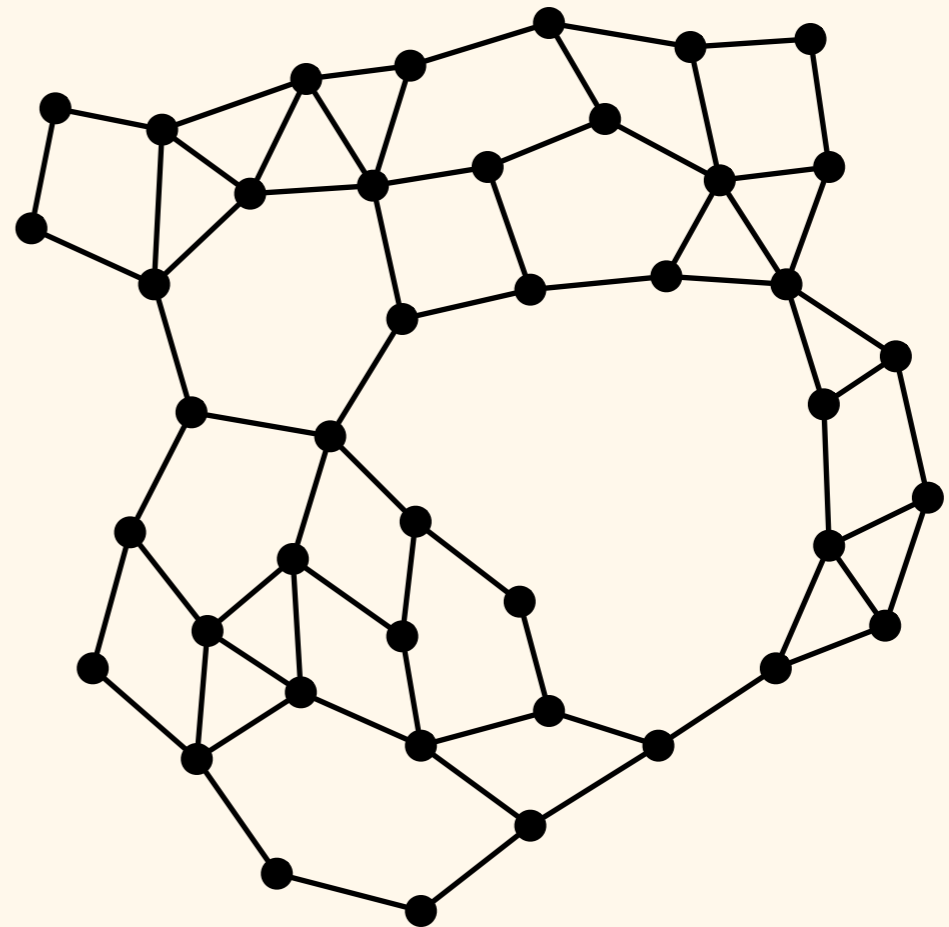


Dealing with Bad News

- Three traditional escapes:
 - Randomised algorithms
 - *Geometric information*
 - “Almost local” algorithms

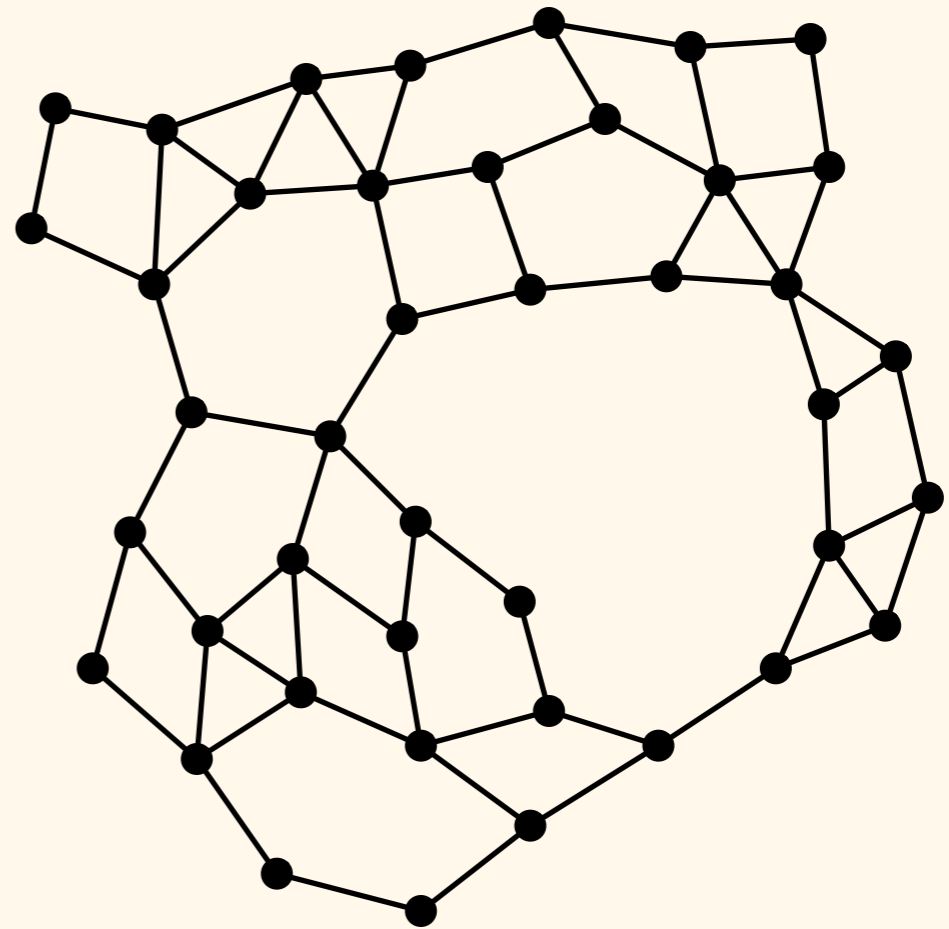
Geometric Graphs

- Assume that nodes are *points in the plane*
- Assume “reasonable” embedding



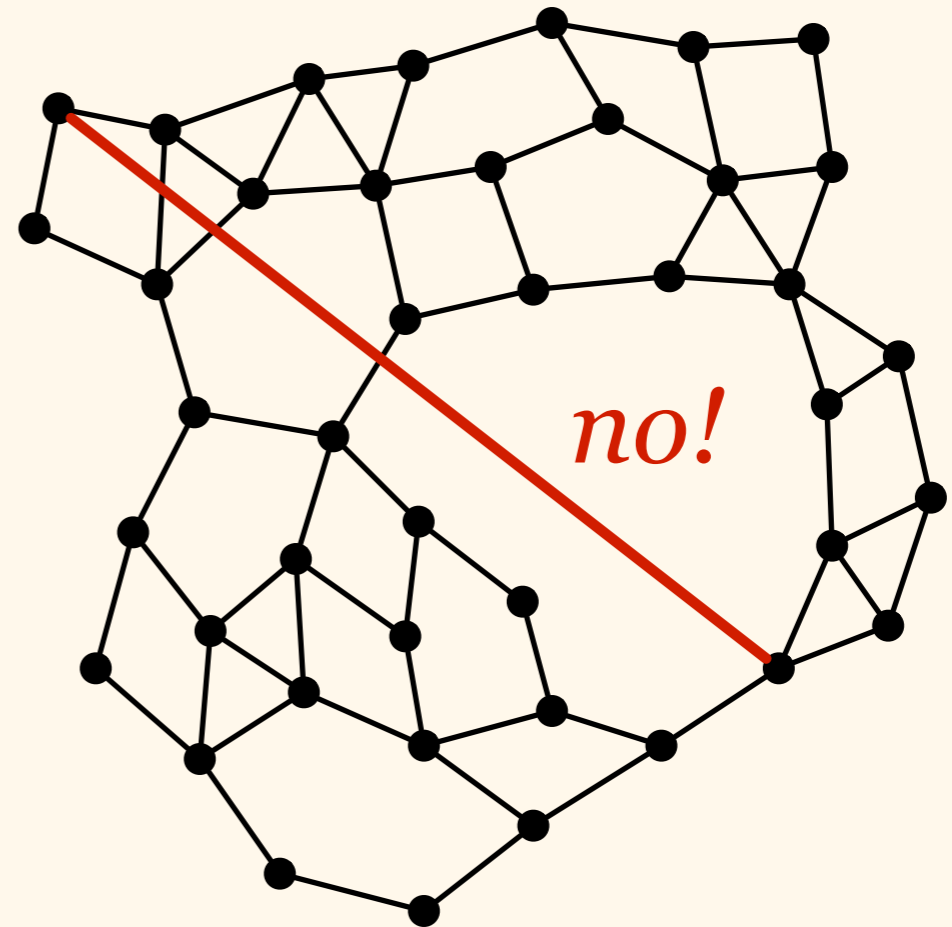
Geometric Graphs

- Assume that nodes are points in the plane
- Assume “reasonable” embedding
 - *Civilised graph*



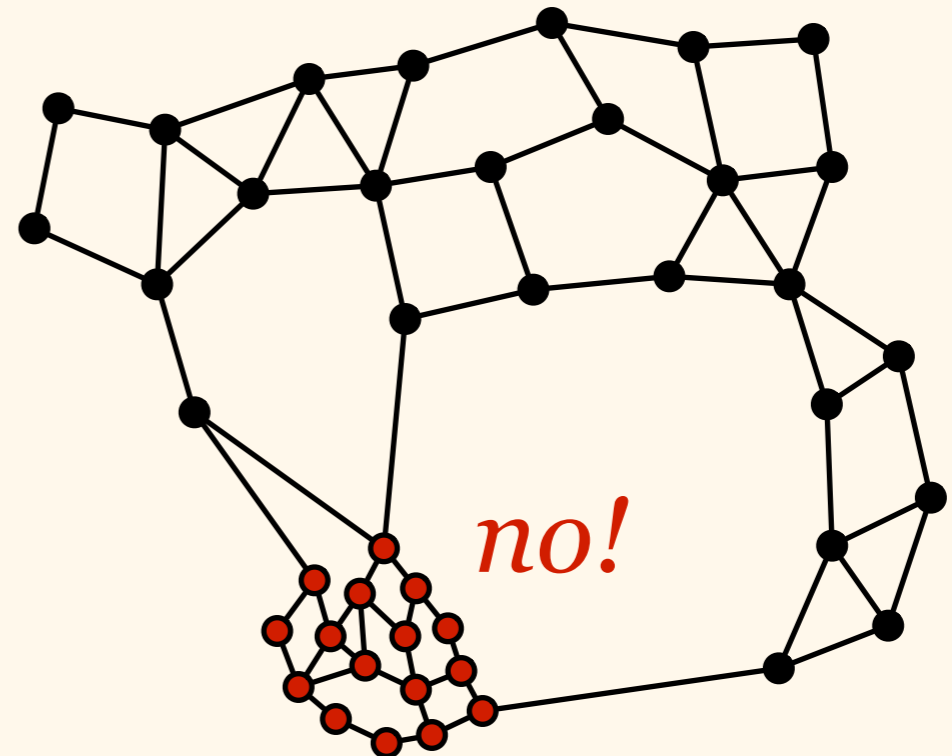
Geometric Graphs

- Assume that nodes are points in the plane
- Assume “reasonable” embedding
 - Civilised graph:
edges not too long...



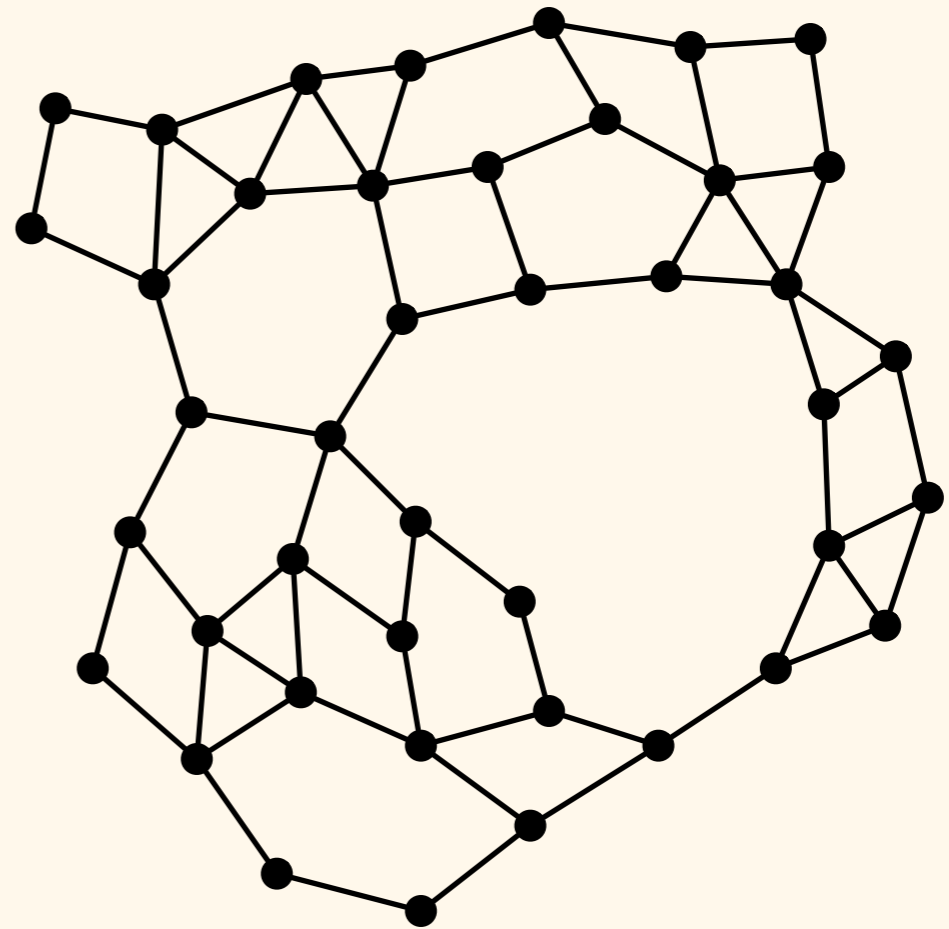
Geometric Graphs

- Assume that nodes are points in the plane
- Assume “reasonable” embedding
 - Civilised graph:
edges not too long,
nodes not in too dense



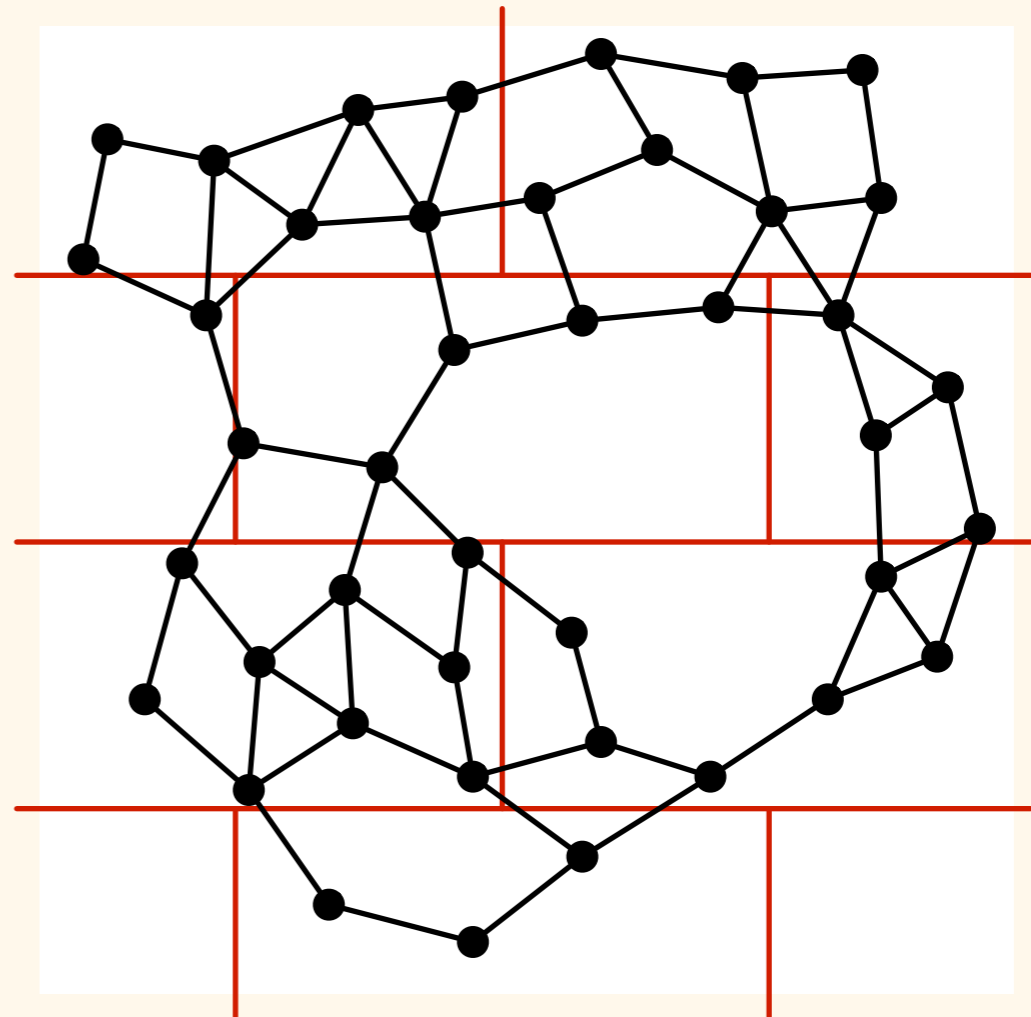
Geometric Graphs

- Assume that nodes are points in the plane
- Assume “reasonable” embedding
 - Civilised graph
 - Unit disk graph
 - Quasi unit disk graph...



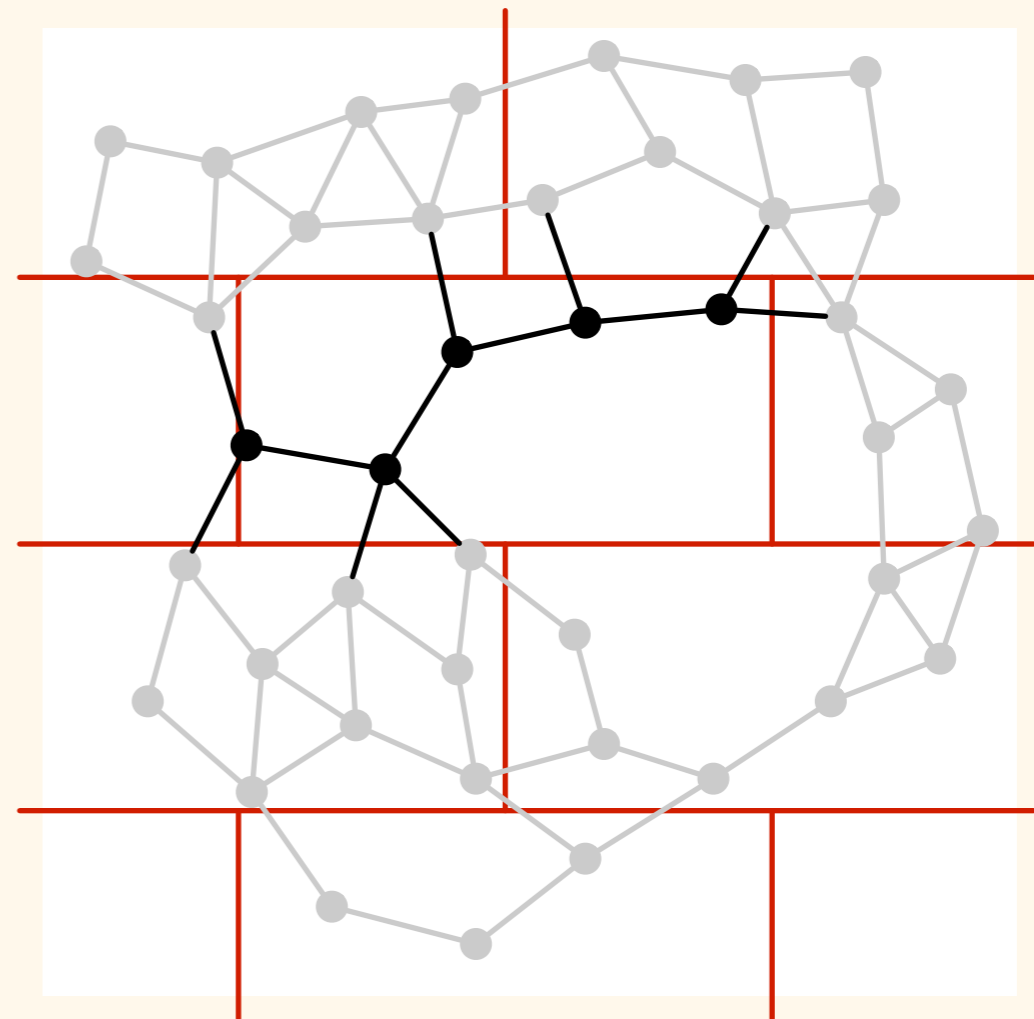
Geometric Graphs

- Exploit coordinates
 - a simple approach:
divide-and-conquer
 - e.g., partition the plane
in rectangular *tiles*



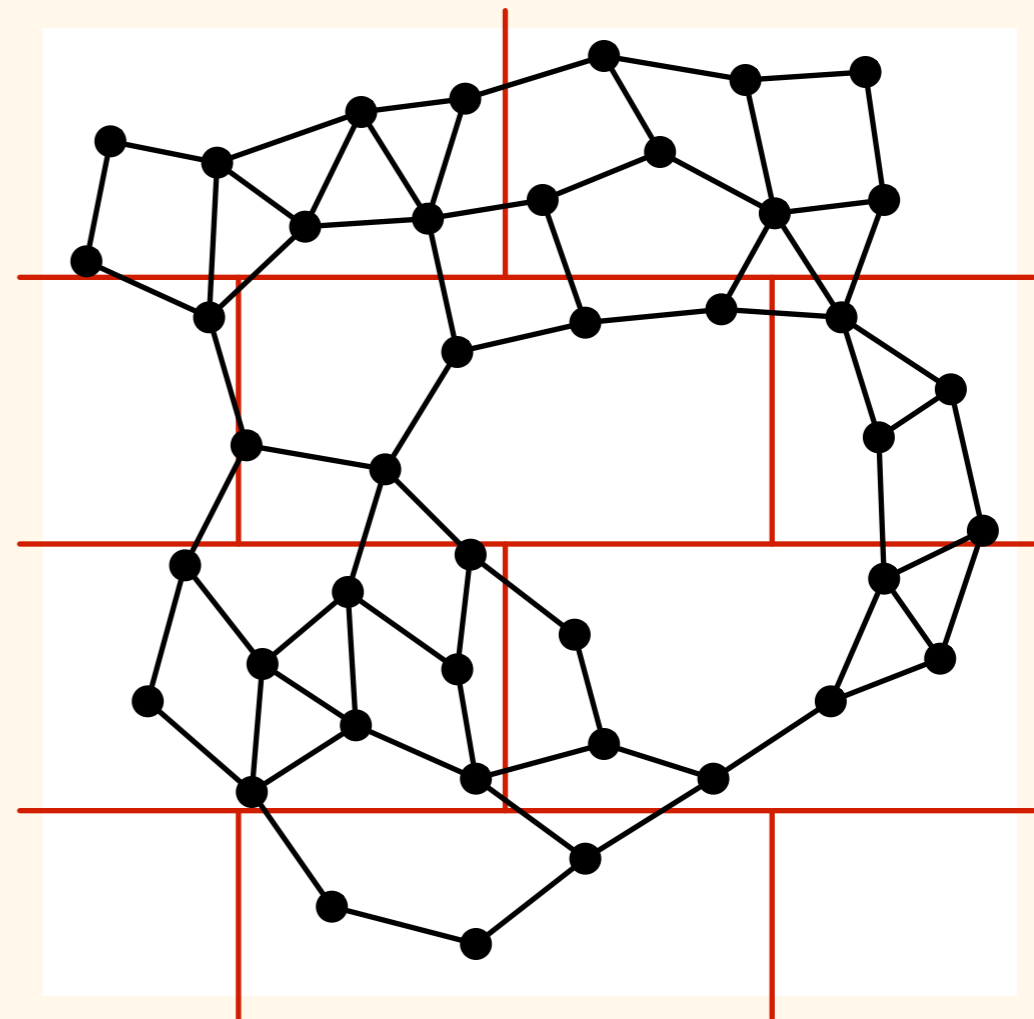
Geometric Graphs

- Exploit coordinates
 - each tile defines a *constant-size subproblem*
 - solve the subproblem locally within each tile (in parallel for all tiles)



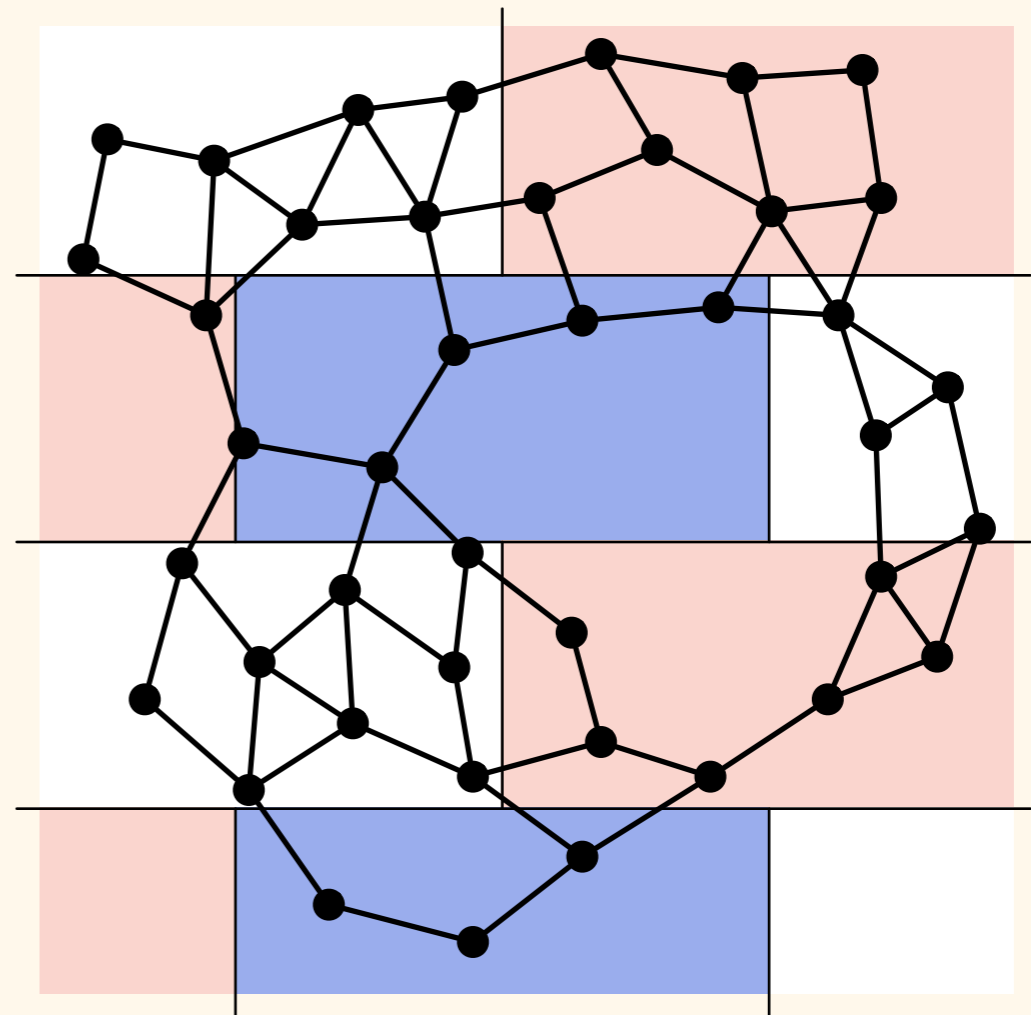
Geometric Graphs

- Exploit coordinates
 - each tile defines a constant-size subproblem
 - solve the subproblem locally within each tile
 - *merge* the solutions of the subproblems



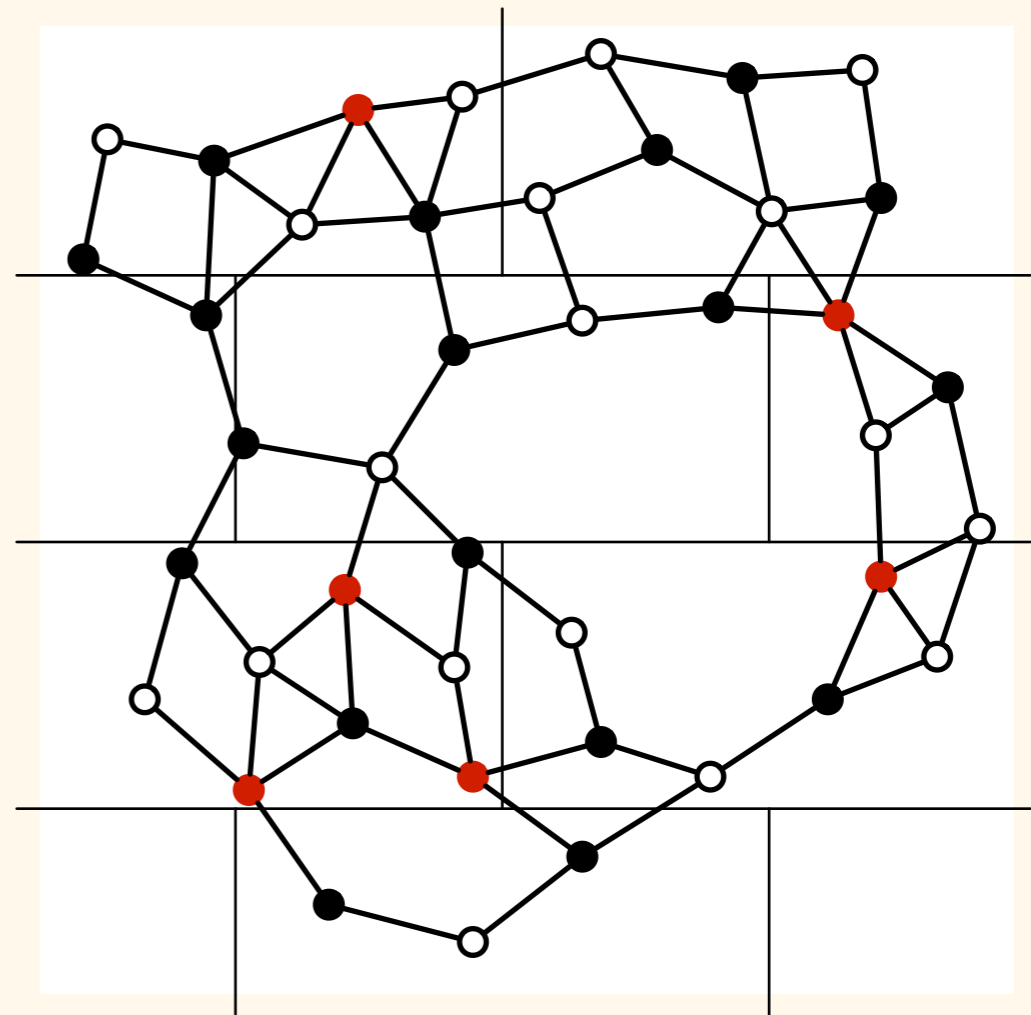
Geometric Graphs

- Graph colouring:
 - $f = 3$ -colouring of tiles
 - all edges are short
 - there is no edge that joins e.g. a blue tile and another blue tile



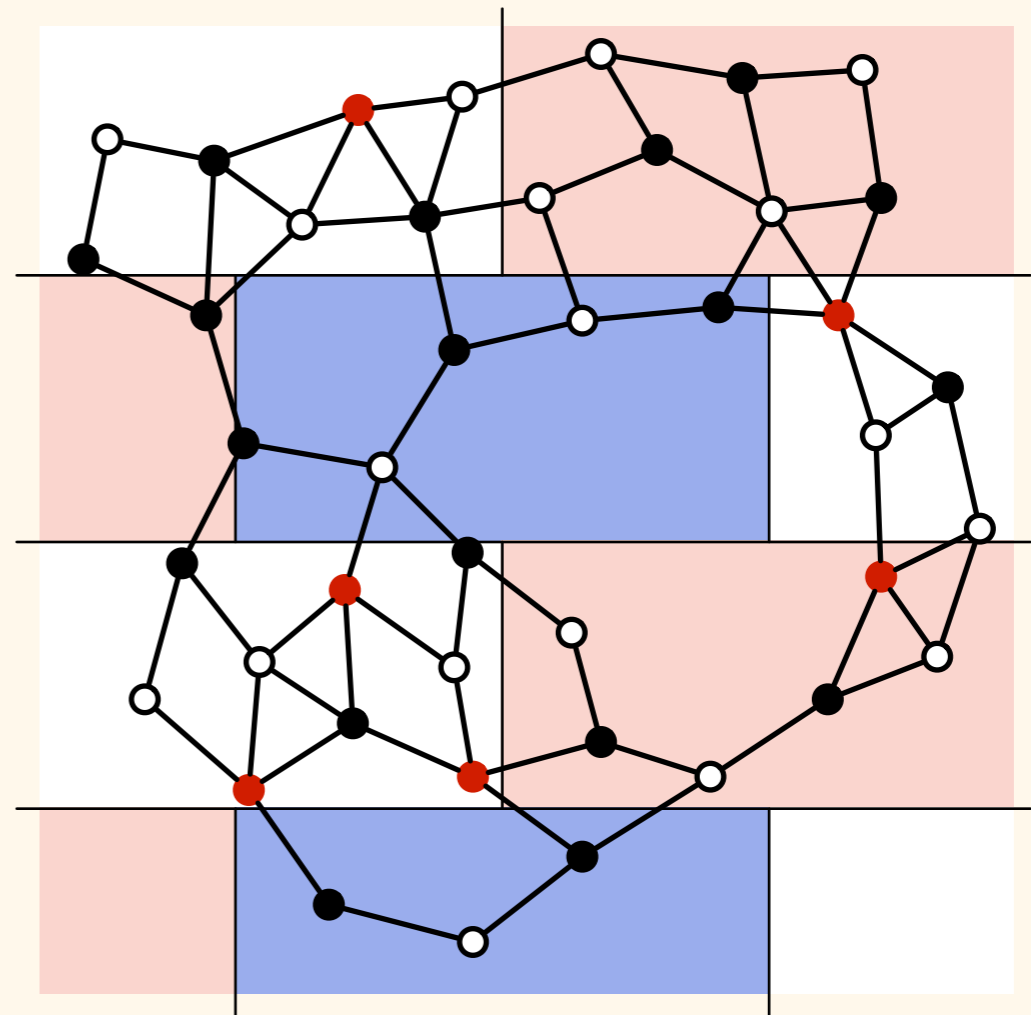
Geometric Graphs

- Graph colouring:
 - $f = 3$ -colouring of tiles
 - $g = k$ -colouring that is valid *inside* each tile
 - can be solved by brute force



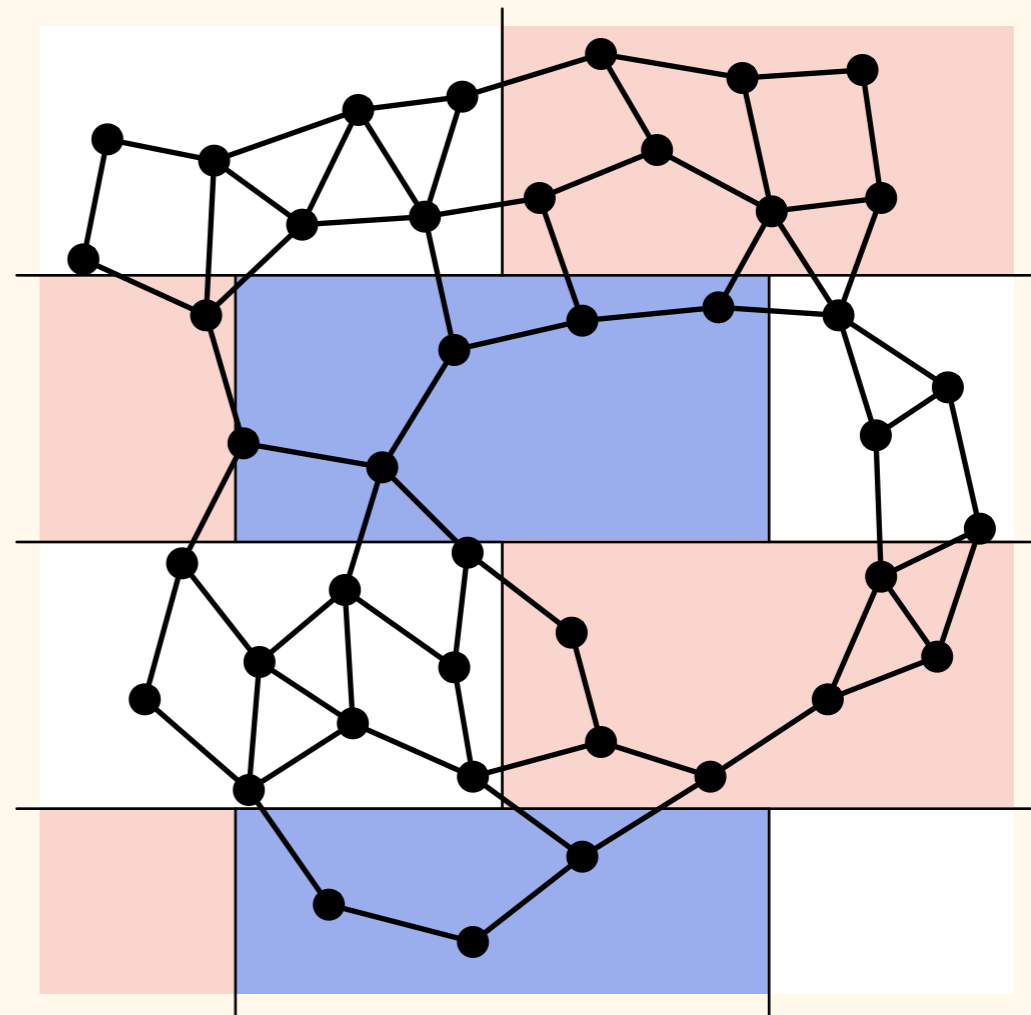
Geometric Graphs

- Graph colouring:
 - $f = 3$ -colouring of tiles
 - $g = k$ -colouring that is valid *inside* each tile
 - Output: (f, g)
 - Valid $3k$ -colouring!



Geometric Graphs

- Simple local algorithms:
 - *maximal* matchings, independent sets, ...
 - *approximation algorithms* for vertex covers, dominating sets, colourings, ...



Dealing with Bad News

- Three traditional escapes:
 - Randomised algorithms
 - Geometric information
 - *“Almost local” algorithms*

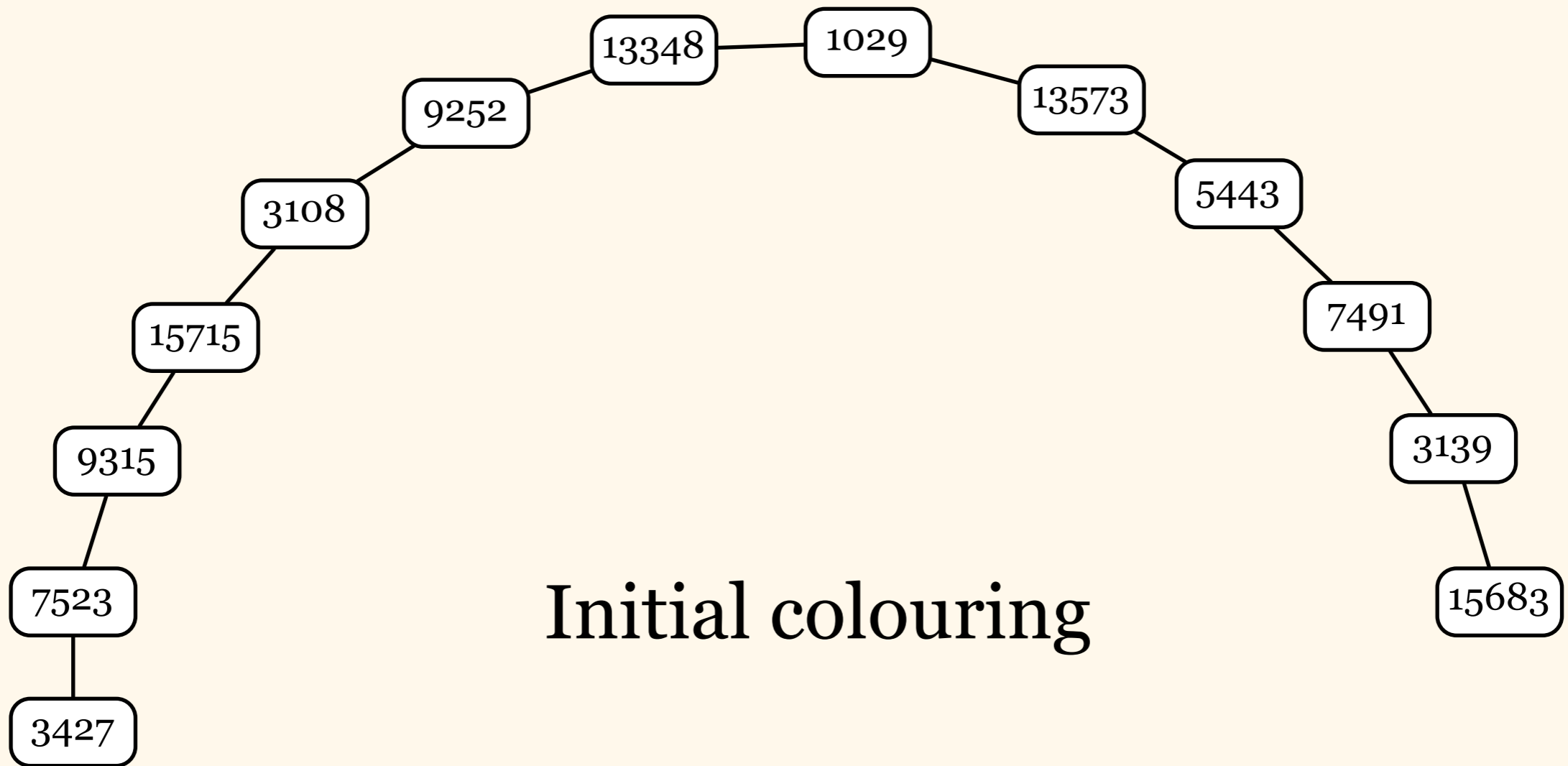
Almost Local Algorithms

- We cannot find non-trivial solutions in a cycle in $O(1)$ rounds
- But we can do it in $O(\log^* n)$ rounds!
 - $\log^* n =$ iterated logarithm
 - $0 \leq \log^* n \leq 7$ for all real-world values of n
 - Good enough?

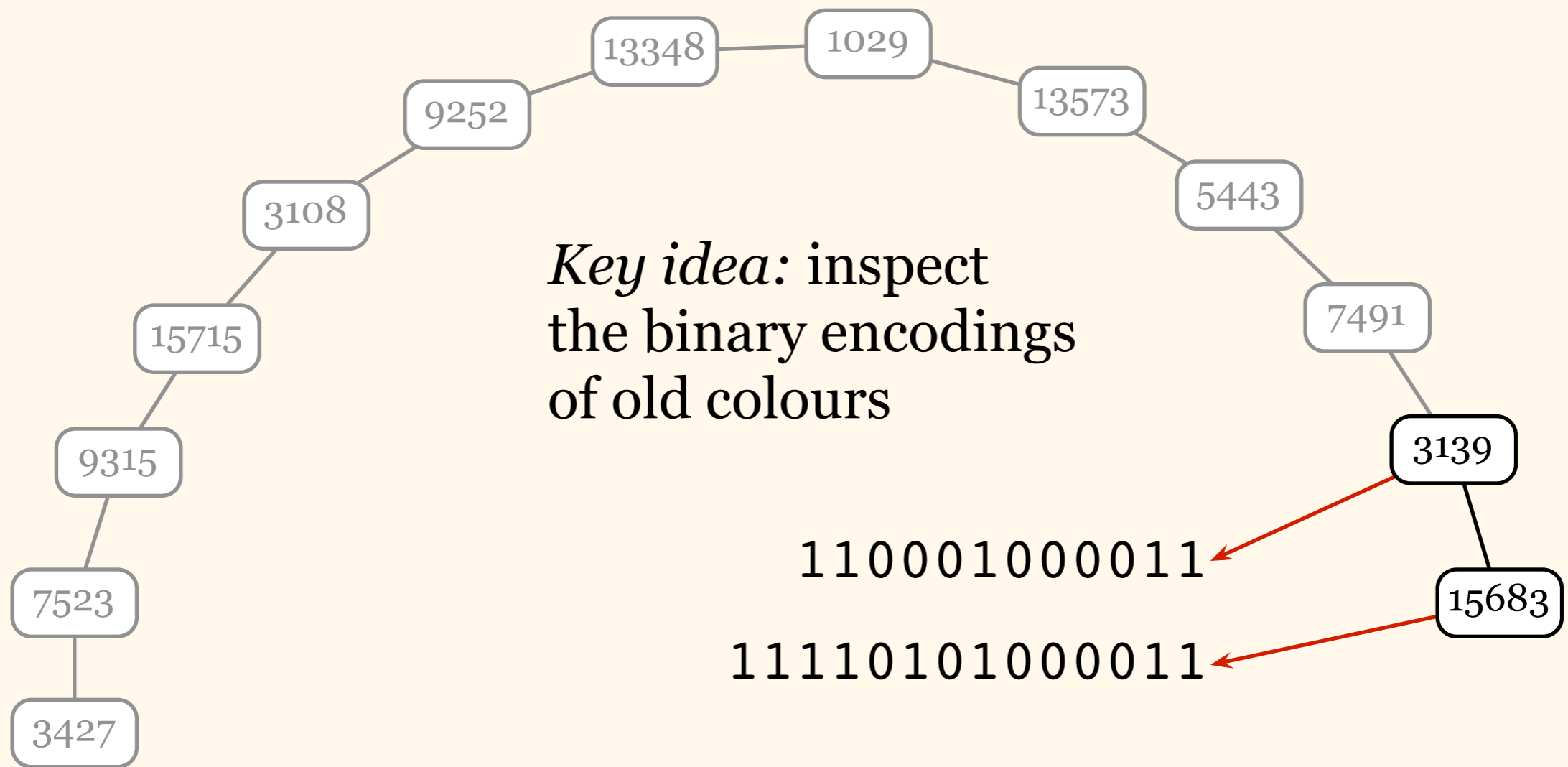
Almost Local Algorithms

- Main tool: colour reduction
 - *Cole & Vishkin* (1986)
 - *Goldberg, Plotkin & Shannon* (1988)
- Bit manipulation trick:
 - From k colours to $O(\log k)$ colours in one step
 - Initially $\text{poly}(n)$ colours: unique identifiers
 - Iterate $O(\log^* n)$ times until $O(1)$ colours

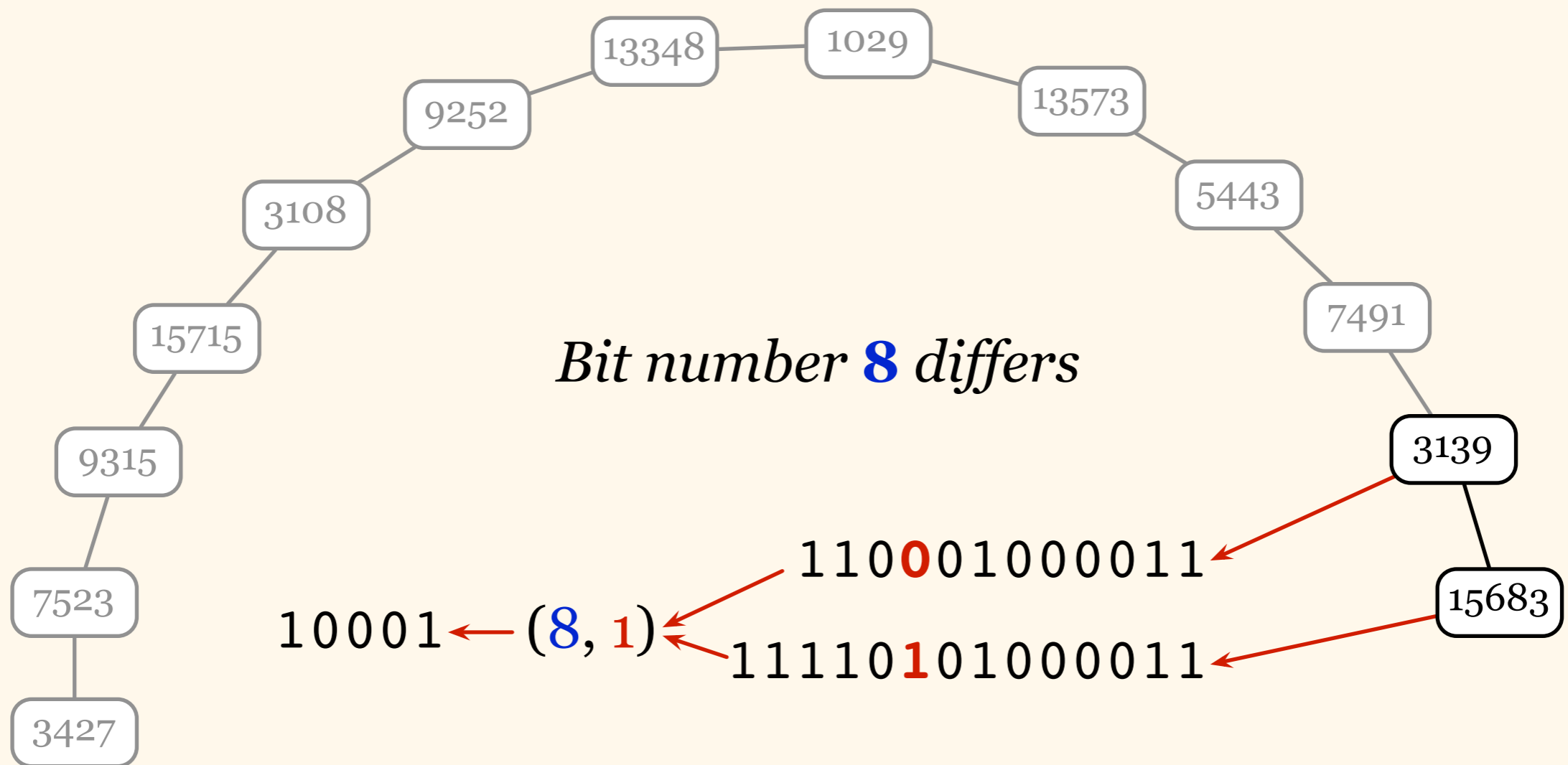
Almost Local Algorithms



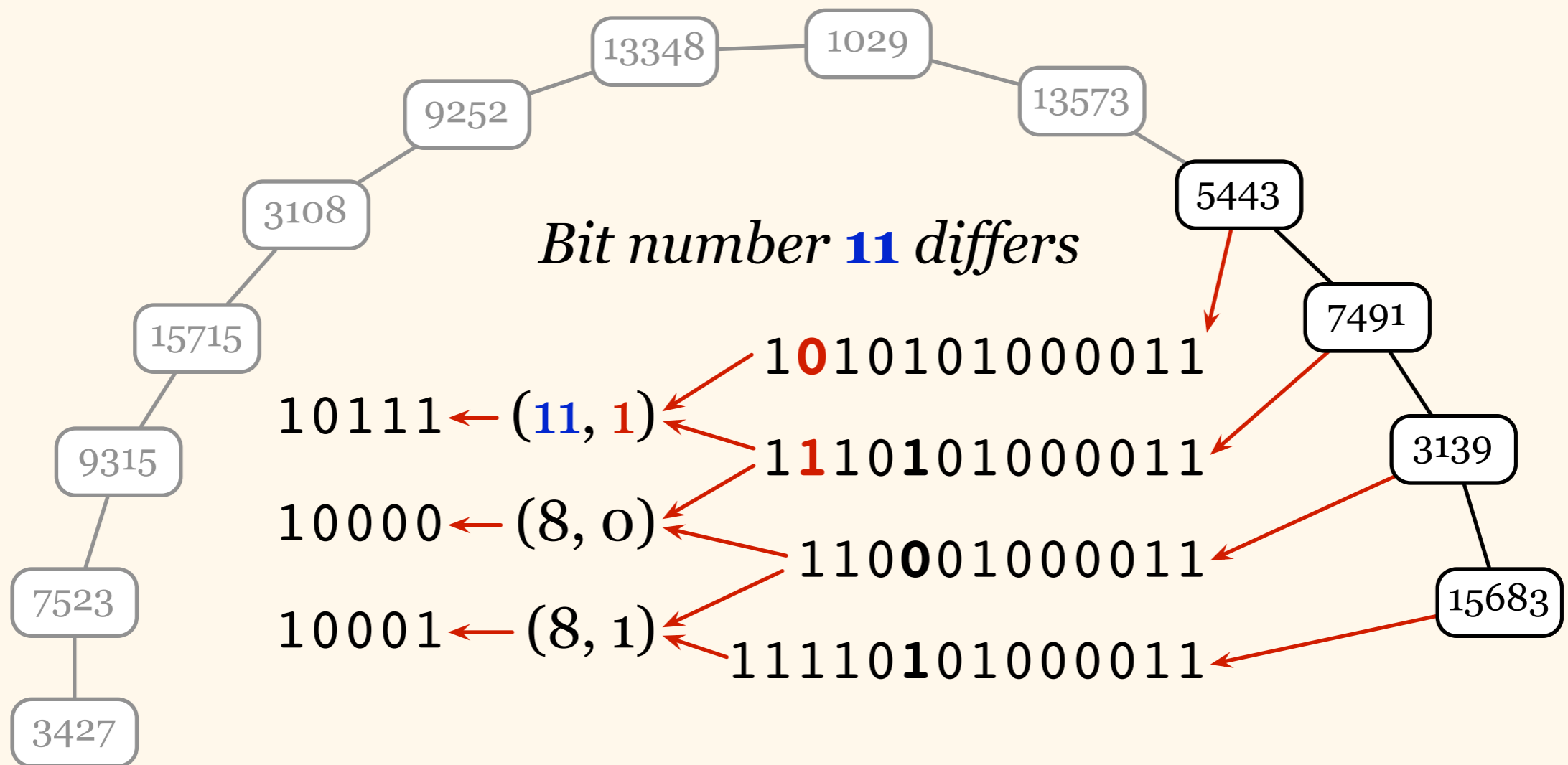
Almost Local Algorithms



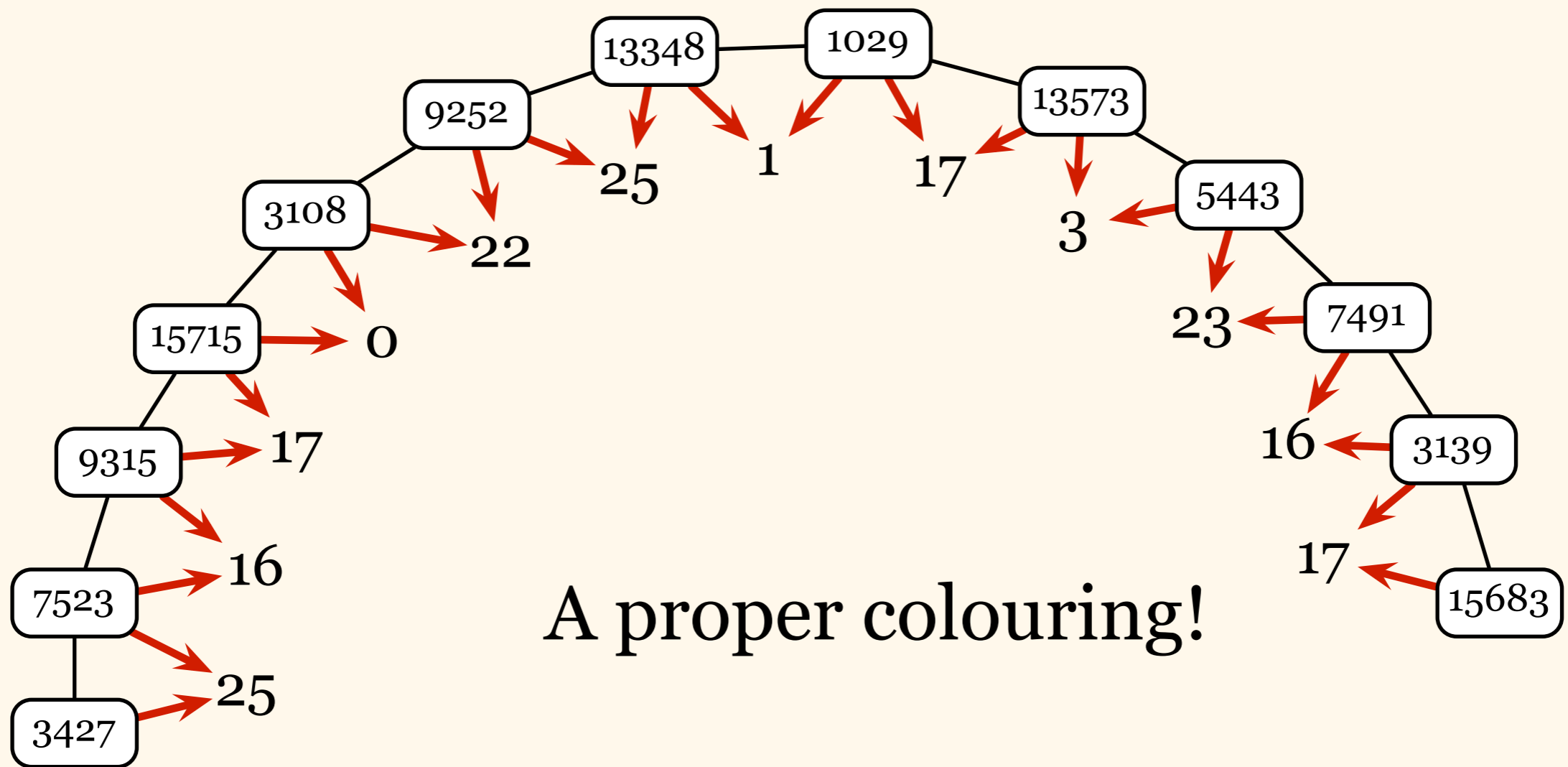
Almost Local Algorithms



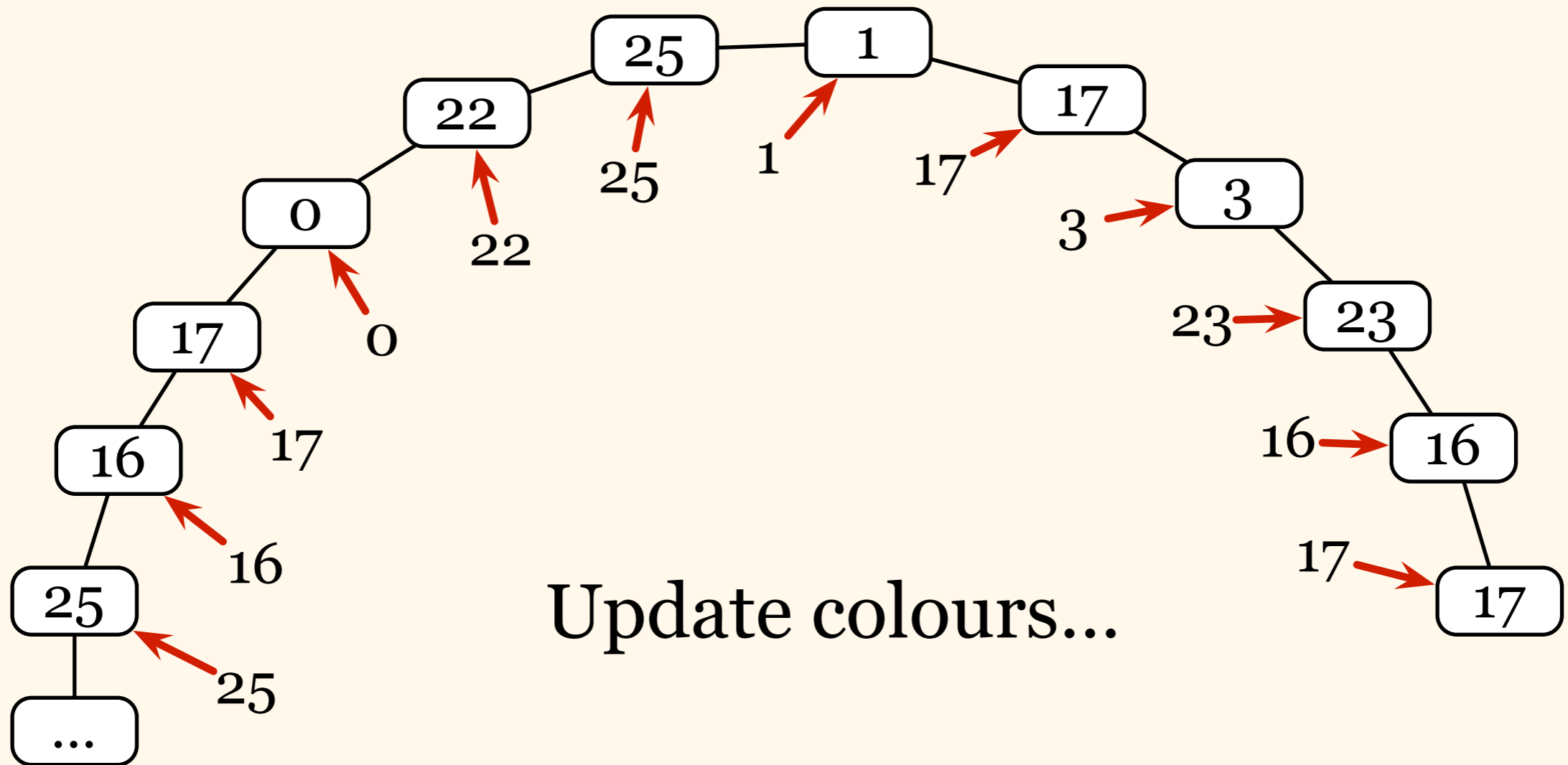
Almost Local Algorithms



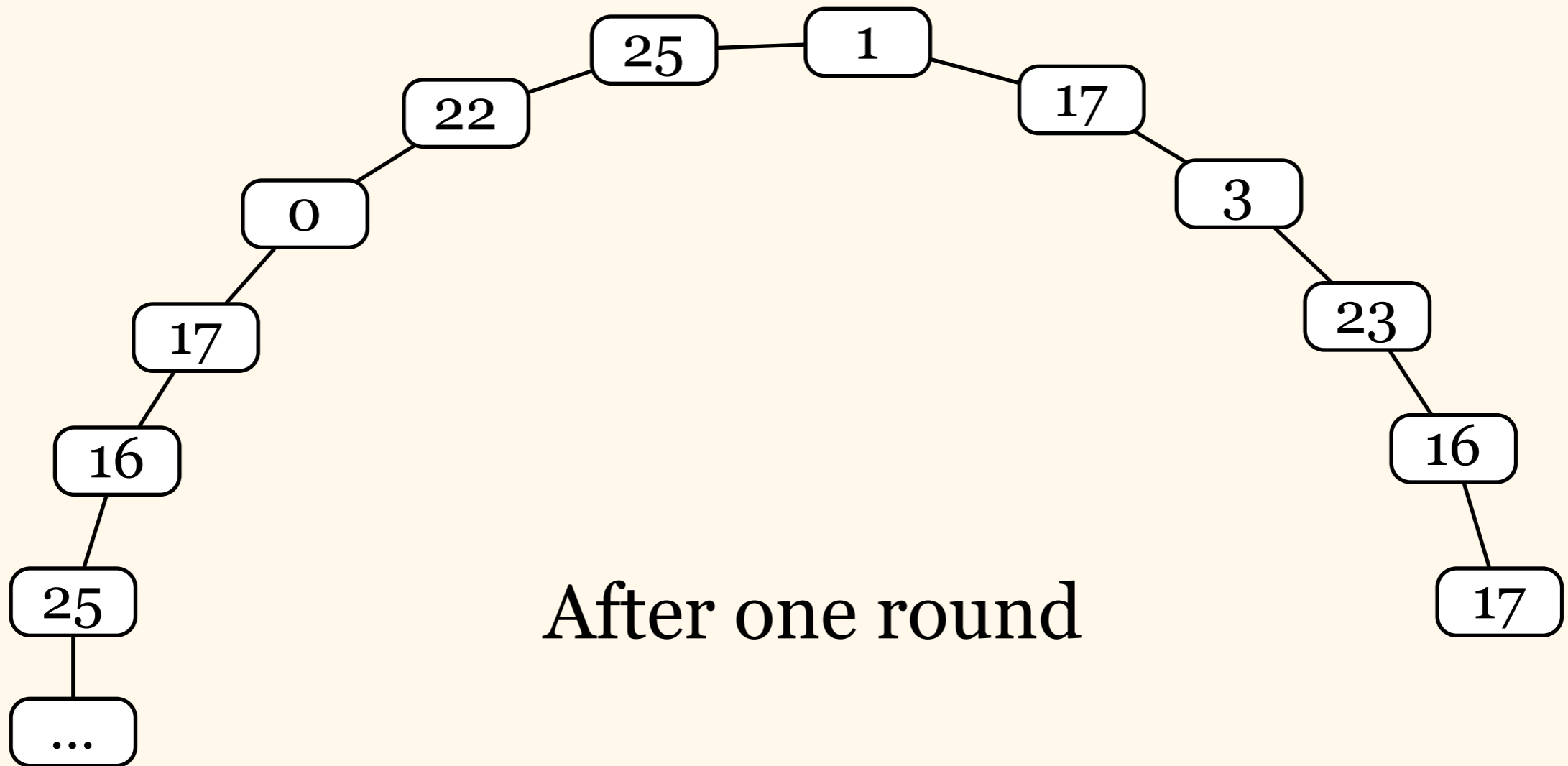
Almost Local Algorithms



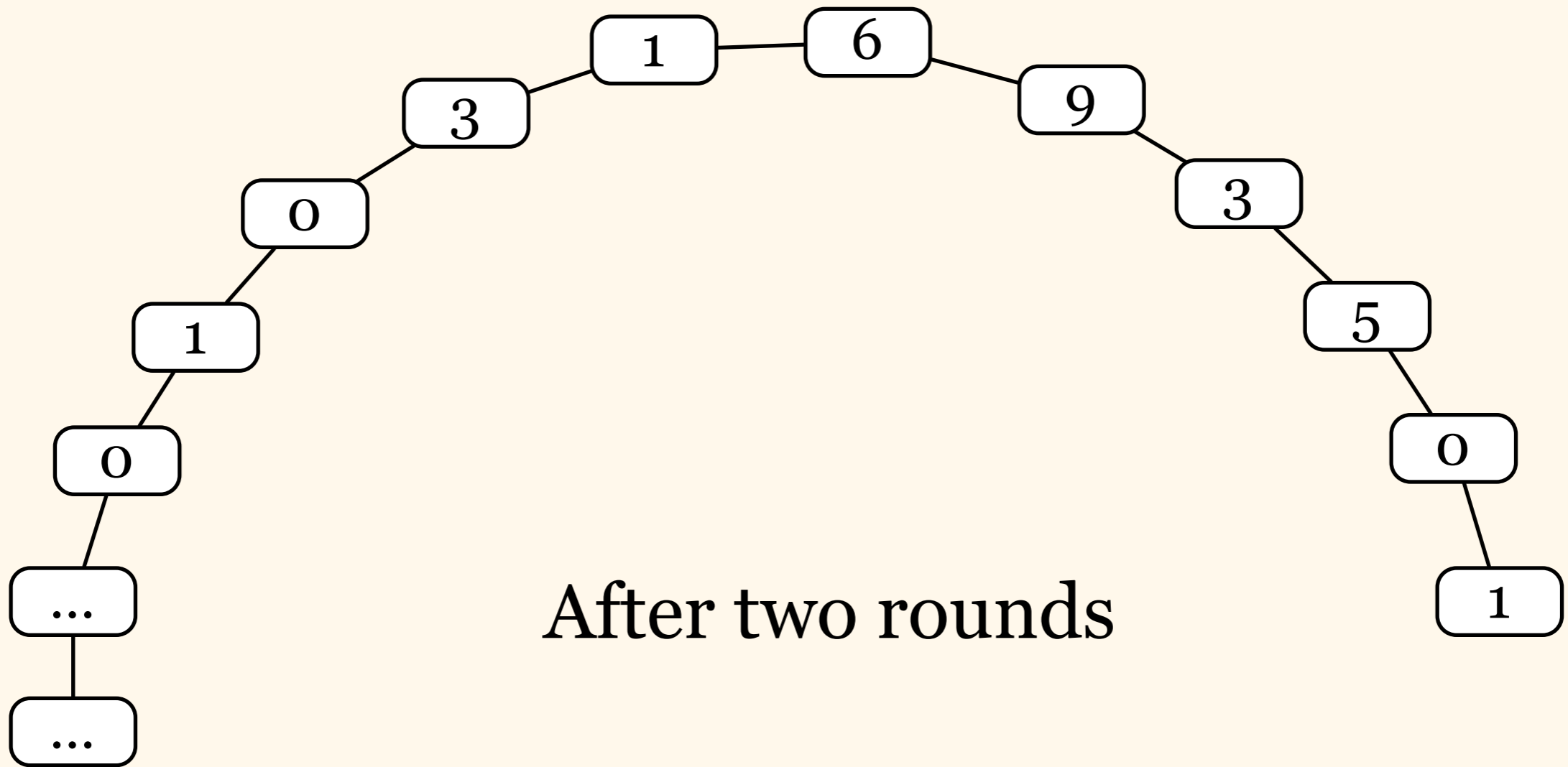
Almost Local Algorithms



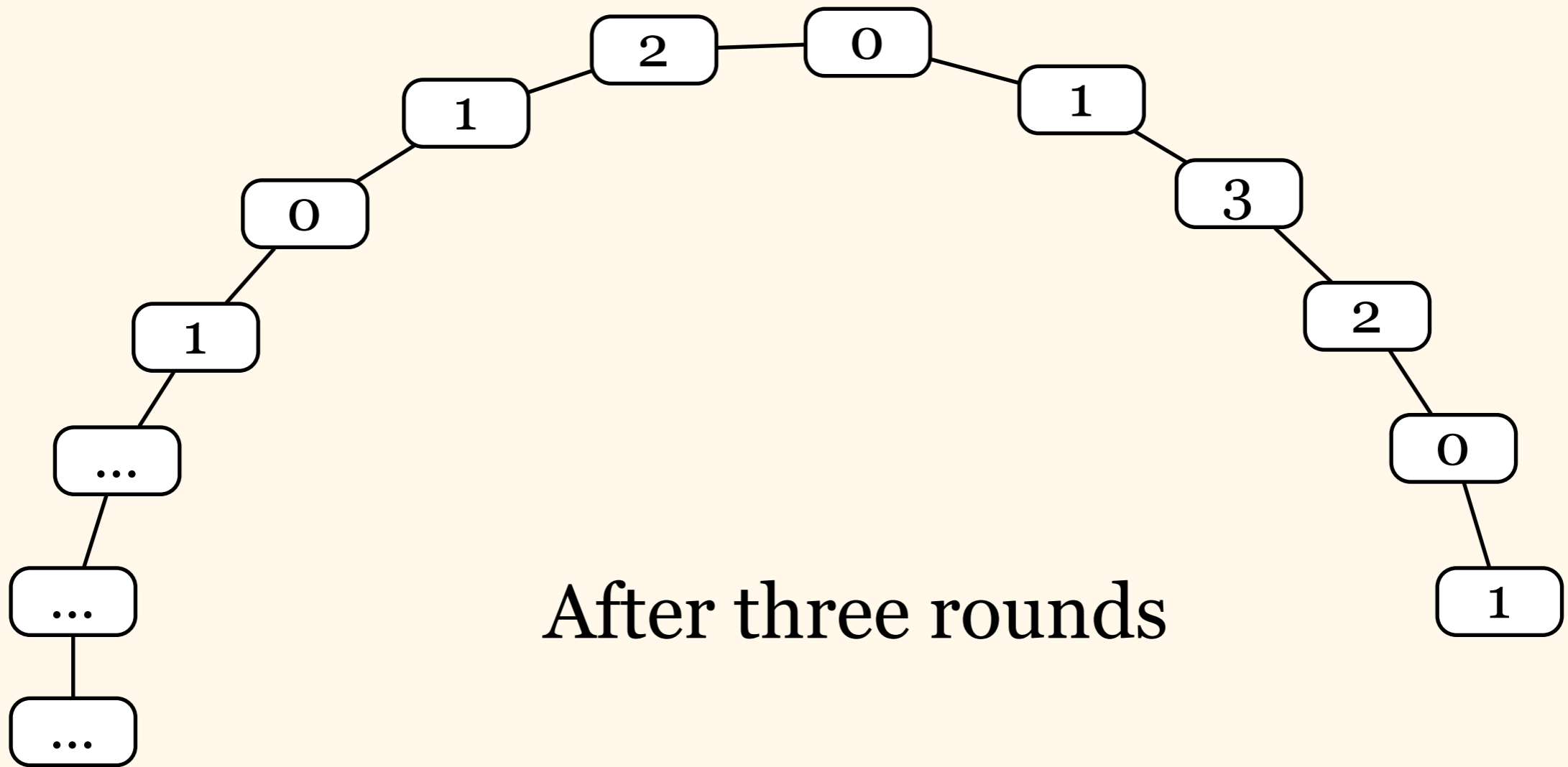
Almost Local Algorithms



Almost Local Algorithms



Almost Local Algorithms



Almost Local Algorithms

- Graph colouring in $O(\log^* n)$ rounds
 - Paths or cycles, 3-colouring
- Generalisations:
 - Trees, bounded-degree graphs, ...
 - Graphs of maximum degree Δ :
($\Delta+1$)-colouring in $O(\Delta + \log^* n)$ rounds

Almost Local Algorithms

- Graph colouring in $O(\log^* n)$ rounds
- Many applications:
 - Maximal independent set:
first try to add nodes of colour 0 (in parallel),
then try to add nodes of colour 1 (in parallel), ...
 - Maximal matching
 - Greedy algorithm for dominating sets

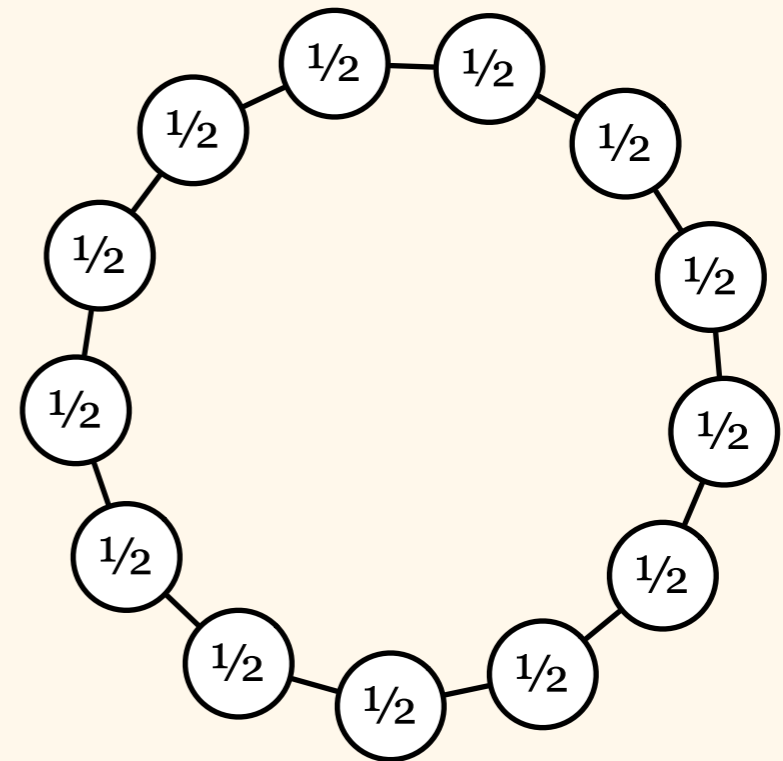
Almost Local Algorithms

- Graph colouring in $O(\log^* n)$ rounds
- Many applications
- Fast, but not strictly local
 - And inherently depends on the existence of small, unique, numerical identifiers

Past: Summary

- Bad news:
 - Cannot break symmetry in cycles
- Three traditional escapes:
 - Randomised algorithms
 - Geometric information
 - “Almost local” algorithms

Present

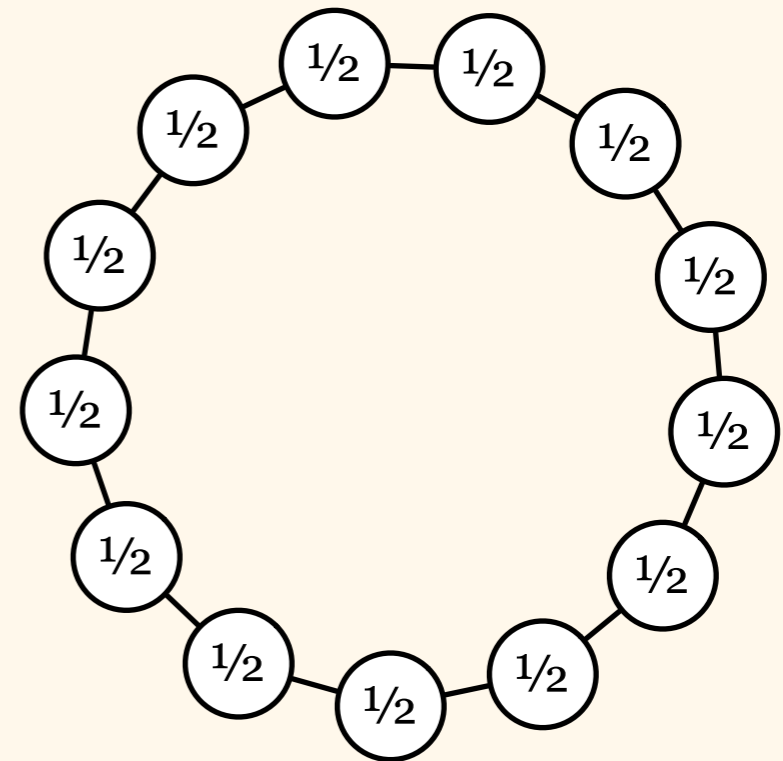


Dealing with Bad News

- You cannot break symmetry in cycles...
- Which problems *do not require* symmetry breaking in cycles?

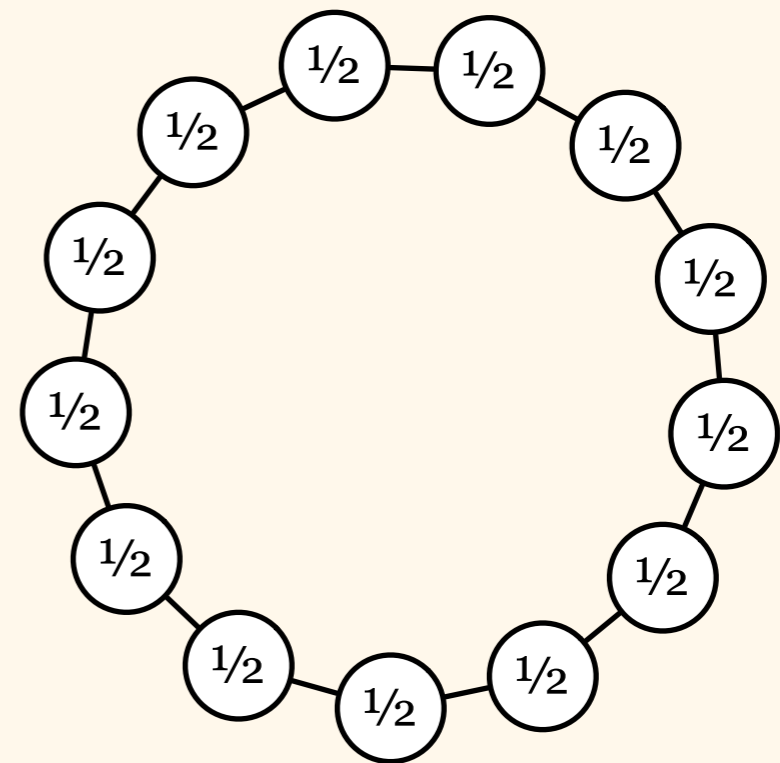
Tractable Problems

- Linear programs (LPs)
 - Many resource-allocation problems can be modelled as LPs
 - If the input is symmetric, a trivial solution is an optimal solution!
 - *Only non-symmetric inputs are challenging...*



Tractable Problems

- Linear programs (LPs)
 - Approximation scheme for packing and covering LPs
 - Local algorithm
 - *Kuhn, Moscibroda & Wattenhofer* (2006)

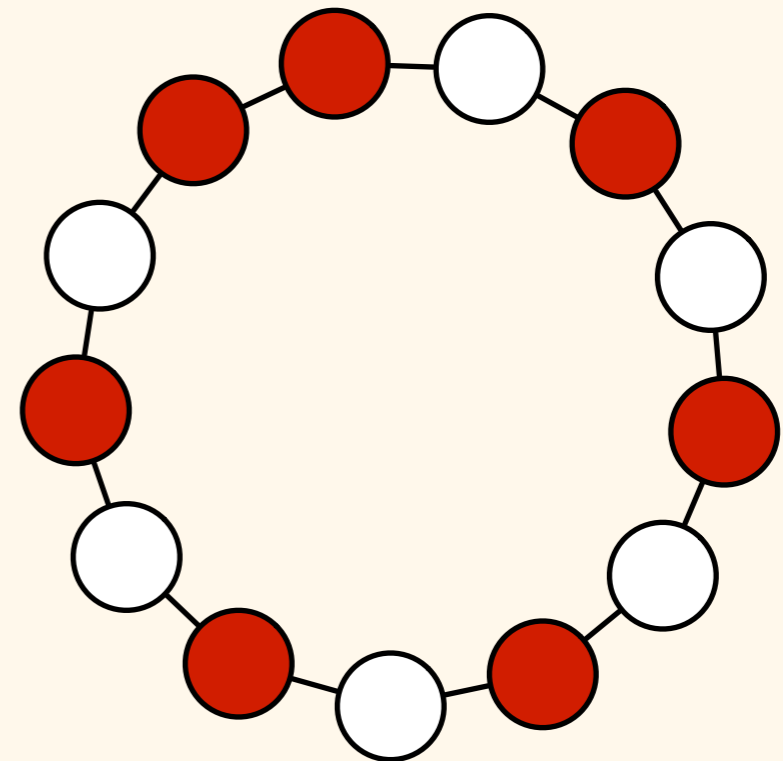


Tractable Problems

- Vertex covers
 - 2-approximation is the best that we can find with *centralised polynomial-time algorithms*
 - Nobody knows how to find 1.9999-approximation efficiently
 - Hence if we could find a 2-approximation with *local algorithms*, it would be amazing!

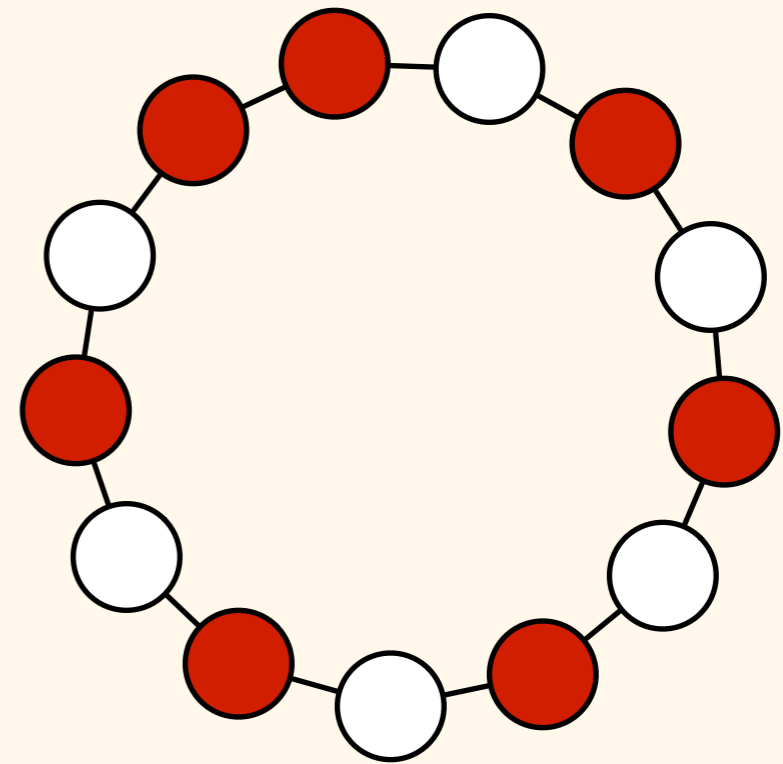
Tractable Problems

- Vertex covers
 - 2-approximation does not require symmetry breaking
 - In a regular graph, trivial solution (all nodes) is 2-approximation
 - Again, *only non-symmetric inputs are challenging...*



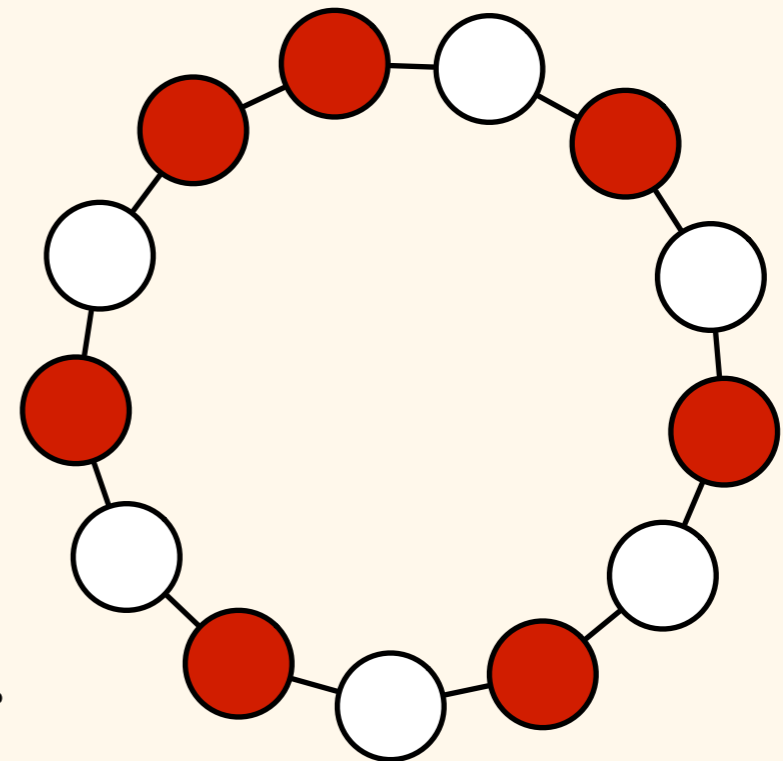
Tractable Problems

- Vertex covers
 - 2-approximation of vertex cover in bounded-degree graphs
 - Local algorithm
 - *Åstrand & Suomela* (2010)



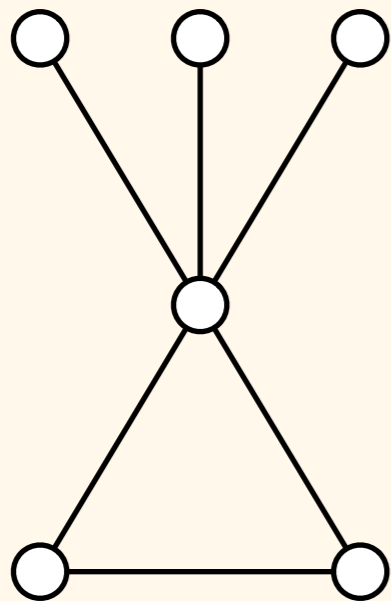
Tractable Problems

- Vertex covers
 - 2-approximation of vertex cover in bounded-degree graphs
 - Local algorithm
 - A bit complicated...
 - Let's have a look at a simpler local algorithm:
3-approximation of vertex cover



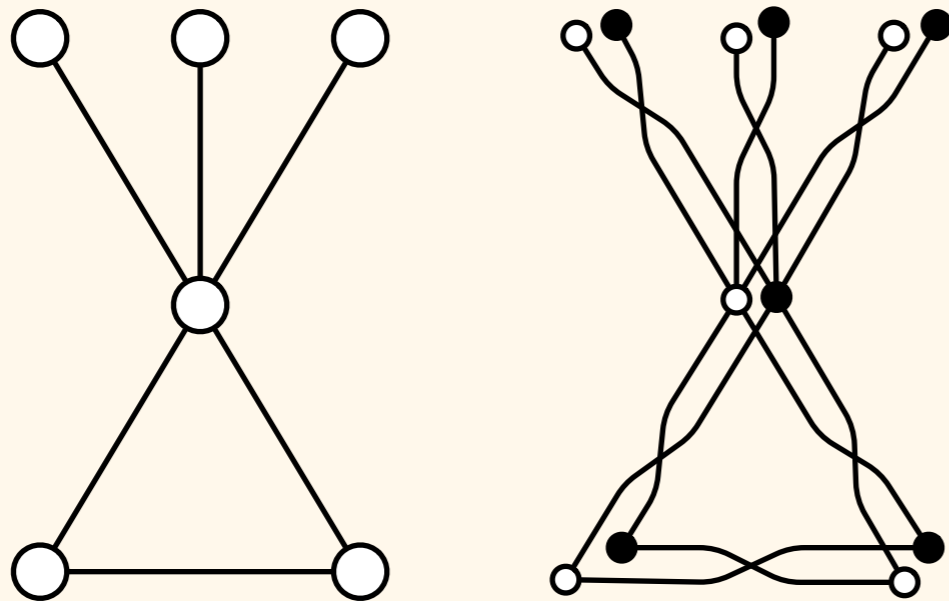
Vertex Cover

A simple local algorithm:
3-approximation of minimum vertex cover



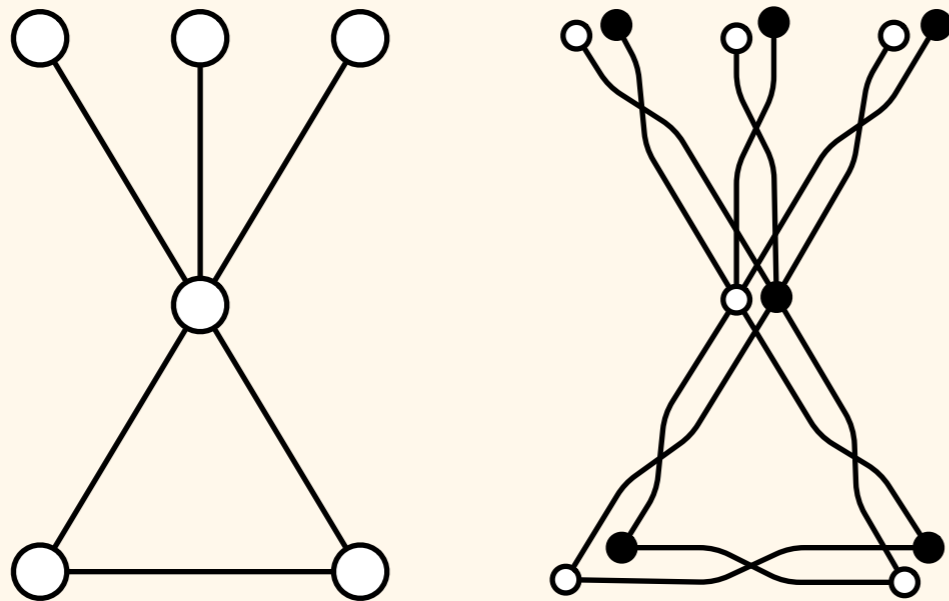
Vertex Cover

Construct a *virtual graph*:
two copies of each node; edges across



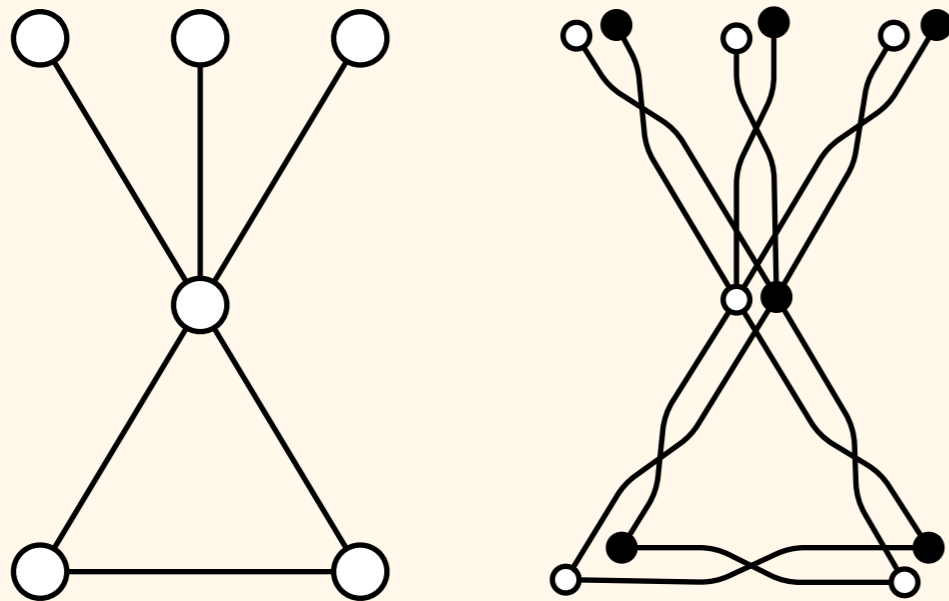
Vertex Cover

The virtual graph is *2-coloured*:
all edges are from white to black



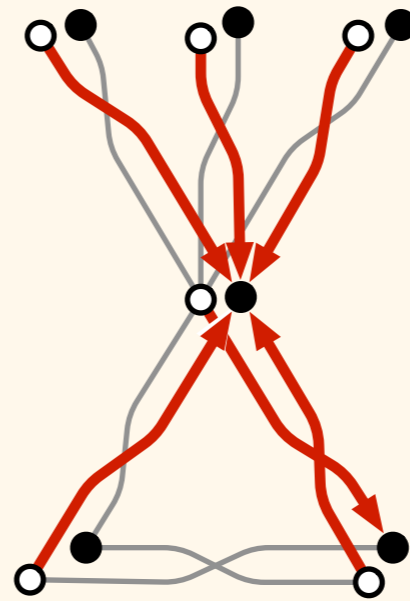
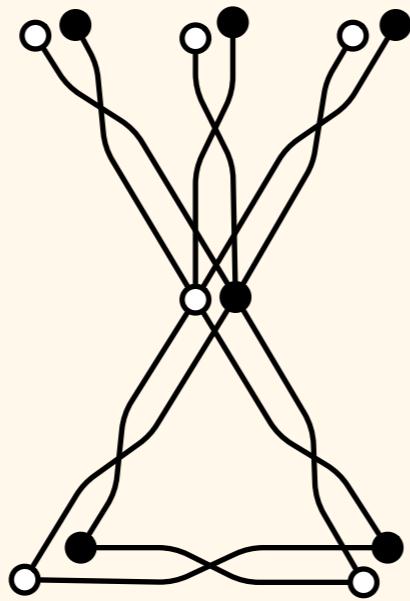
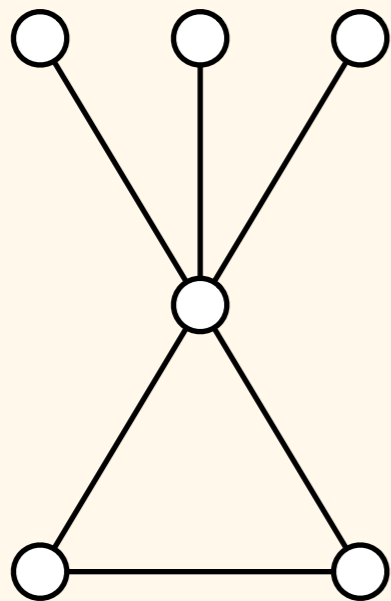
Vertex Cover

The virtual graph is 2-coloured –
therefore we can find a *maximal matching*!



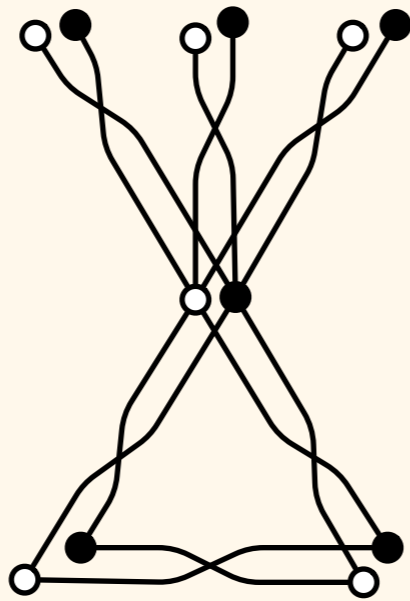
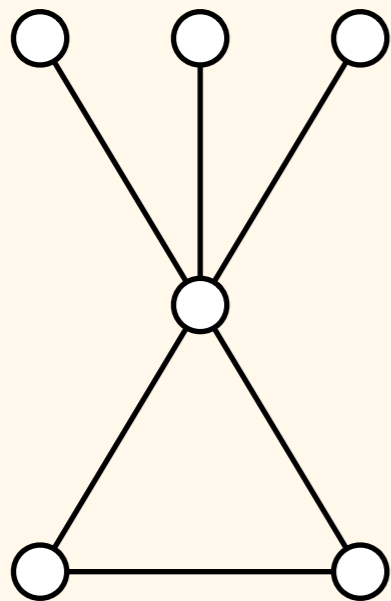
Vertex Cover

White nodes send *proposals* to their black neighbours



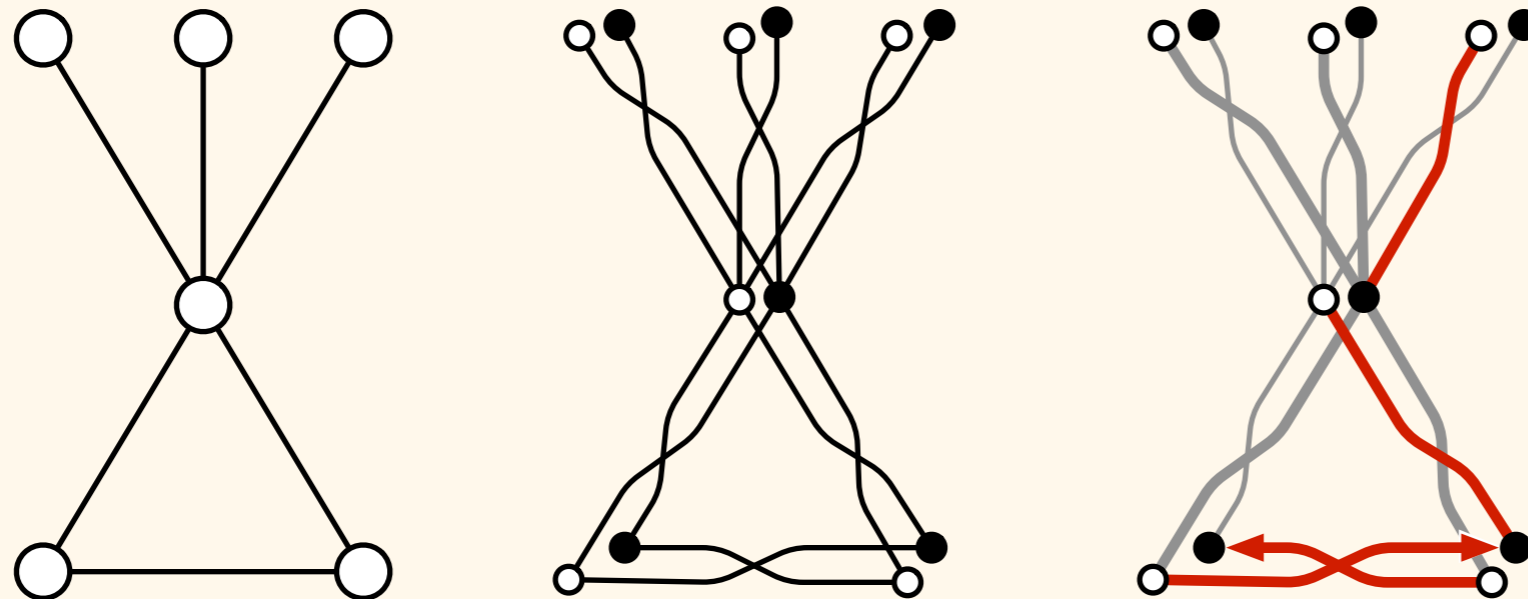
Vertex Cover

Black nodes *accept* one of the proposals



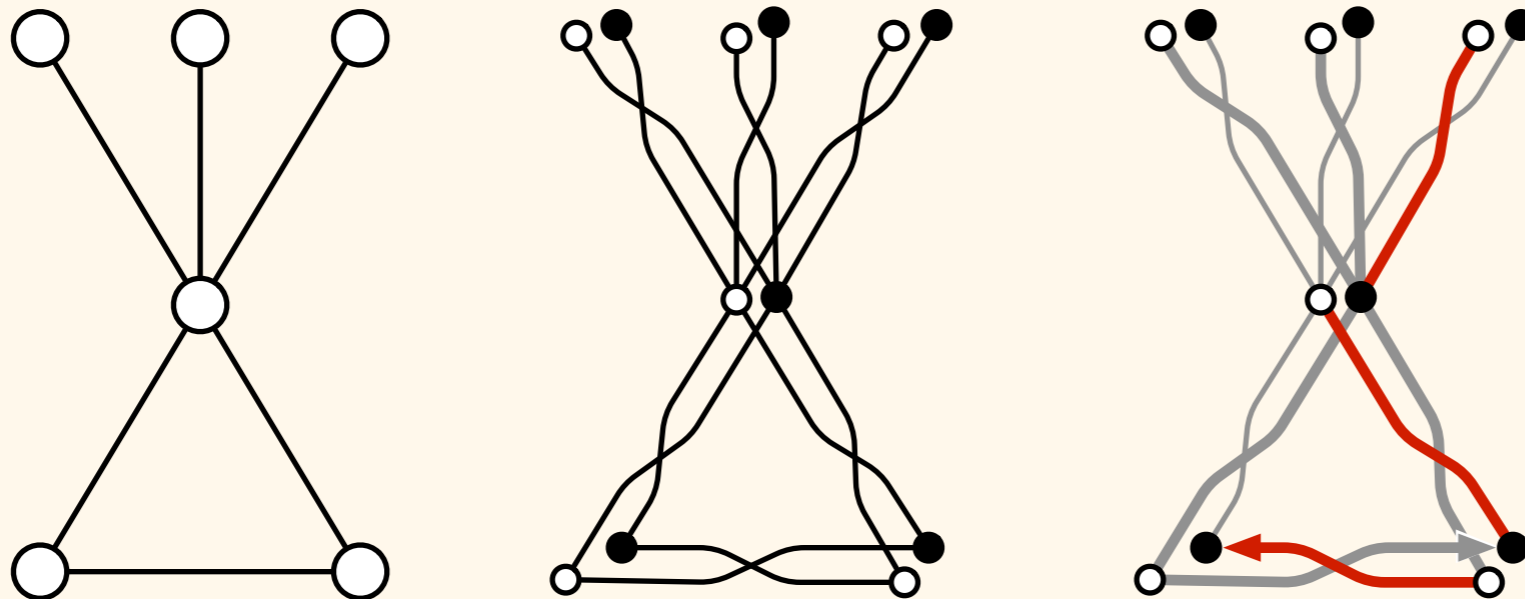
Vertex Cover

White nodes send *proposals* to another black neighbour if they were rejected



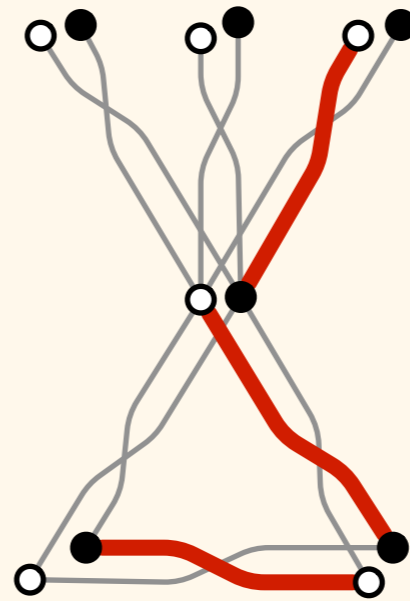
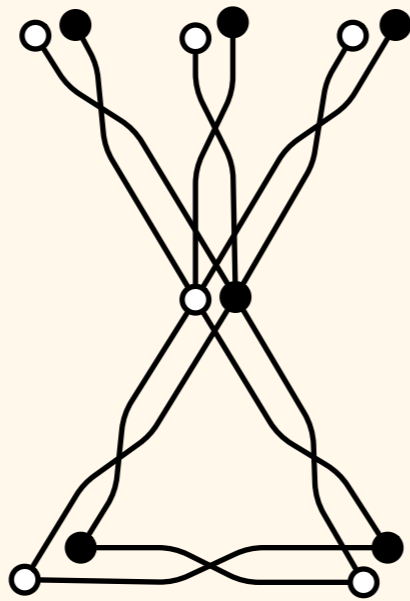
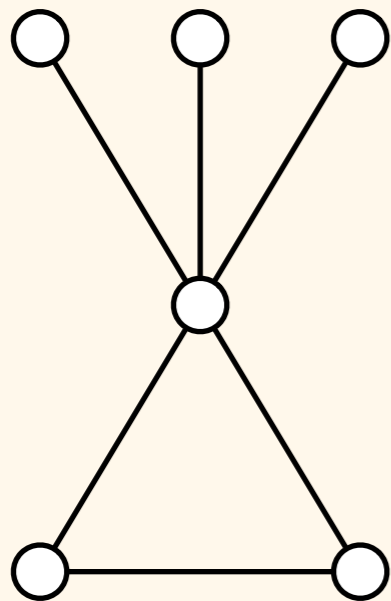
Vertex Cover

Again, black nodes *accept* one proposal – unless they were already matched



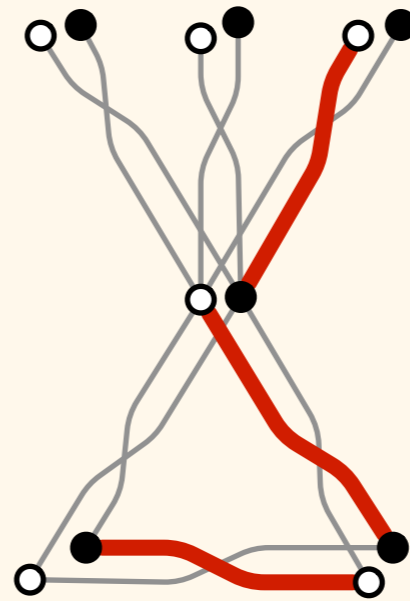
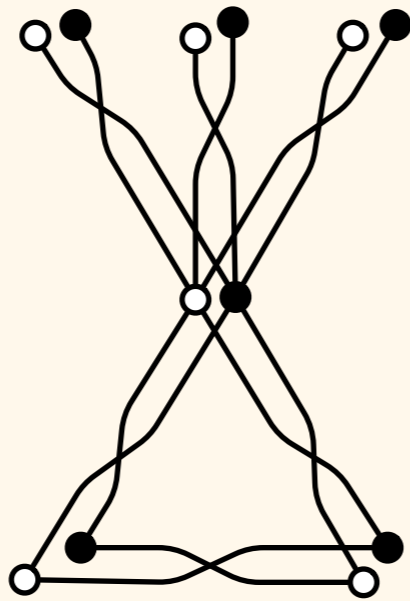
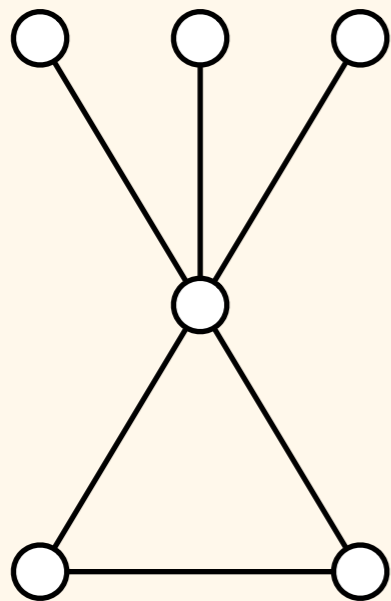
Vertex Cover

Continue until all white nodes are matched –
or they are rejected by all black neighbours



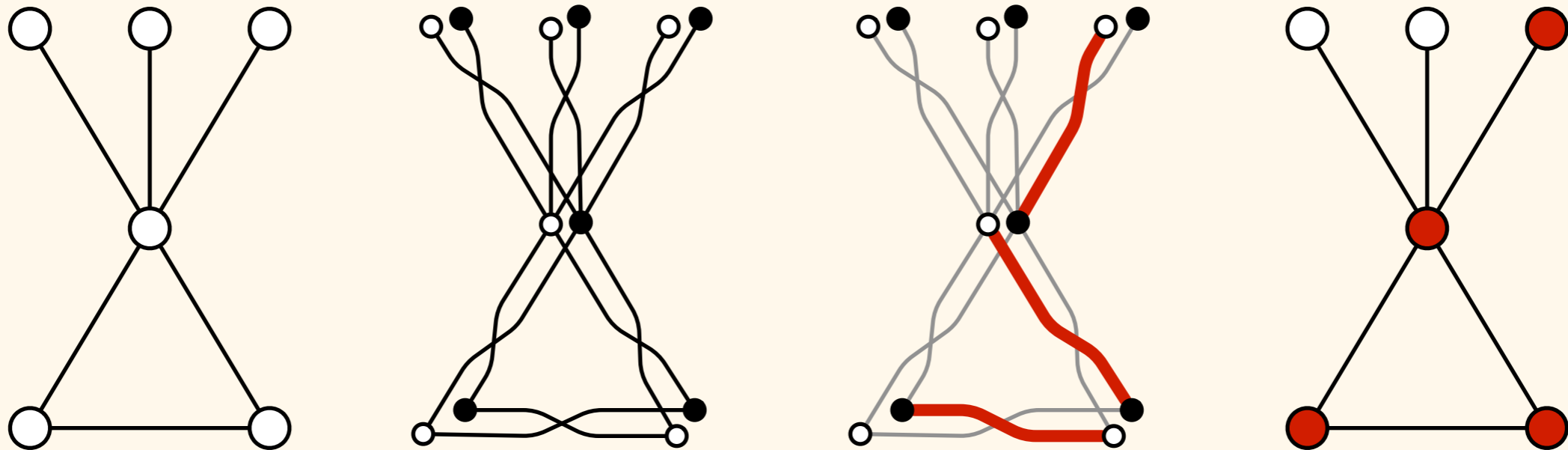
Vertex Cover

End result: a *maximal matching* in the virtual graph



Vertex Cover

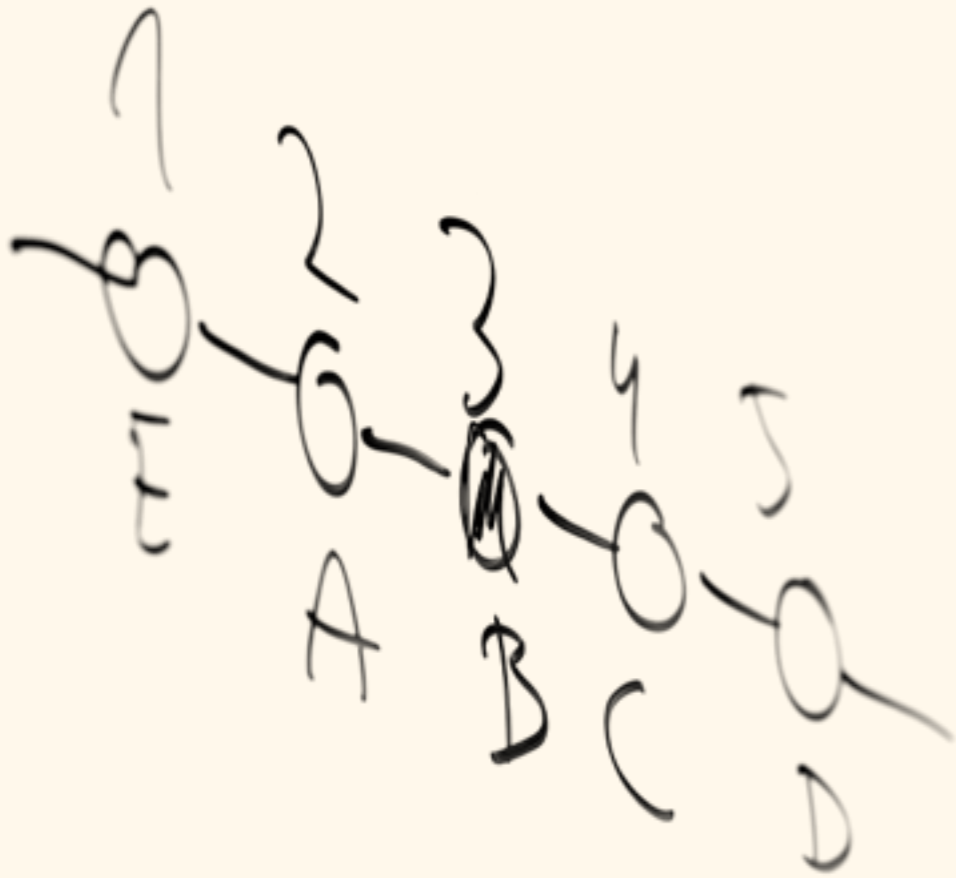
Take all original nodes that were matched –
3-approximation of minimum vertex cover!



Present: Summary

- You cannot break symmetry in cycles...
- But we can study problems that *do not require* symmetry breaking!
 - *Linear programs*: local approximation schemes
 - *Vertex covers*: local 2-approximation algorithm
 - *Edge dominating sets*: local approximation algorithm
 - ...

Future

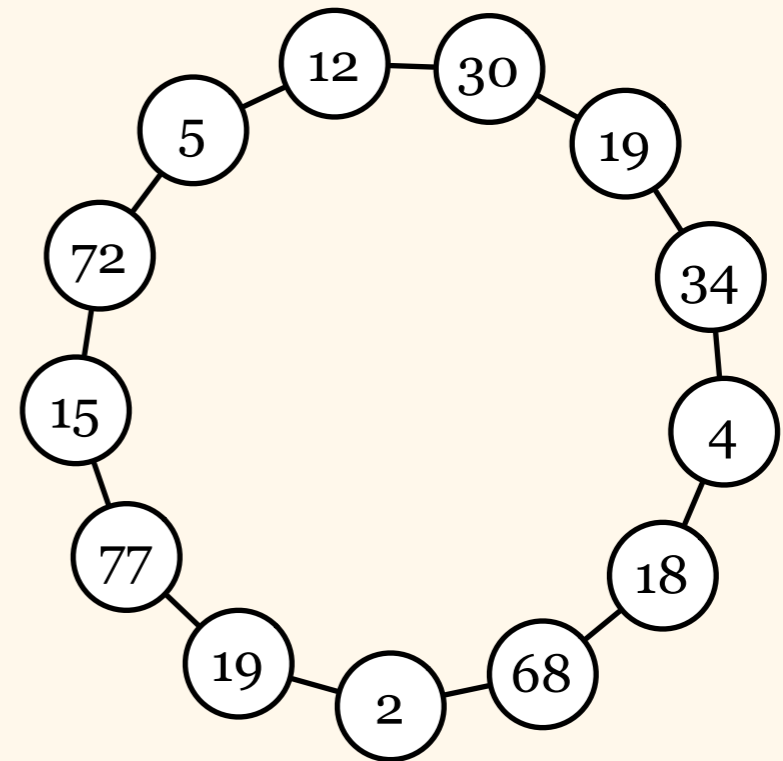


Dealing with Bad News

- Let's have a fresh look at the lower bounds!
 - Exactly what was proved?

Lower Bounds

- Only trivial solutions in cycles
- *Assumption:*
constant-size output
 - Each node outputs constant number of bits
- Innocuous?



Output Size

- Vertex cover, independent set, dominating set, cut: *1 bit per node*
- Matching, edge dominating set, edge cover: *1 bit per edge*
 - In a cycle, this is $O(1)$ bits per node

Output Size

- Graph colouring:
 - $O(1)$ colours should be enough in a cycle
 - Hence *$O(1)$ bits per node* is enough to encode the solution
- Linear programs:
 - For a near-optimal solution, we can use *finite-precision rational numbers*

Output Size

- Natural problems seem to have constant-size output
- Hence the negative results apply
 - Unique identifiers do not help in cycles
 - We can only produce trivial solutions in cycles
 - We can only solve problems that do not require symmetry-breaking

Output Size

- Natural problems seem to have constant-size output
- Hence the negative results apply
- *Did we miss anything?*

Scheduling Problems

- Local approximation algorithms
 - Scheduling problems:
fractional graph colouring,
fractional domatic partition, ...
 - First example of a local algorithm that
actually requires unique numerical identifiers
 - *Hasemann, Hirvonen, Rybicki & Suomela*
(work in progress)

More New Directions

- Deterministic local algorithm
 - cf. deterministic Turing machine – class P
- Randomised local algorithm
 - cf. probabilistic Turing machine – class BPP, etc.
- *Nondeterministic* local algorithm
 - cf. nondeterministic Turing machine – class NP

Decision Problems

- Back to very basics: *decision problems*
 - Is this graph bipartite? Acyclic? Hamiltonian? Eulerian? Connected? 3-colourable? Symmetric?
 - Decision problems form the foundation of classical complexity theory...

Decision Problems

- Decision problems in distributed setting:
 - *yes*-instance: all nodes happy
 - *no*-instance: at least one node raises alarm
- Few decision problems can be solved with deterministic local algorithms
 - But now we have a very natural extension...

Decision Problems

- *Nondeterministic* local algorithms
 - Yes-instances have a compact certificate that can be verified with a local algorithm
 - “*locally checkable proof*”
- Cf. class NP:
 - Yes-instances have a compact certificate that can be verified in P

Locally Checkable Proofs

- Key question: what is the size of the proof?
 - *How many bits per node are needed?*
 - For example, it is easy to show that a graph is bipartite: just give a 2-colouring, 1 bit per node
 - How do you prove that a graph is *not* bipartite?

Locally Checkable Proofs

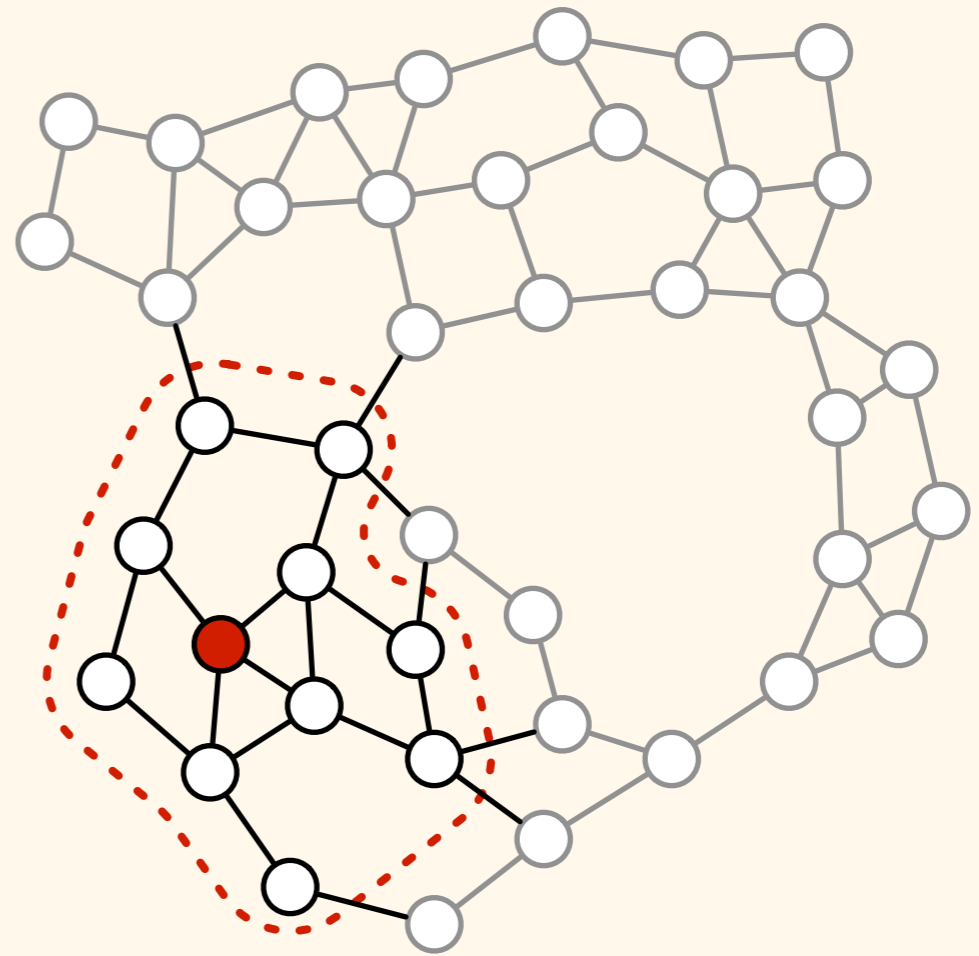
- Key question: what is the size of the proof?
 - *How many bits per node are needed?*
 - For example, it is easy to show that a graph is bipartite: just give a 2-colouring, 1 bit per node
 - How do you prove that a graph is *not* bipartite?
 - Find an odd cycle, and prove that it exists
 - $O(\log n)$ bits is enough, $\Omega(\log n)$ bits necessary

Locally Checkable Proofs

- Natural hierarchy of proof complexities:
 - 2-colourable graphs: $\Theta(1)$ bits per node
 - Non-2-colourable graphs: $\Theta(\log n)$ bits per node
 - Non-3-colourable graphs: $\text{poly}(n)$ bits per node
 - *Göös & Suomela* (2011)

Summary

- Local algorithms
- Strong lower bounds
 - Nevertheless,
a lot of progress!
- Latest hot topics
 - Scheduling problems
 - Nondeterministic models



www.hiit.fi/jukka.suomela/

