Computing in Distributed Systems
Chapter 3: The Colouring Problem

Akka Zemmari

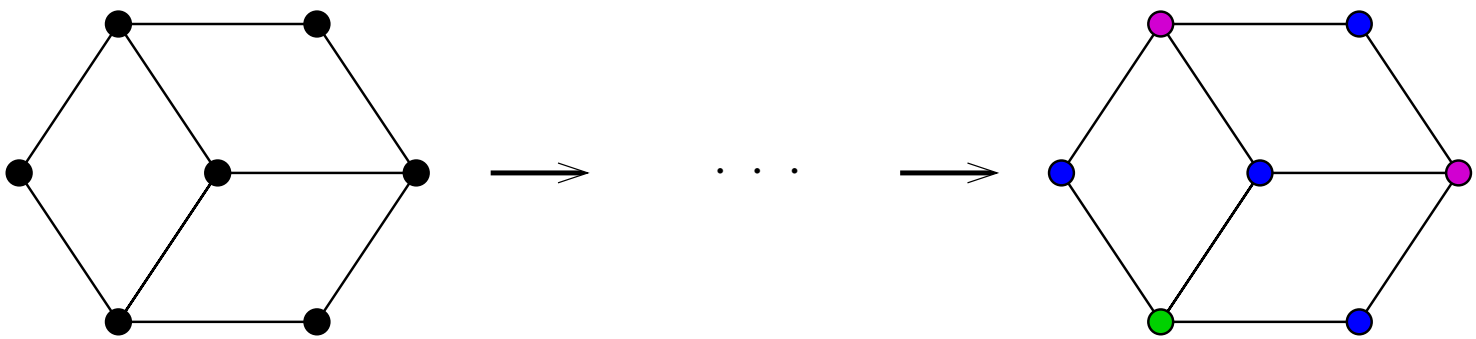
LaBRI - Université de Bordeaux

Content

The Colouring Problem

Definition.

Let $G = (V, E)$ be a simple connected undirected graph. A proper vertex colouring for G is an assignment of a colour $c(v)$ to each vertex v, such that any two adjacent vertices have a different colour, i.e., $c(v) \neq c(u)$ for every $\{u, v\} \in E$.

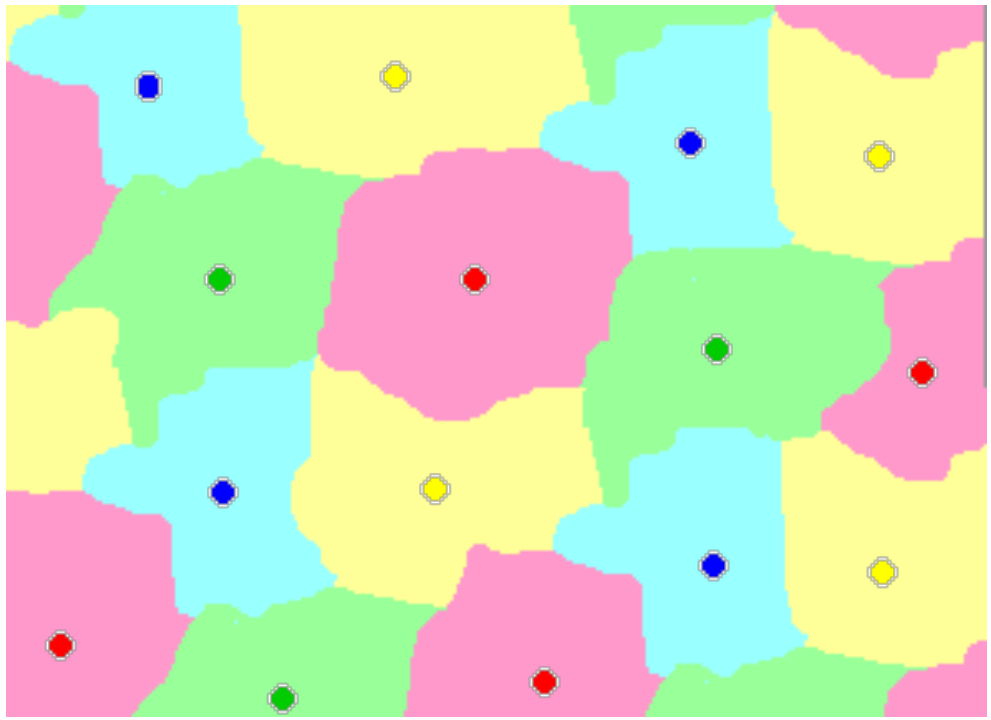


Remark

Distributed system $\rightarrow$ Nodes of the system have to colour themselves.

Why Color a Network

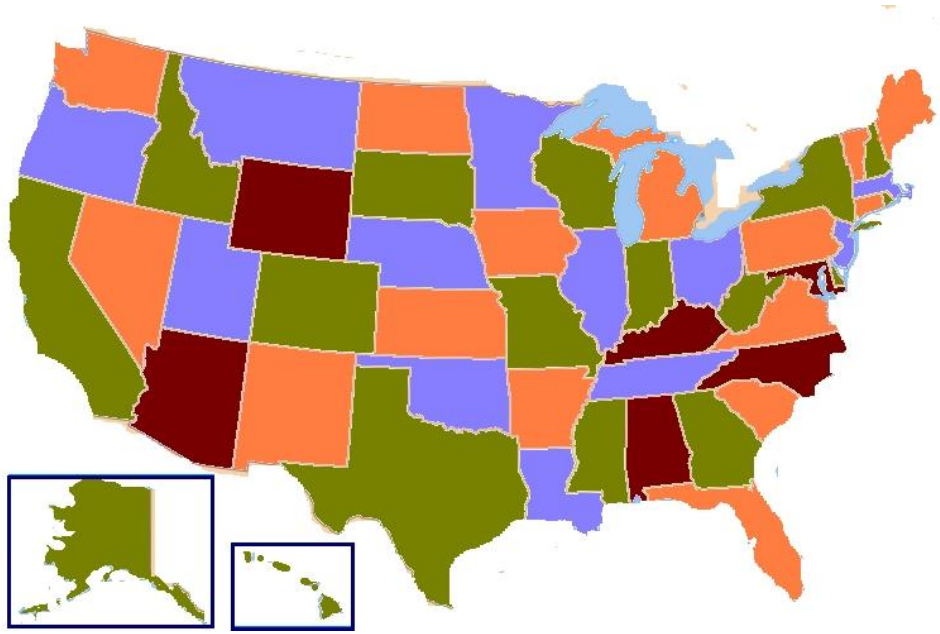
Medium access: reuse frequencies in wireless networks at certain spatial distance such that there is no interference.



Why Color a Network

4-Color Theorem

Can color each map using 4 colours only: no two adjacent countries have same color. ?

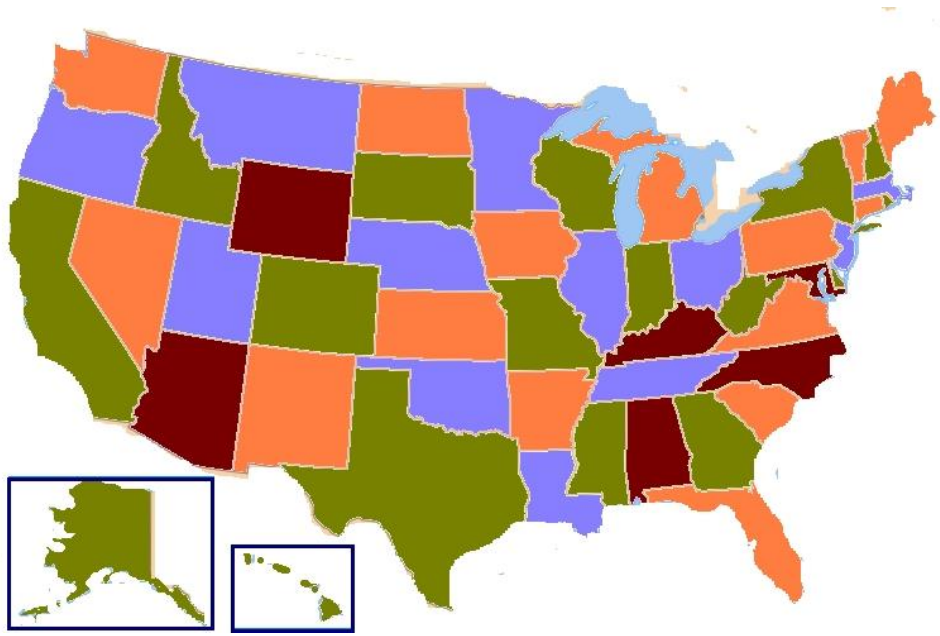


Why Color a Network

4-Color Theorem

Can color each map using 4 colours only: no two adjacent countries have same color.

$\Rightarrow$ First conjecture 1852, first proof with 5 colours 1890. First computer proof 1976 (Appel+Haken). Another proof using Coq assistant 2005.



A Simple (Sequential) Colouring Algorithm

Algorithme 1.

1. While $\exists$ a non-coloured vertex v do
2. color v using the smallest available colours (the smallest colour which is not used by any of the neighbours)
3. End While

Lemma.

The algorithm is correct, terminates in at most n rounds and uses at most $\Delta + 1$ colours (Δ is the maximum degree of the corresponding graph).

Solving the colouring problem in a distributed system ?

- Which assumptions we make ? $\rightarrow$ can we do it deterministically ? shall we use some randomisation ? ...
- Design a (randomised) distributed algorithm (an algorithm that nodes of the system will execute)
- Analyse the designed algorithm:
  - Time complexity: number of rounds ?
  - Message complexity ?
  - Size of the messages ?
  - Number of colours ?
  - ...

A First Distributed Algorithm (Wattenhofer)

We consider the following procedure:

FirstFree

Require: Nodes have different IDs

Colour v using the smallest available colour.

A First Distributed Algorithm (Wattenhofer)

Now we consider the following algorithm:

Reduce

Require: Nodes have different IDs

1. v sends its ID to its neighbours;
2. v receives the IDs of its neighbours
3. While v has an uncoloured neighbour with higher ID

3.1. v sends "undecided" to its neighbours;

4. v executes FirstFree;
5. v sends its decision to its neighbours.

A First Distributed Algorithm (Wattenhofer)

Analysis of the algorithm:

Theorem

The algorithm is correct and terminates in n steps. It uses $\Delta + 1$ colours.

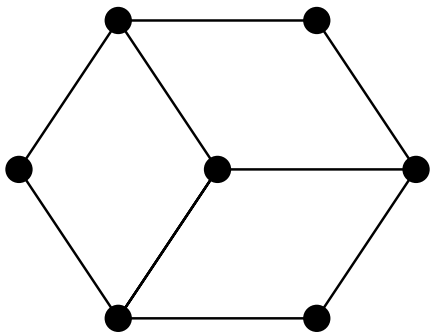
Observations

- The network is not anonymous (we use the node's IDs)
- Uses messages of size $O(\log n)$ (assuming nodes have IDs ranging from 1 to a polynomial on n)
- Its time complexity is $O(n)$.
+ Distributed algorithm.
+ Uses $\Delta + 1$ colours.

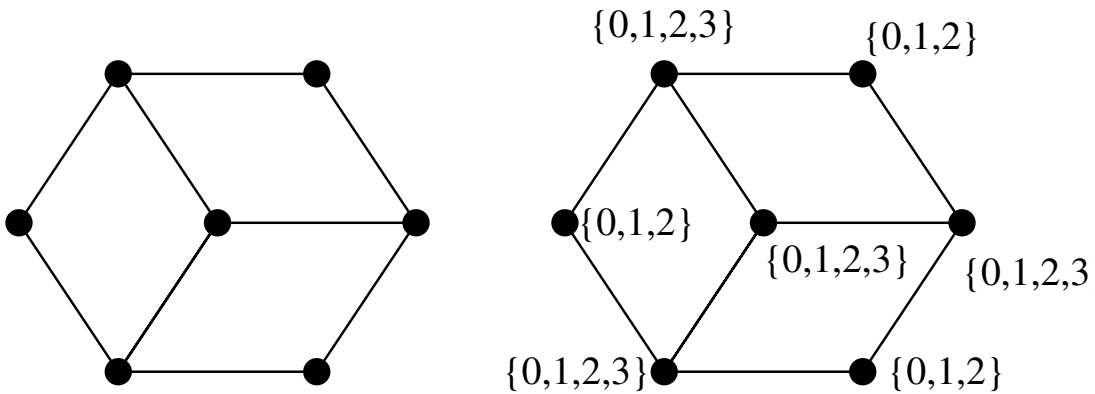
Colouring Anonymous Networks

- Assumption: The network is anonymous and nodes can send messages of size $O(\log n)$ bits
- **Theorem.** [Pel00] *There is no deterministic distributed algorithm for anonymous graphs for solving the colouring problem assuming all vertices wake up simultaneously.*
  $\Rightarrow$ use randomisation.

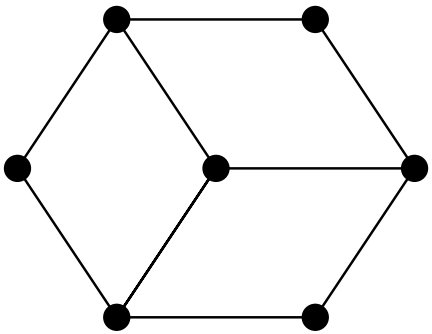
Johansson's Algorithm

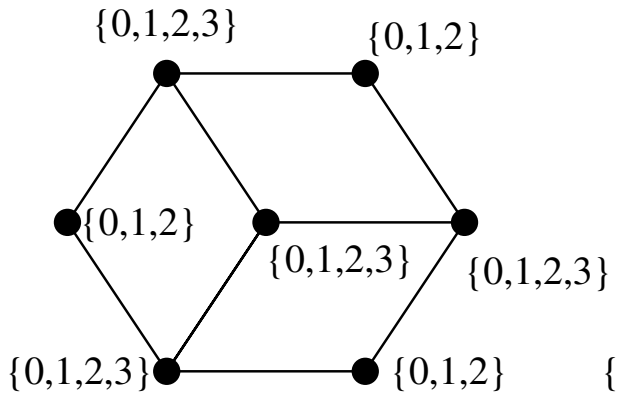


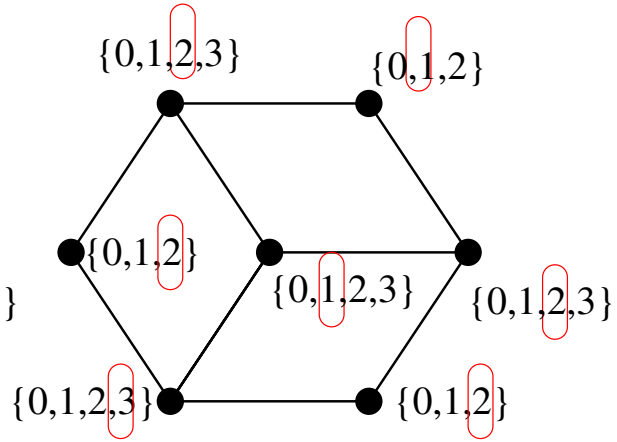
Johansson's Algorithm



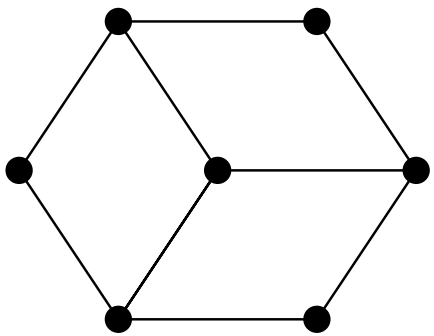
Johansson's Algorithm

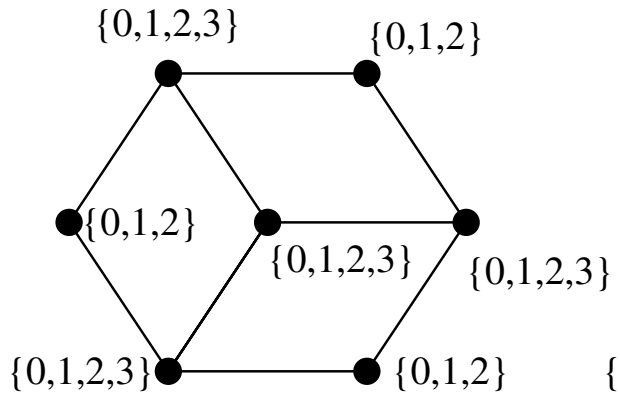


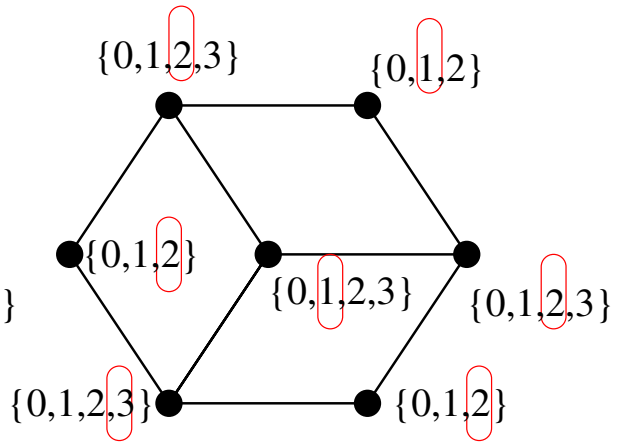


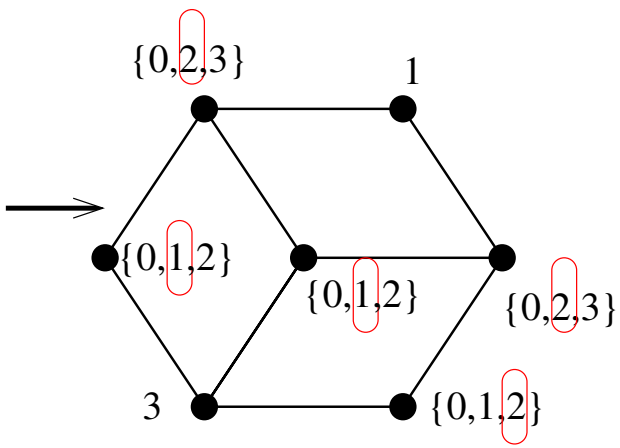


Johansson's Algorithm

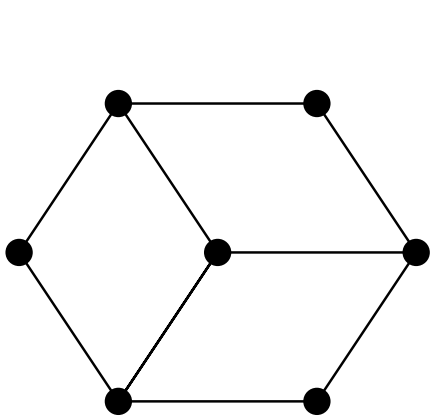


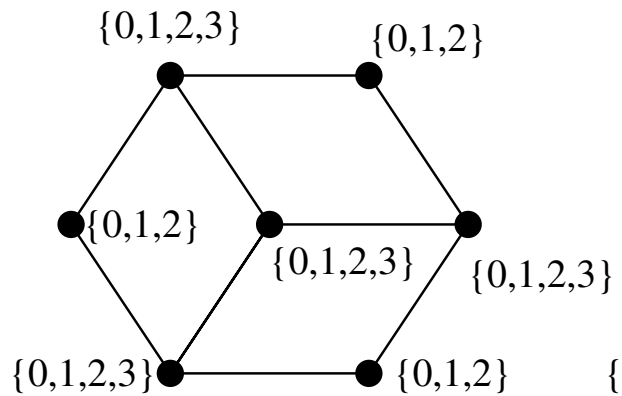


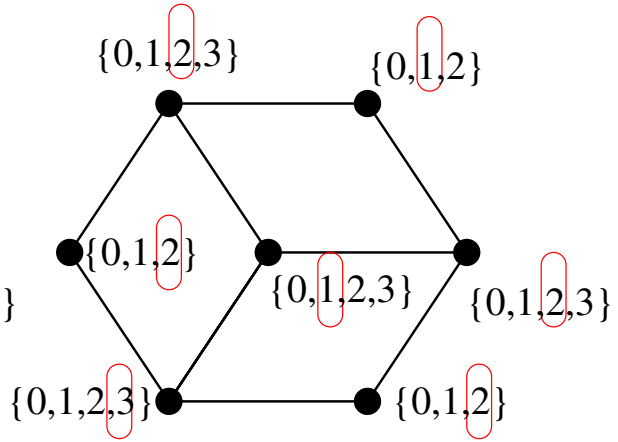


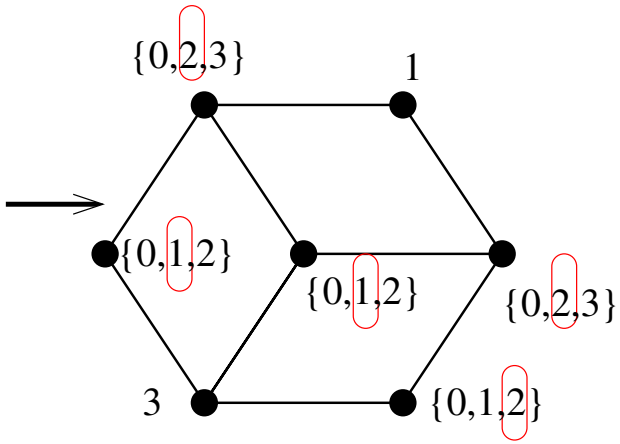


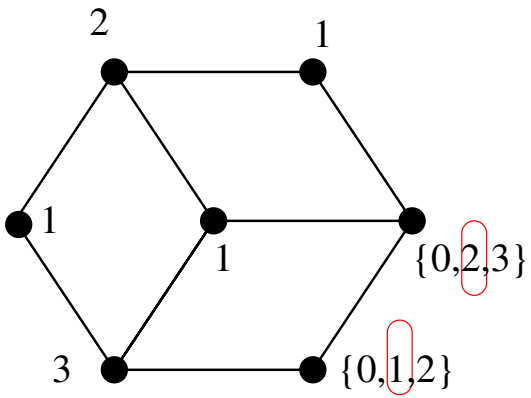
Johansson's Algorithm



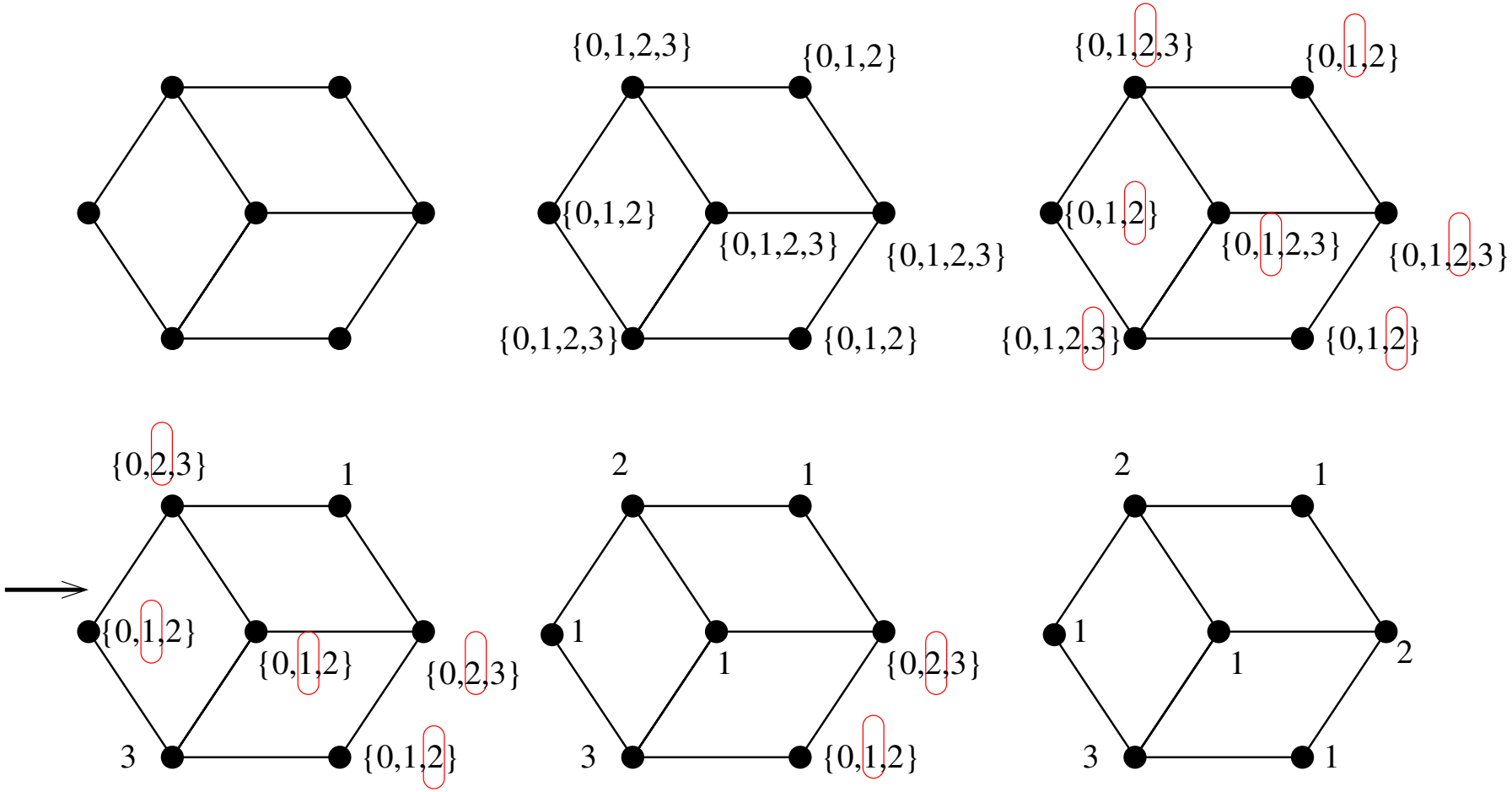




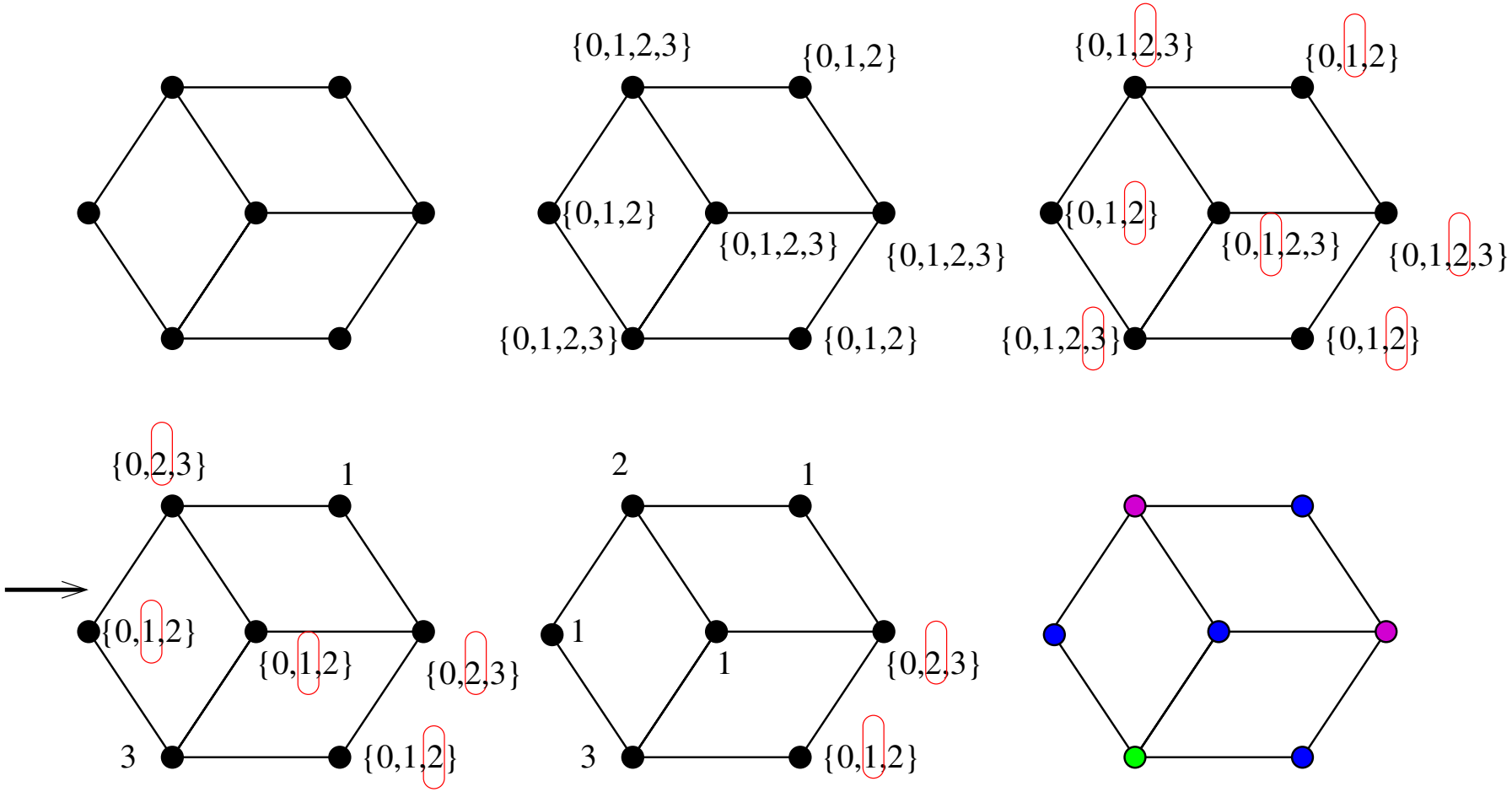




Johansson's Algorithm



Johansson's Algorithm



Johansson's Algorithm

Theorem.
Johansson's Algorithm runs in $O(\log n)$ rounds on average and w.h.p.

Remark.
Messages are of size $O(\log n)$.

Johansson's Algorithm

Theorem.
Johansson's Algorithm runs in $O(\log n)$ rounds on average and w.h.p.

Remark.
Messages are of size $O(\log n)$.

Question: Can we do the same using messages of size 1-bit ?

YES !

Question: Can we do the same using messages of size 1-bit ?

YES !

Algorithm *Fast_Colour*

var:

$colour_v$: *word* **Init** *empty-word*;

$actives_v$: $\subseteq N(v)$ **Init** $N(v)$;

b_v: $\in \{0,1\}\}$;

While $actives_v \neq \emptyset$ **do**

$b_v \leftarrow flip(0,1)$;

$colour_v \leftarrow b_v \oplus colour_v$;

For all $u \in actifs_v$ **do**

send b_v to u;

receive b_u from u;

If $b_v \neq b_u$**then**

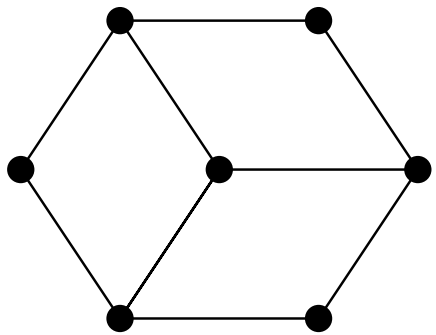
$active_v \leftarrow active_v \setminus \{u\}$;

End If

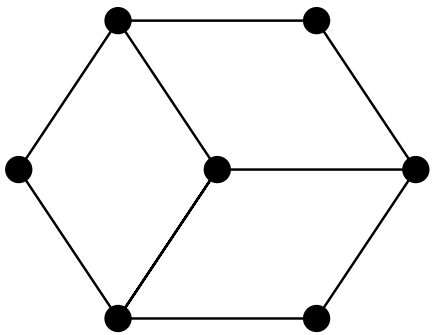
End For

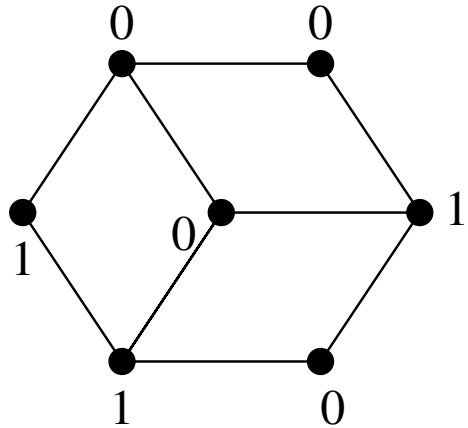
End While

Algorithm *Fast_Colour*

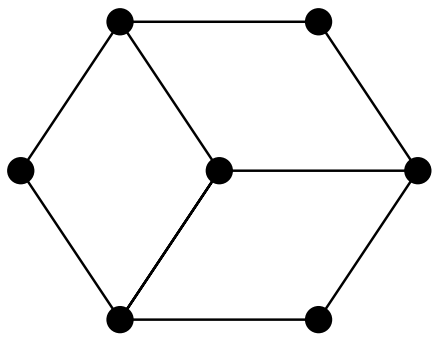


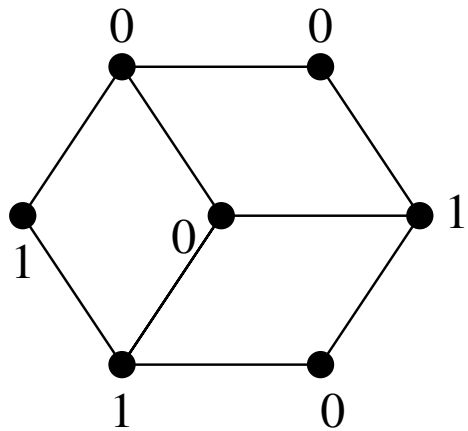
Algorithm *Fast_Colour*

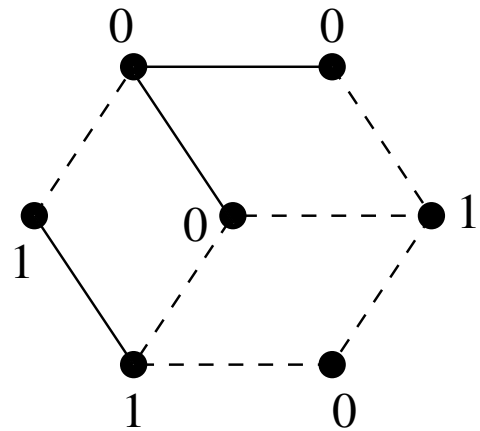




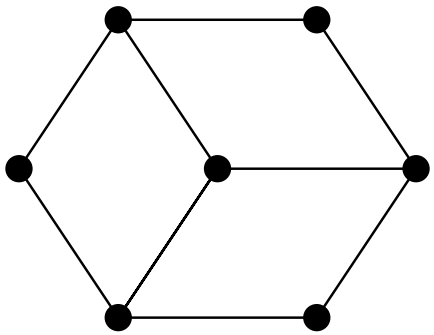
Algorithm *Fast_Colour*

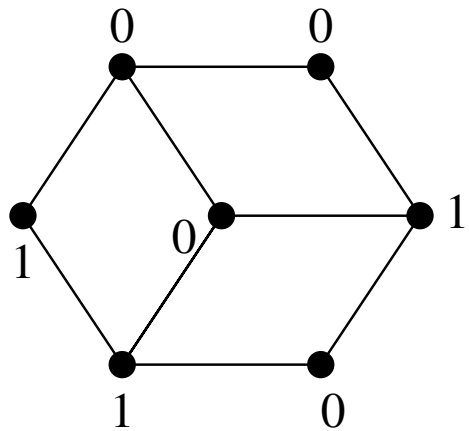


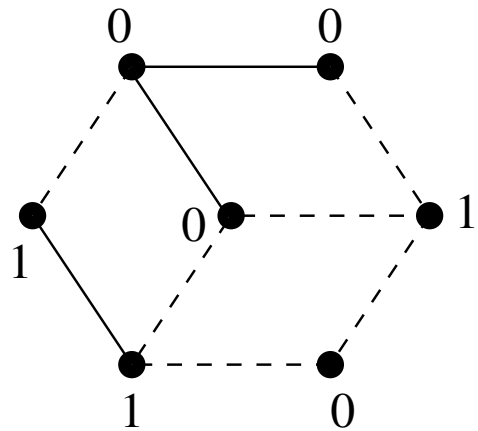


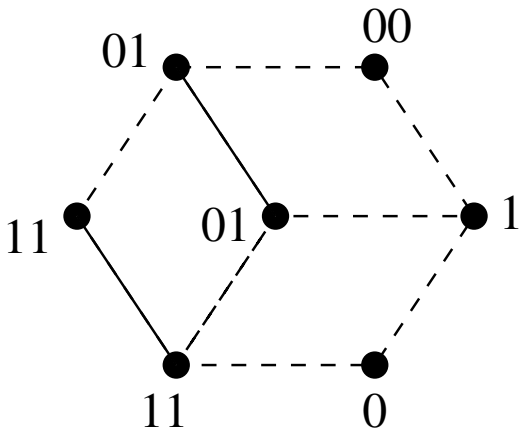


Algorithm *Fast_Colour*

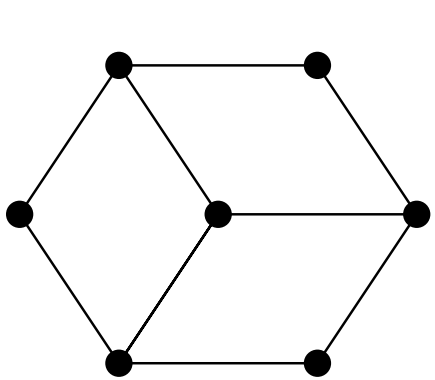


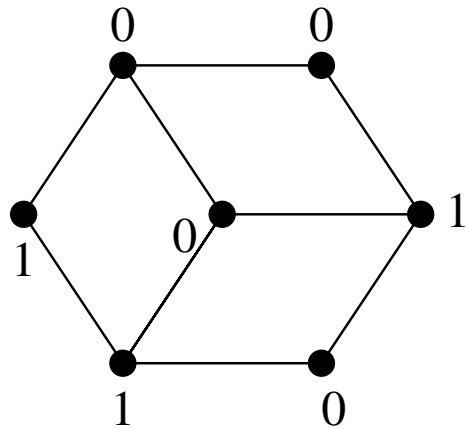


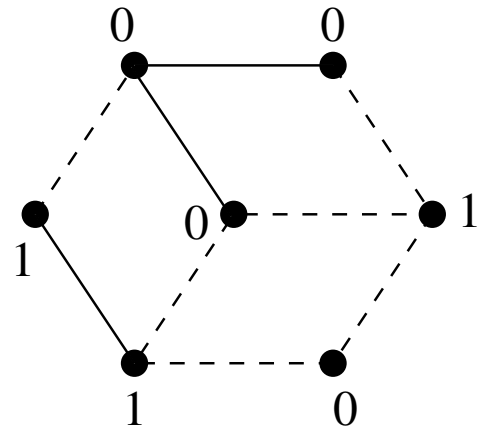


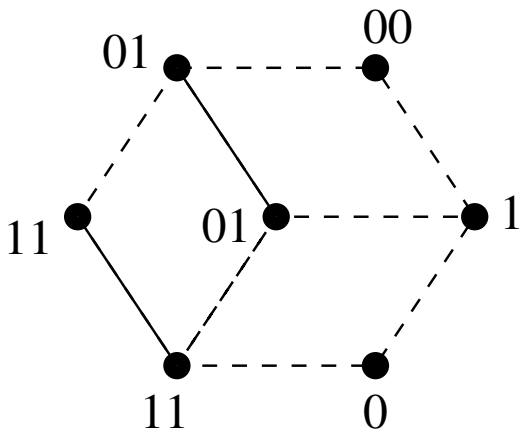


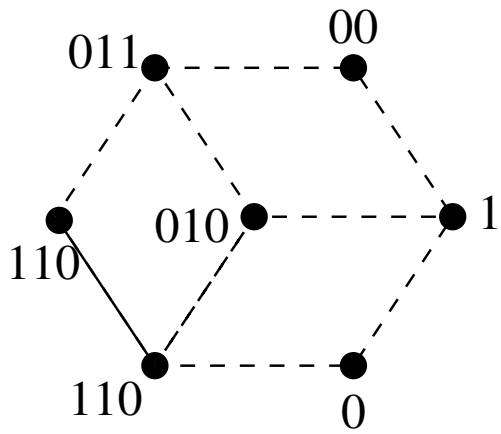
Algorithm *Fast_Colour*



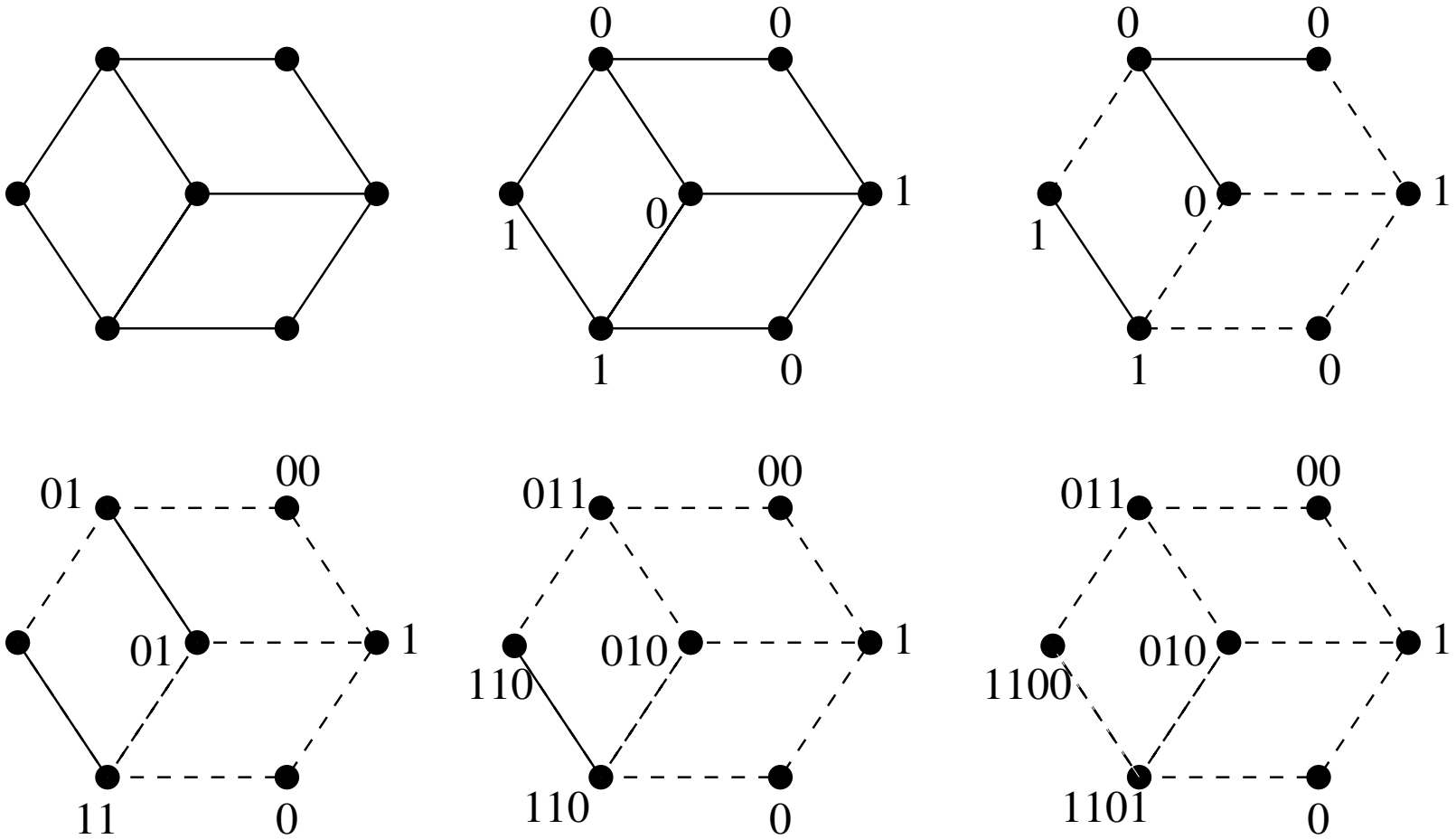




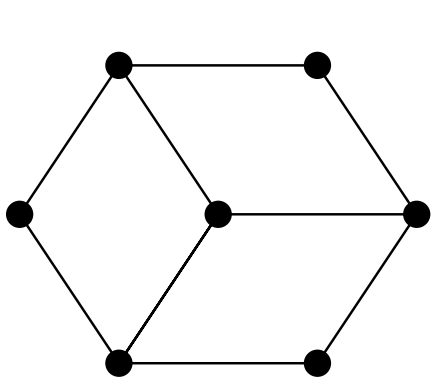


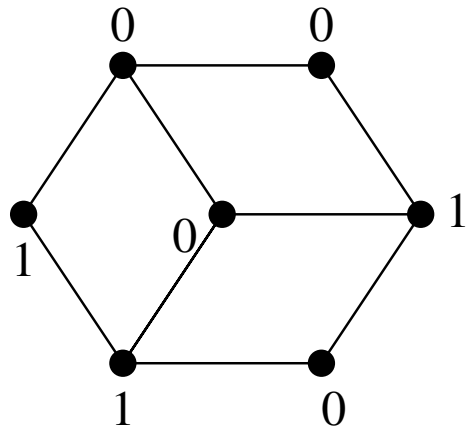


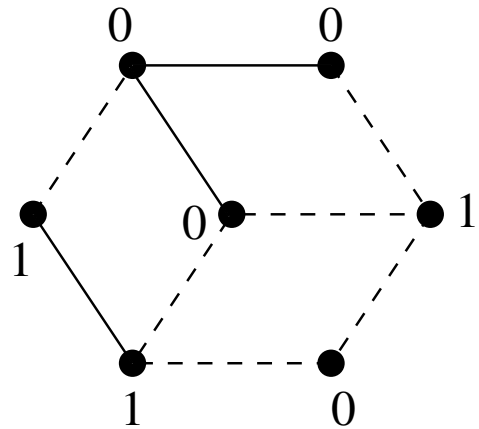
Algorithm *Fast_Colour*

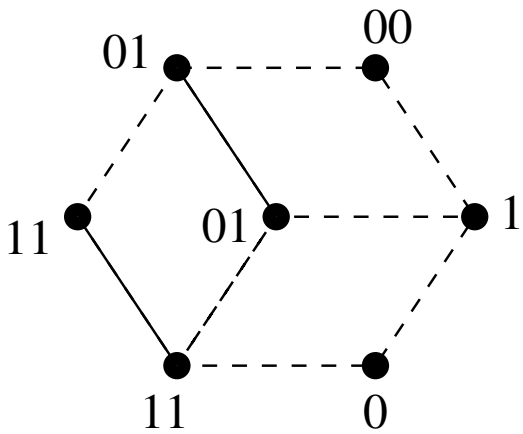


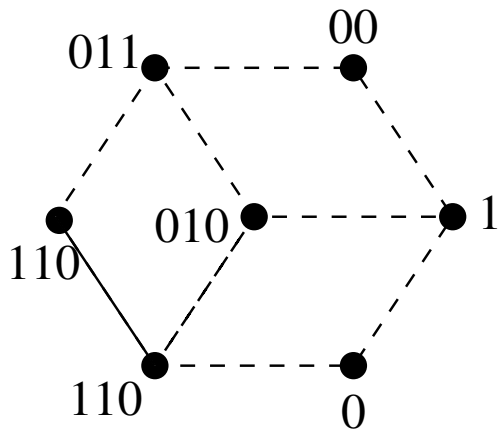
Algorithm *Fast_Colour*

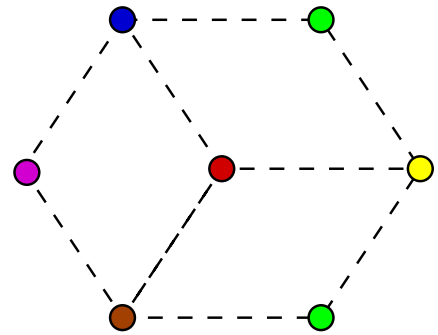












Analysis of the Algorithm

Lemma.
In any phase of the algorithm, the expected number of edges removed from the residual graph G is half the number of its edges.

Corollary.
There are constants k_1 and K_1 such that for any graph G of n vertices, the number of phases to remove all edges from G is:

- less than $k_1 \log n$ on average,
- less than $K_1 \log n$ w.h.p.

Analysis of the Algorithm

Lemma.

In any phase of the algorithm, the expected number of edges removed from the residual graph G is half the number of its edges.

Corollary.

There are constants k_1 and K_1 such that for any graph G of n vertices, the number of phases to remove all edges from G is:

- less than $k_1 \log n$ on average,
- less than $K_1 \log n$ w.h.p.

Analysis of the Algorithm

Theorem.
Algorithm *Fast_Colour* computes a colouring for any arbitrary graph of size n in time $O(\log n)$ w.h.p., each message containing 1 bit.

- What about the number of colours ?
- Can we do better ?
  - YES !

Analysis of the Algorithm

Theorem.
Algorithm *Fast_Colour* computes a colouring for any arbitrary graph of size n in time $O(\log n)$ w.h.p., each message containing 1 bit.

- What about the number of colours ?
- Can we do better ?
  - YES !

Analysis of the Algorithm

Theorem.
Algorithm *Fast_Colour* computes a colouring for any arbitrary graph of size n in time $O(\log n)$ w.h.p., each message containing 1 bit.

- What about the number of colours ?
- Can we do better ?
  - YES !

Analysis of the Algorithm

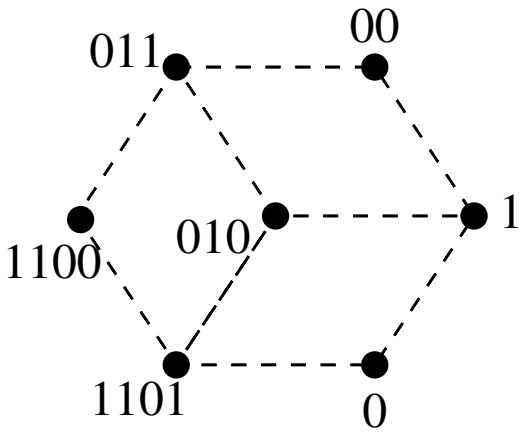
Theorem.
Algorithm *Fast_Colour* computes a colouring for any arbitrary graph of size n in time $O(\log n)$ w.h.p., each message containing 1 bit.

- What about the number of colours ?
- Can we do better ?
  - YES !

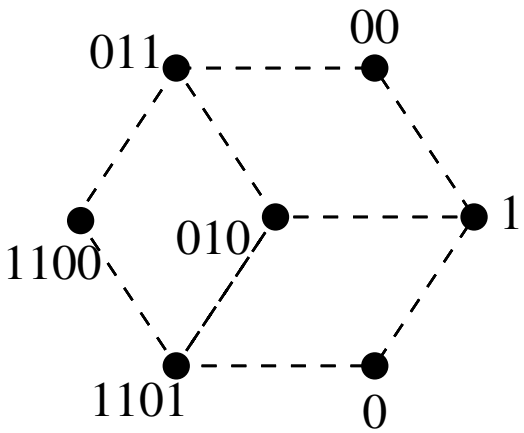
Reducing the Number of Colours - Algorithm $\mathcal{R}$

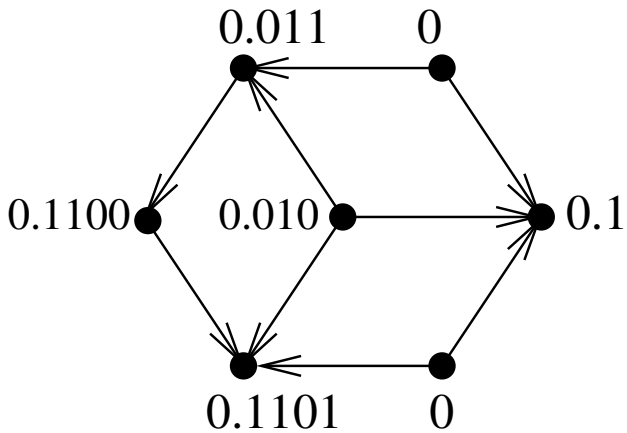
var:
  FC_u : integer **Init** $couleur_v$; (* final colour of v *)
  IN_v : $\subset N(v)$ **Init** $\emptyset$;
  OUT_v : $\subset N(u)$ **Init** $\emptyset$;
  $Colours_v$: set of possible colours **Init** $\{0, 1, \cdots, d(v)\}$;
For each $v \in N(v)$
  If $(x(u) < x(v))$ **then** $OUT_v = OUT_v \cup \{v\}$;
  Else $IN_v = IN_v \cup \{v\}$;
  End If;
End For
While $IN_v \neq \emptyset$ **do**
  receive $< c, w >$ from a neighbour w;
  $IN_v = IN_v \setminus \{w\}$;
  $Colours_v = Colours_v \setminus \{c\}$;
End While
$FC_v = \min\{Colours_v\}$;
For Each $u \in OUT_v$
  send $< FC_v, u >$ to v;
End For;

Reducing the Number of Colours - Algorithm $\mathcal{R}$

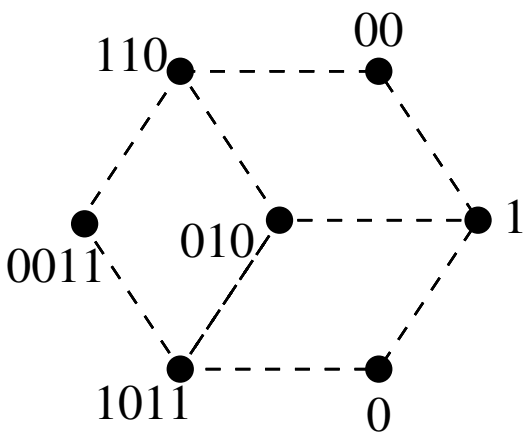


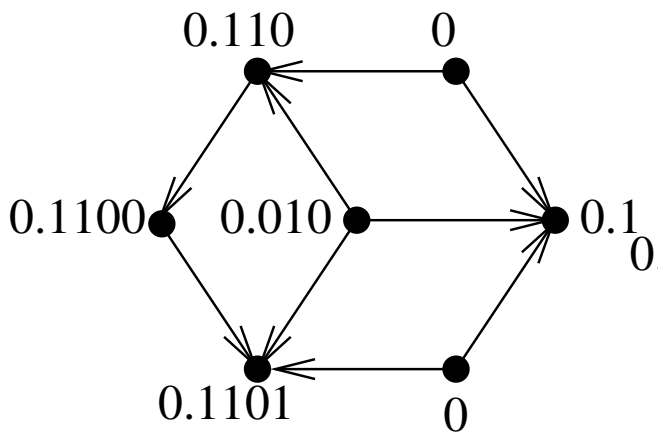
Reducing the Number of Colours - Algorithm $\mathcal{R}$

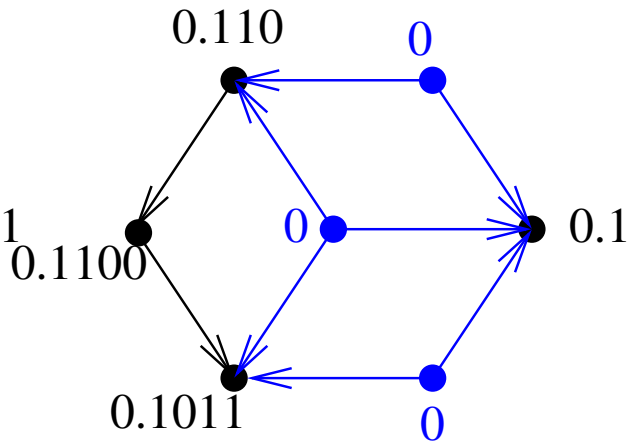




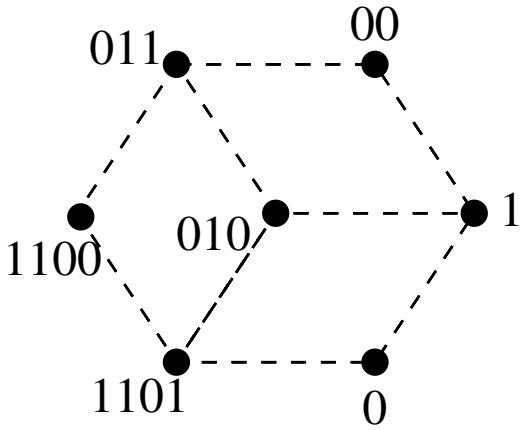
Reducing the Number of Colours - Algorithm $\mathcal{R}$

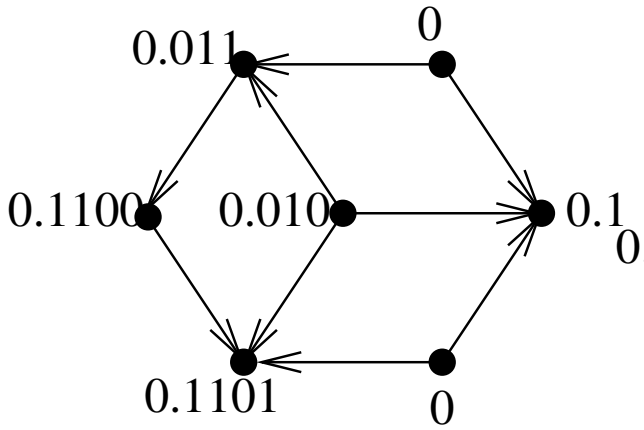


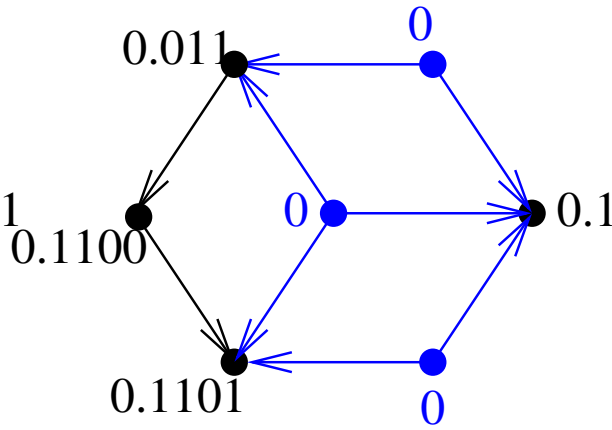


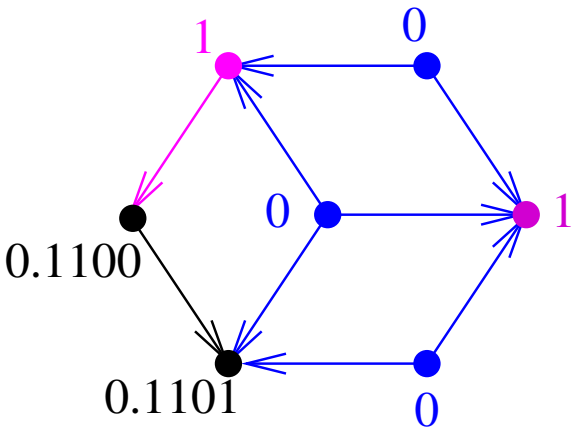


Reducing the Number of Colours - Algorithm $\mathcal{R}$

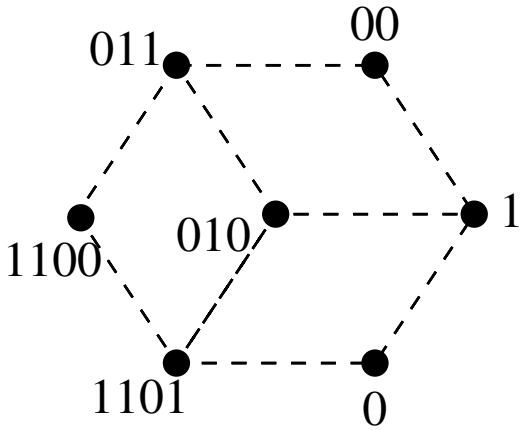


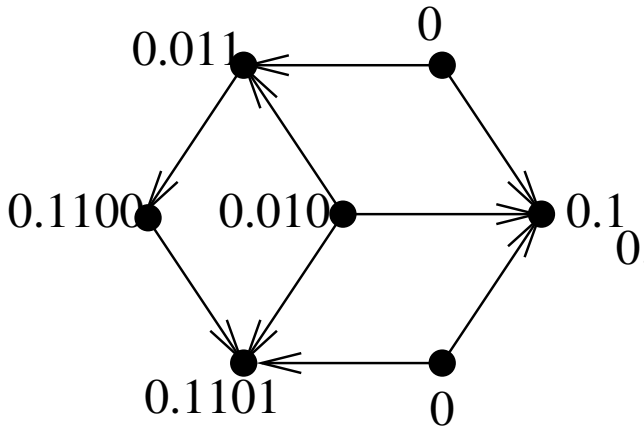


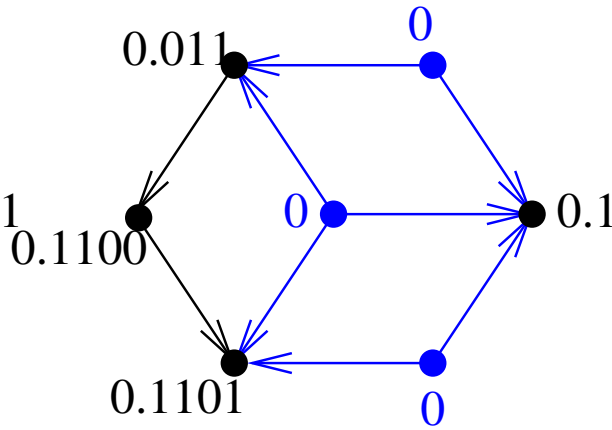


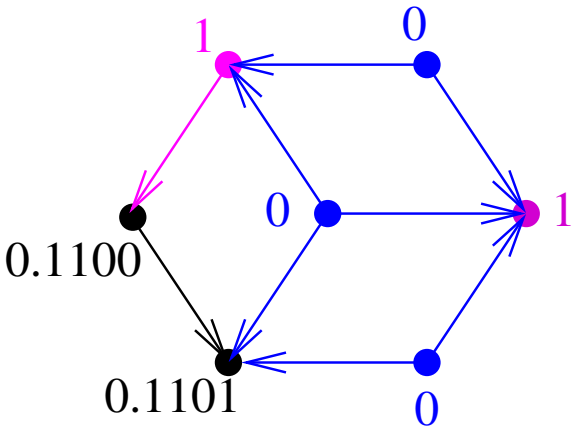


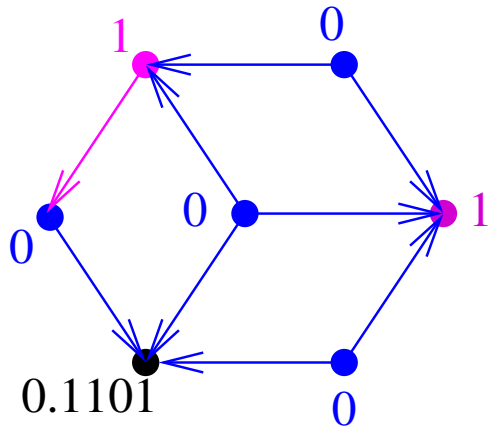
Reducing the Number of Colours - Algorithm $\mathcal{R}$



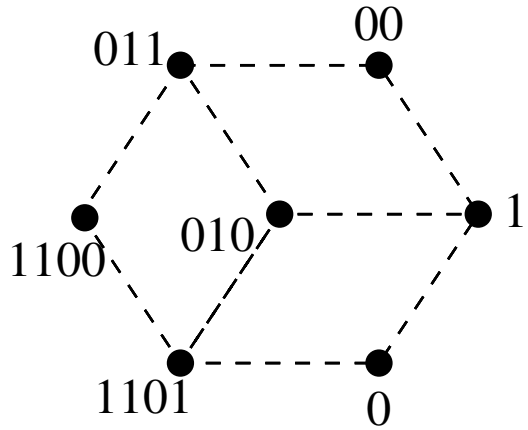


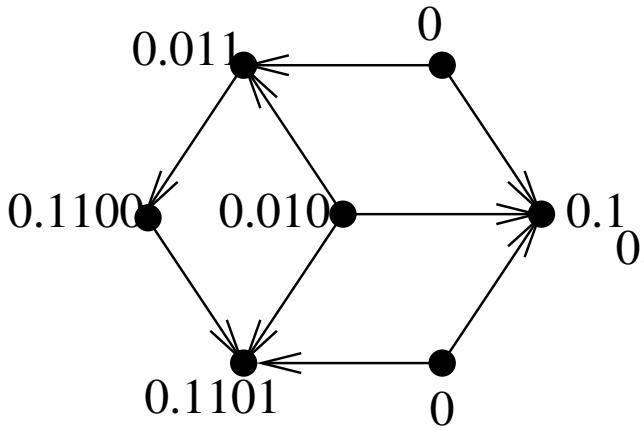


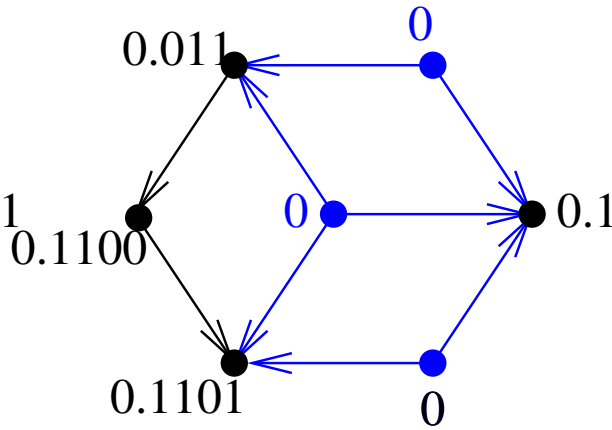


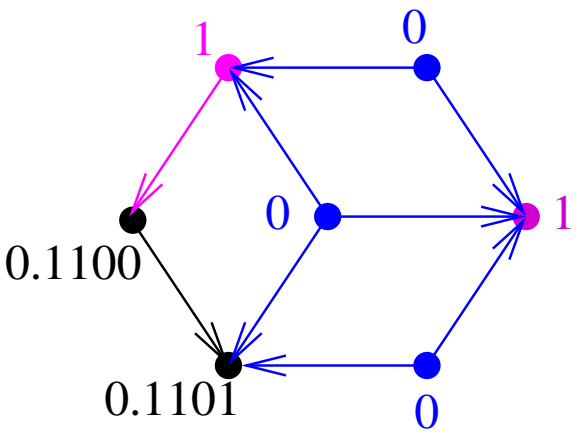


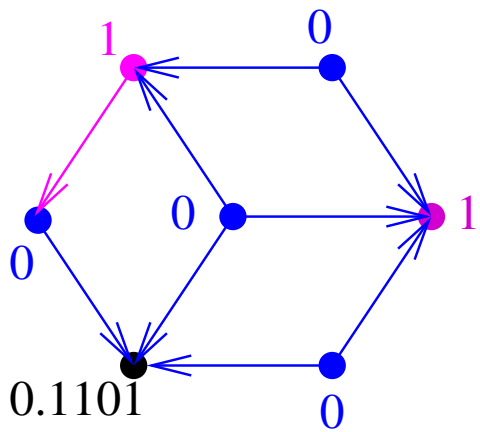
Reducing the Number of Colours - Algorithm $\mathcal{R}$

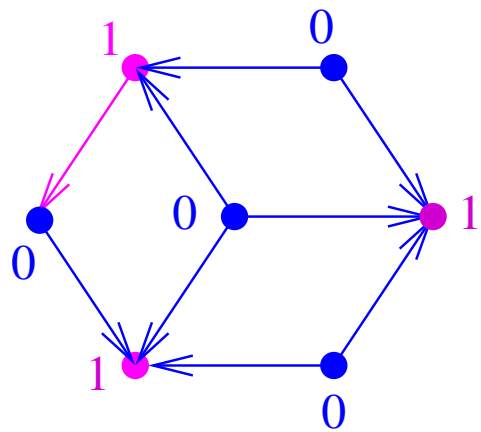












Analysis of the Algorithm

Theorem

For any graph $G = (V, E)$ having a maximum degree Δ, Algorithm $\mathcal{R}$ achieves a $(\Delta + 1)-$colouring of G in at most $e\Delta + 3 \log n$ rounds w.h.p.

Bibliography I

H. Attiya and J. Welch.

Distributed Computing.
Wiley, 2004.

D. Peleg.

Distributed computing - A Locality-sensitive approach.
SIAM Monographs on discrete mathematics and applications, 2000.

G. Tel.

Introduction to distributed algorithms.
Cambridge University Press, 2000.

R. Motwani and P. Raghavan.

Randomized Algorithms.
Cambridge University Press, 1995.

K. Kothapalli, M. Onus, C. Scheideler, and C. Schindelhauer.

Distributed coloring in $O(\sqrt{\log n})$ bit rounds.
In *20th International Parallel and Distributed Processing Symposium (IPDPS 2006), Proceedings, 25-29 April 2006, Rhodes Island, Greece*. IEEE, 2006.

Y. Métivier, J.M. Robson, N. Saheb-Djahromi and A. Zemmari.

About randomised distributed graph coloring and graph partition algorithms.
Information and Computation. 208 :1296-1304, 2010.