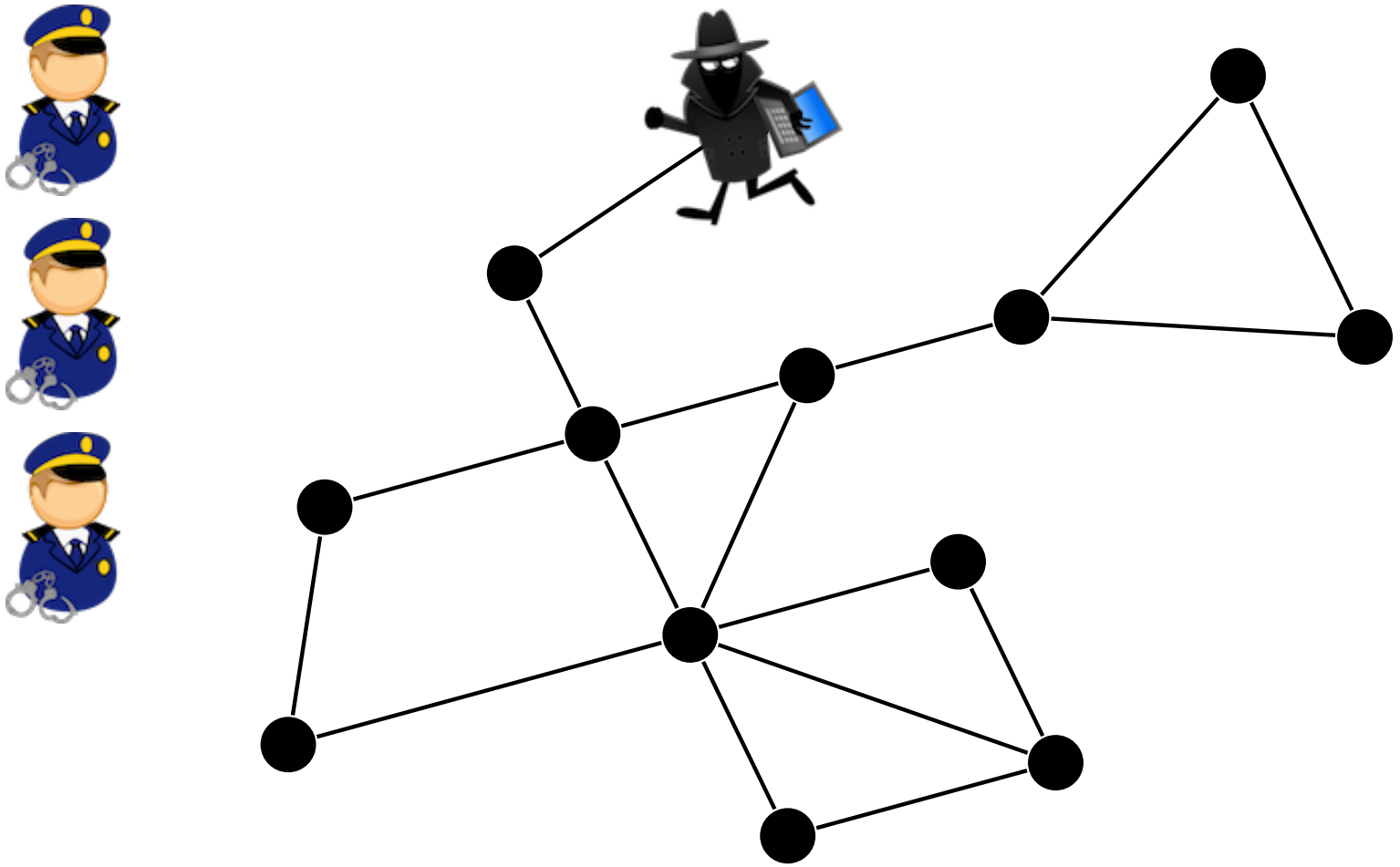
Jeu des gendarmes et des voleurs et largeur arborescente

Marie Fortin

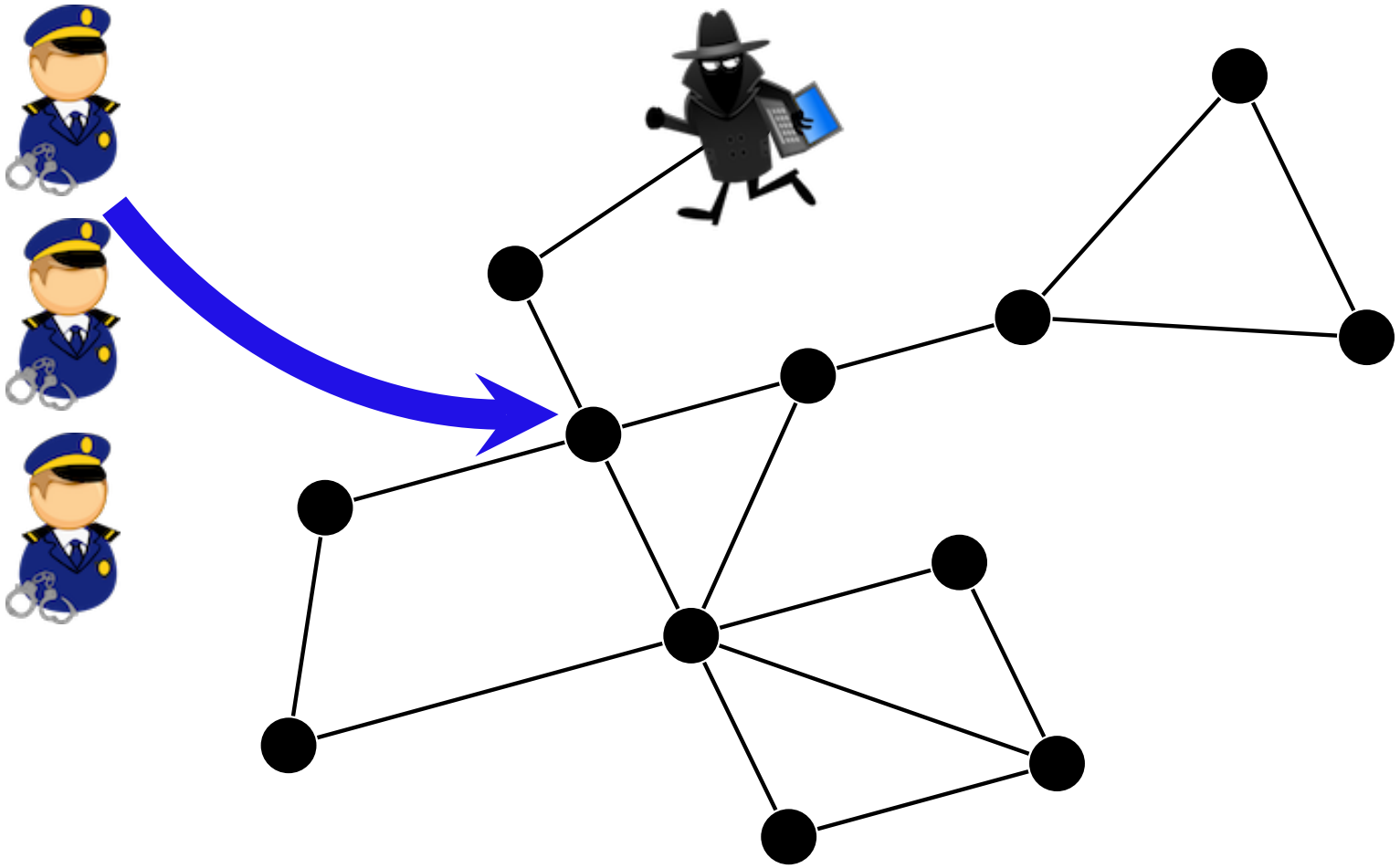
LSV, ENS Paris-Saclay

Portes Ouvertes LSV, 7 septembre 2017

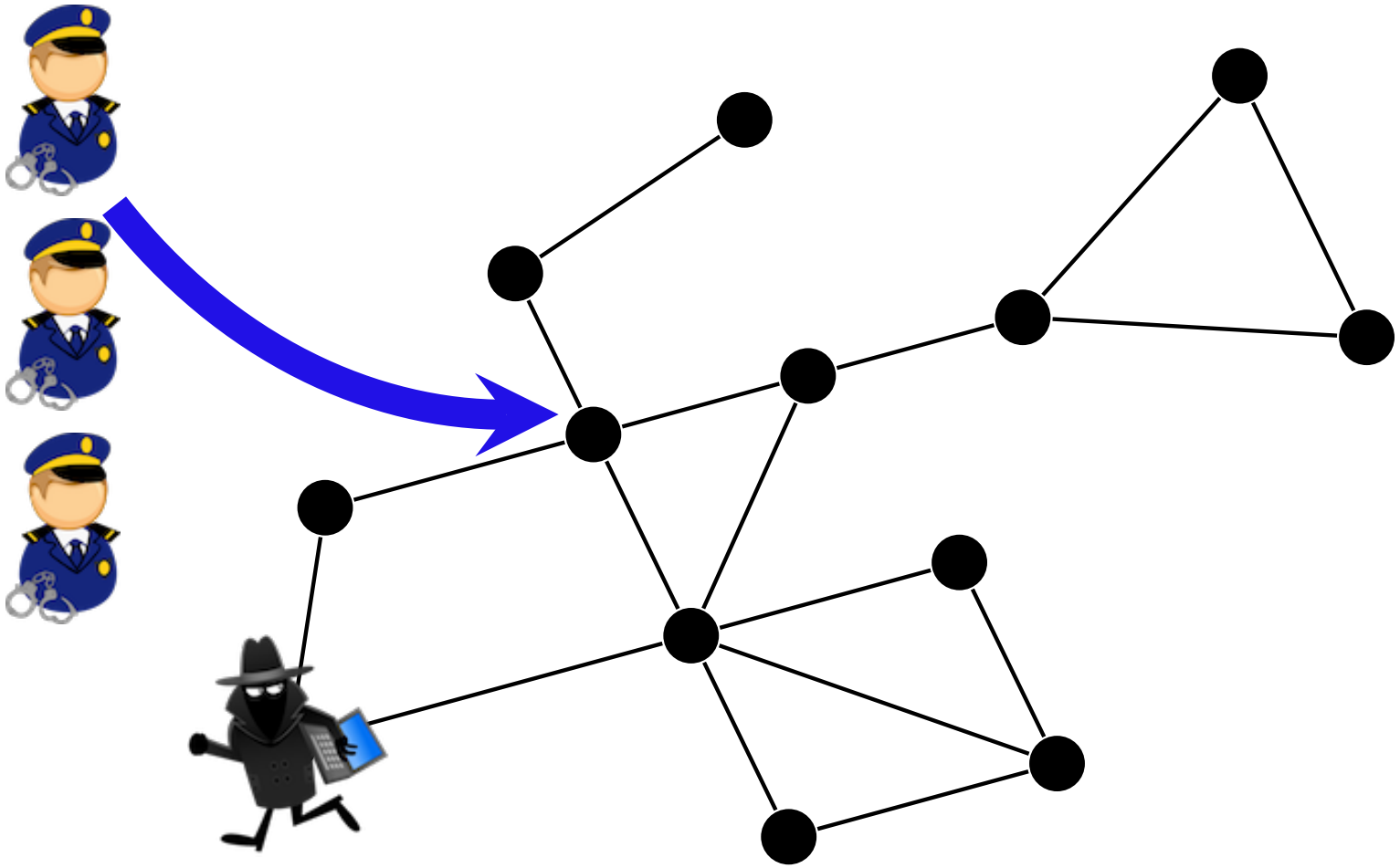
Les gendarmes et le voleur



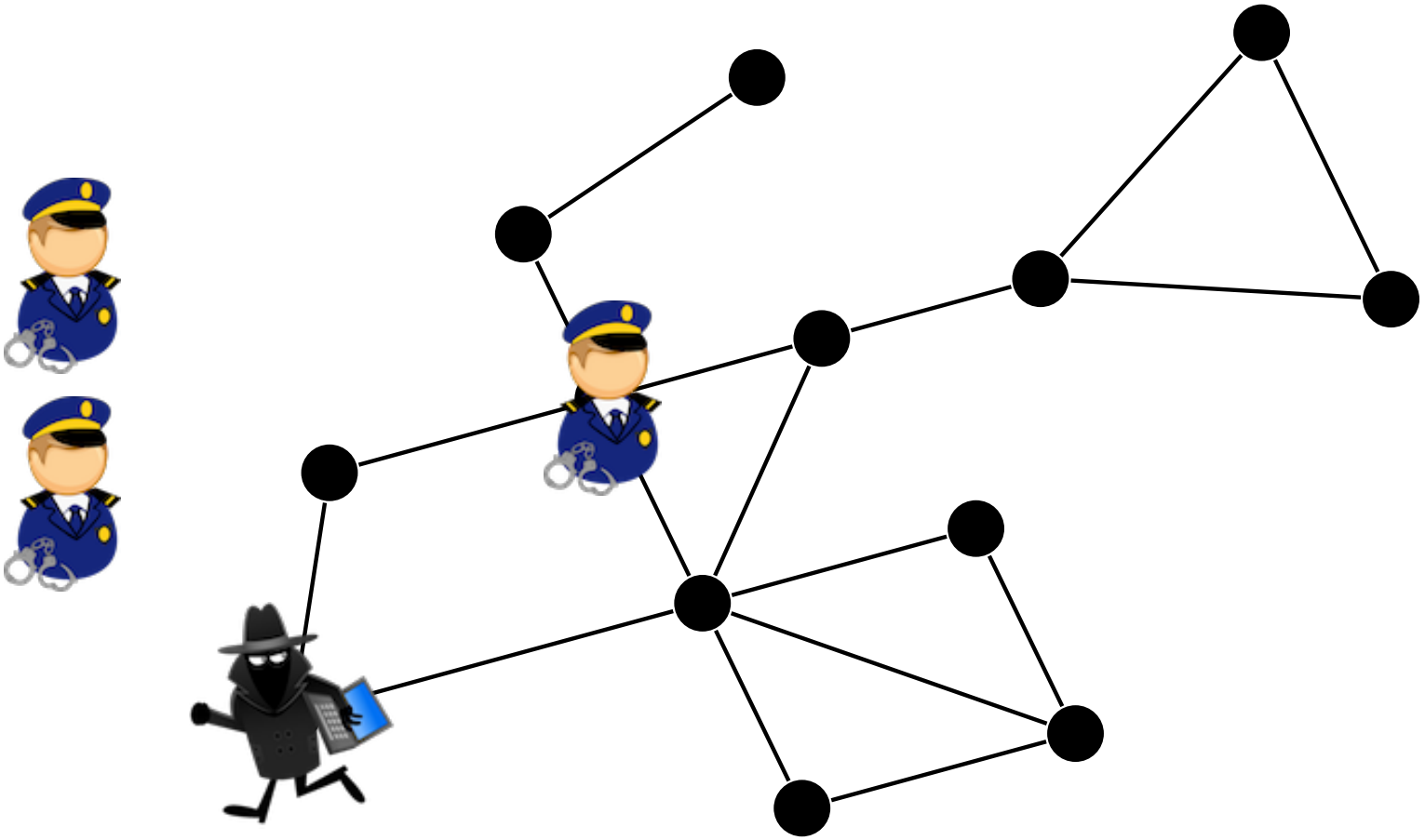
Les gendarmes et le voleur



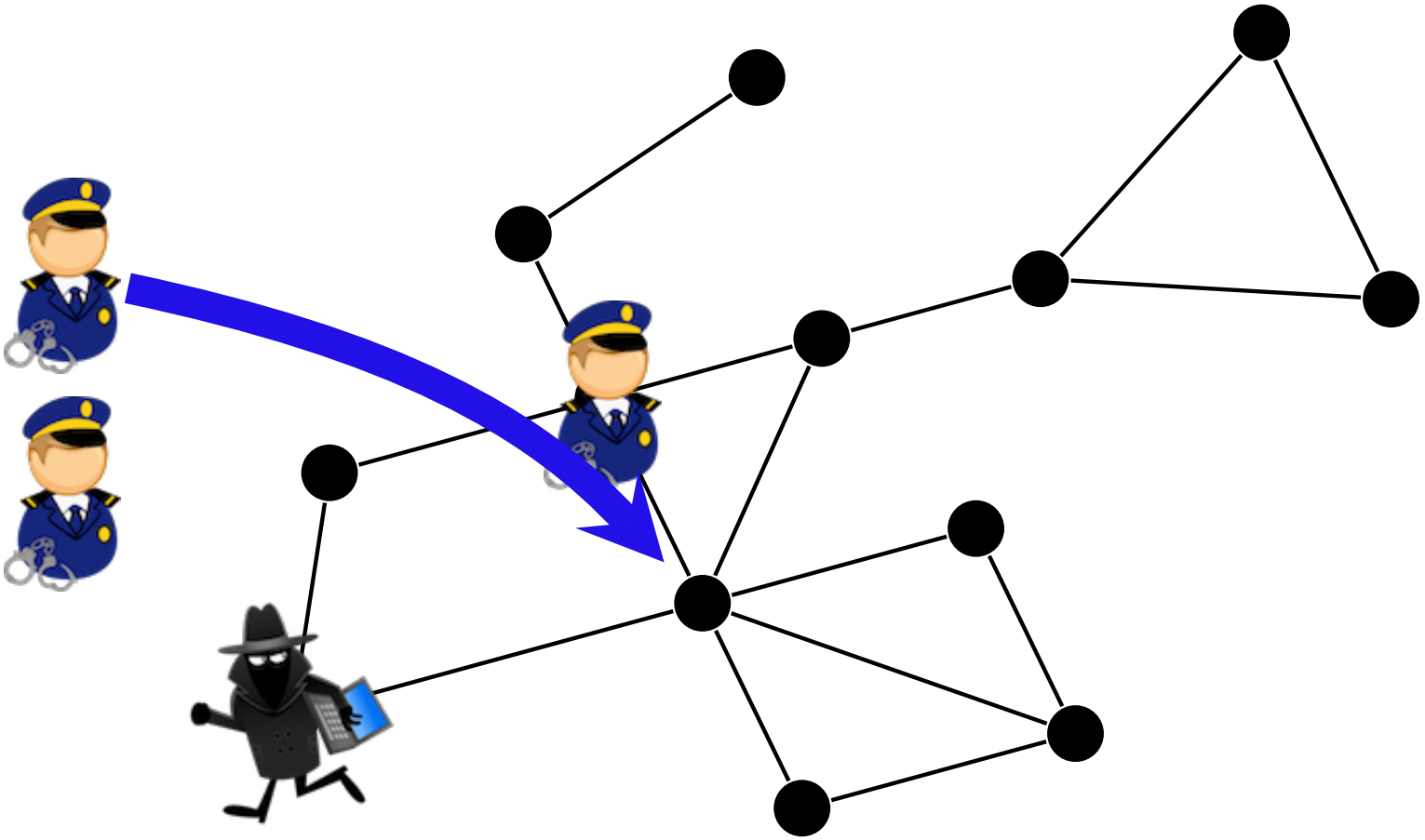
Les gendarmes et le voleur



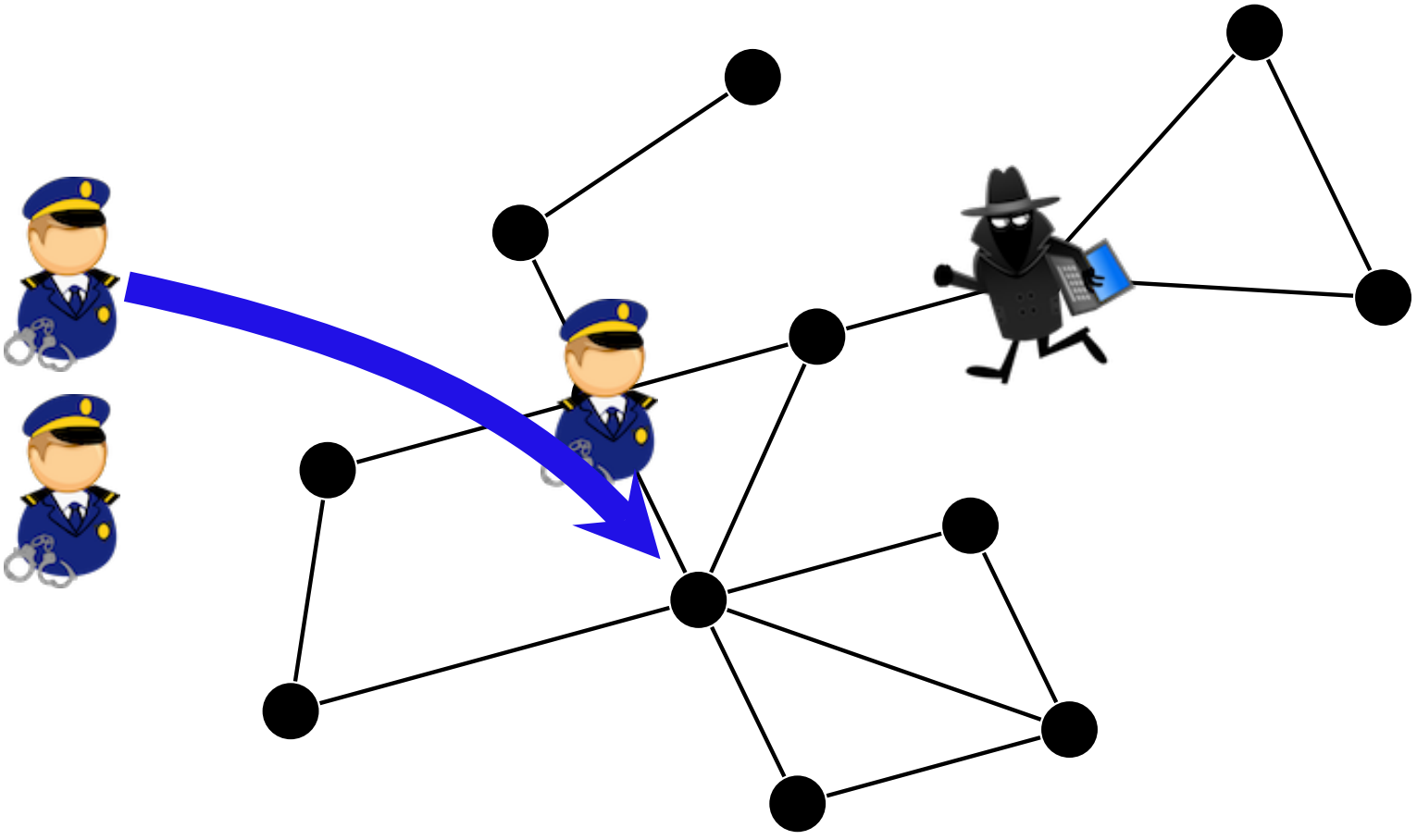
Les gendarmes et le voleur



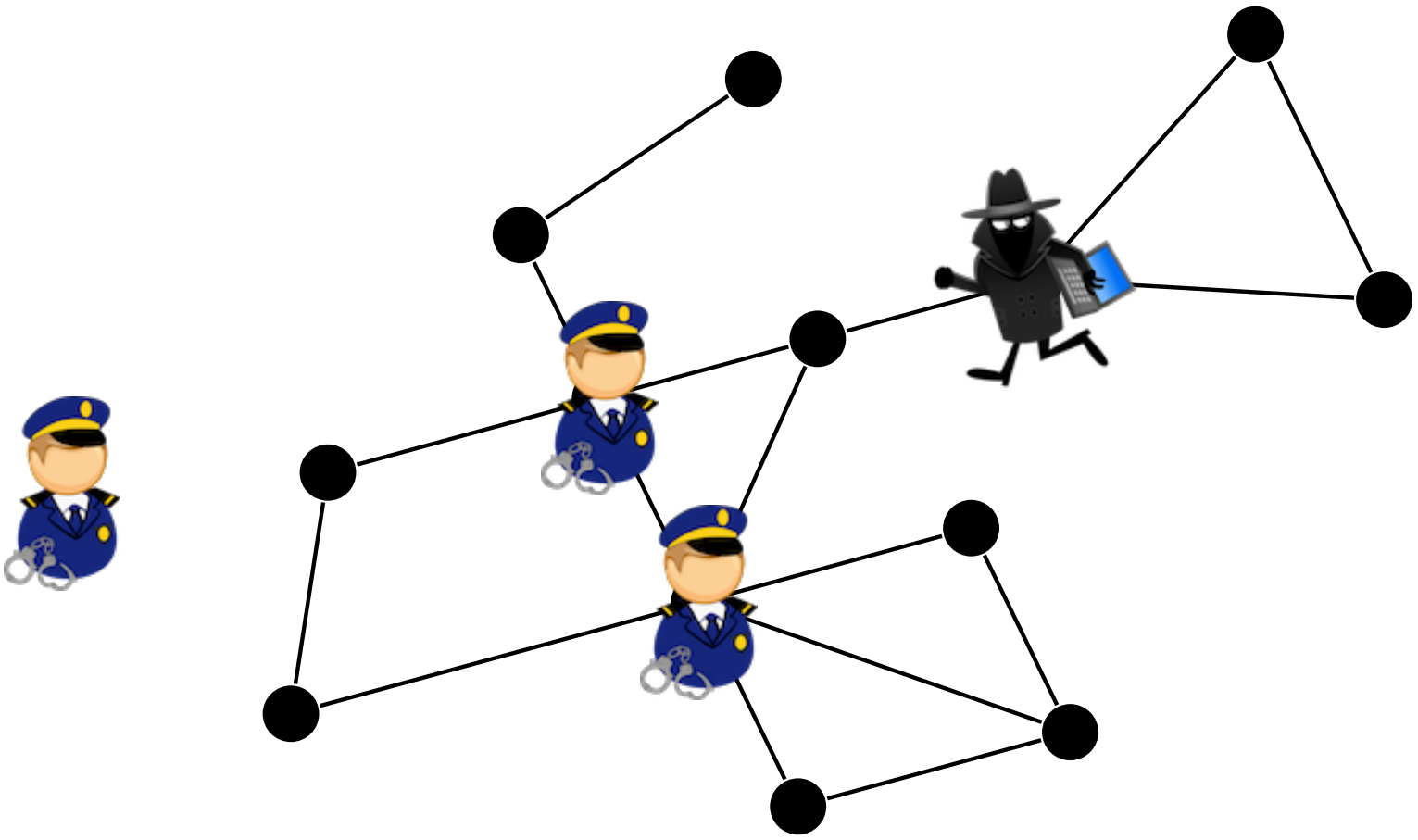
Les gendarmes et le voleur



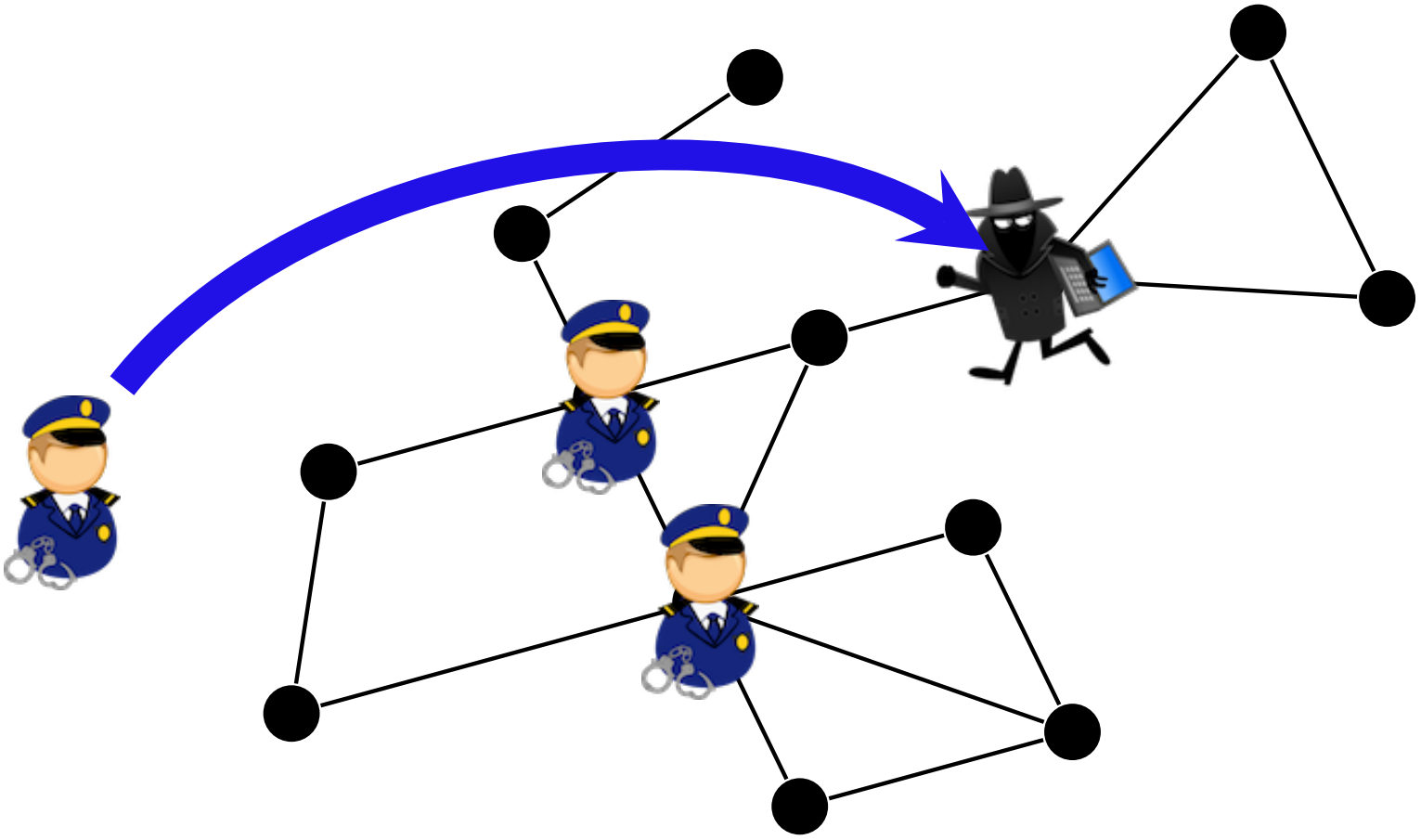
Les gendarmes et le voleur



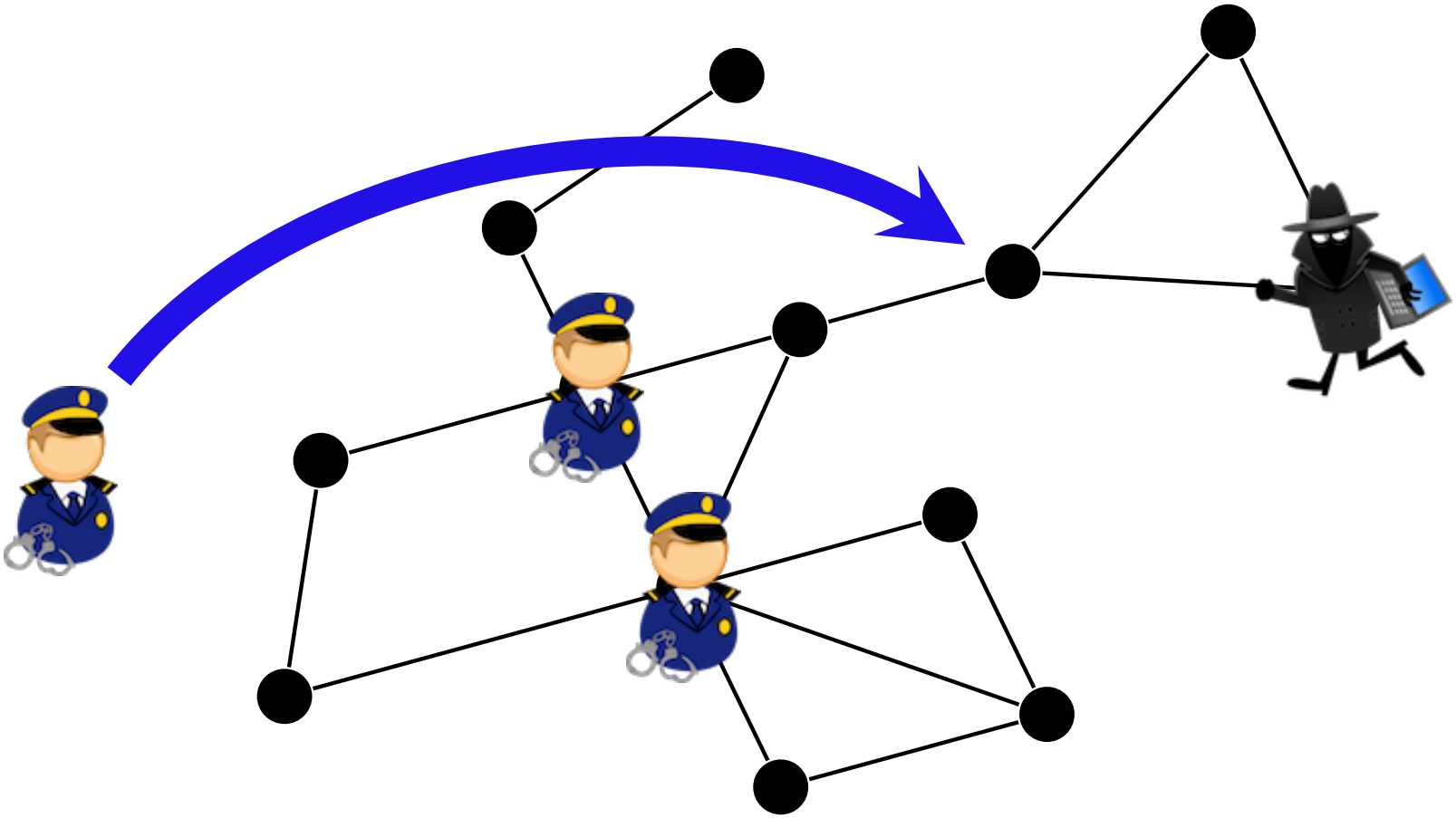
Les gendarmes et le voleur



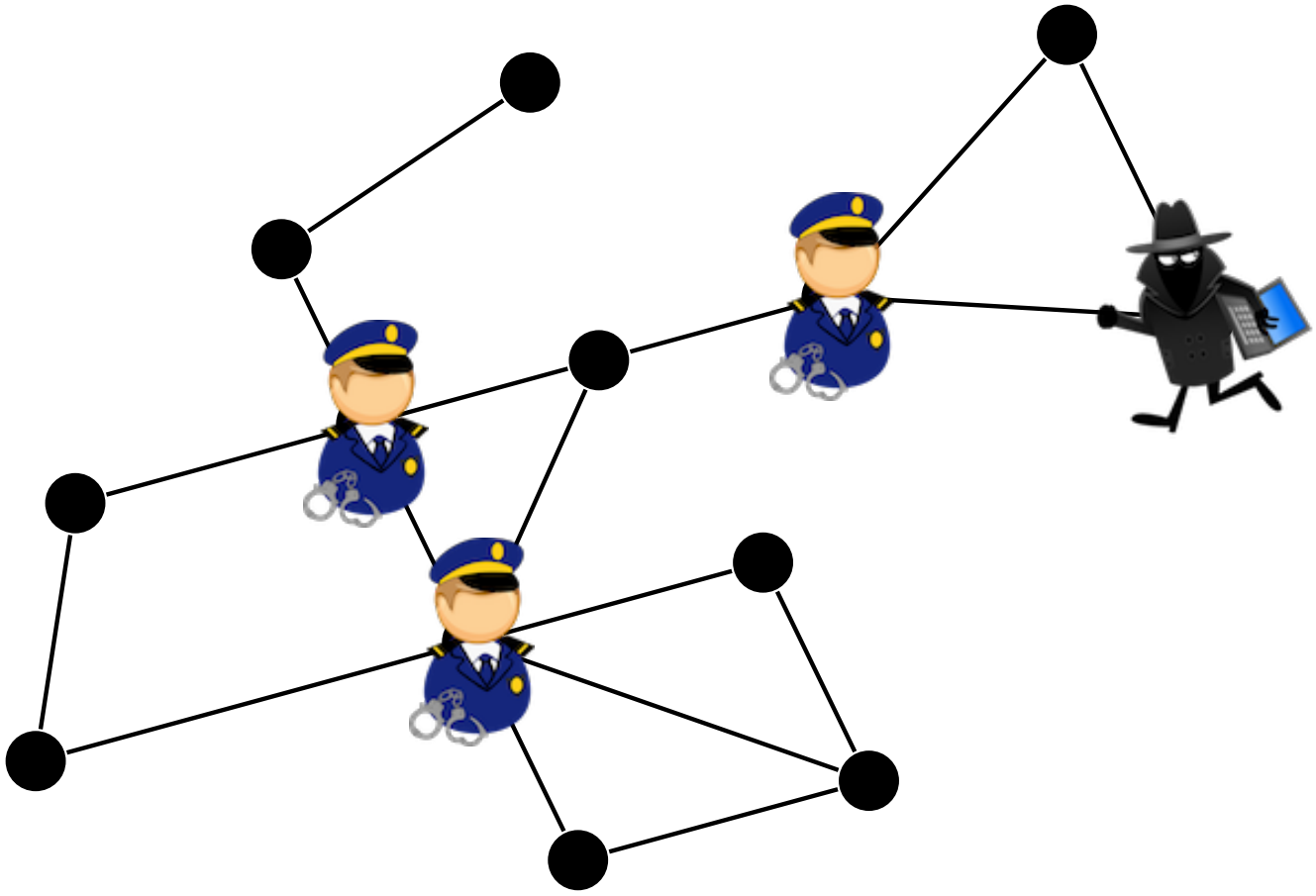
Les gendarmes et le voleur



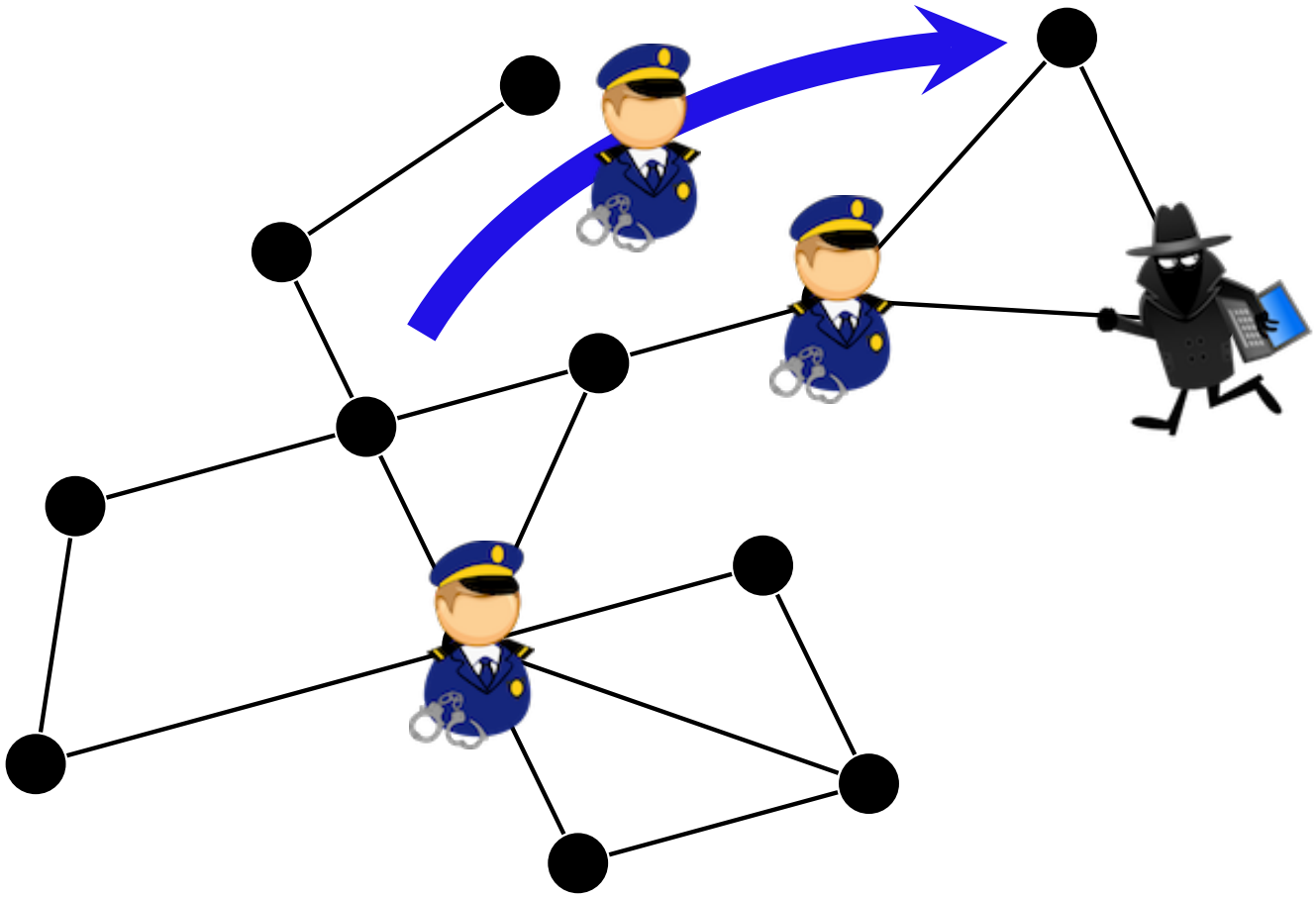
Les gendarmes et le voleur



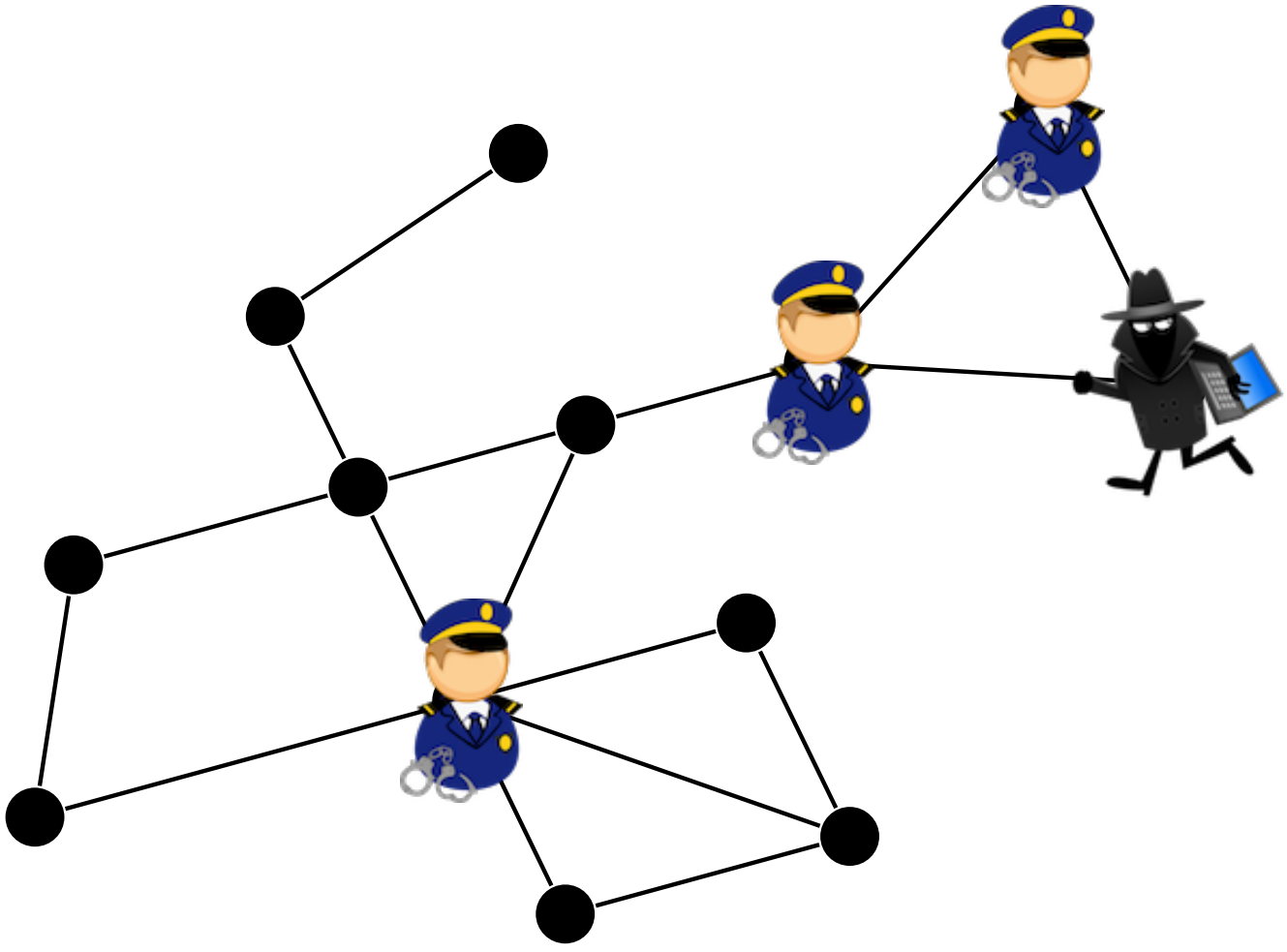
Les gendarmes et le voleur



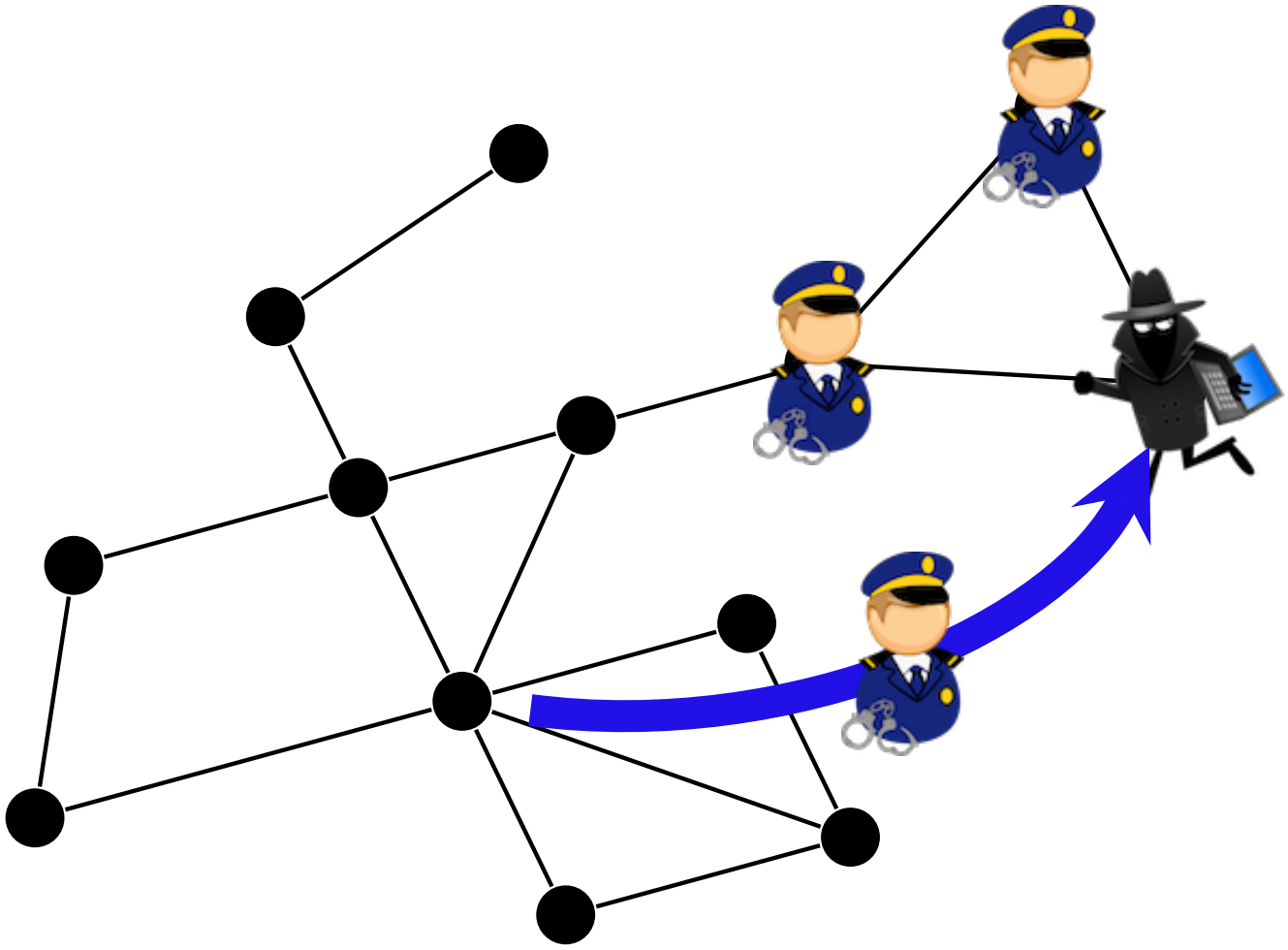
Les gendarmes et le voleur



Les gendarmes et le voleur

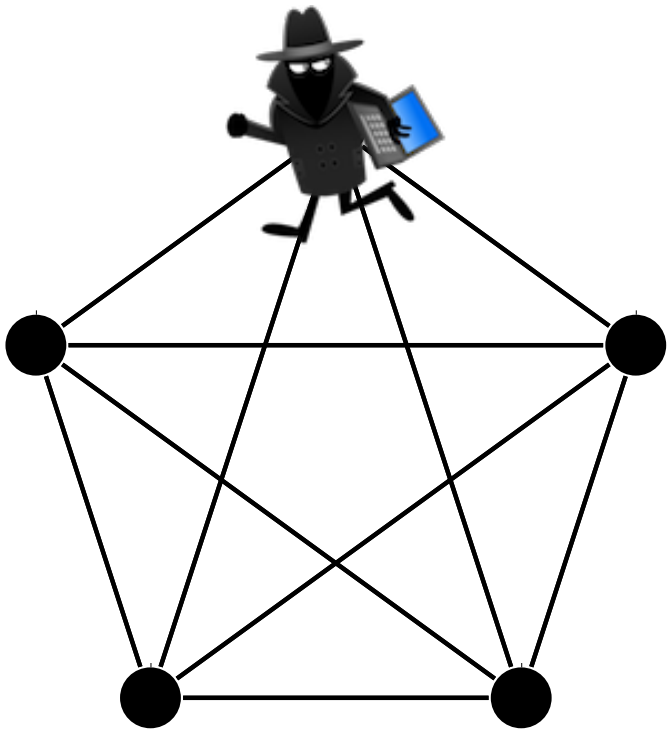


Les gendarmes et le voleur

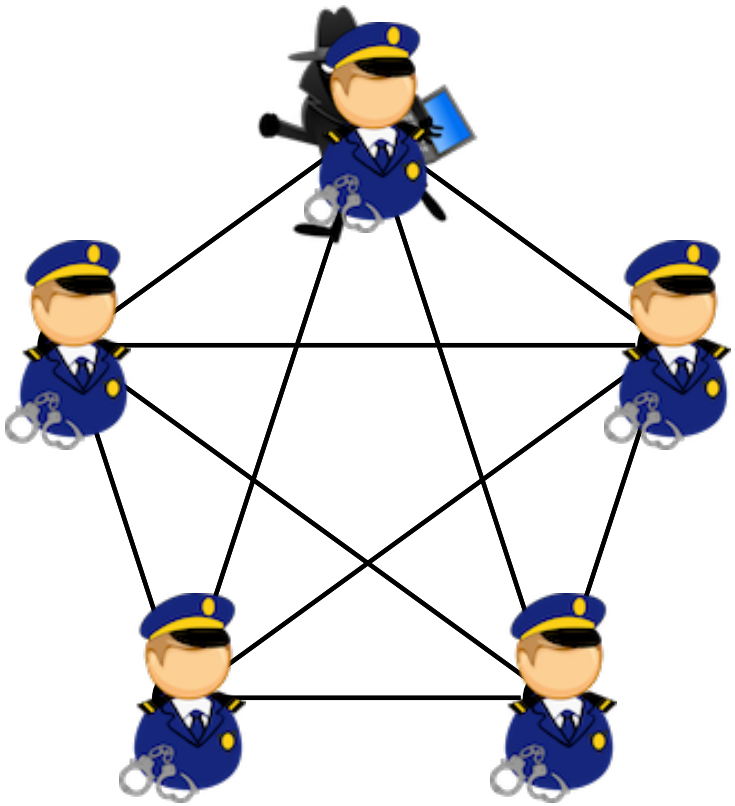


Les gendarmes et le voleur

Combien faut-il de gendarmes dans une clique ?

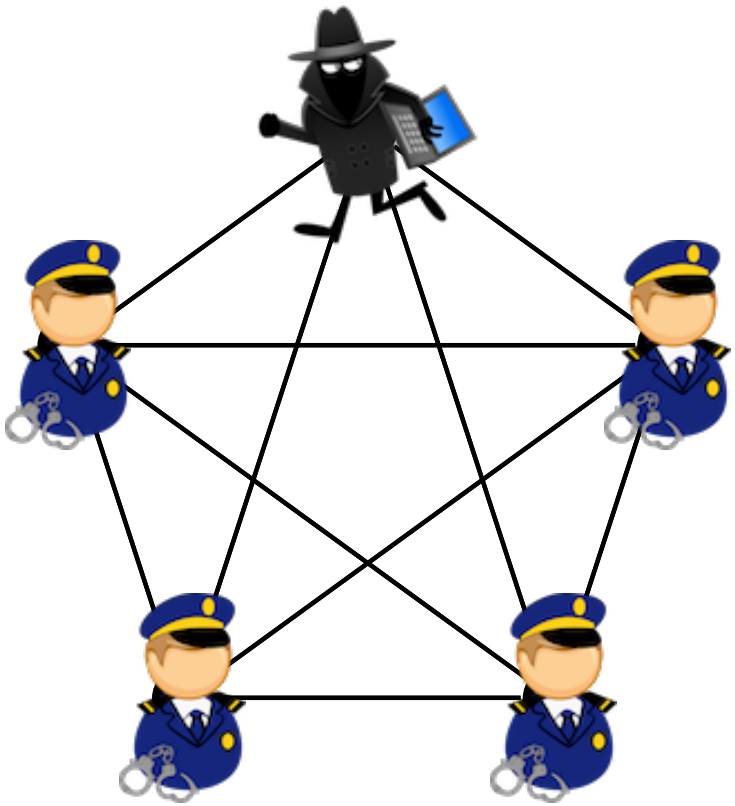


Combien faut-il de gendarmes dans une clique ?



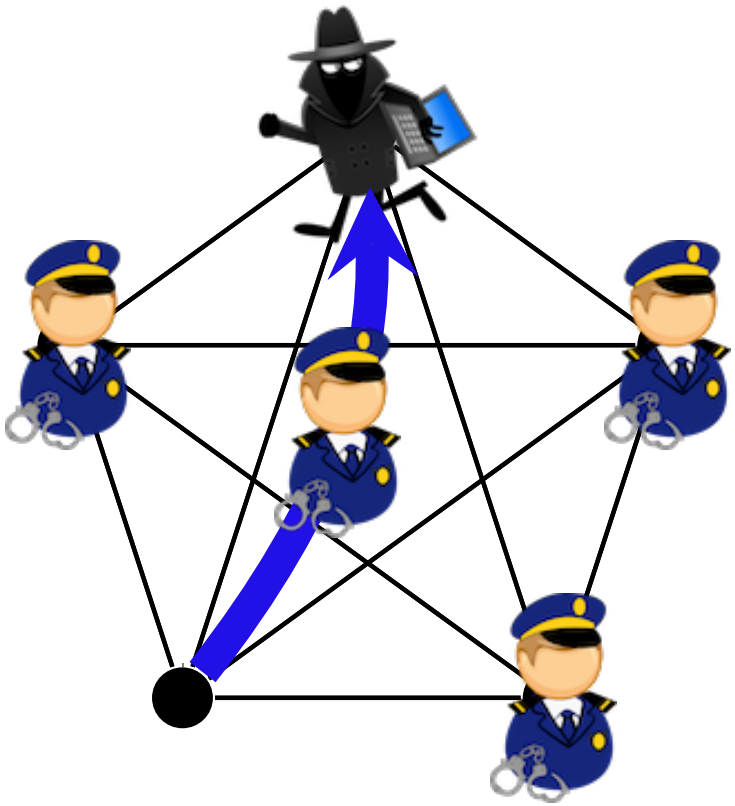
- n ✓

Combien faut-il de gendarmes dans une clique ?



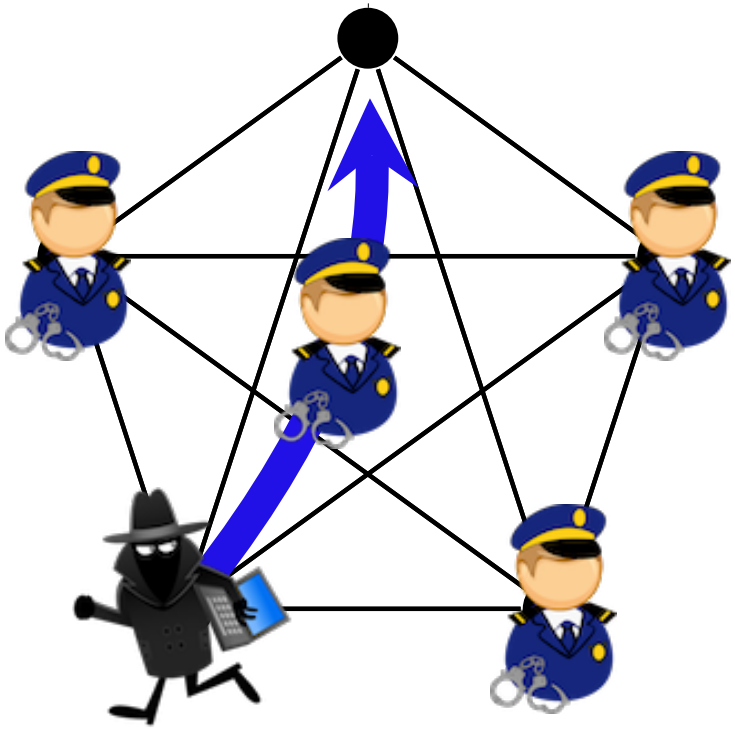
- n ✓
- $n-1$?

Combien faut-il de gendarmes dans une clique ?



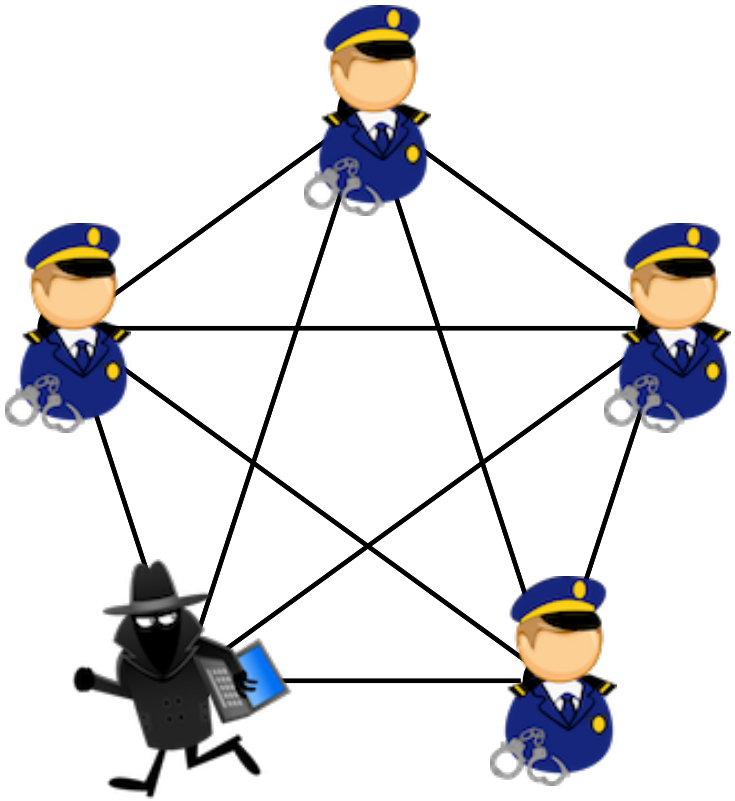
- n ✓
- $n-1$?

Combien faut-il de gendarmes dans une clique ?



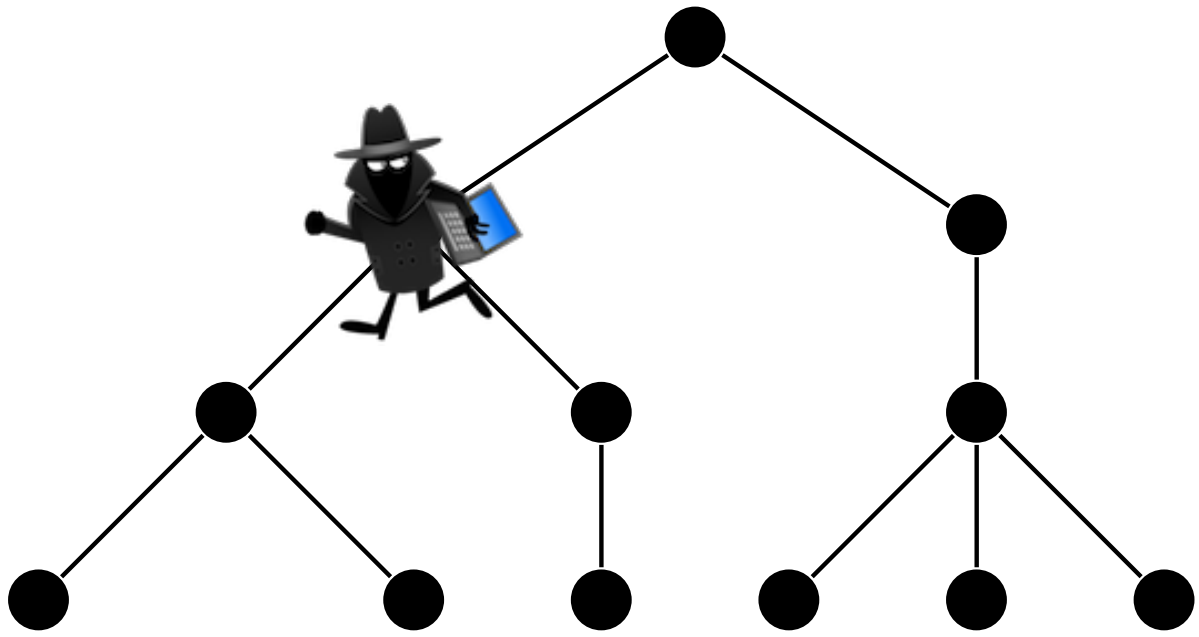
- n ✓
- $n-1$?

Combien faut-il de gendarmes dans une clique ?

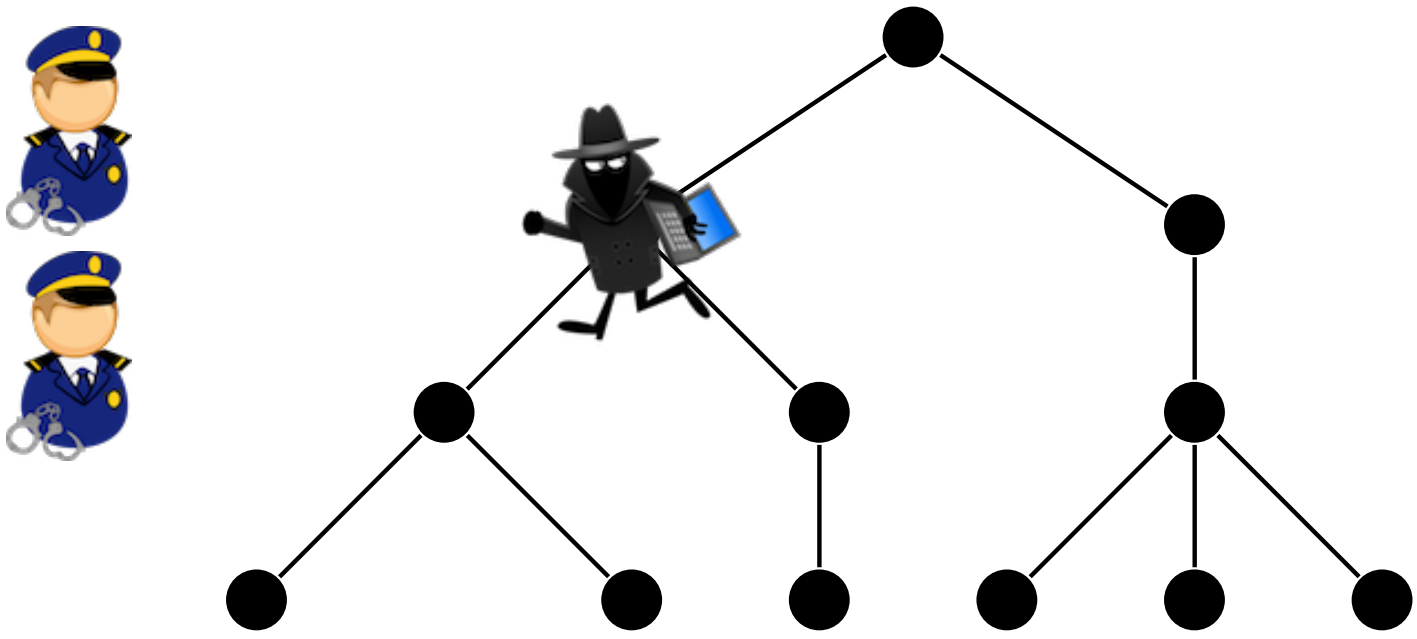


- n ✓
- $n-1$? ✗

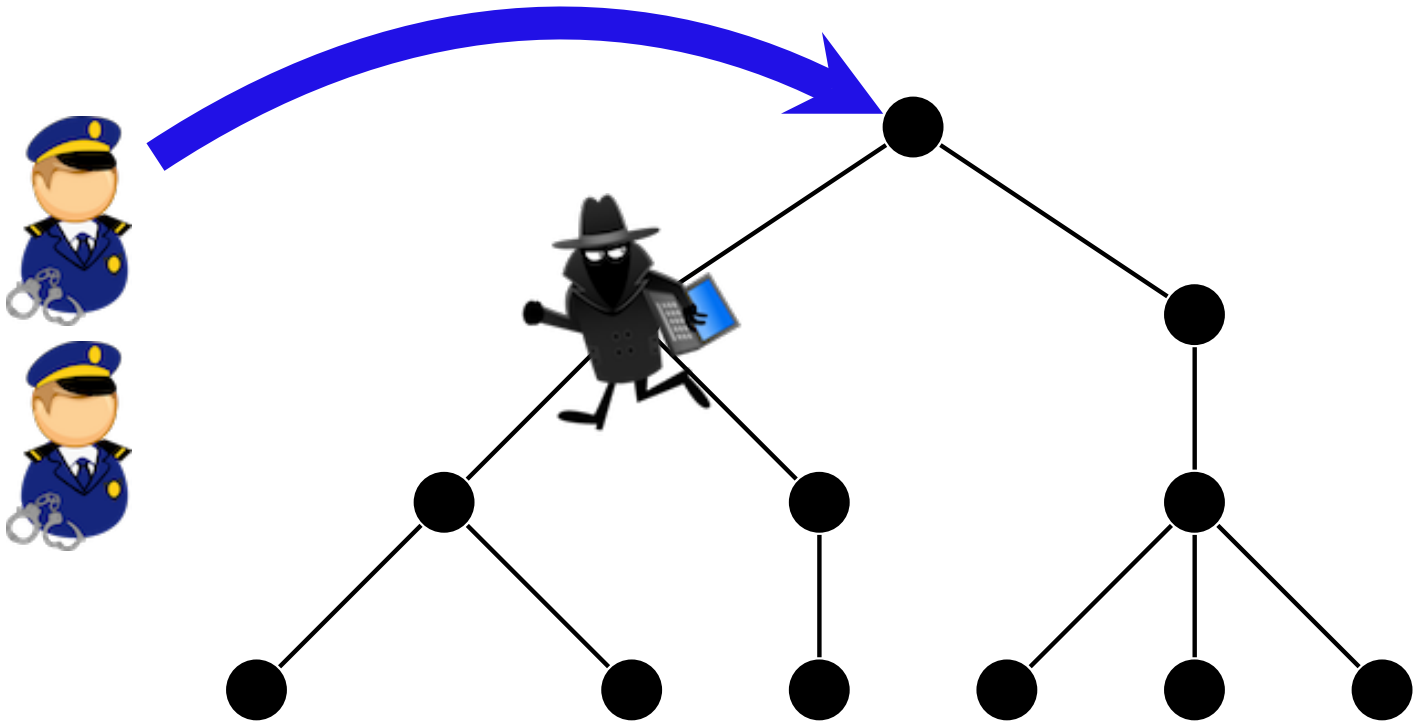
Et dans un arbre ?



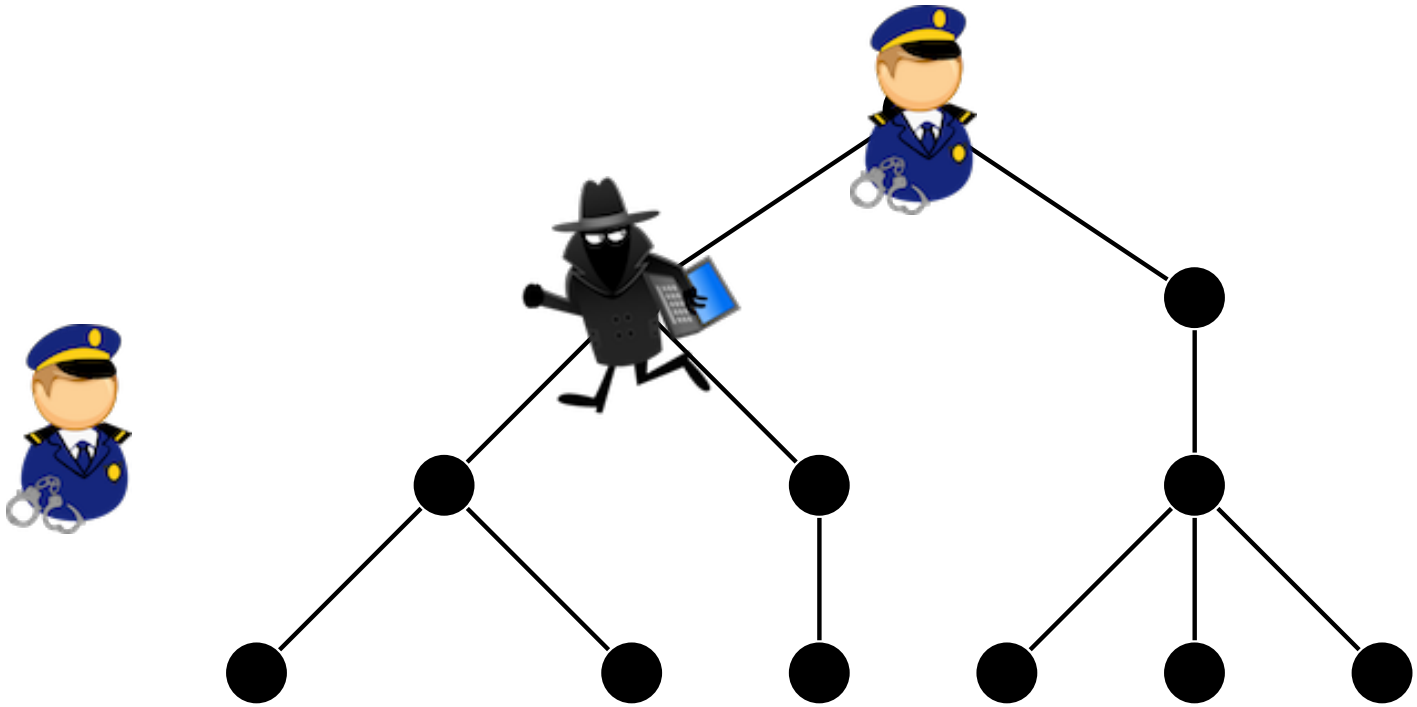
Et dans un arbre ?



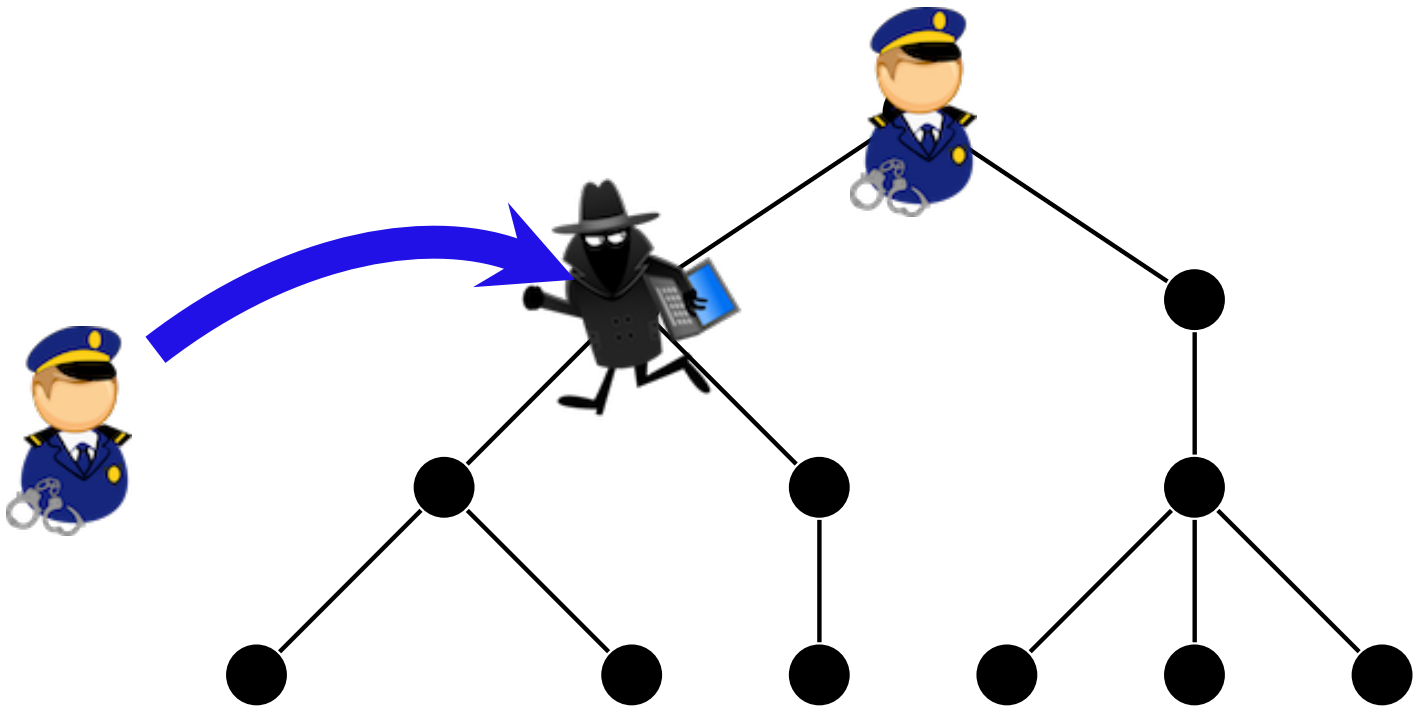
Et dans un arbre ?



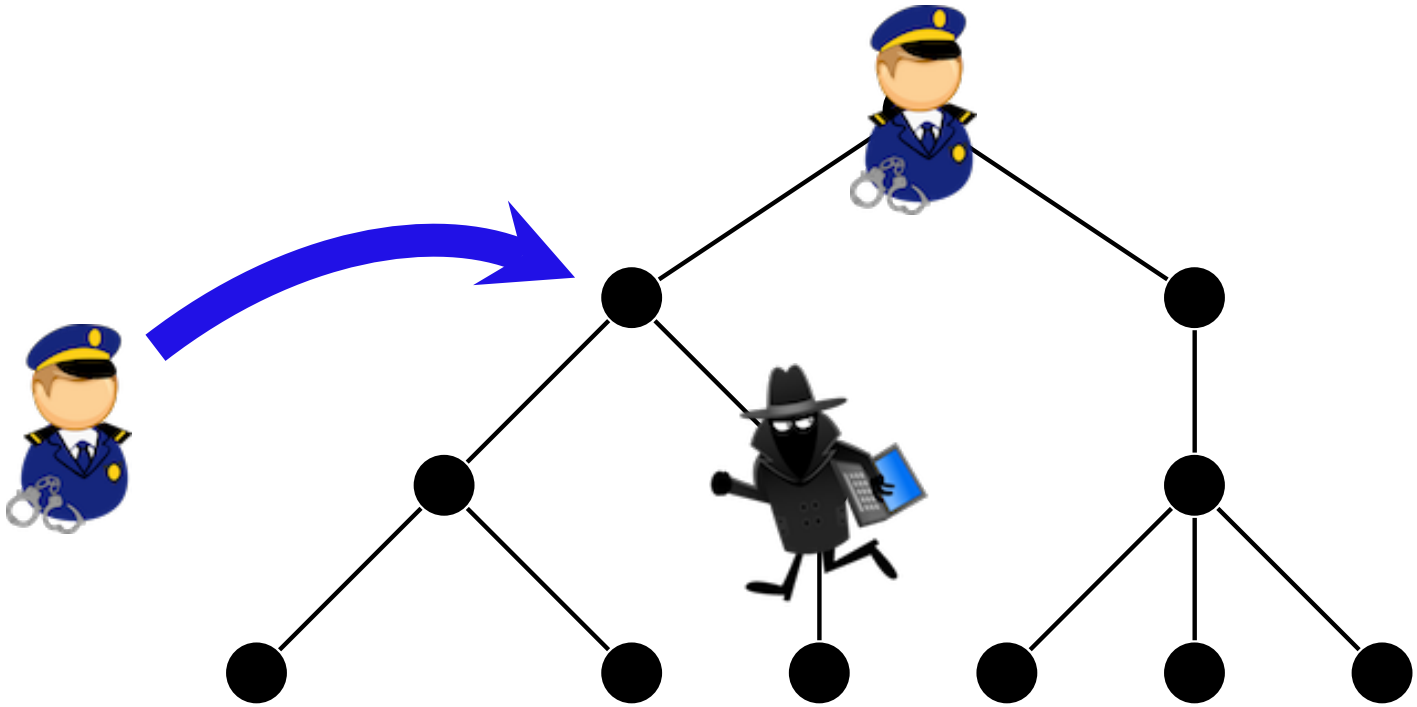
Et dans un arbre ?



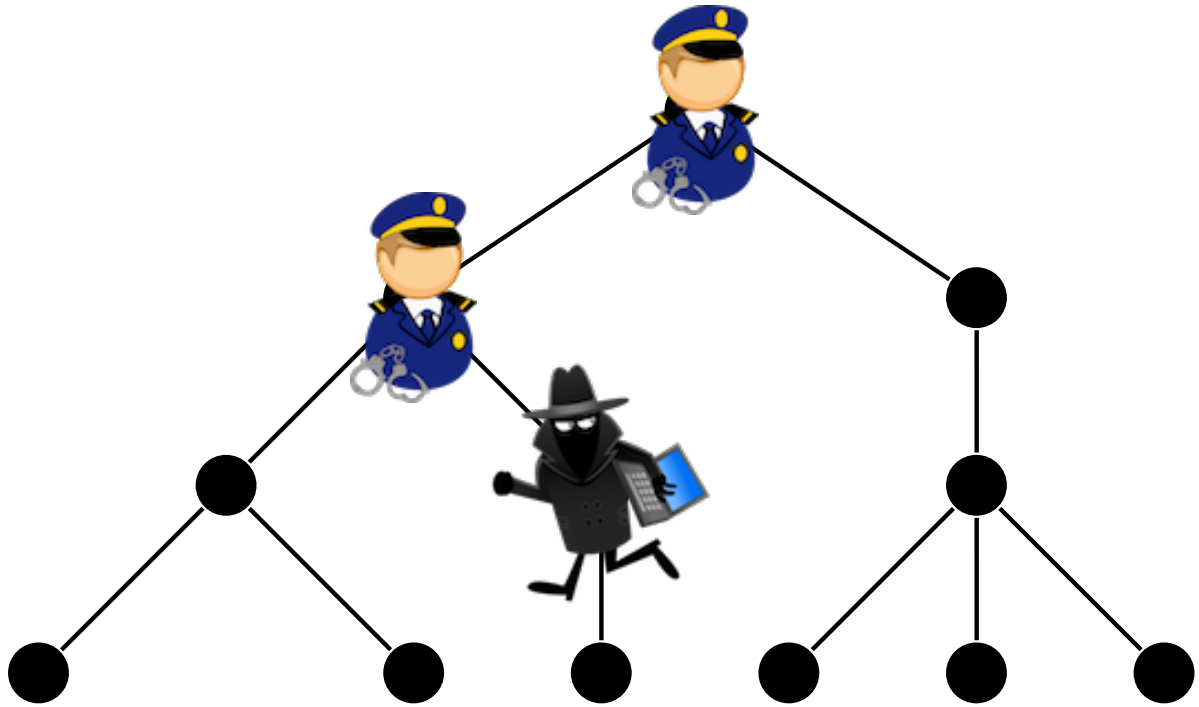
Et dans un arbre ?



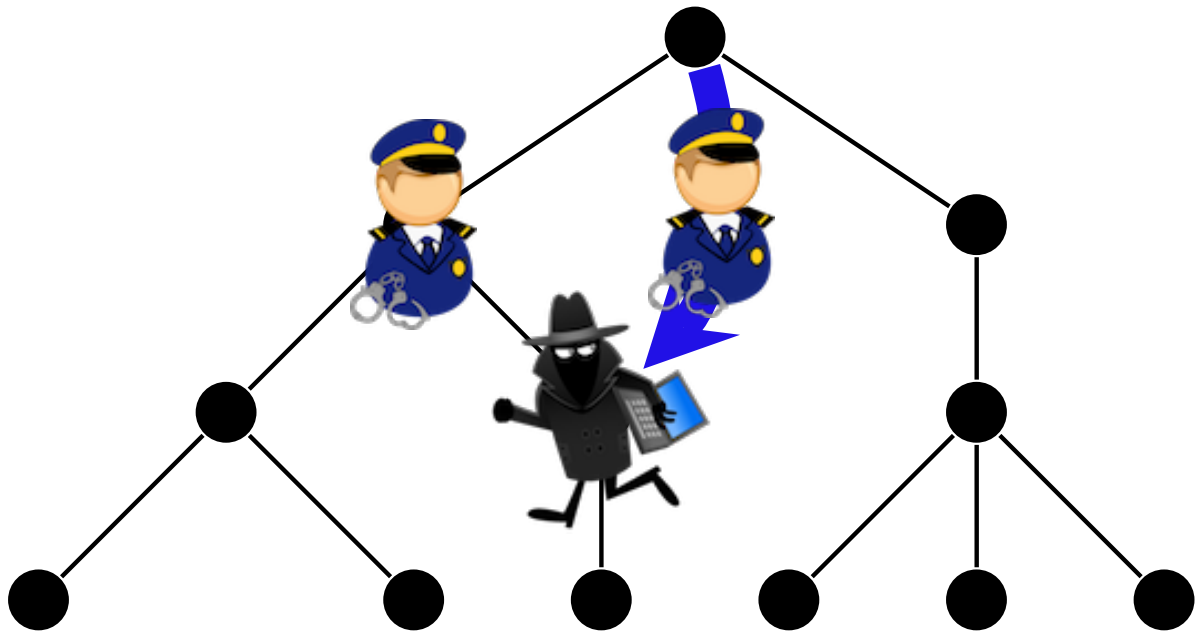
Et dans un arbre ?



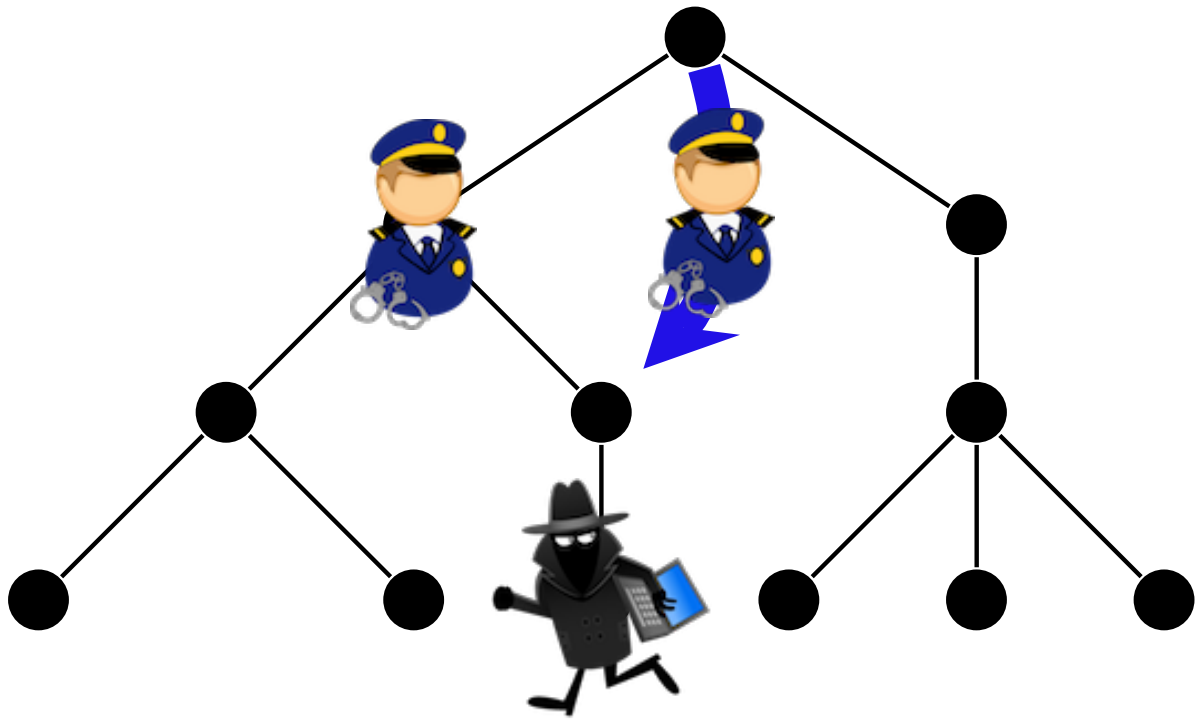
Et dans un arbre ?



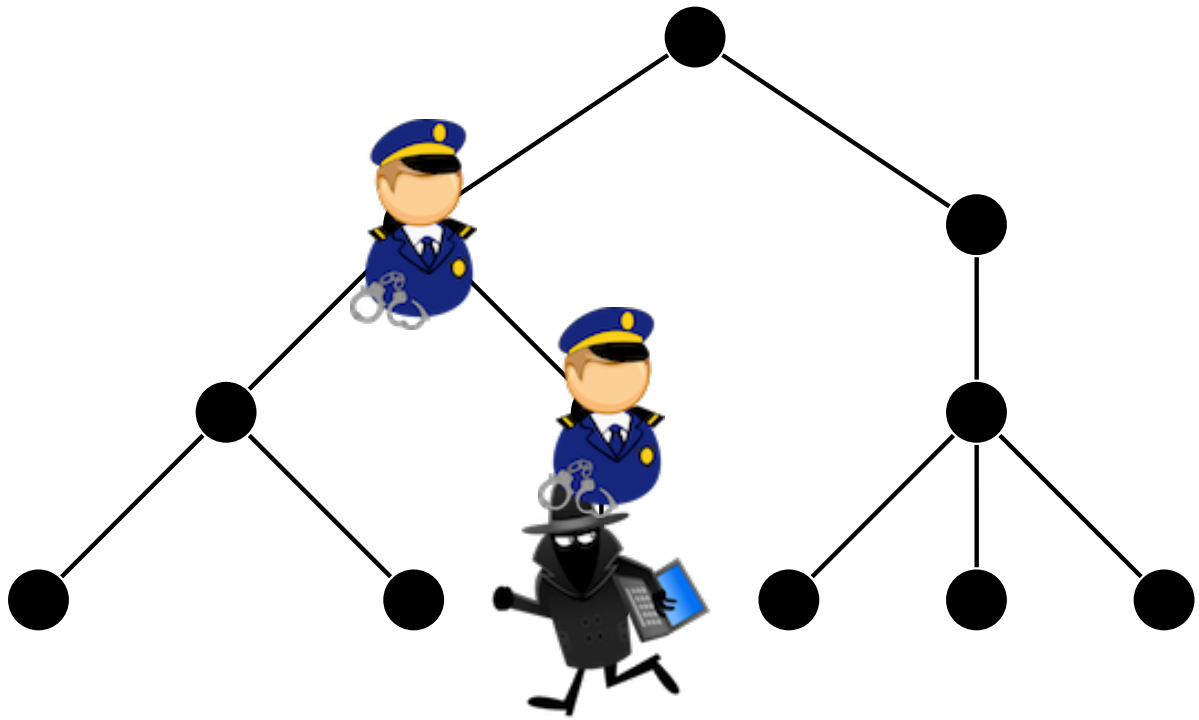
Et dans un arbre ?



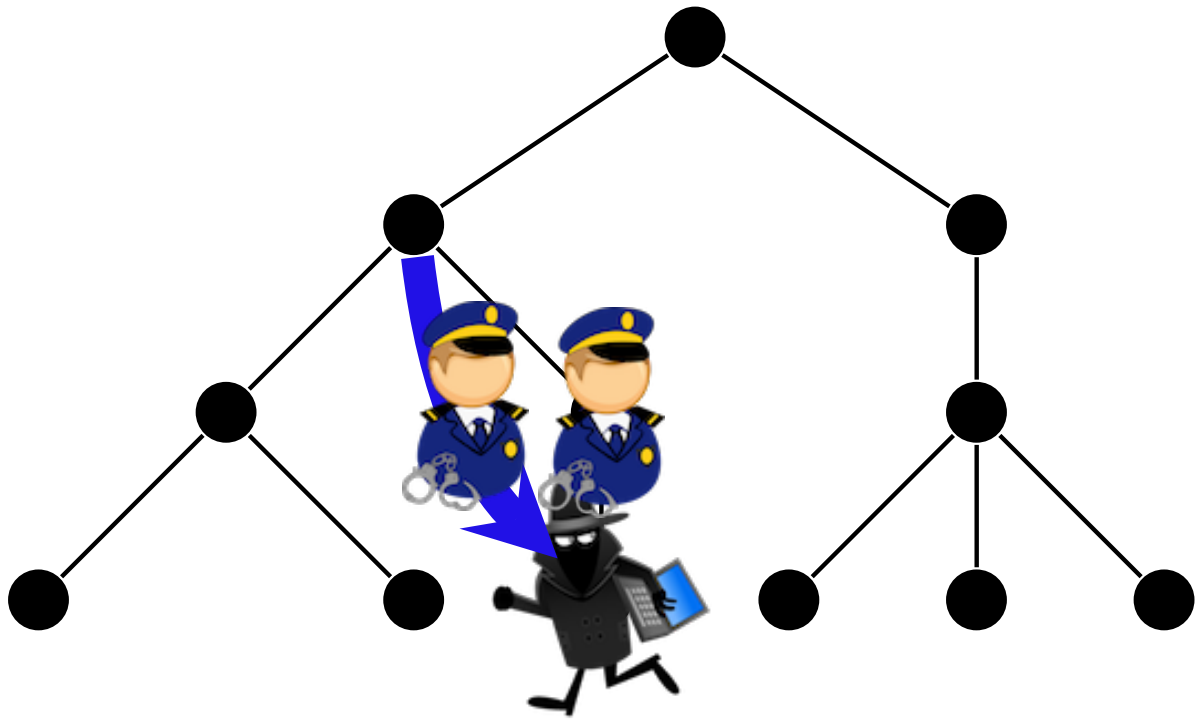
Et dans un arbre ?



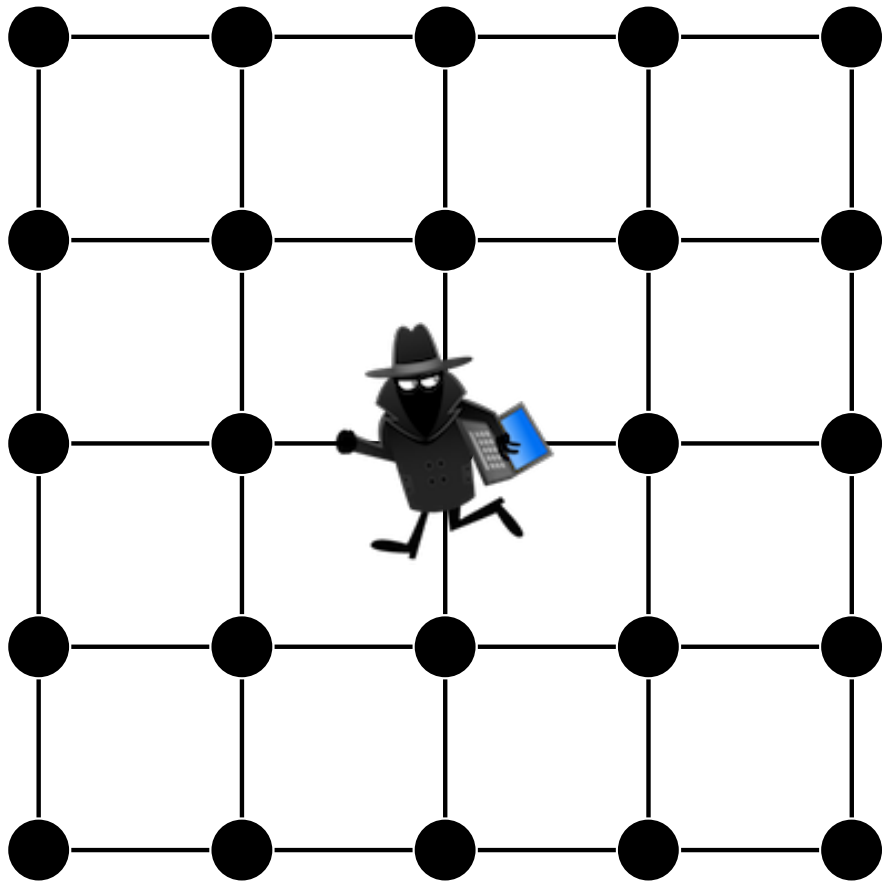
Et dans un arbre ?



Et dans un arbre ?

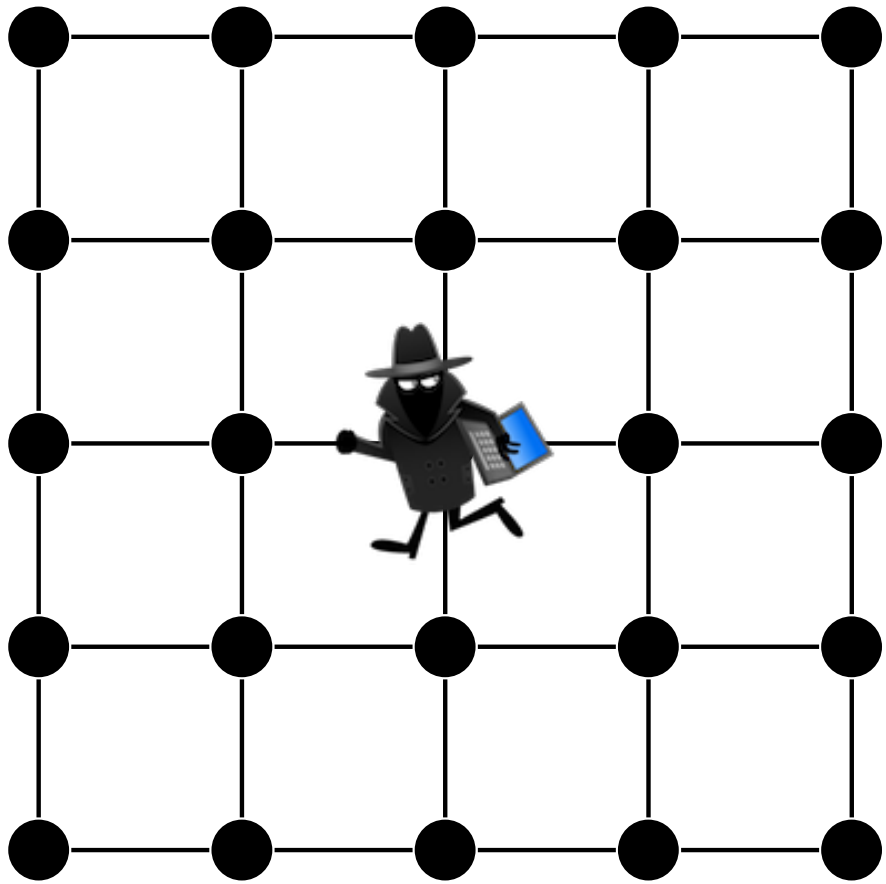


Grille $n \times n$

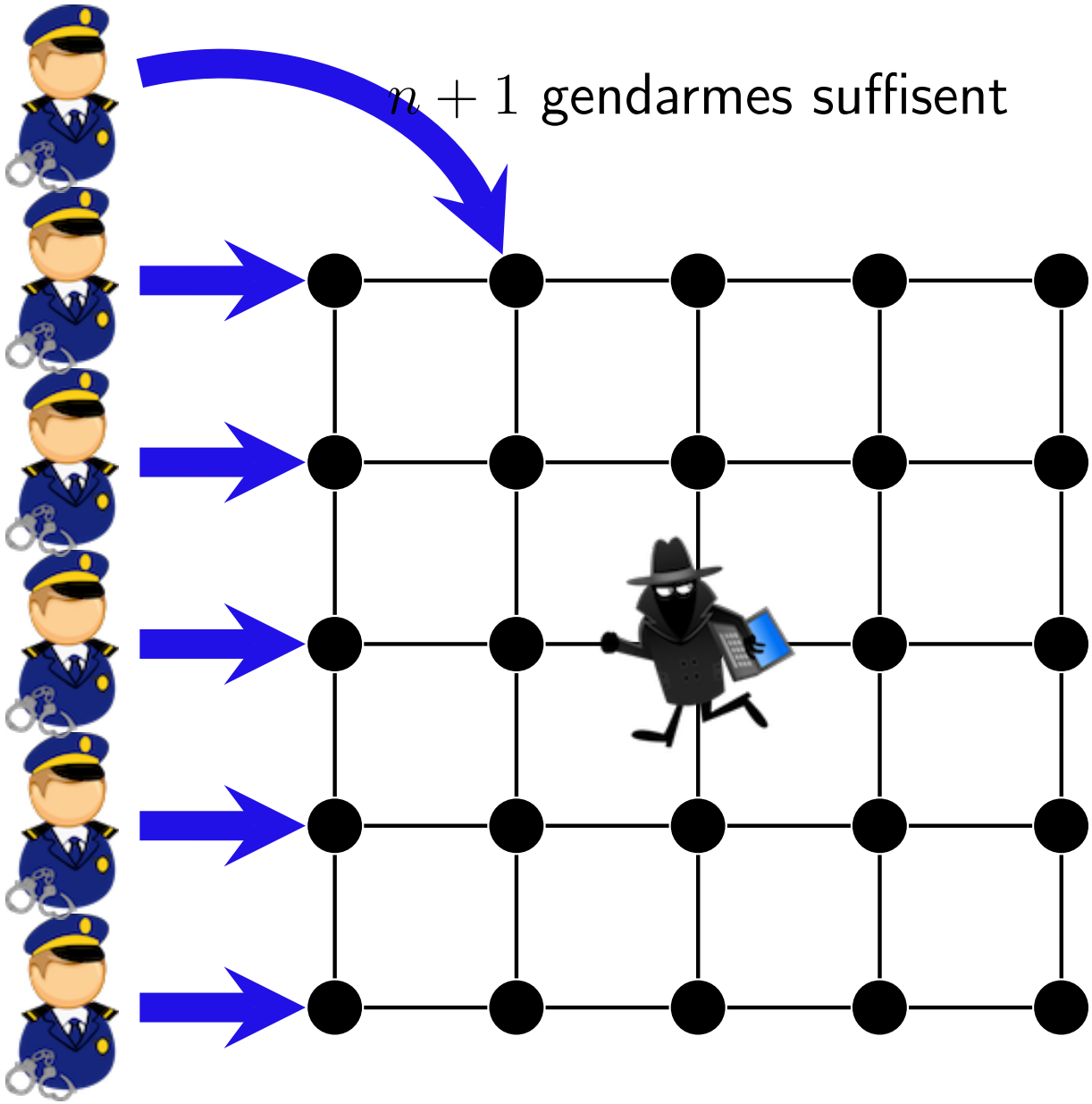


Grille $n \times n$

$n + 1$ gendarmes suffisent

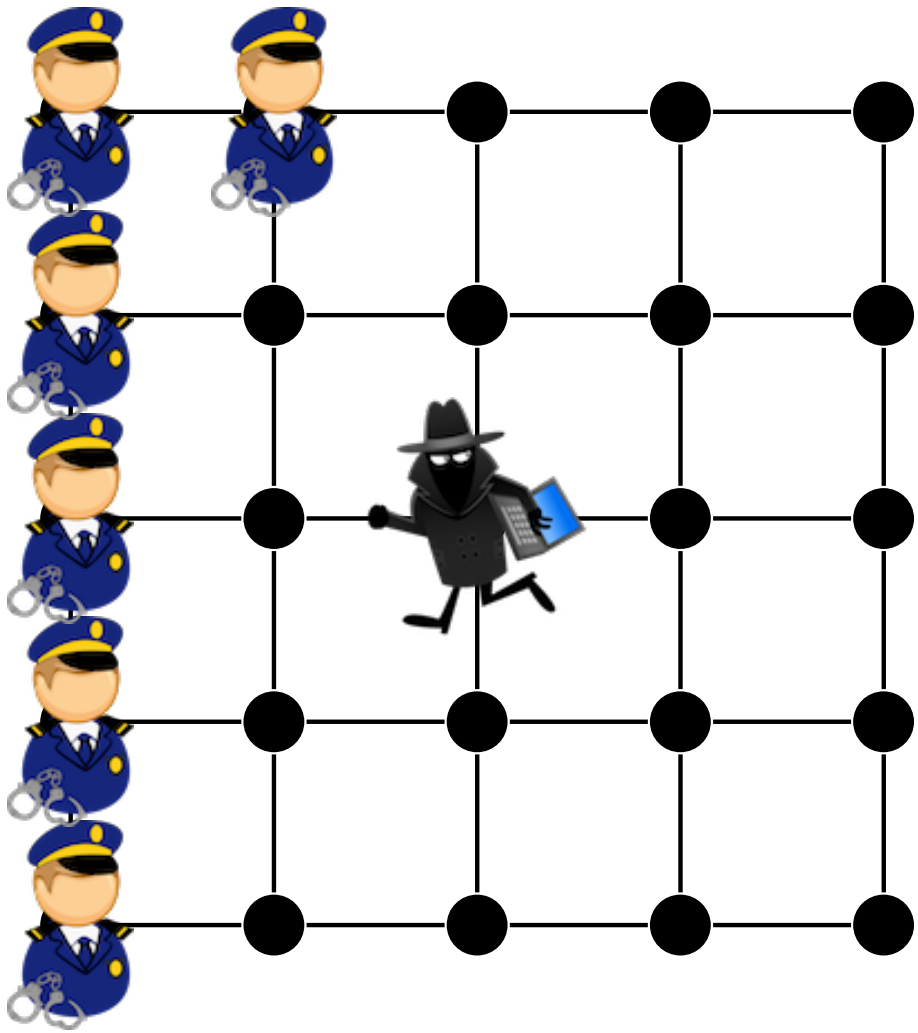


Grille $n \times n$



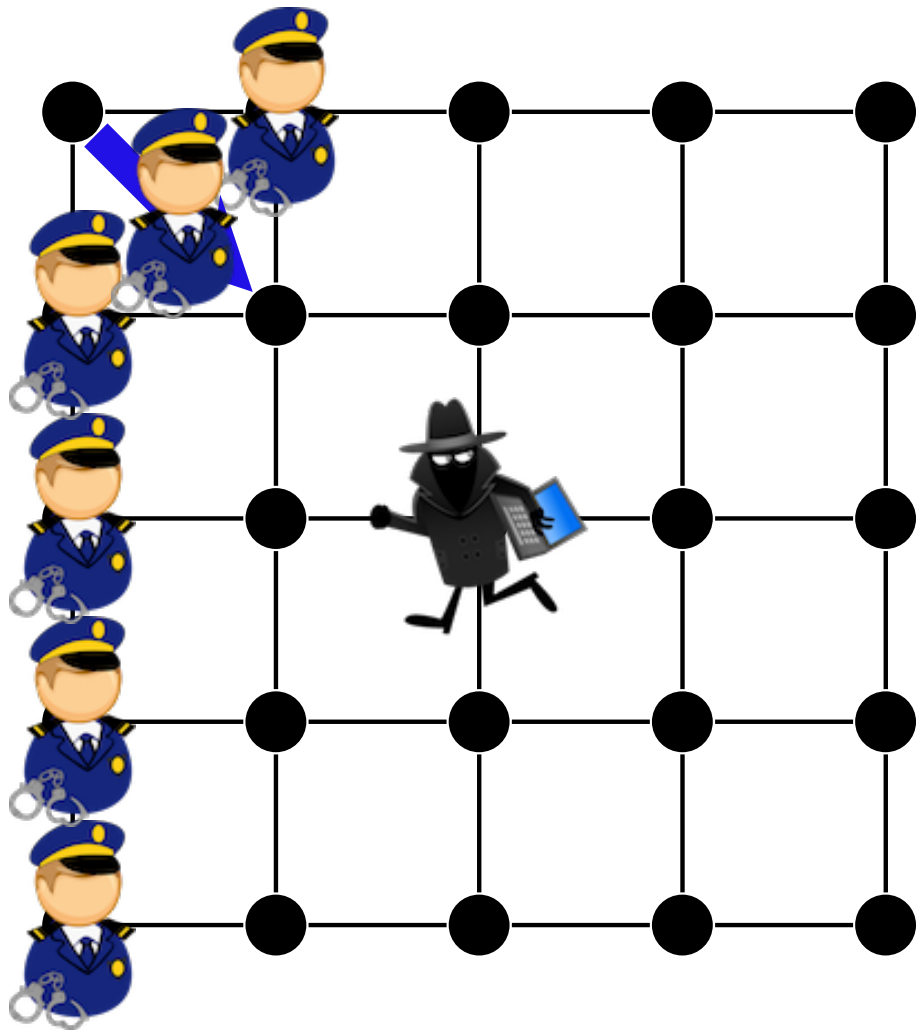
Grille $n \times n$

$n + 1$ gendarmes suffisent



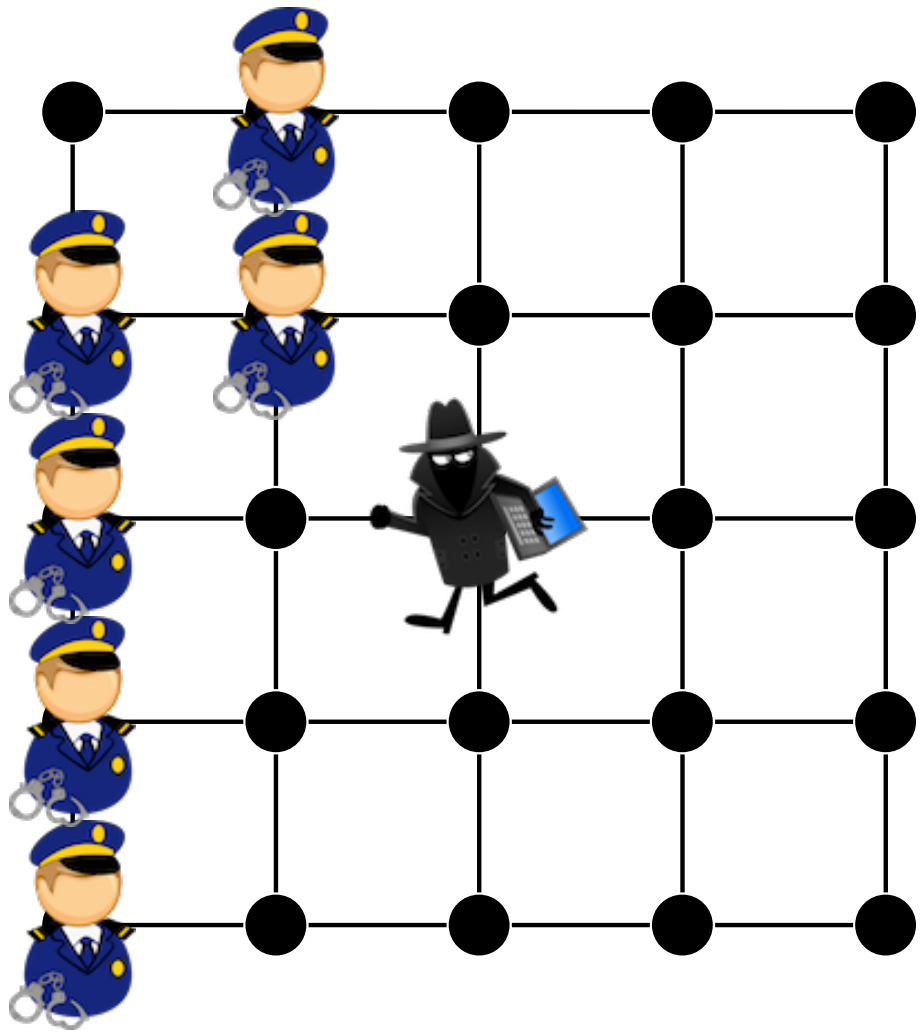
Grille $n \times n$

$n + 1$ gendarmes suffisent



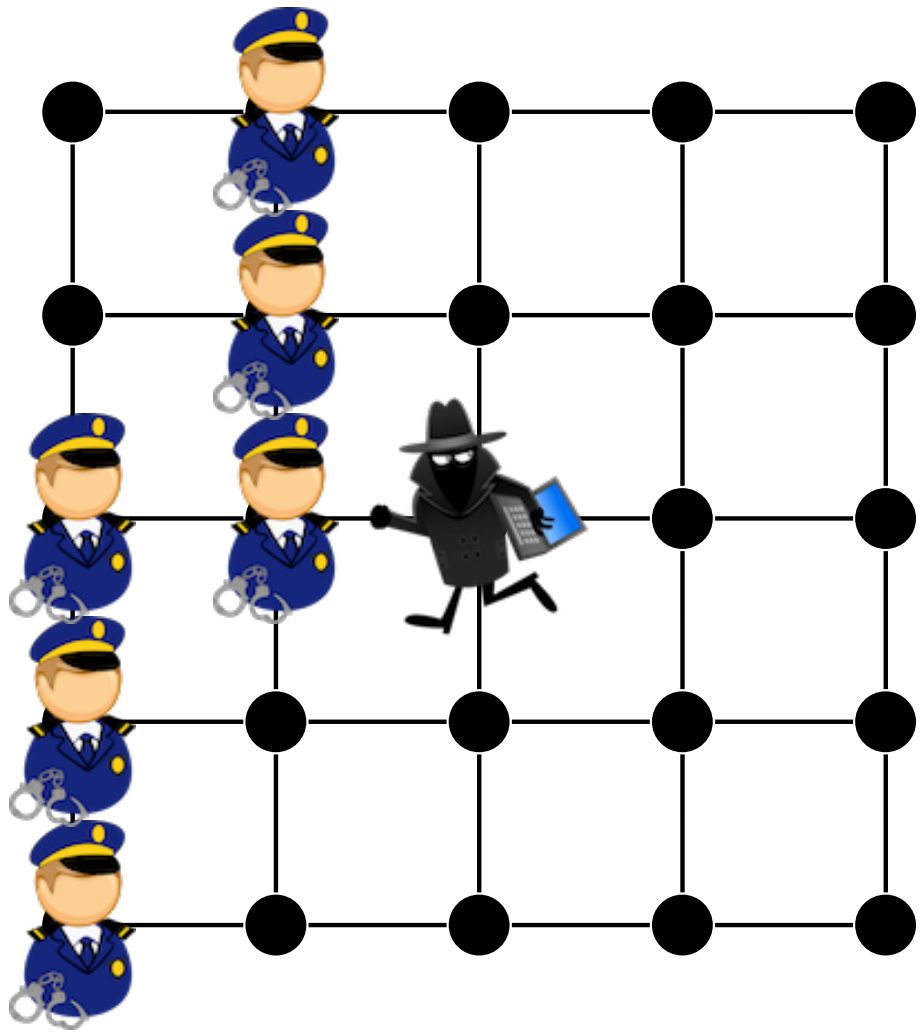
Grille $n \times n$

$n + 1$ gendarmes suffisent



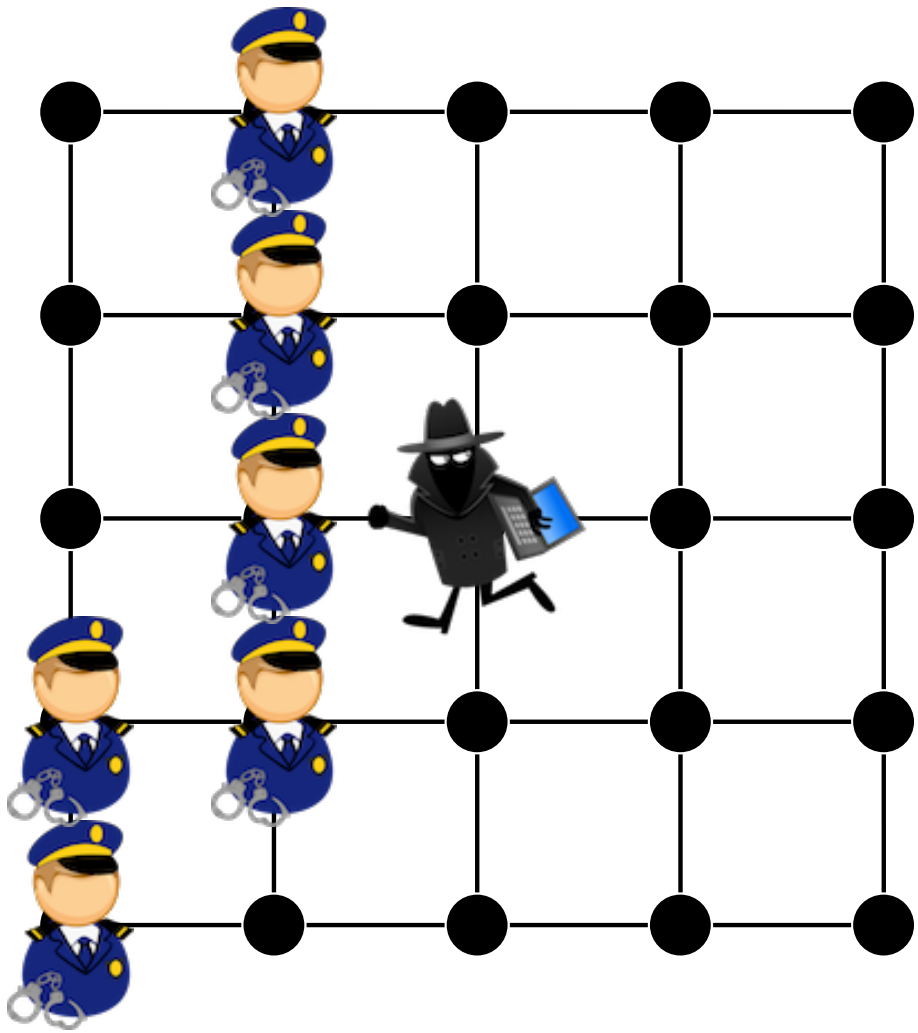
Grille $n \times n$

$n + 1$ gendarmes suffisent



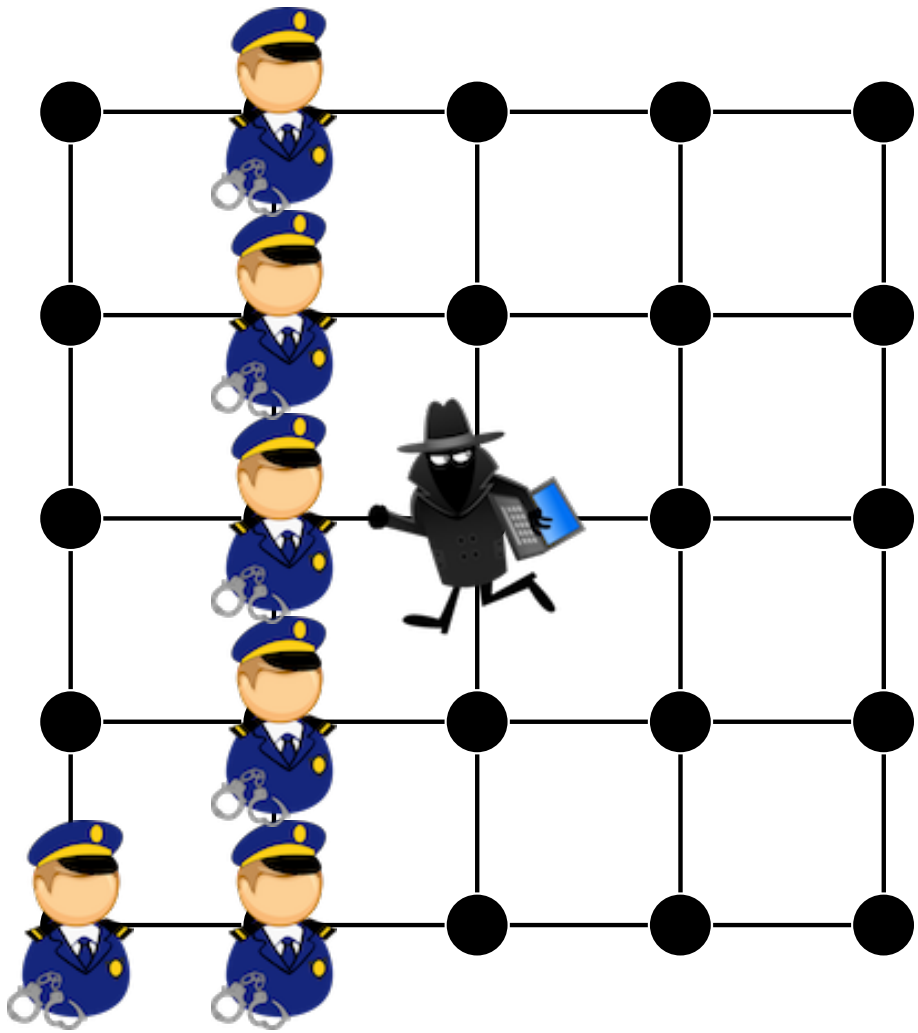
Grille $n \times n$

$n + 1$ gendarmes suffisent



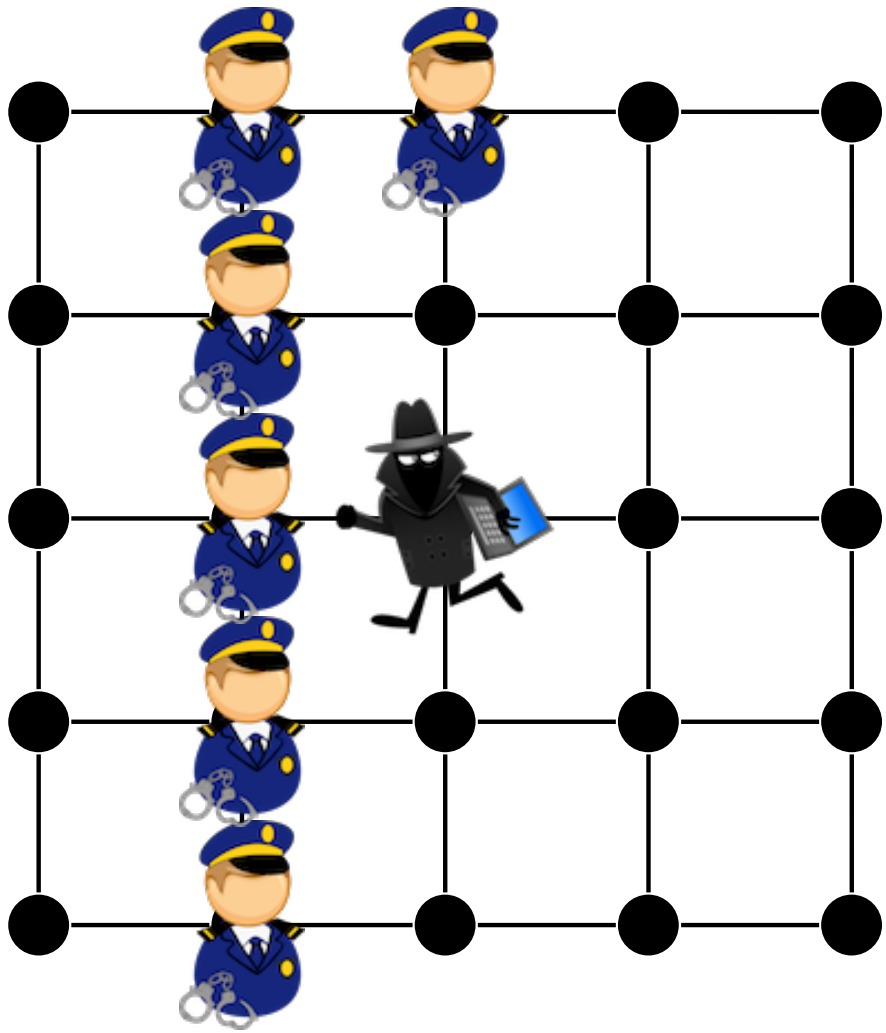
Grille $n \times n$

$n + 1$ gendarmes suffisent



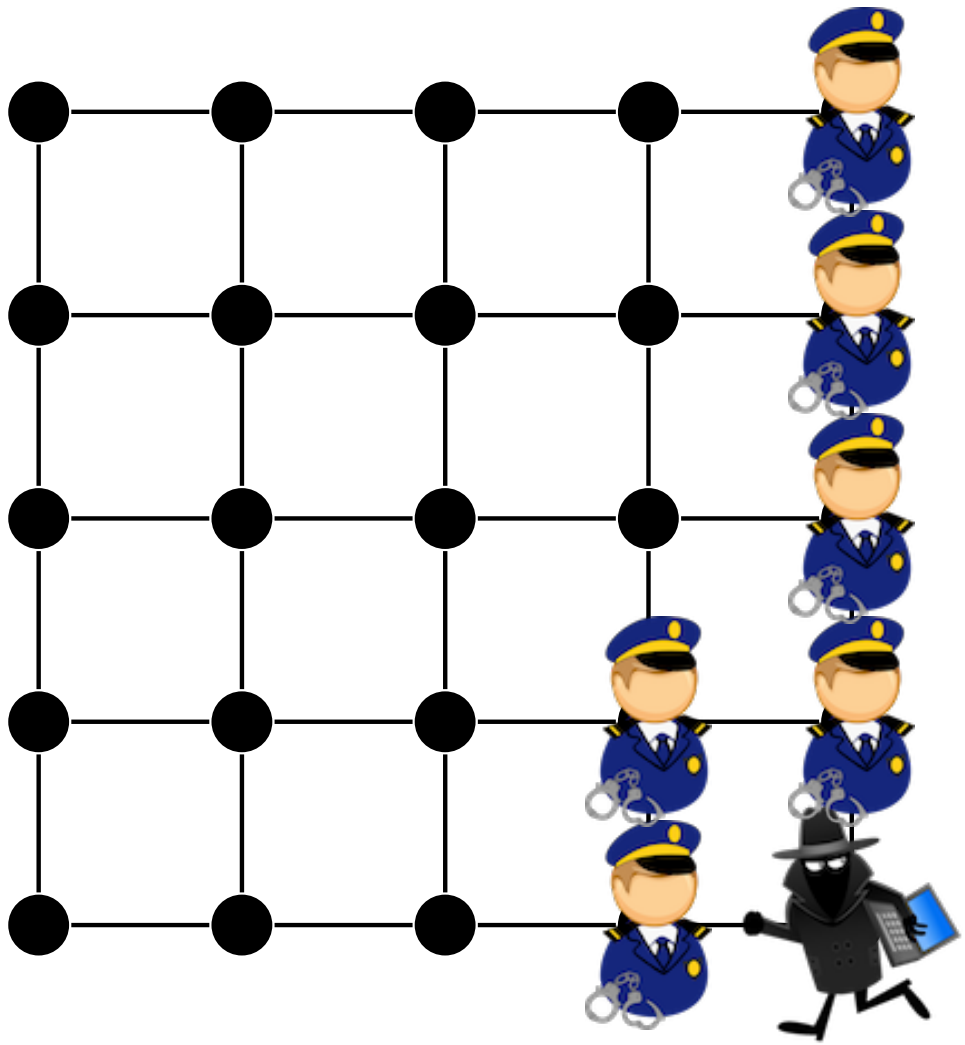
Grille $n \times n$

$n + 1$ gendarmes suffisent



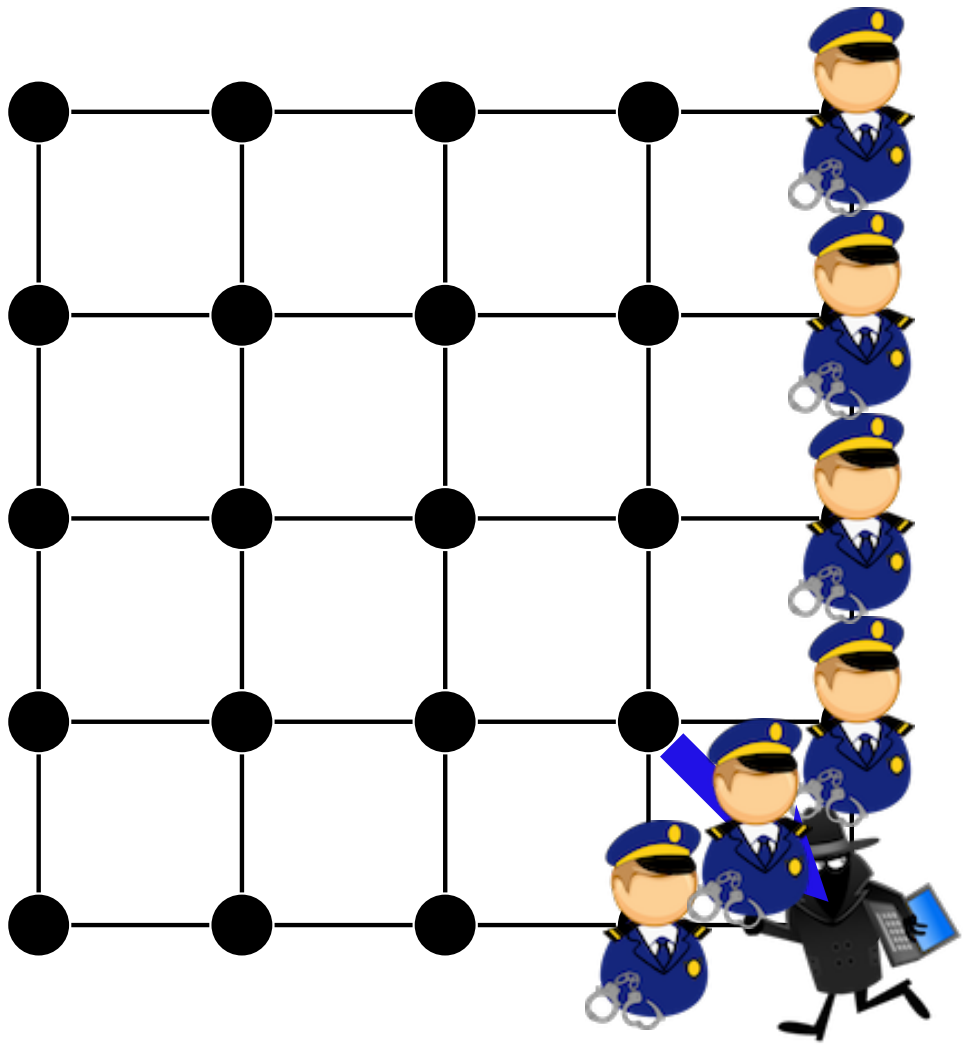
Grille $n \times n$

$n + 1$ gendarmes suffisent

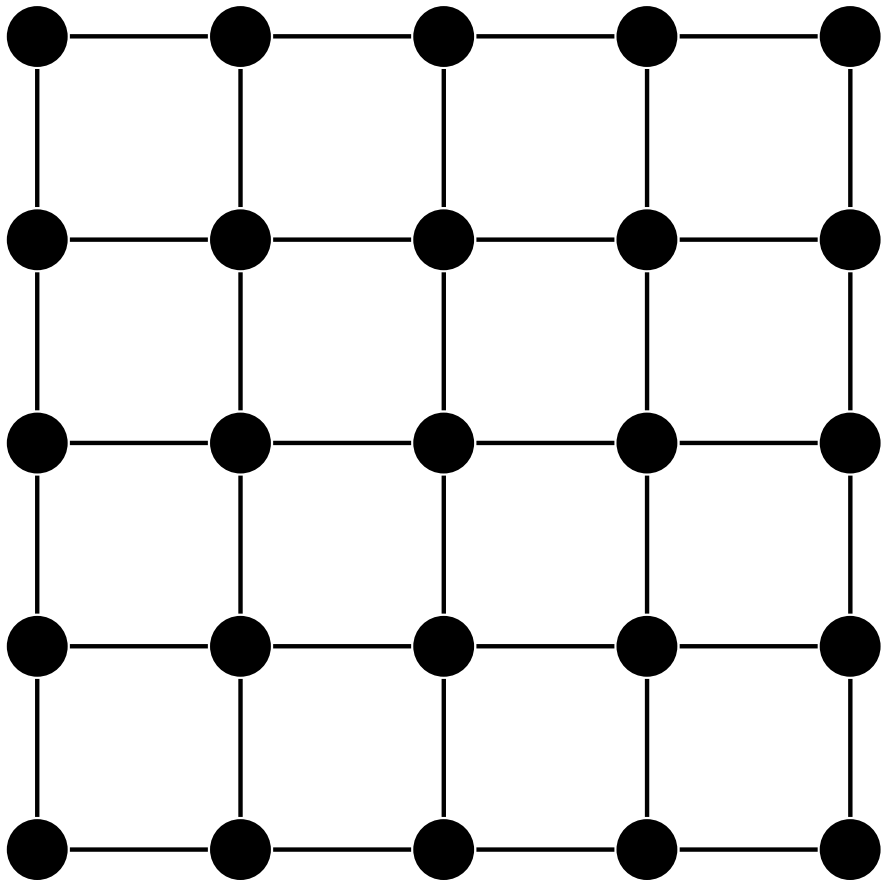


Grille $n \times n$

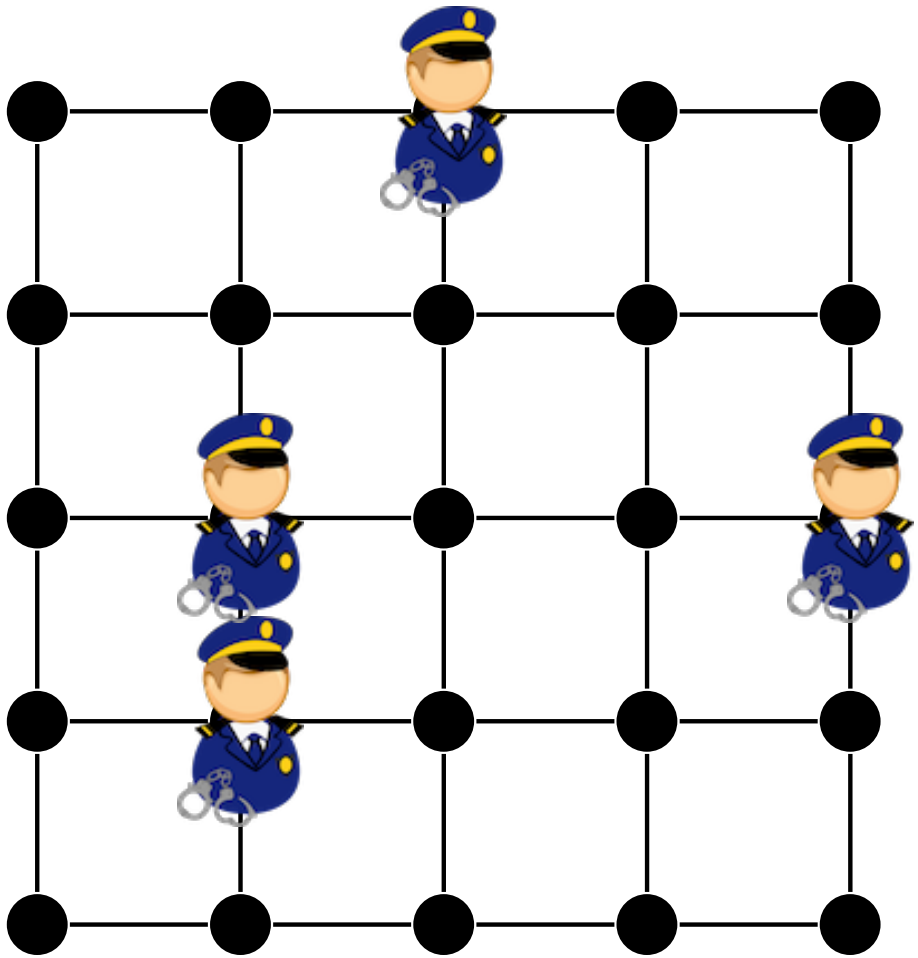
$n + 1$ gendarmes suffisent



Grille $n \times n$

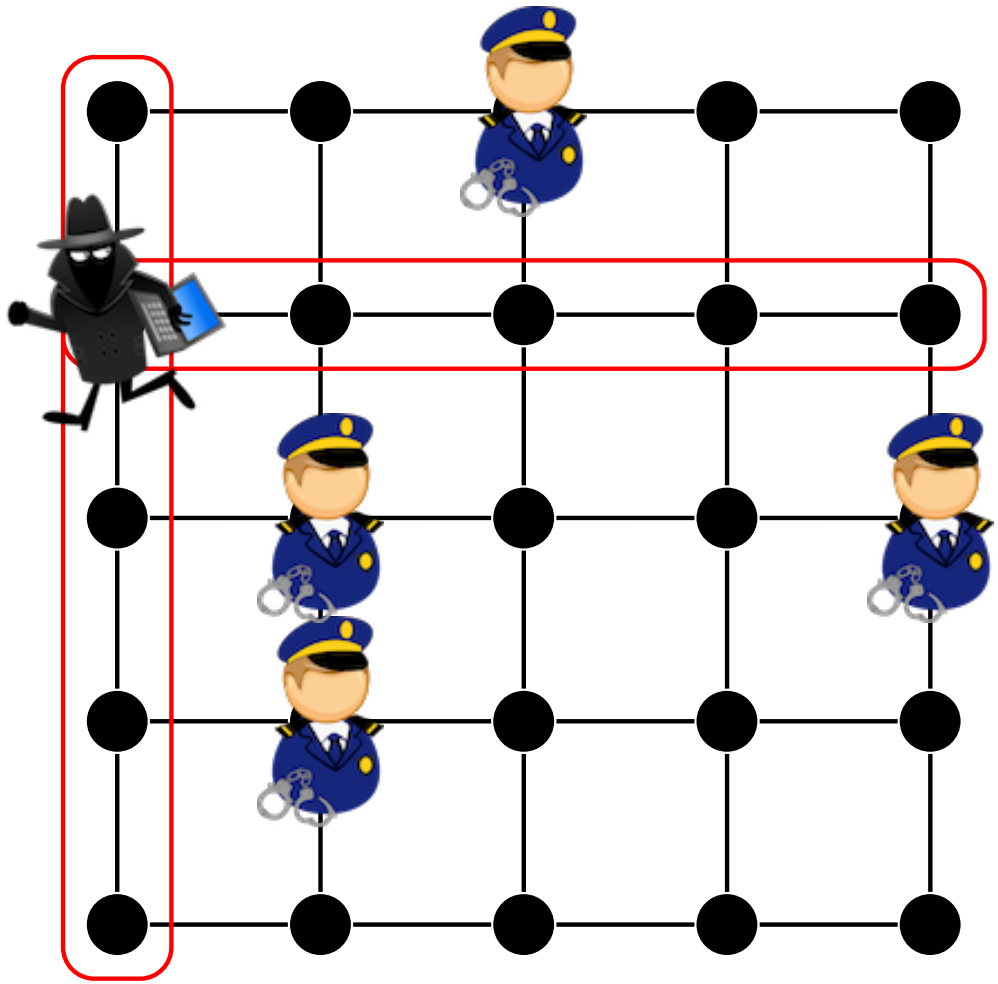


Grille $n \times n$



Grille $n \times n$

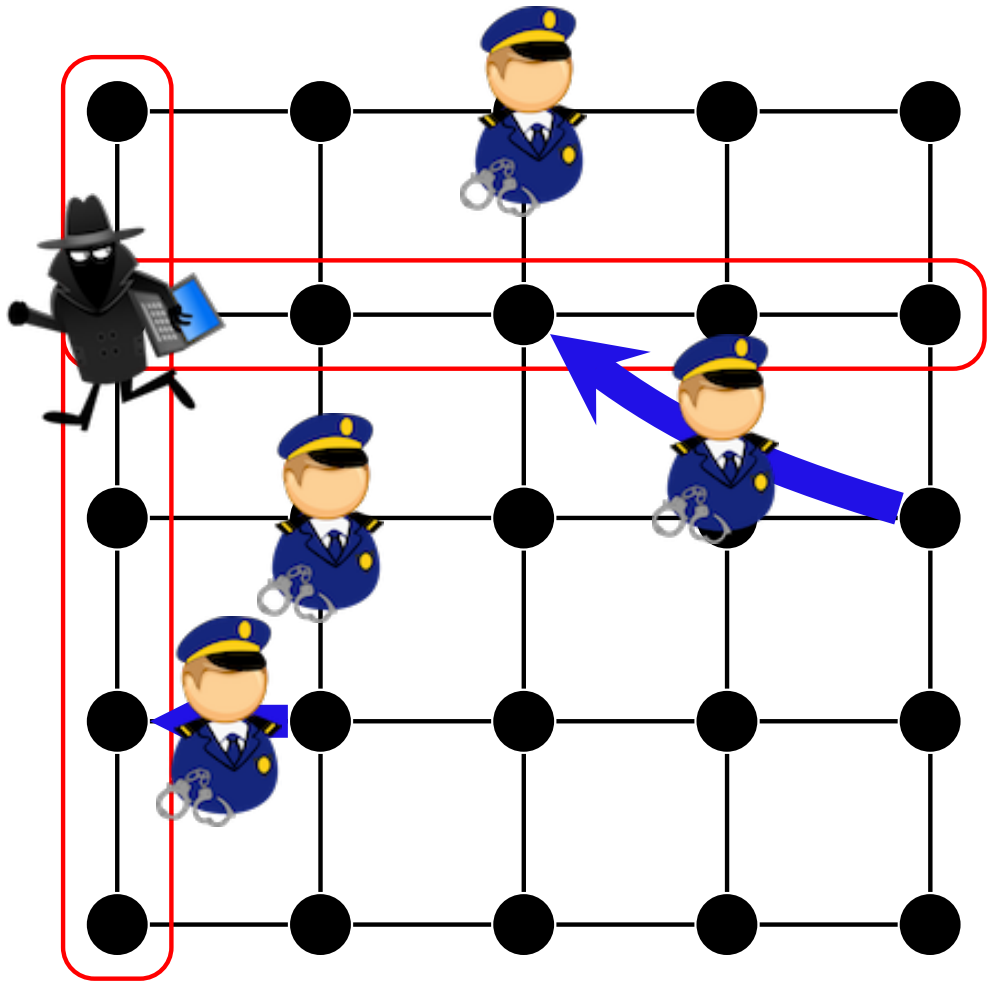
mais pas $n - 1$!



Invariant: au moins une ligne + une colonne libres

Grille $n \times n$

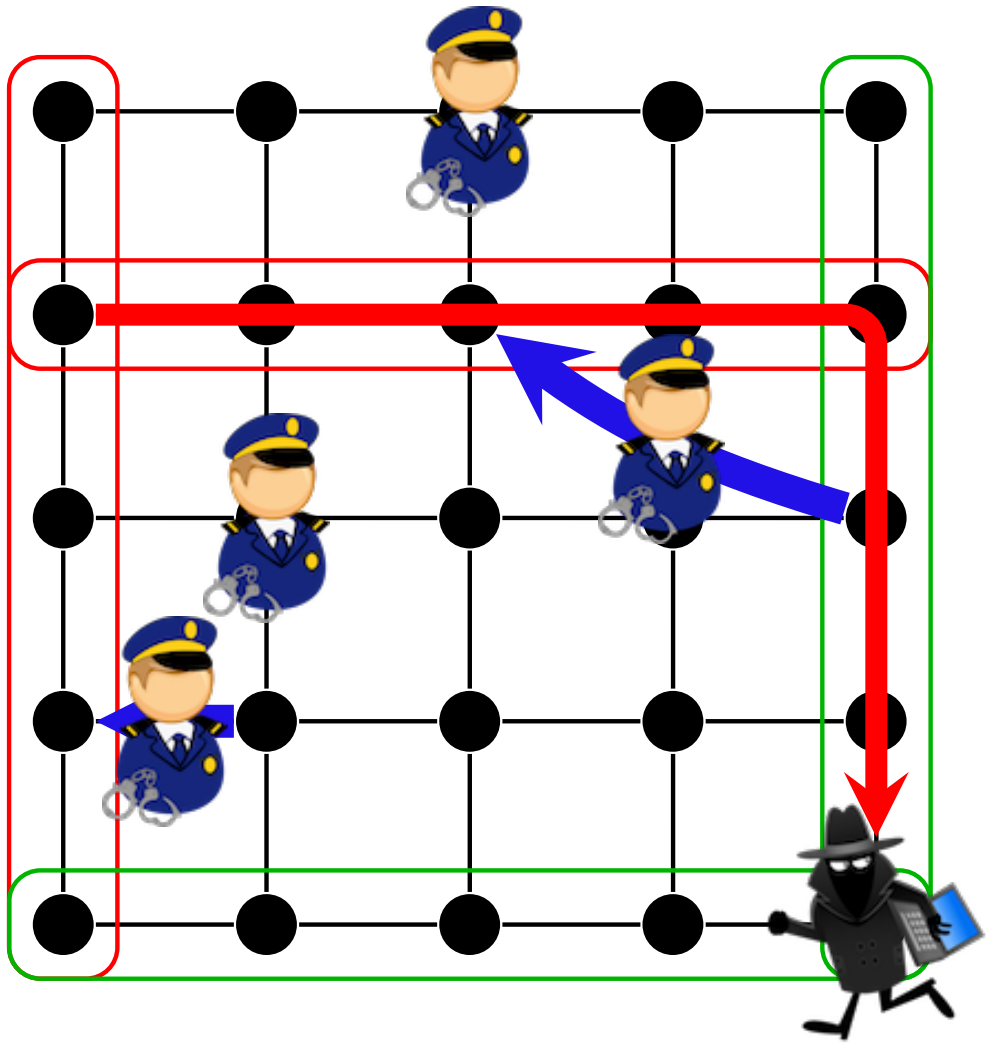
mais pas $n - 1$!



Invariant: au moins une ligne + une colonne libres

Grille $n \times n$

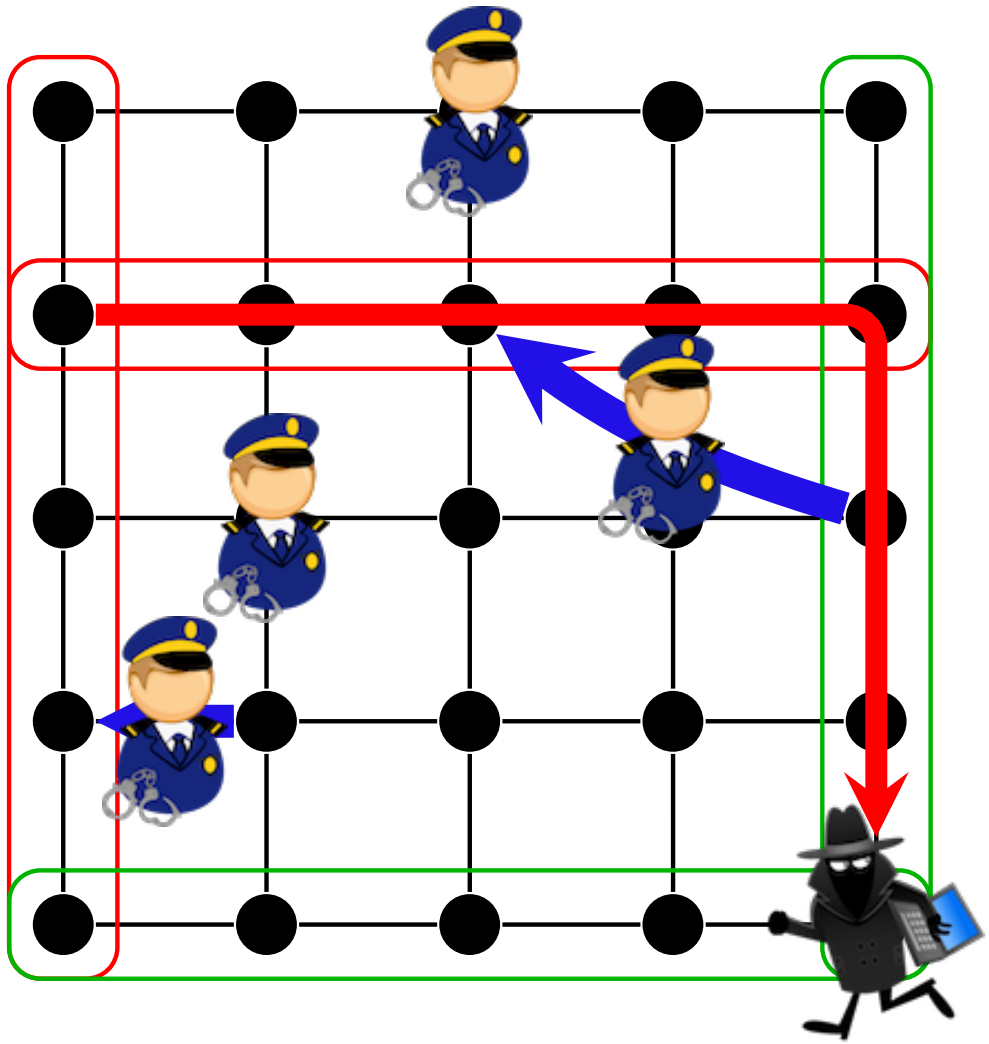
mais pas $n - 1$!



Invariant: au moins une ligne + une colonne libres

Grille $n \times n$

mais pas $n - 1$! . . . ni n



Invariant: au moins une ligne + une colonne libres

Combien de gendarmes sont nécessaires ?

- Arbres $\rightarrow 2$
- Clique avec n sommets $\rightarrow n$
- $(n \times n)$-grille $\rightarrow n + 1$

Combien de gendarmes sont nécessaires ?

- Arbres $\rightarrow 2$
- Clique avec n sommets $\rightarrow n$
- $(n \times n)$-grille $\rightarrow n + 1$

Plus un graphe est proche d'un arbre, moins il faut de gendarmes pour attraper le voleur.

Combien de gendarmes sont nécessaires ?

- Arbres $\rightarrow 2$
- Clique avec n sommets $\rightarrow n$
- $(n \times n)$-grille $\rightarrow n+1$

Plus un graphe est proche d'un arbre, moins il faut de gendarmes pour attraper le voleur.

Dans le jeu avec k gendarmes sur G, les gendarmes ont une stratégie pour capturer le voleur ssi G a une largeur arborescente $\leq k-1$.

Combien de gendarmes sont nécessaires ?

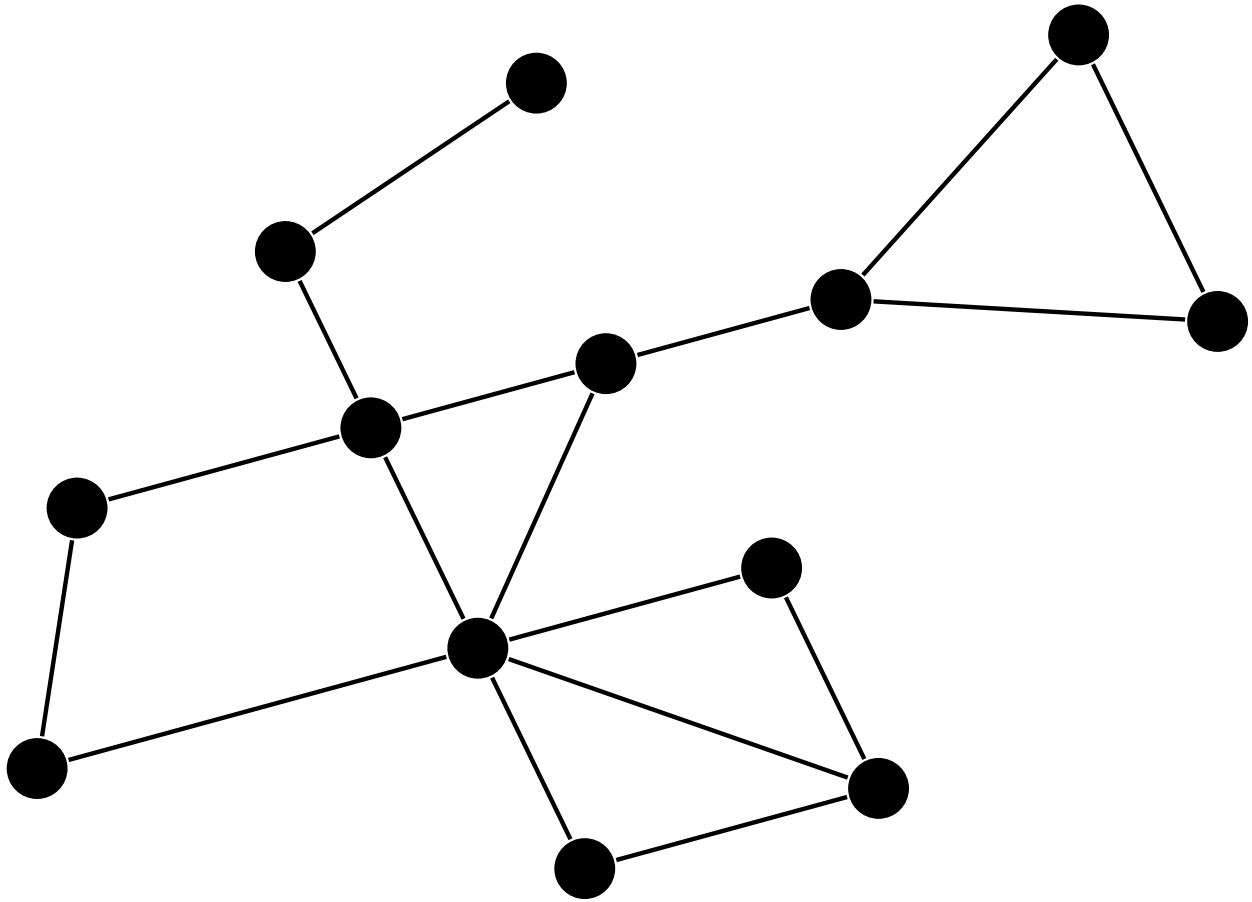
- Arbres $\rightarrow 2$
- Clique avec n sommets $\rightarrow n$
- $(n \times n)$-grille $\rightarrow n+1$

Plus un graphe est proche d'un arbre, moins il faut de gendarmes pour attraper le voleur.

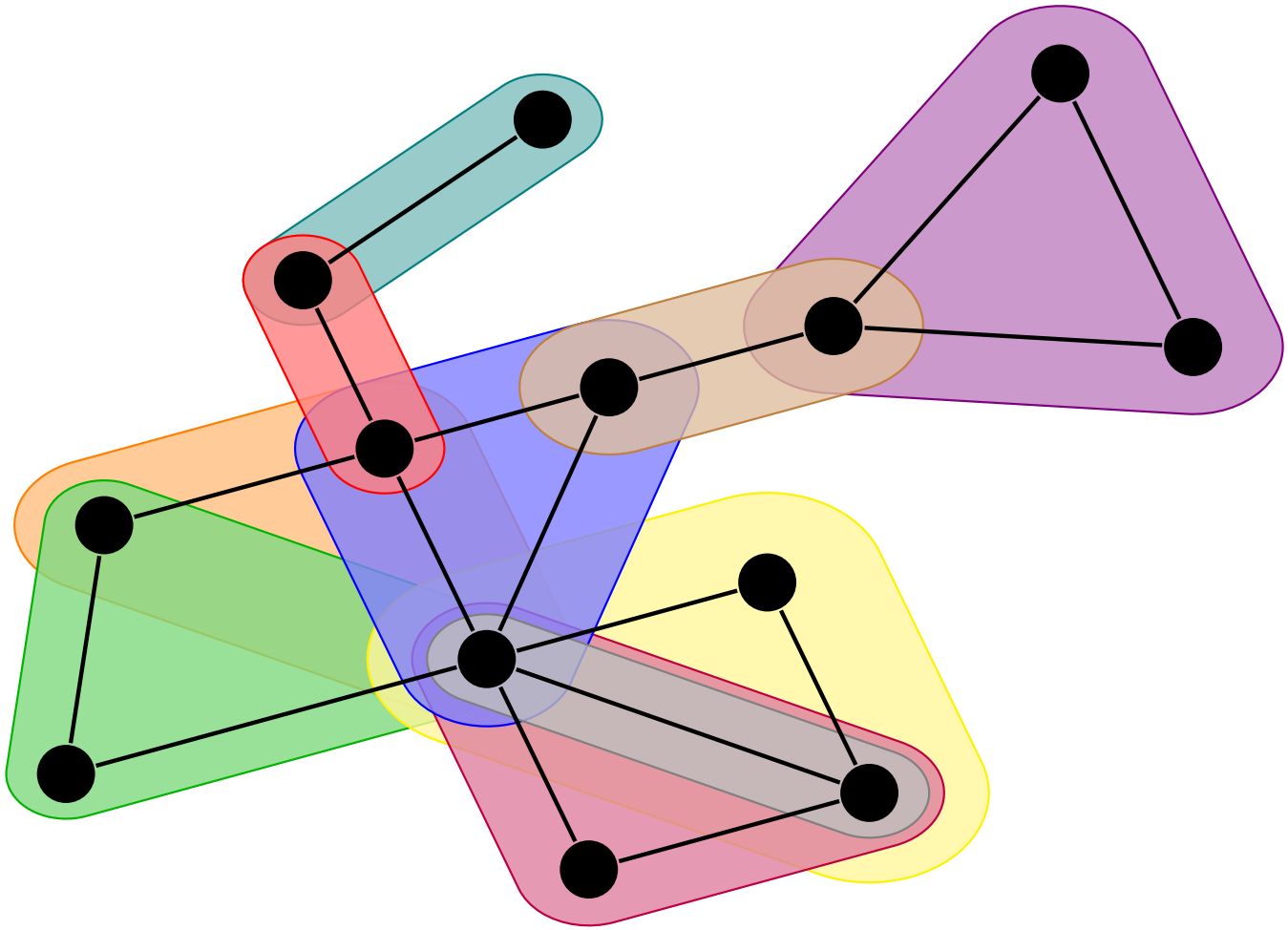
Dans le jeu avec k gendarmes sur G, les gendarmes ont une stratégie pour capturer le voleur ssi G a une largeur arborescente $\leq k-1$.

G a une largeur arborescente ≤ 1 ssi c'est un arbre ou une forêt.

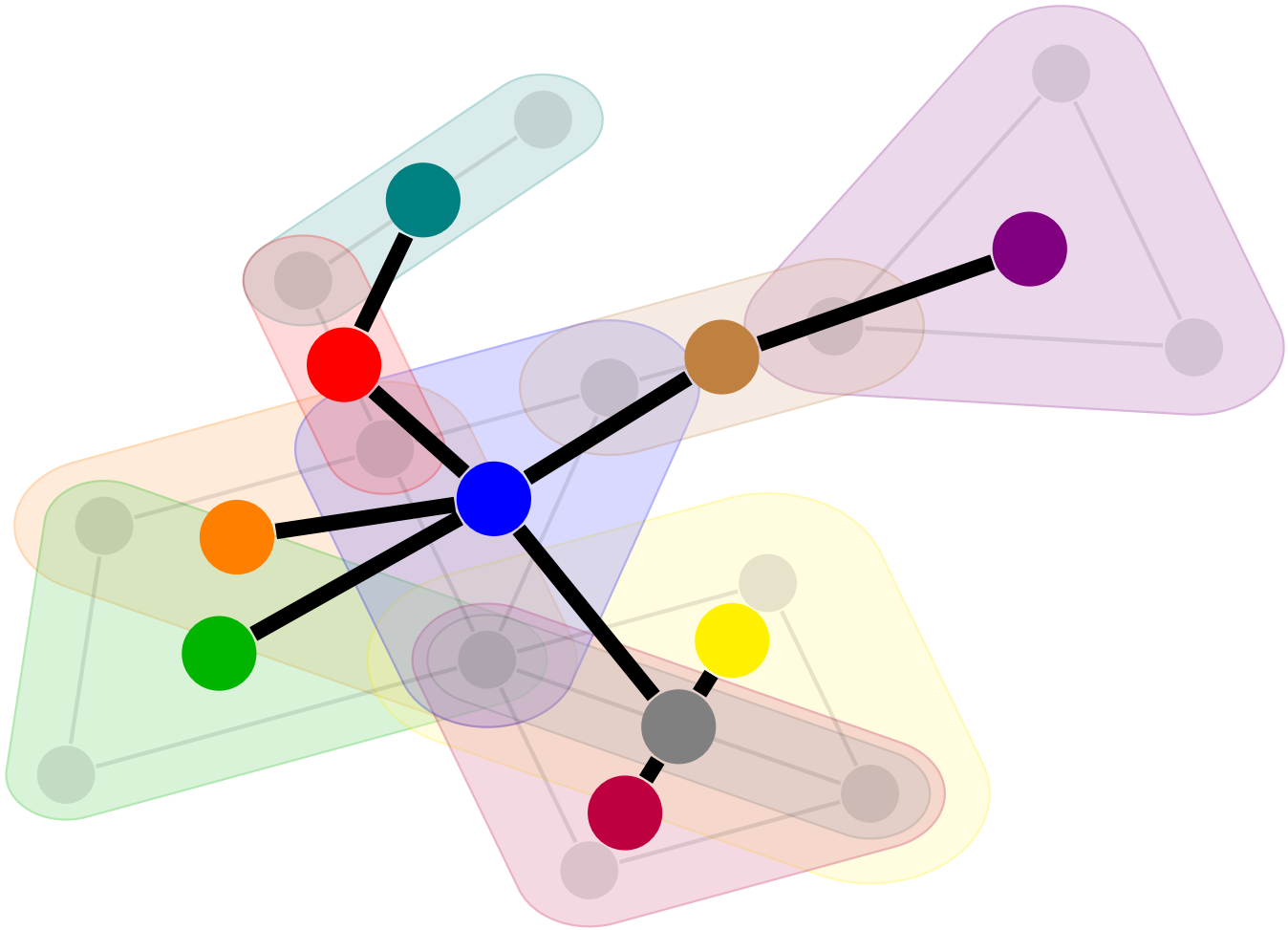
Décomposition en arbre



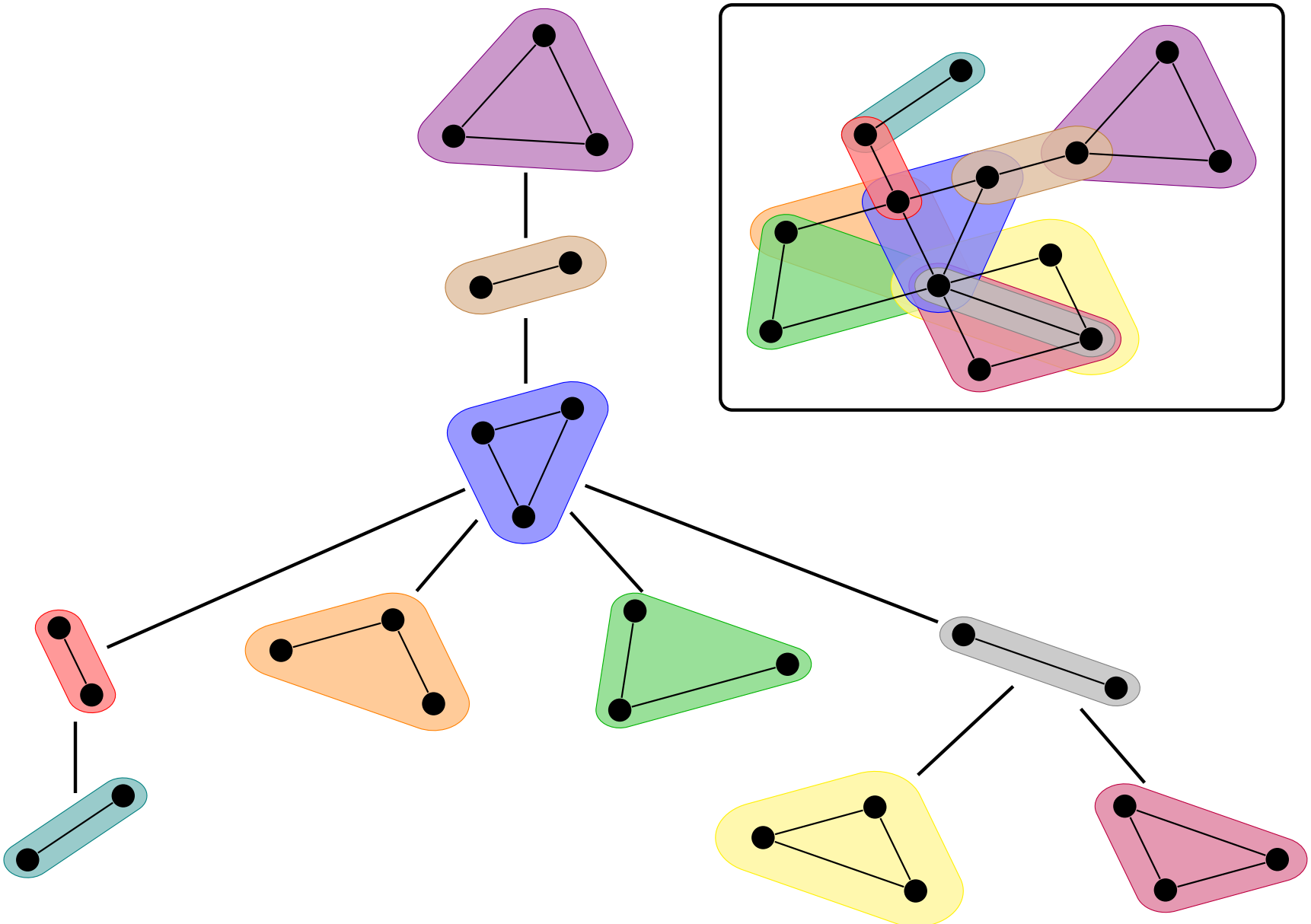
Décomposition en arbre



Décomposition en arbre

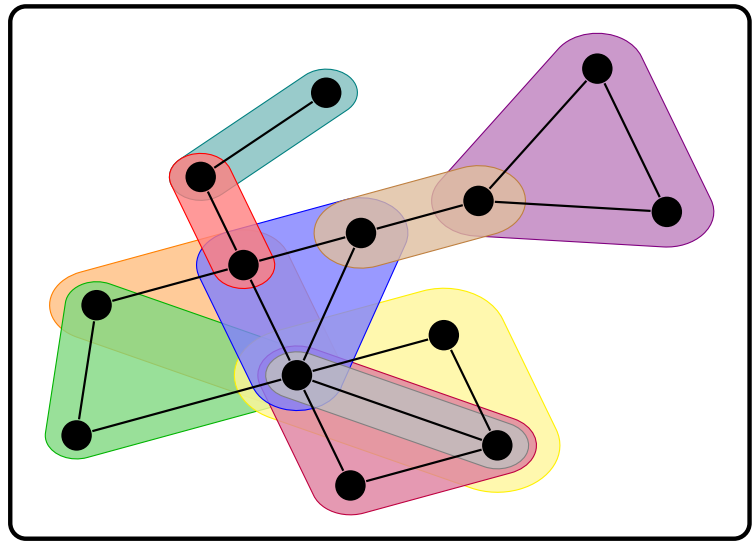


Décomposition en arbre



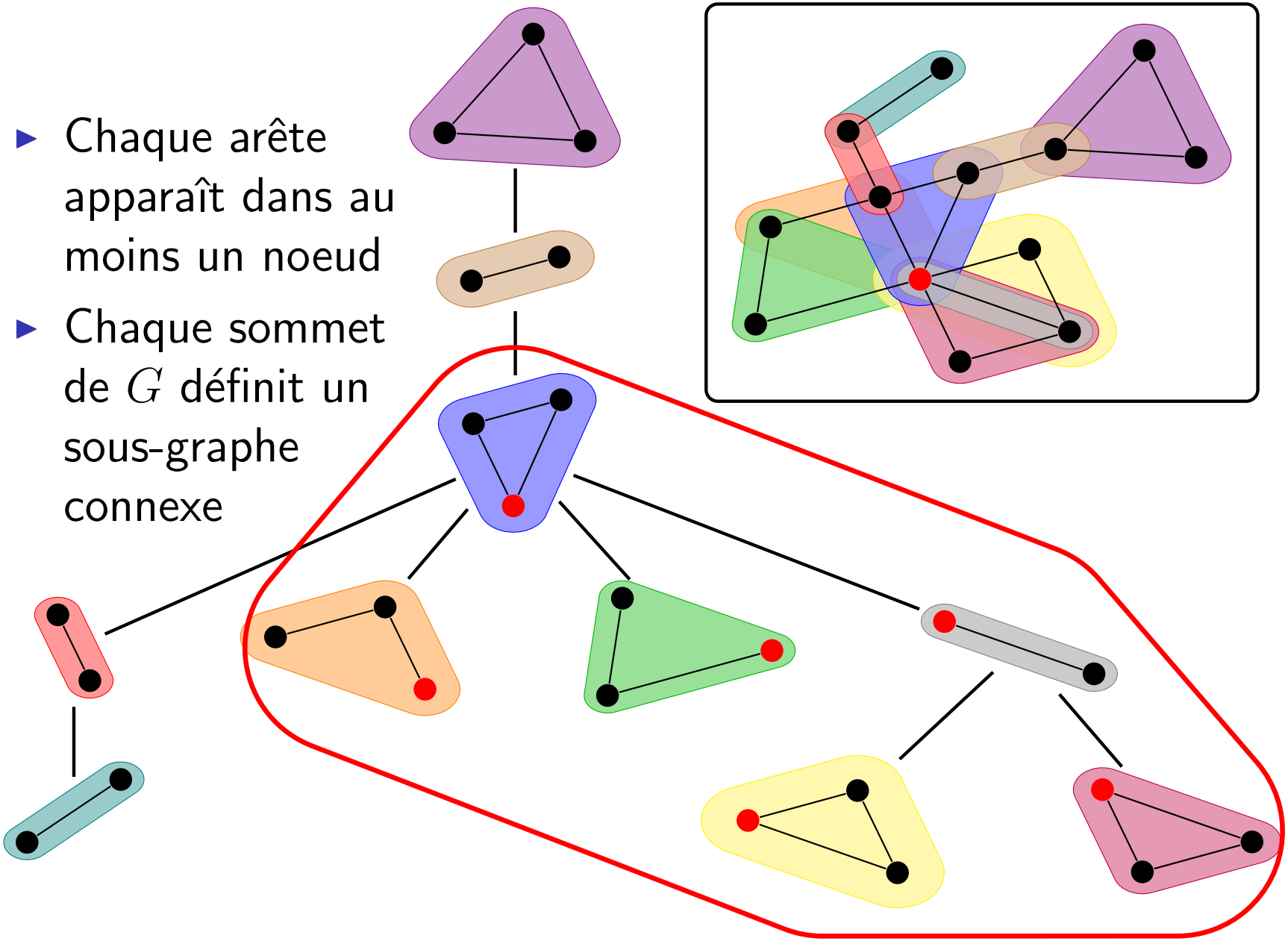
Décomposition en arbre

- Chaque arête apparaît dans au moins un noeud



Décomposition en arbre

- Chaque arête apparaît dans au moins un noeud
- Chaque sommet de G définit un sous-graphe connexe



Largeur arborescente

La largeur arborescente d'un graphe G est le plus petit k tel que G admette une décomposition en arbre avec au plus $k+1$ sommets de G dans chaque noeud.

Largeur arborescente

La largeur arborescente d'un graphe G est le plus petit k tel que G admette une décomposition en arbre avec au plus $k+1$ sommets de G dans chaque noeud.

$\rightarrow$ Deux définitions équivalentes de la largeur arborescente :

- plus petit k tel que avec $k+1$ gendarmes, les gendarmes ont une stratégie pour capturer le voleur sur G.
- avec les décompositions en arbre.

Applications

Observation : beaucoup de problèmes sont difficiles sur les graphes, mais faciles sur les arbres.

Applications

Observation : beaucoup de problèmes sont difficiles sur les graphes, mais faciles sur les ~~arbres.~~ graphes proches d'arbres.

Applications

Observation : beaucoup de problèmes sont difficiles sur les graphes, mais faciles sur les ~~arbres.~~ graphes proches d'arbres.

- Algorithmique: recherche d'un ensemble indépendant maximum, d'un cycle hamiltonien, k-coloriage, ...

Applications

Observation : beaucoup de problèmes sont difficiles sur les graphes, mais faciles sur les ~~arbres.~~ graphes proches d'arbres.

- Algorithmique: recherche d'un ensemble indépendant maximum, d'un cycle hamiltonien, k-coloriage, ...
- Logique: satisfaisabilité d'une formule, ...

Applications

Observation : beaucoup de problèmes sont difficiles sur les graphes, mais faciles sur les ~~arbres.~~ graphes proches d'arbres.

- Algorithmique: recherche d'un ensemble indépendant maximum, d'un cycle hamiltonien, k-coloriage, ...
- Logique: satisfaisabilité d'une formule, ...
- Vérification: vérification de systèmes distribués, ...

Applications

Observation : beaucoup de problèmes sont difficiles sur les graphes, mais faciles sur les ~~arbres.~~ graphes proches d'arbres.

- Algorithmique: recherche d'un ensemble indépendant maximum, d'un cycle hamiltonien, k-coloriage, ...
- Logique: satisfaisabilité d'une formule, ...
- Vérification: vérification de systèmes distribués, ...
- ...

Applications

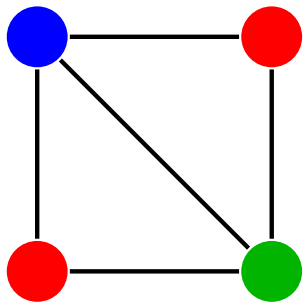
Observation : beaucoup de problèmes sont difficiles sur les graphes, mais faciles sur les ~~arbres.~~ graphes proches d'arbres.

- Algorithmique: recherche d'un ensemble indépendant maximum, d'un cycle hamiltonien, k-coloriage, ...
- Logique: satisfaisabilité d'une formule, ...
- Vérification: vérification de systèmes distribués, ...
- ...

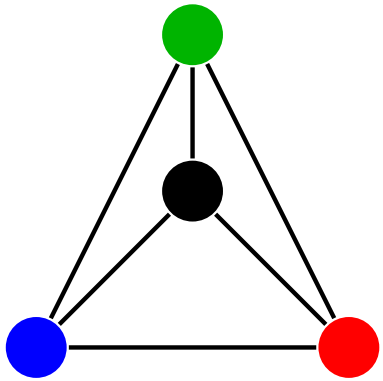
On rencontre souvent des familles de graphes avec une petite largeur arborescente.

Exemple : 3-coloration

Objectif: algorithme qui prend en entrée un graphe et répond "oui" ssi le graphe est coloriable avec trois couleurs.



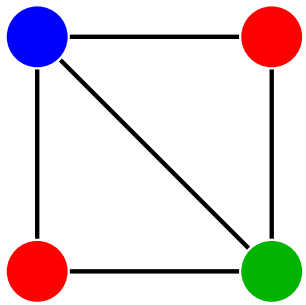
3-coloriable



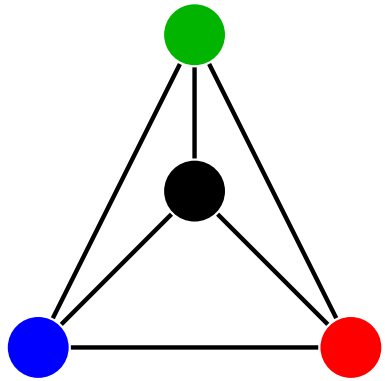
Pas 3-coloriable

Exemple : 3-coloration

Objectif: algorithme qui prend en entrée un graphe et répond "oui" ssi le graphe est coloriable avec trois couleurs.



3-coloriable

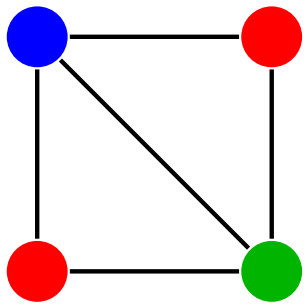


Pas 3-coloriable

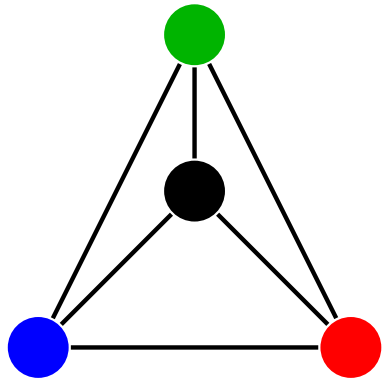
- Pour des graphes quelconques, NP-complet !

Exemple : 3-coloration

Objectif: algorithme qui prend en entrée un graphe et répond "oui" ssi le graphe est coloriable avec trois couleurs.



3-coloriable



Pas 3-coloriable

- Pour des graphes quelconques, NP-complet !
- Pour des graphes de largeur arborescente fixée, algo linéaire

Programmation dynamique

Idée: résoudre successivement des sous-problèmes associés aux sous-arbres de la décomposition, en remontant des feuilles vers la racine.

Programmation dynamique

Idée: résoudre successivement des sous-problèmes associés aux sous-arbres de la décomposition, en remontant des feuilles vers la racine.

Il faut

- faire en sorte qu'on puisse facilement calculer toutes les solutions du sous-problème associé à un noeud x si on connait toutes celles des noeuds fils $x_1, \ldots, x_n$ de x.

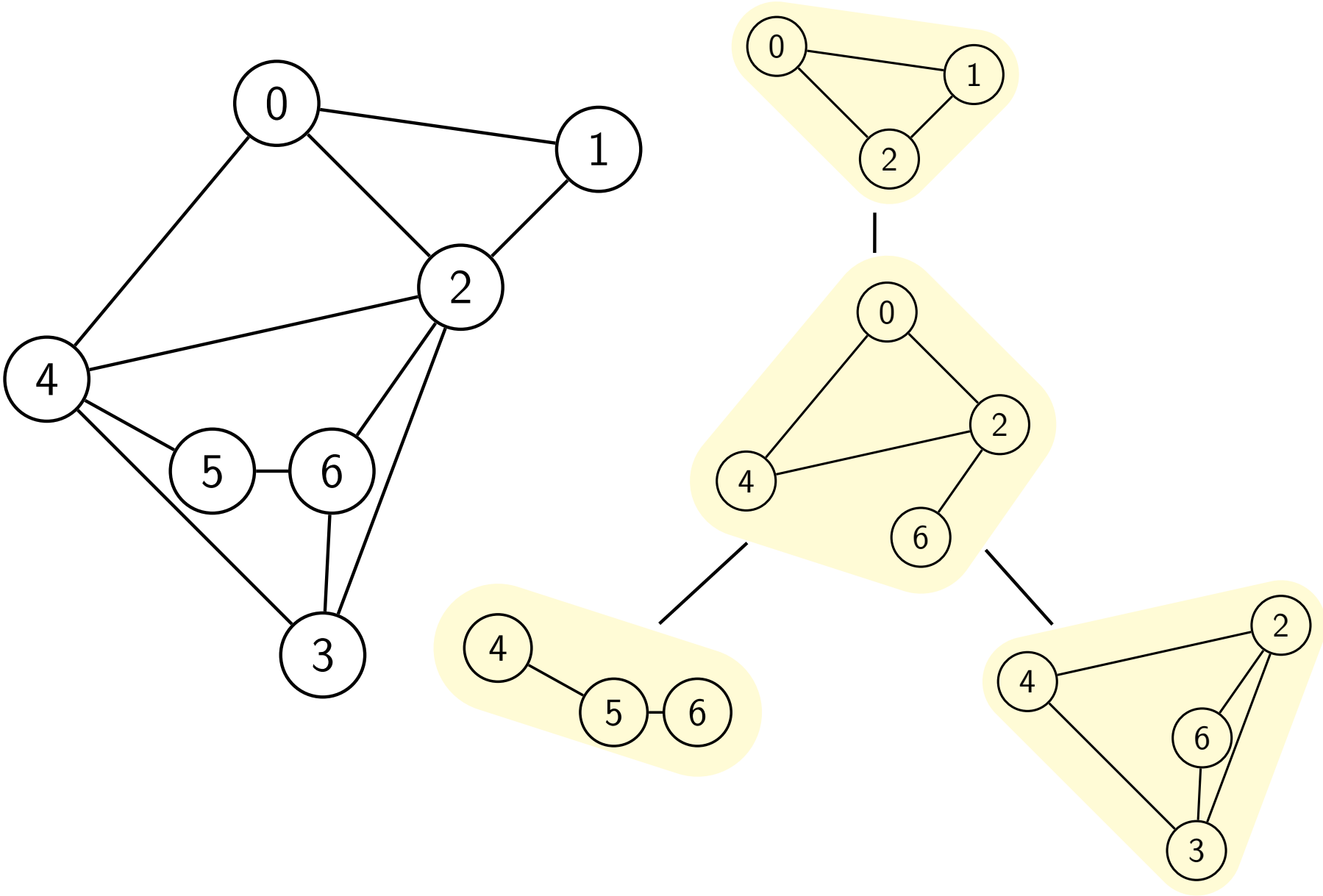
Programmation dynamique

Idée: résoudre successivement des sous-problèmes associés aux sous-arbres de la décomposition, en remontant des feuilles vers la racine.

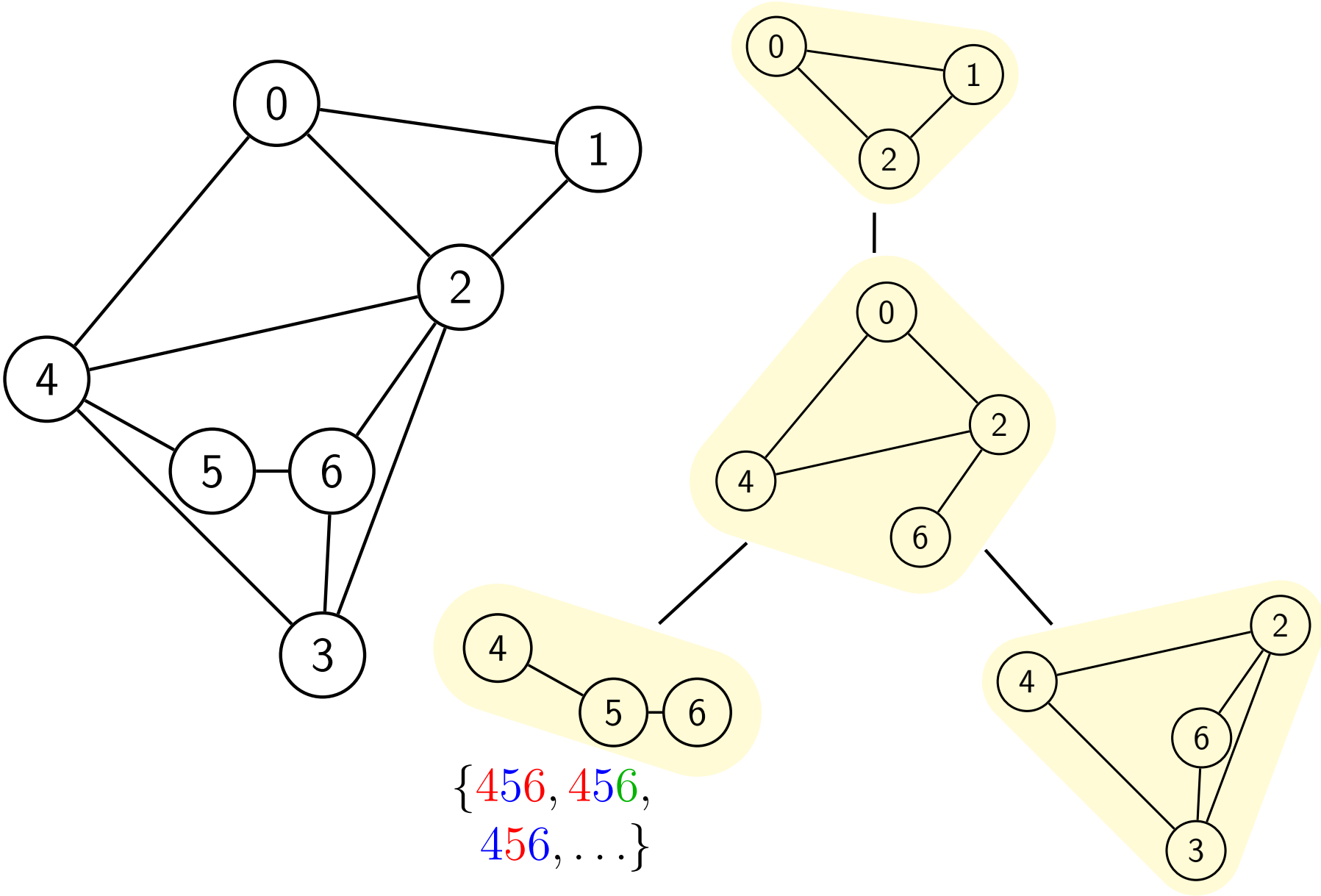
Il faut

- faire en sorte qu'on puisse facilement calculer toutes les solutions du sous-problème associé à un noeud x si on connait toutes celles des noeuds fils $x_1, \ldots, x_n$ de x.
- minimiser la quantité d'information qu'on retient à propos des solutions.

Exemple

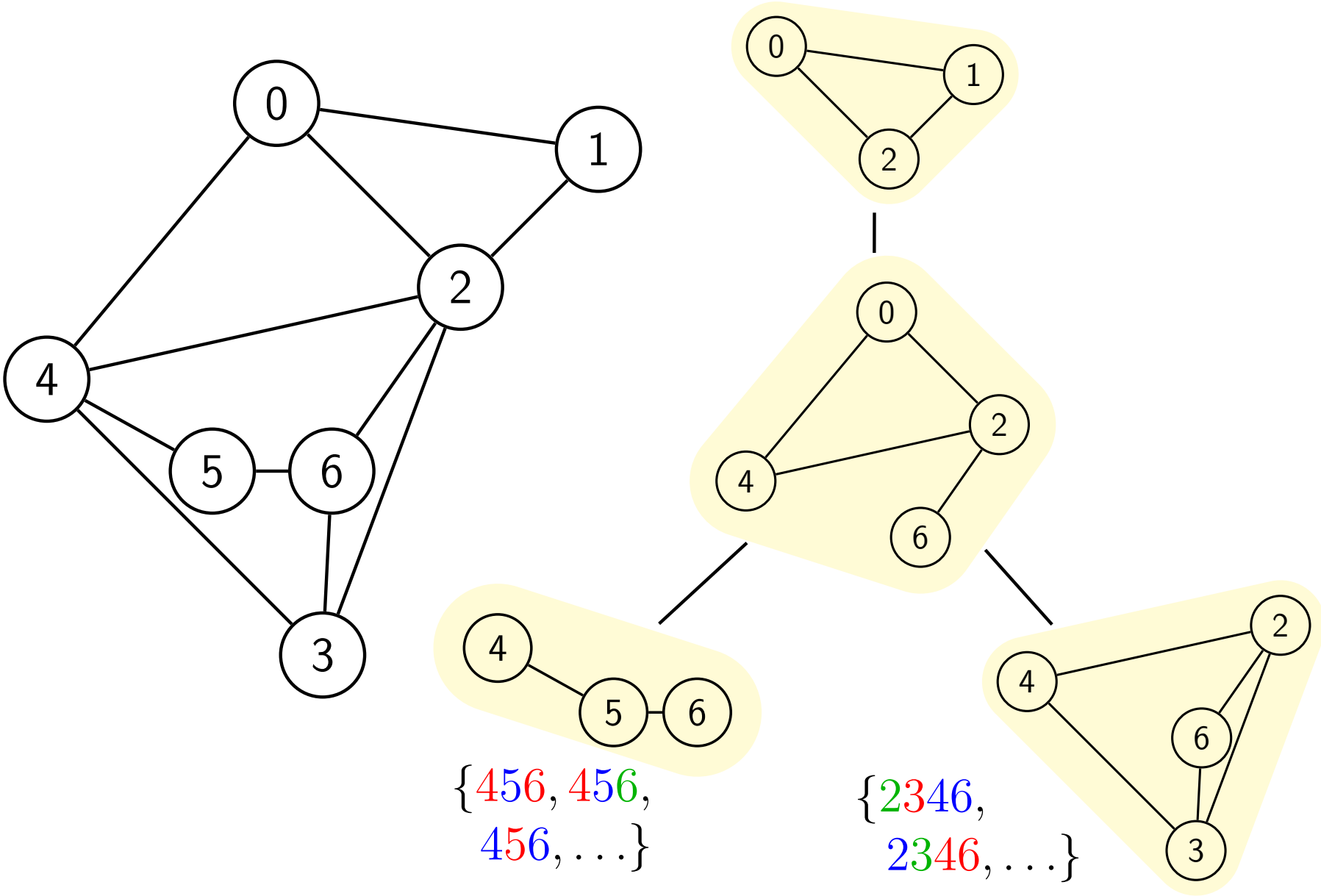


Exemple



$\{456, 456,$
$456, \ldots\}$

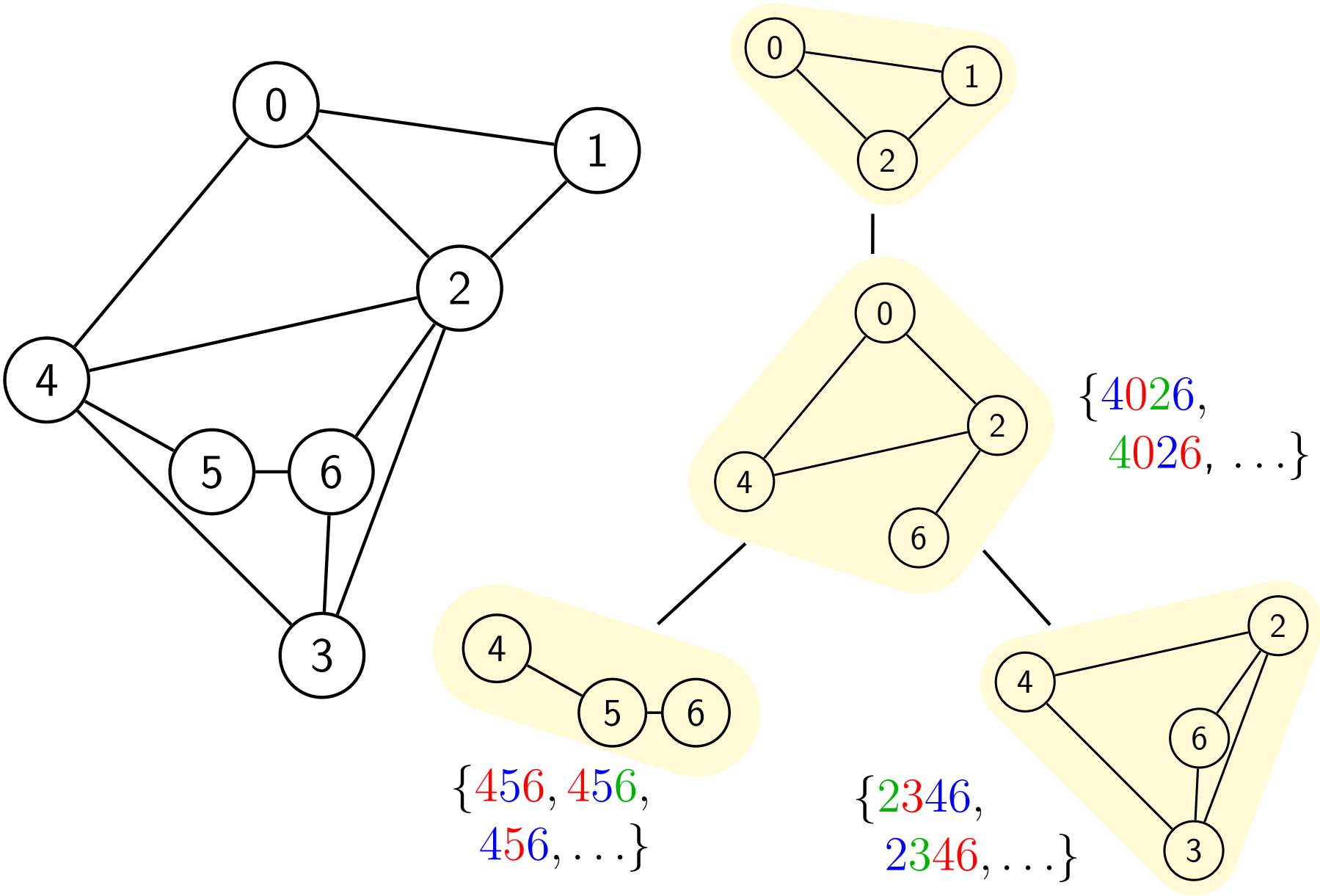
Exemple



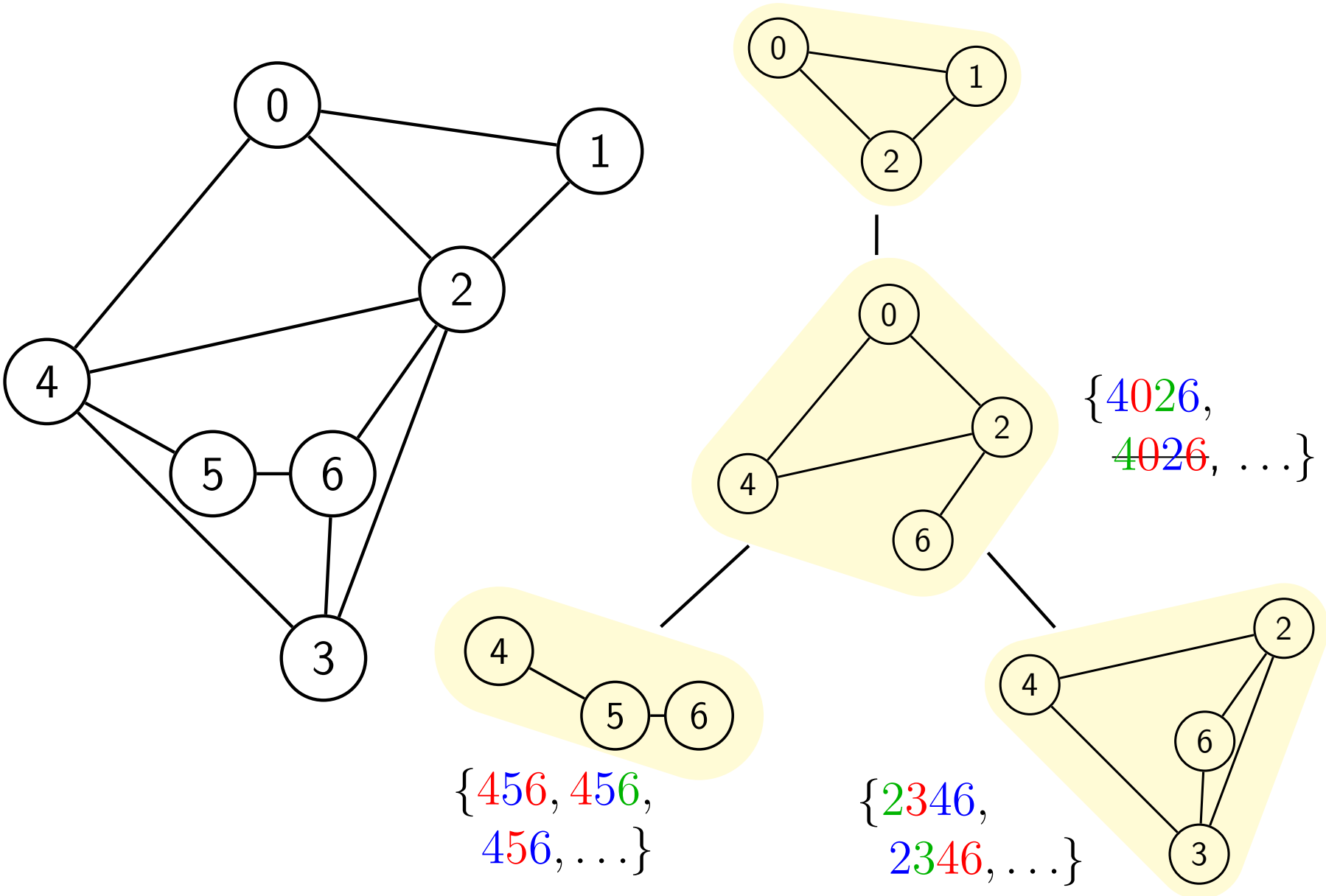
$\{456, 456, 456, \ldots\}$

$\{2346, 2346, \ldots\}$

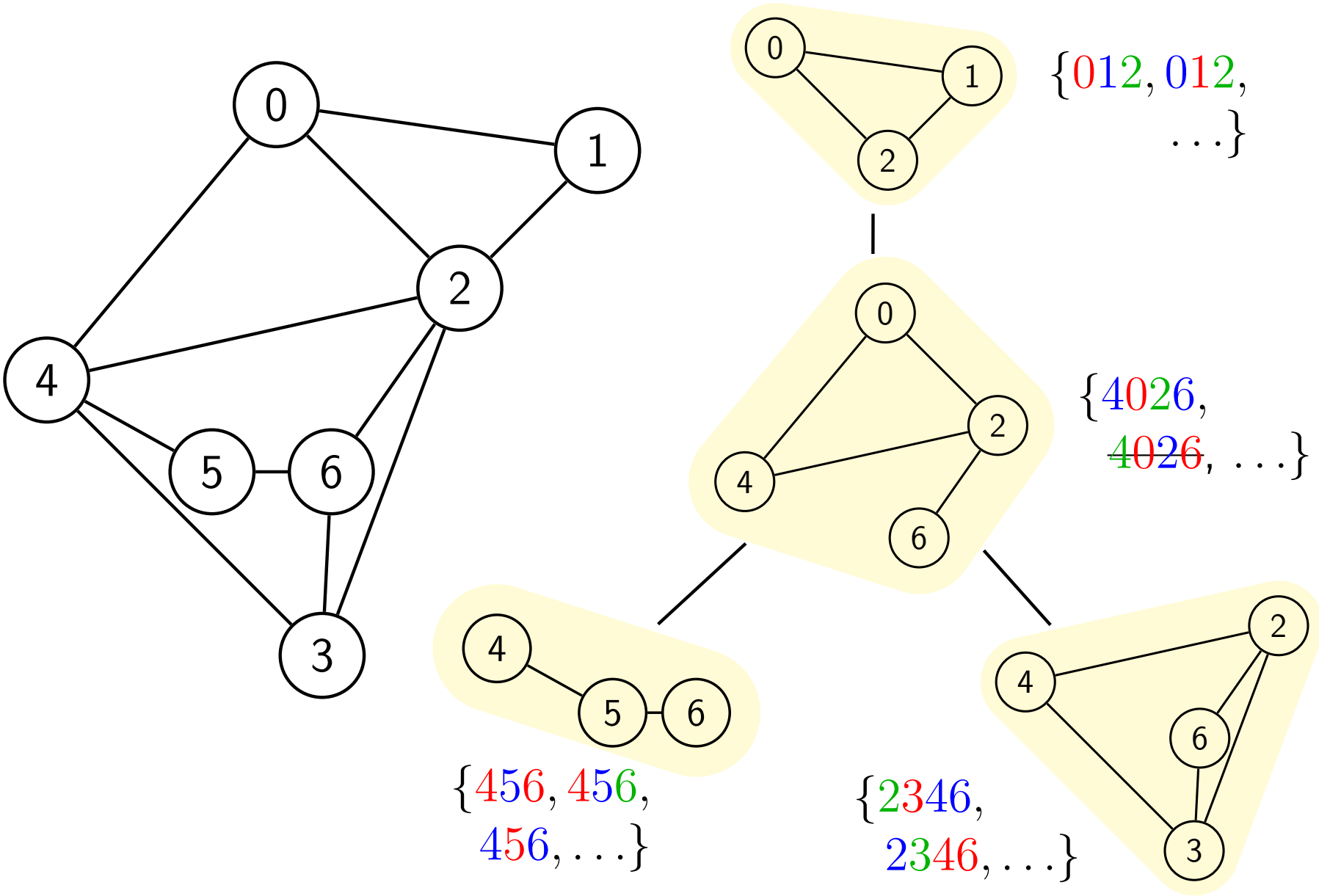
Exemple



Exemple



Exemple



Merci !

Questions ?