Algorithmes Distribués Probabilistes

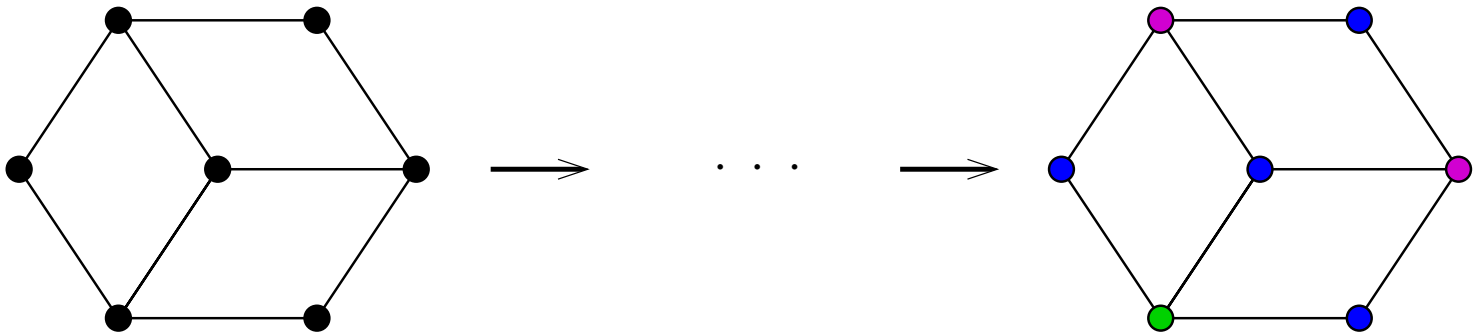
Akka Zemmari
zemmari@labri.fr

25 novembre 2013

Définitions.

Soit $G = (V, E)$ un graphe. Une *coloration propre* de G est une fonction qui à chaque sommet v affecte une couleur $c(v)$ telles que

$$\forall \{u, v\} \in E, \;\; c(v) \neq c(u).$$



Algorithme séquentiel

Algorithme 1.

1. Tant que $\exists$ un sommet non colorié v faire
2. colorier v avec la plus petite couleur qui n'est pas en conflit avec les voisins déjà coloriés
3. Fin tant que

Théorème

L'algorithme est correct et termine en n pas. L'algorithme utilise $\Delta + 1$ couleurs.

Un premier algorithme distribué (Wattenhofer)

On considère la procédure suivante

FirstFree

Require : Les sommets ont des identifiants (ID) différents
Colorier v avec la plus petite couleur encore disponible.

Remarque : on doit s'assurer que deux voisins ne sont pas coloriés au même moment !

Un premier algorithme distribué (Wattenhofer)

On considère l'algorithme :

Reduce

Require : Les sommets ont des identifiants (ID) différents

Chaque noeud v exécute les instructions suivantes :

v envoie son *ID* à tous ses voisins;

v reçoit les *ID*s de ses voisins;

Tant que v a un voisin non colorié avec un *ID* plus grand **faire**

v envoie *undecided* à tous ses voisins ;

Fin tant que

v choisit une couleur libre en utilisant *FirstFree*;

v informe tous ses voisins de son choix;

Un premier algorithme distribué (Wattenhofer)

Théorème

L'algorithme est correct et termine en n pas. L'algorithme utilise $\Delta + 1$ couleurs.

Observations

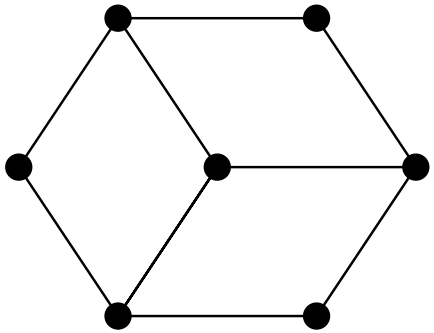
- nécessite la connaissance des *ID*s
- utilise des messages de taille $\log n$ (en supposant que les *ID*s sont de simples numéros de 1 à n
- complexité en n
+ algorithme distribué
+ utilise $\Delta + 1$ couleurs.

L'algoritrhme *FastColor*

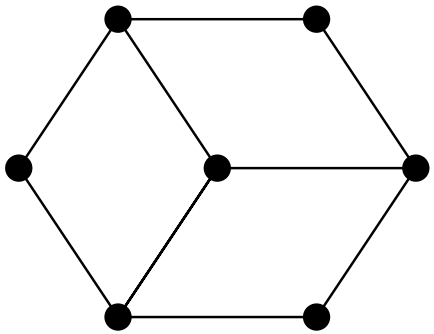
var:

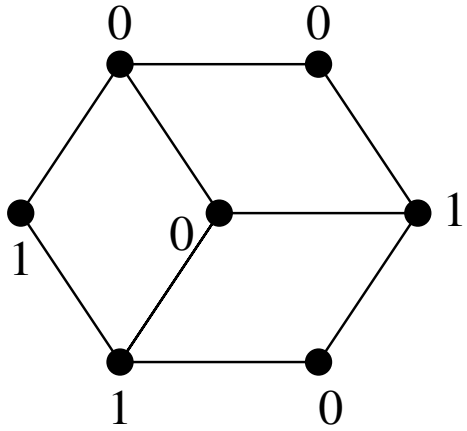
$couleur_v$: *mot* **Init** *mot-vide*;
$actifs_v$: $\subseteq N(v)$ **Init** $N(v)$;
b_v: $\in \{0,1\}\}$;
Tant que $actifs_v \neq \emptyset$ **faire**
$b_v \leftarrow flip(0,1)$;
$couleur_v \leftarrow b_v \oplus couleur_v$;
Pour tout $u \in actifs_v$ **Faire**
envoyer b_v à u;
recevoir b_u de u;
Si $b_v \neq b_u$**alors**
$active_v \leftarrow active_v \setminus \{u\}$;
Fin Si
Fin Pour
Fin Tant que

L'algorithme *Fast_Colour*

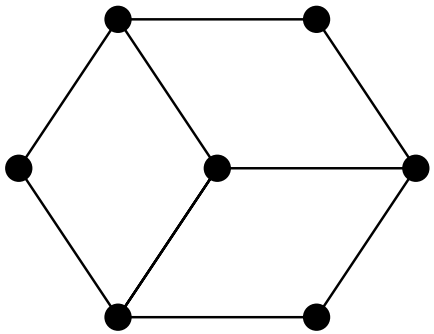


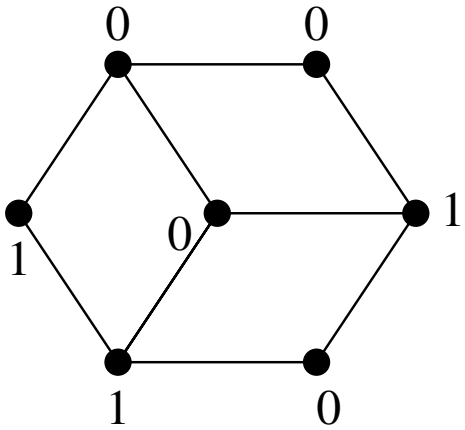
L'algorithme *Fast_Colour*

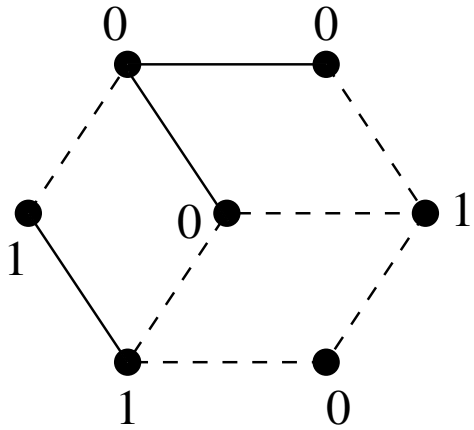




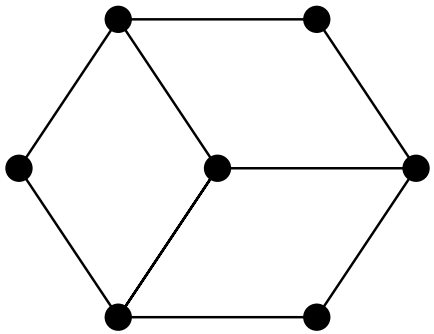
L'algorithme *Fast_Colour*

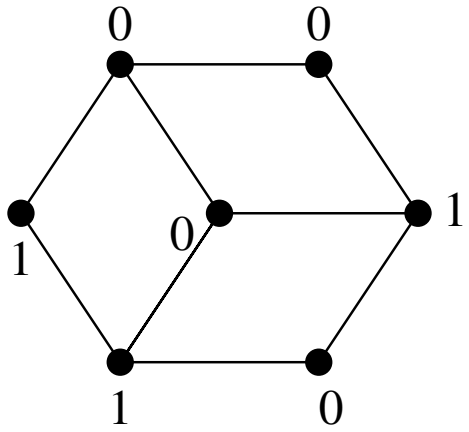


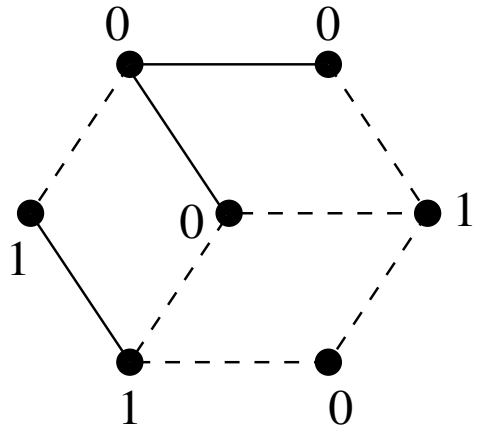


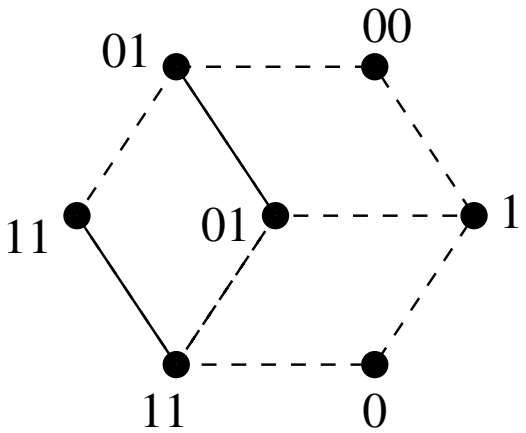


L'algorithme *Fast_Colour*

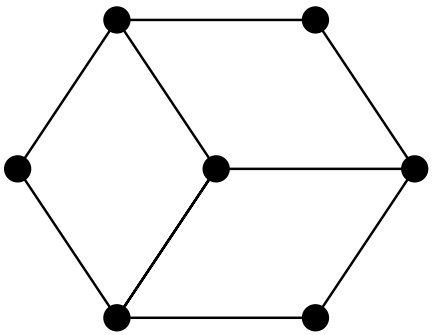


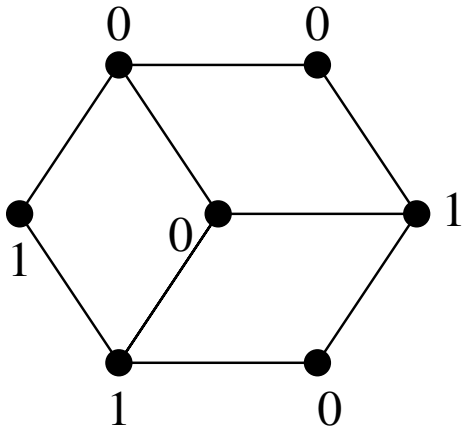


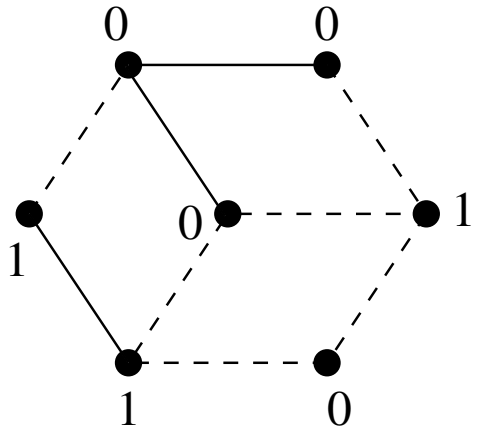


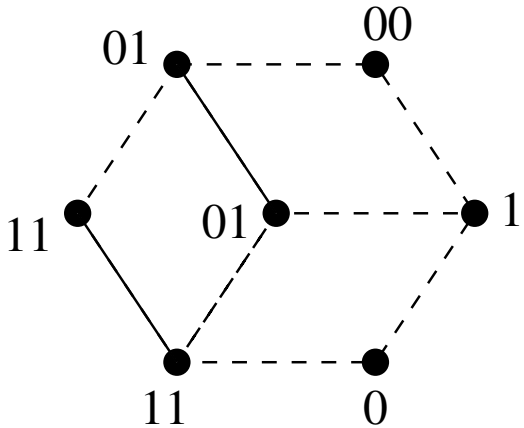


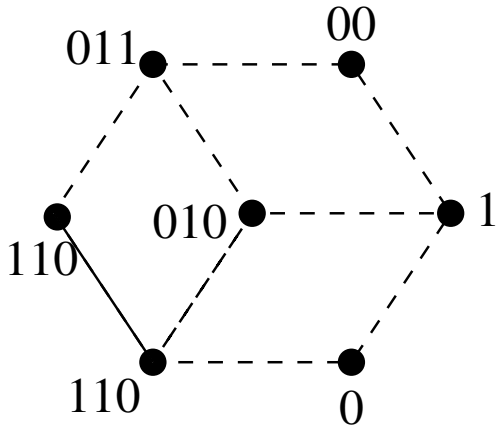
L'algorithme *Fast_Colour*



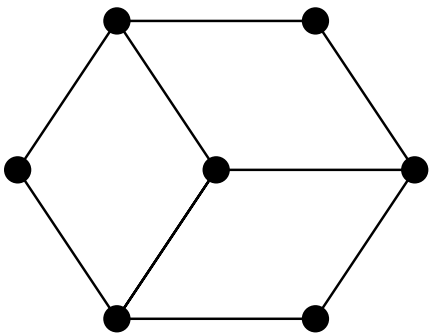


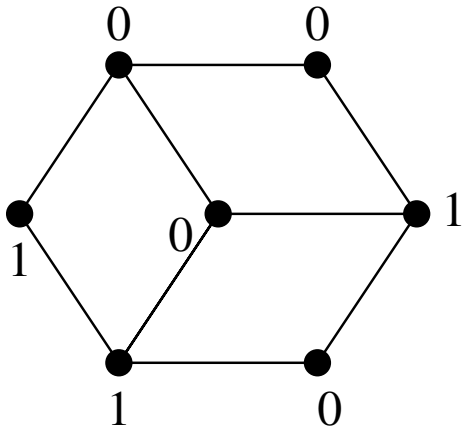


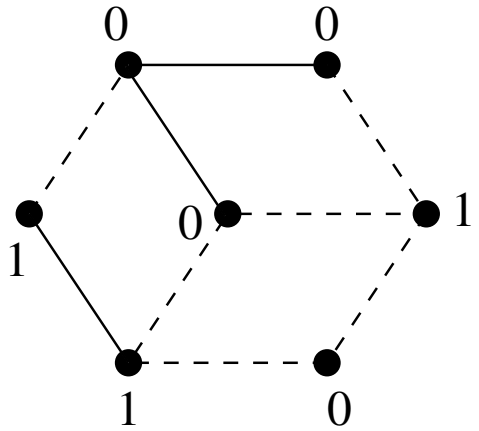


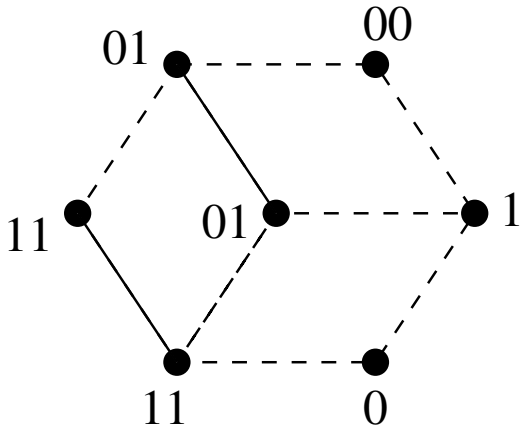


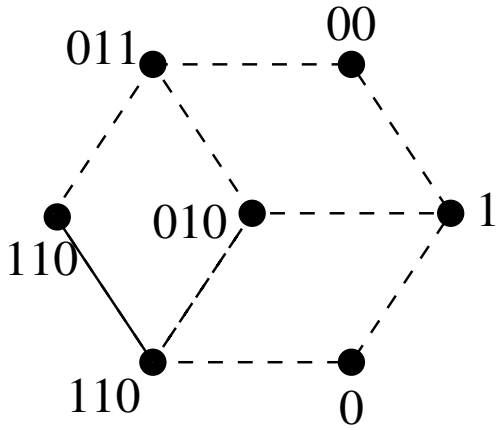
L'algorithme *Fast_Colour*

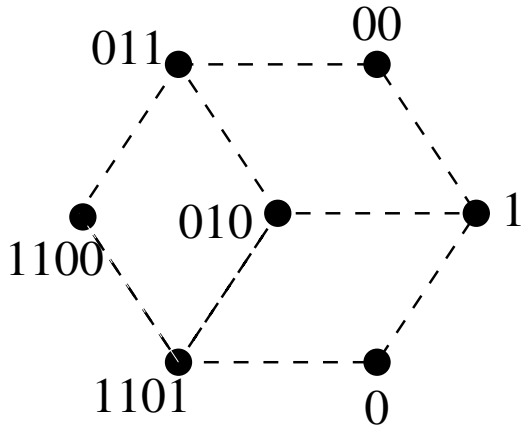




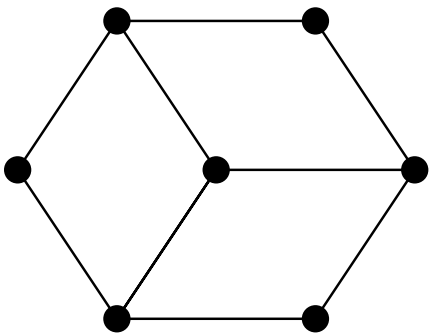


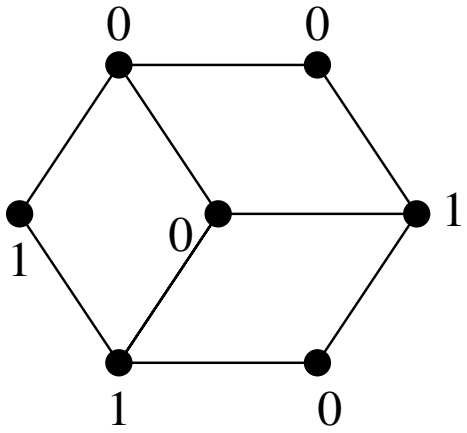


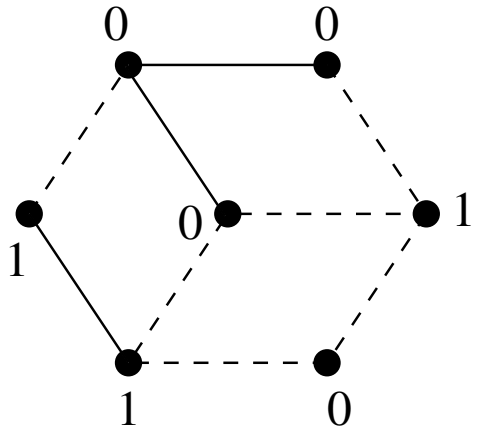


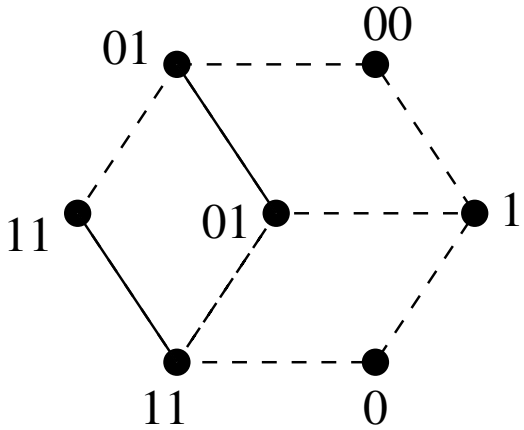


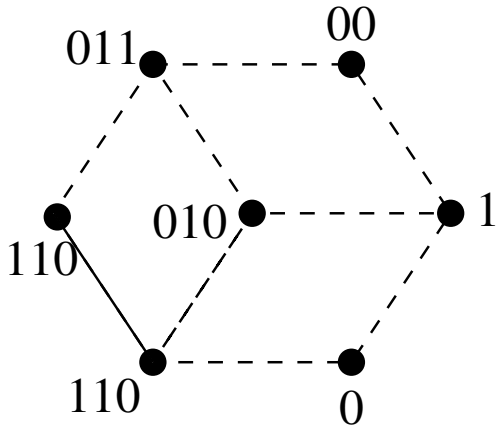
L'algorithme *Fast_Colour*

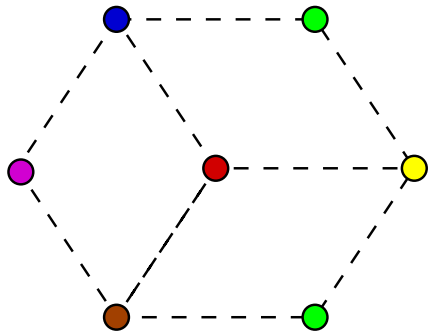












Analyse de l'algorithme

Lemma.

En moyenne, chaque phase supprime la moitié des arêtes du graphe risiduel.

Théorème.

Il existe deux constantes k_1 et K_1 telles que pour tout graphe G à n sommets, le nombre de phases nécessaires pour supprimer toutes les arêtes de G est

- inférieur à $k_1 \log n$ en moyenne,
- inférieur à $K_1 \log n$ avec forte probabilité $(1 - o(\frac{1}{n}))$.

Analyse de l'algorithme

Lemma.

En moyenne, chaque phase supprime la moitié des arêtes du graphe risiduel.

Théorème.

Il existe deux constantes k_1 et K_1 telles que pour tout graphe G à n sommets, le nombre de phases nécessaires pour supprimer toutes les arêtes de G est

- inférieur à $k_1 \log n$ en moyenne,
- inférieur à $K_1 \log n$ avec forte probabilité $(1 - o(\frac{1}{n}))$.

Observations :

+ Le réseau peut être anonyme
+ utilise des messages de taille 1 bit
+ complexité en $O(\log n)$
+ algorithme distribué
- utilise n couleurs

Problème : Nombre de couleurs ....

Réduction du nombre de couleurs - L'algorithme $\mathcal{R}$

var:

FC_u : entier **Init** $couleur_v$; (* couleur finale de v *)

IN_v : $\subset N(v)$ **Init** $\emptyset$;

OUT_v : $\subset N(u)$ **Init** $\emptyset$;

$Couleurs_v$: ensemble de couleurs **Init** $\{0, 1, \cdots, d(v)\}$;

Pour chaque $v \in N(v)$

Si $(x(u) < x(v))$ **alors** $OUT_v = OUT_v \cup \{v\}$;

Sinon $IN_v = IN_v \cup \{v\}$;

Fin Si;

Fin Pour

Tant que $IN_v \neq \emptyset$ **faire**

recvoir $< c, w >$ d'un voisin w;

$IN_v = IN_v \setminus \{w\}$;

$Colours_v = Colours_v \setminus \{c\}$;

Fin Tant que

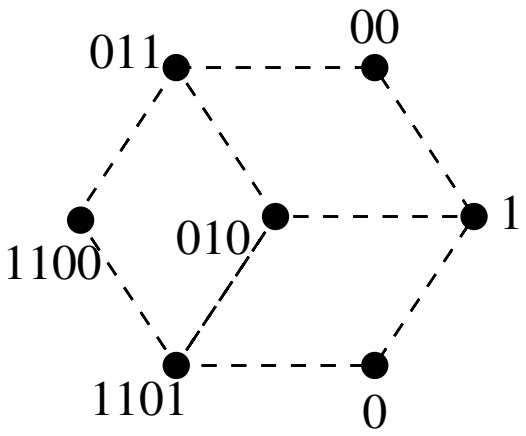
$FC_v = \min\{Colours_v\}$;

Pour chaque $u \in OUT_v$

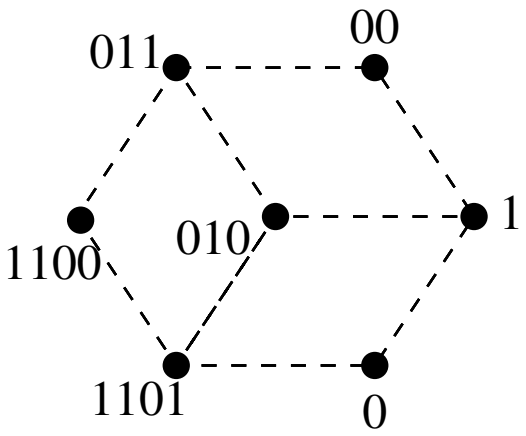
envoyer $< FC_v, u >$ à v;

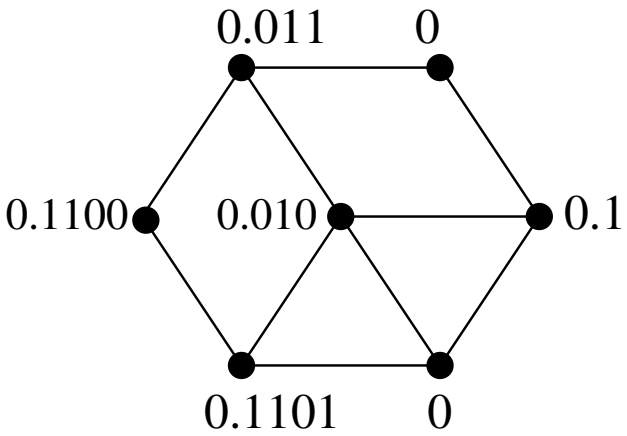
Fin Pour;

Réduction du nombre de couleurs - L'algorithme $\mathcal{R}$

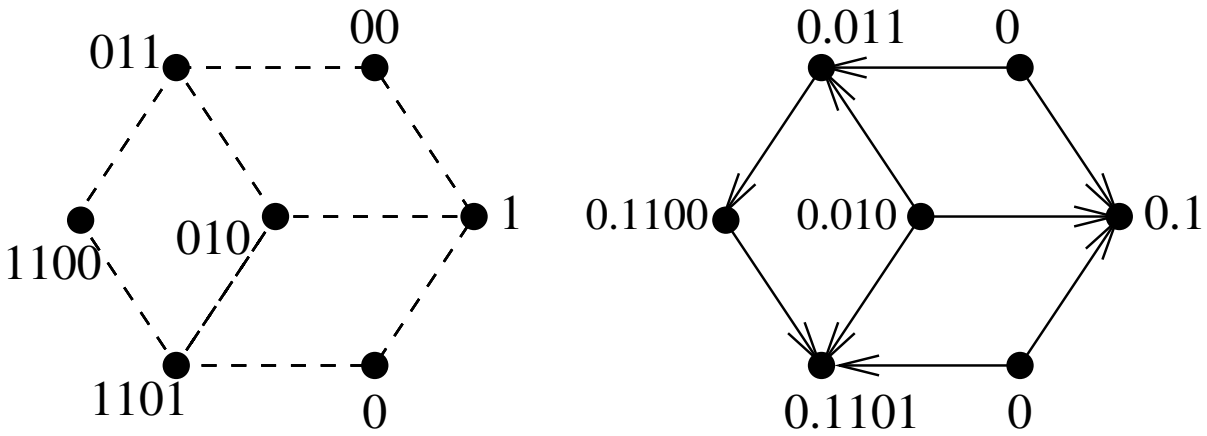


Réduction du nombre de couleurs - L'algorithme $\mathcal{R}$

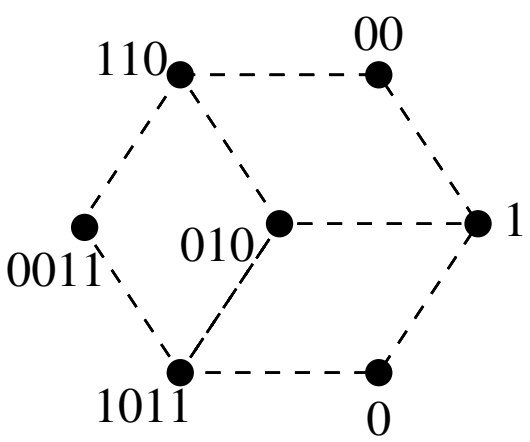


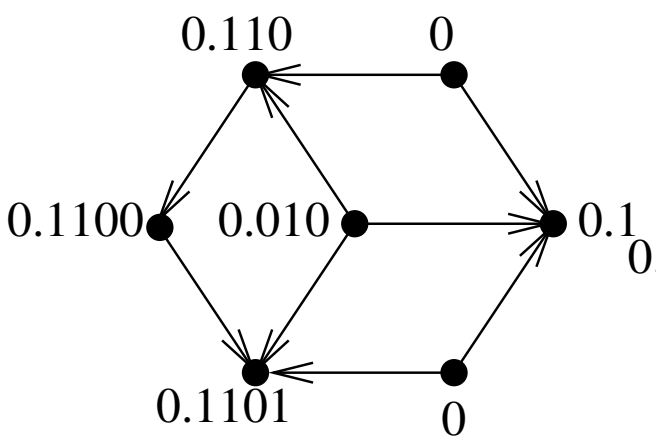


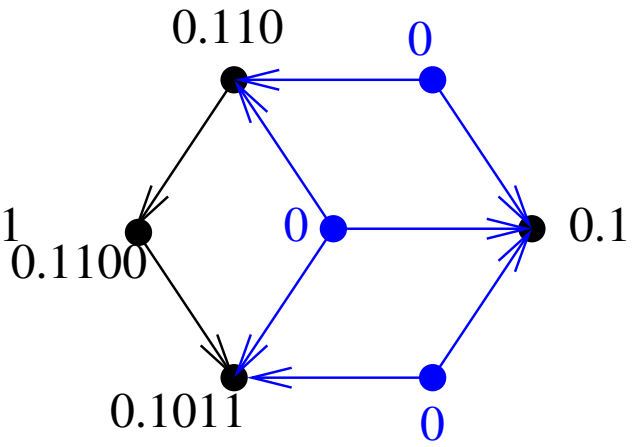
Réduction du nombre de couleurs - L'algorithme $\mathcal{R}$



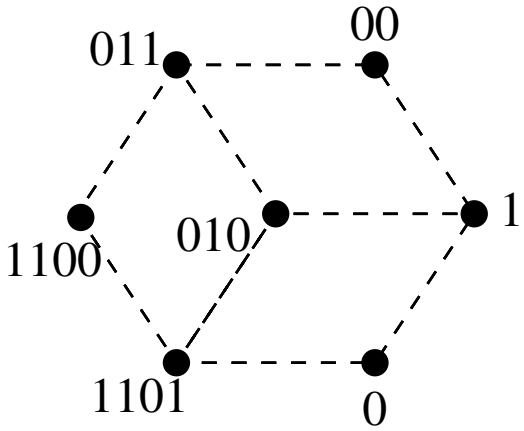
Réduction du nombre de couleurs - L'algorithme $\mathcal{R}$

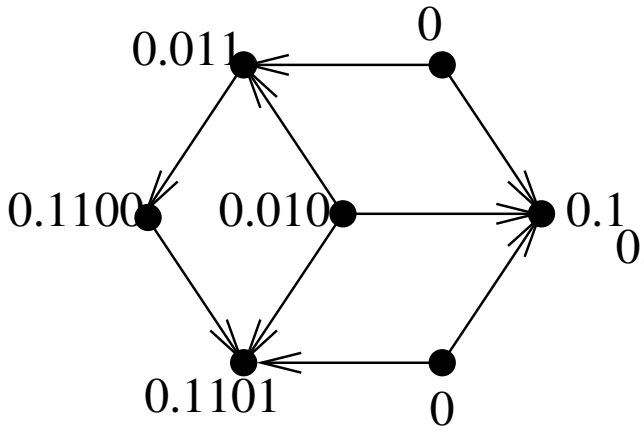


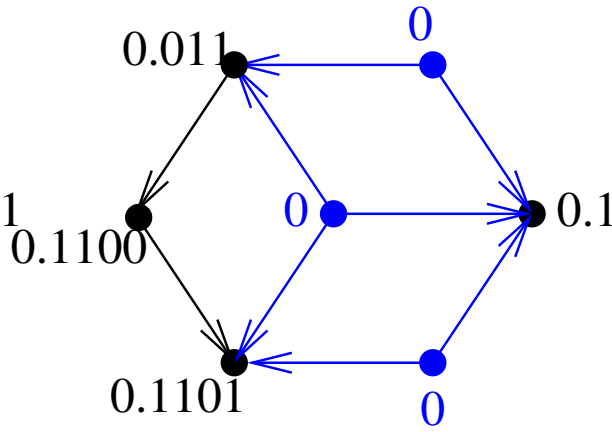


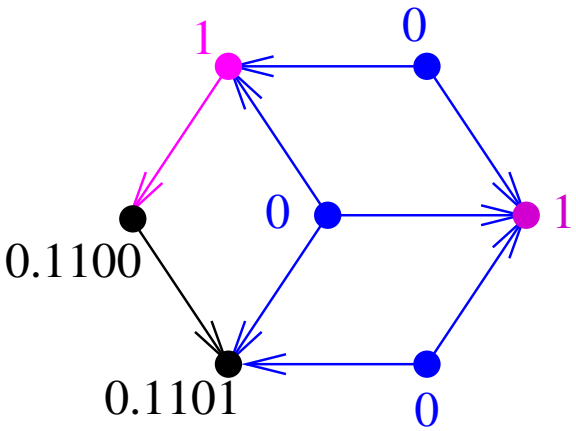


Réduction du nombre de couleurs - L'algorithme $\mathcal{R}$

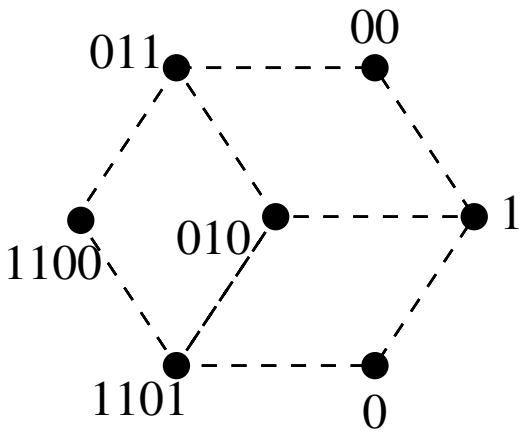


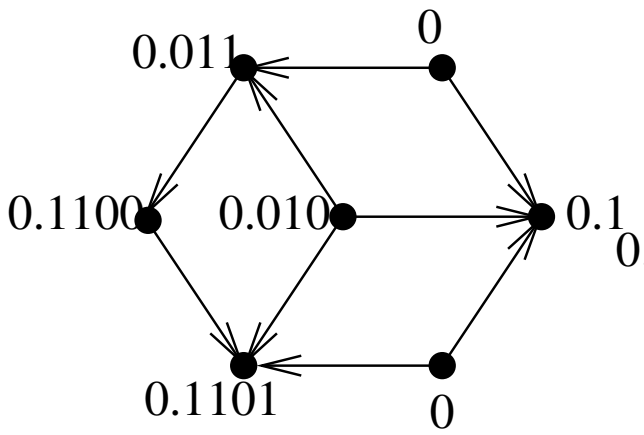


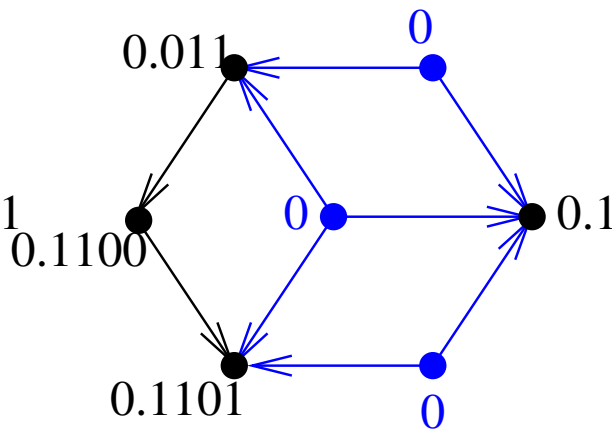


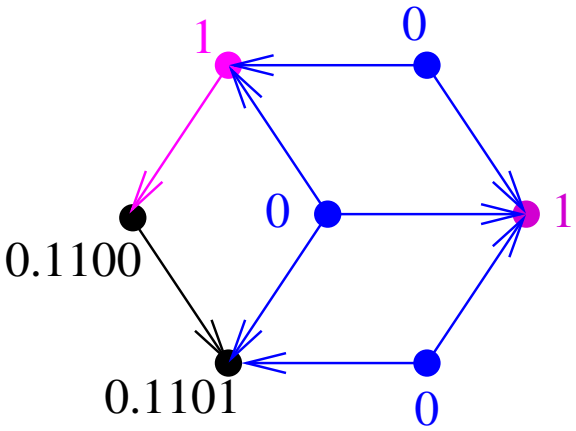


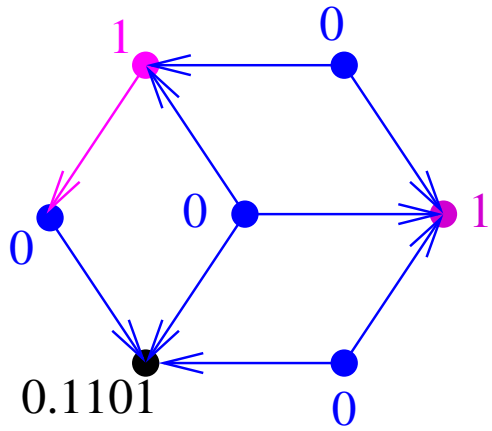
Réduction du nombre de couleurs - L'algorithme $\mathcal{R}$



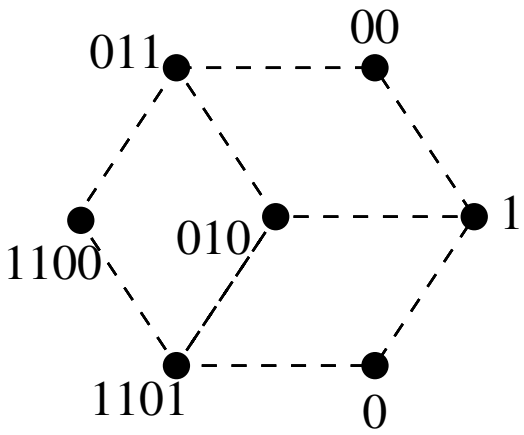


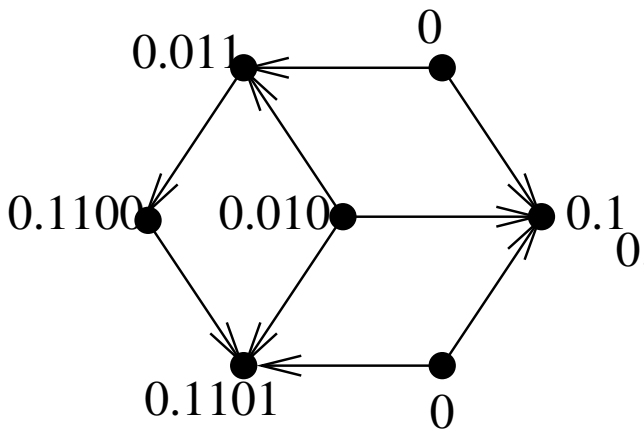


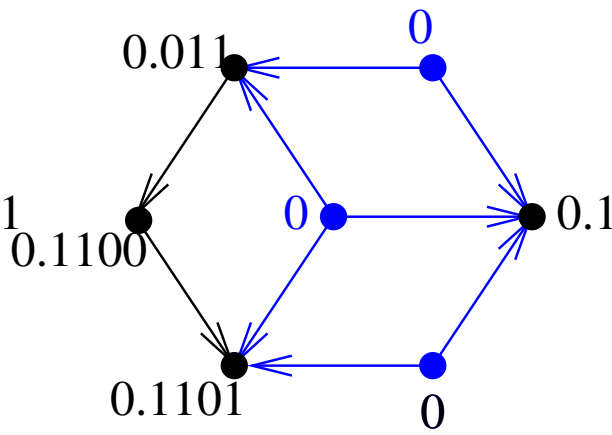


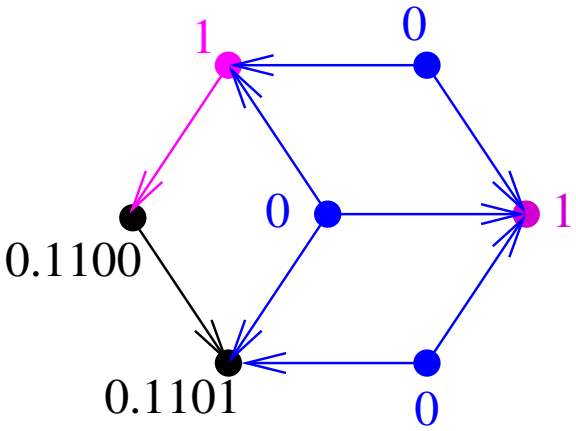


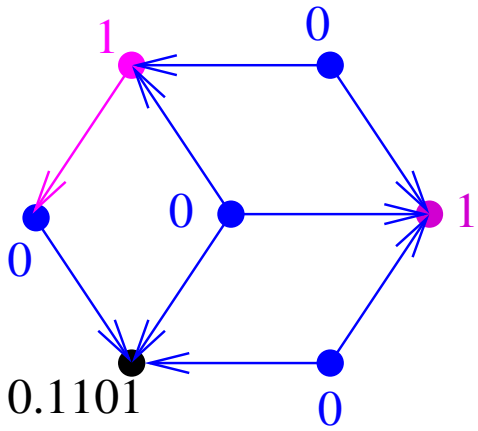
Réduction du nombre de couleurs - L'algorithme $\mathcal{R}$

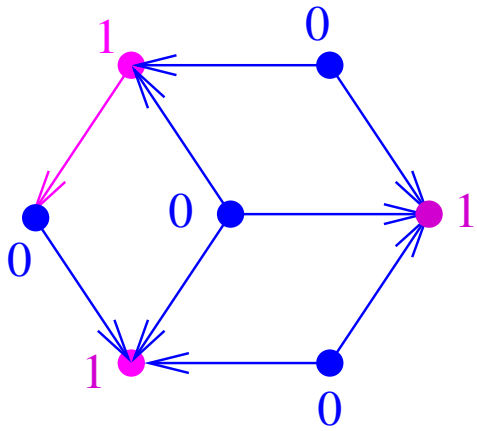




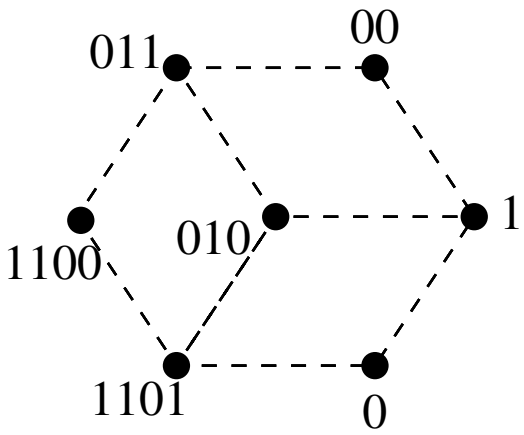


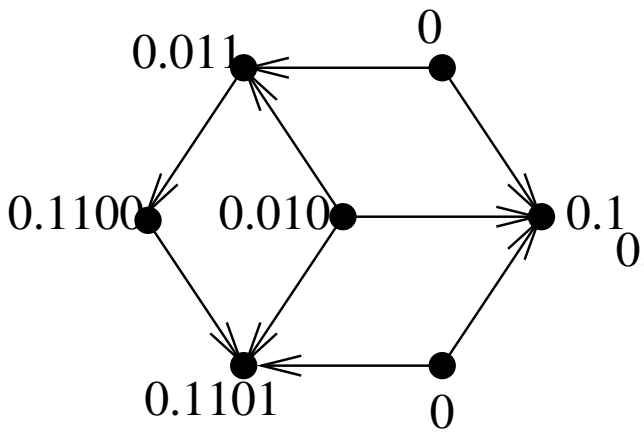


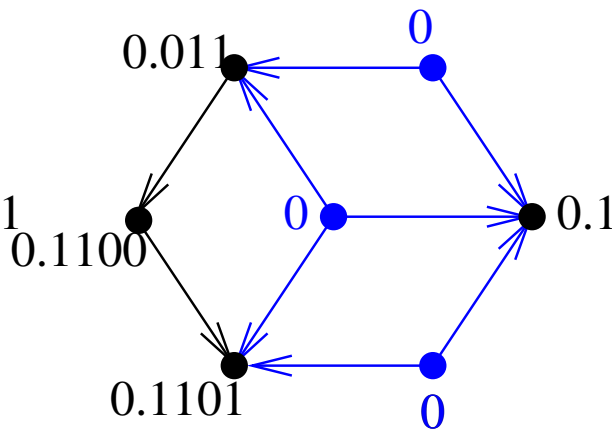


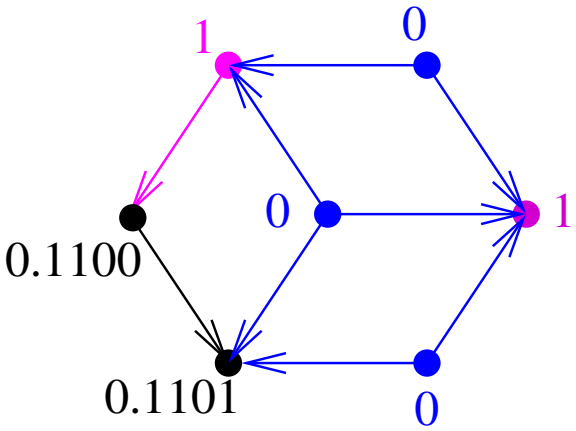


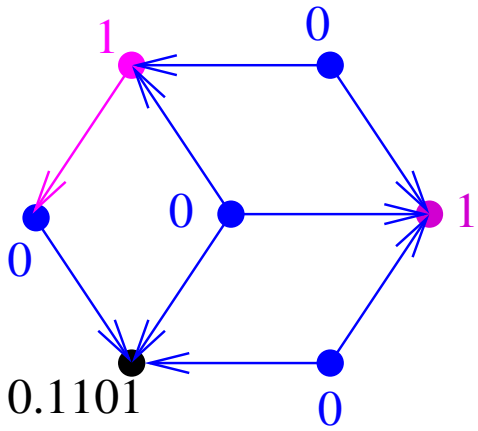
Réduction du nombre de couleurs - L'algorithme $\mathcal{R}$

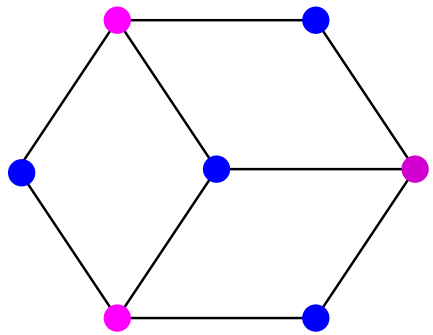












Analyse de l'algorithm

Théorème

Pour tout graphe $G = (V, E)$ à degré maximum Δ, l'algorithme $\mathcal{R}$ réalise une $(\Delta + 1)-$coloration de G en au plus $e\Delta + 3 \log n$ rounds et ce avec forte probabilité.

Analyse de l'algorithm

En réalité, on a le théorème suivant :

Théorème

Le problème du coloriage pour les graphes anonymes de taille n a une complexité $\Theta(\log n)$ rounds avec forte probabilité.