Encerclement connexe dans les graphes

Nicolas Nisse

LRI, Orsay, Université Paris-Sud.

séminaire, LIP, 12 avril 2006

Plan

Encerclement dans les graphes

But

Dans un réseau

- contaminé par un gaz toxique, un virus...,
- une équipe d'agents mobiles doit nettoyer le graphe.

Trouver une stratégie qui nettoie le graphe
en utilisant le moins de ressources possible.

Encerclement dans les graphes

But (Alternative)

Dans un réseau

- envahi par un fugitif invisible, omniscient et arbitrairement rapide,
- une équipe d'agents mobiles doit capturer le fugitif.

Trouver une stratégie qui capture le fugitif
en utilisant le moins d'agents possible.

Motivations

Applications

- sécurité dans les réseaux de type internet
- maintenance de réseaux de pipelines
- opération de secours dans des souterrains

Aspects fondamentaux

L'encerclement est en étroite relation avec des paramètres statiques des graphes.

- largeur arborescente
- largeur linéaire

Plan

Stratégies d' Encerclement, Parson. [GTC,78]
Variante de Kirousis et Papadimitriou. [TCS,86]

Séquence de deux opérations élémentaires, . . .

1. Placer un agent sur un sommet du graphe ;
2. Supprimer un agent d'un sommet du graphe.

. . . qui doit aboutir à la capture du fugitif

Le fugitif est capturé lorsqu'il occupe (ou croise) un sommet occupé par un agent. Une arête est dite nettoyée, lorsque ses extrémités sont occupées par des agents.

Il faut minimiser le nombre d'agents.

Soit $s(G)$ le nombre minimum d'agents nécessaires pour capturer un fugitif invisible dans le graphe G.

Stratégies d' Encerclement, Parson. [GTC,78]
Variante de Kirousis et Papadimitriou. [TCS,86]

Séquence de deux opérations élémentaires,. . .

1. Placer un agent sur un sommet du graphe ;
2. Supprimer un agent d'un sommet du graphe.

. . . qui doit aboutir à la capture du fugitif

Le fugitif est capturé lorsqu'il occupe (ou croise) un sommet occupé par un agent. Une arête est dite nettoyée, lorsque ses extrémités sont occupées par des agents.

Il faut minimiser le nombre d'agents.

Soit $s(G)$ le nombre minimum d'agents nécessaires pour capturer un fugitif invisible dans le graphe G.

Stratégies d' Encerclement, Parson. [GTC,78] Variante de Kirousis et Papadimitriou. [TCS,86]

Séquence de deux opérations élémentaires,. . .

1. Placer un agent sur un sommet du graphe ;
2. Supprimer un agent d'un sommet du graphe.

. . . qui doit aboutir à la capture du fugitif

Le fugitif est capturé lorsqu'il occupe (ou croise) un sommet occupé par un agent. Une arête est dite nettoyée, lorsque ses extrémités sont occupées par des agents.

Il faut minimiser le nombre d'agents.

Soit $s(G)$ le nombre minimum d'agents nécessaires pour capturer un fugitif invisible dans le graphe G.

Stratégies d' Encerclement, Parson. [GTC,78] Variante de Kirousis et Papadimitriou. [TCS,86]

Séquence de deux opérations élémentaires,. . .

1. Placer un agent sur un sommet du graphe ;
2. Supprimer un agent d'un sommet du graphe.

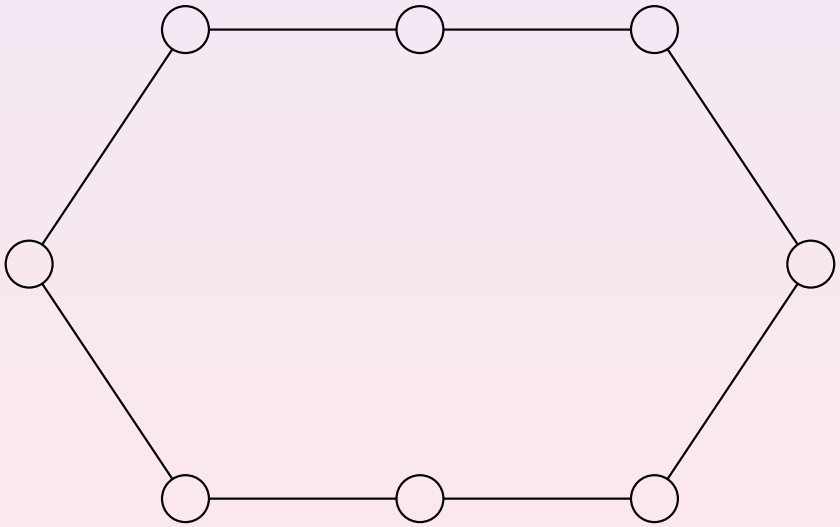
. . . qui doit aboutir à la capture du fugitif

Le fugitif est capturé lorsqu'il occupe (ou croise) un sommet occupé par un agent. Une arête est dite nettoyée, lorsque ses extrémités sont occupées par des agents.

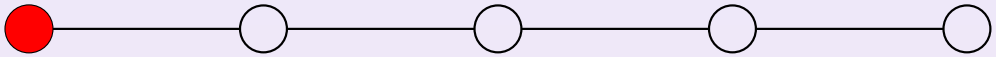
Il faut minimiser le nombre d'agents.

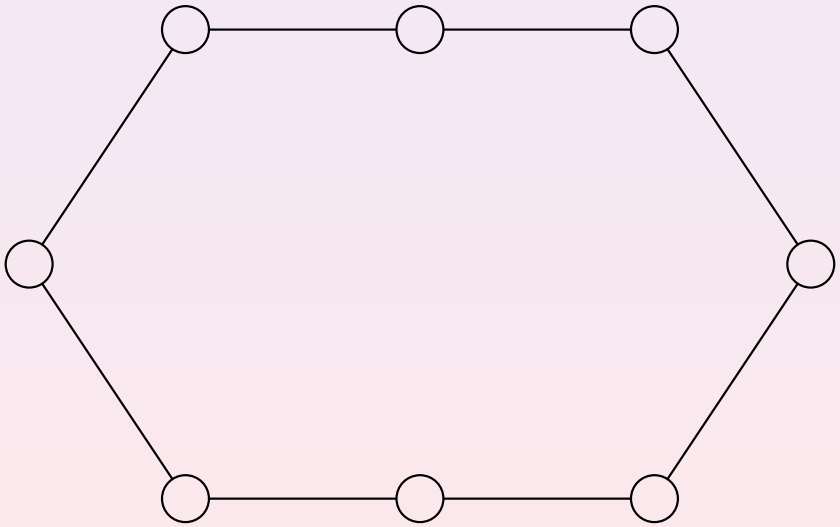
Soit $s(G)$ le nombre minimum d'agents nécessaires pour capturer un fugitif invisible dans le graphe G.

Exemples simples : le chemin et l'anneau

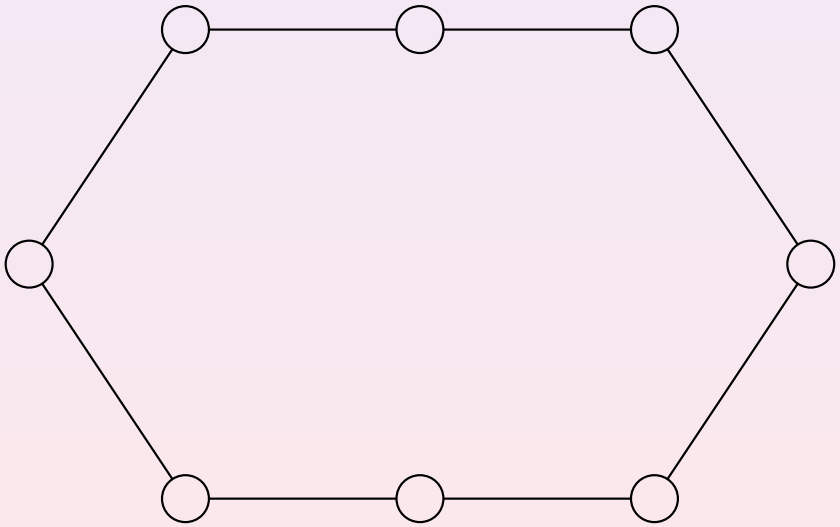


Exemples simples : le chemin et l'anneau

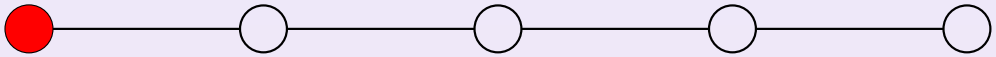


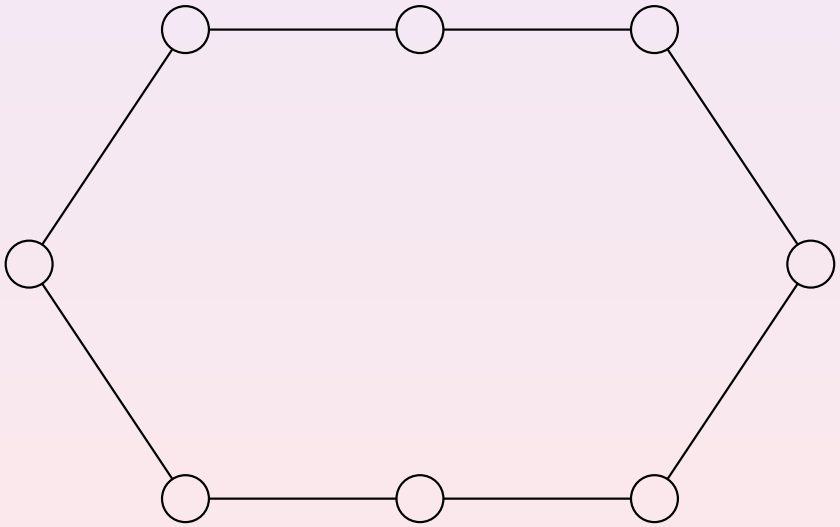


Exemples simples : le chemin et l'anneau

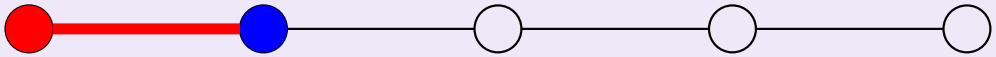


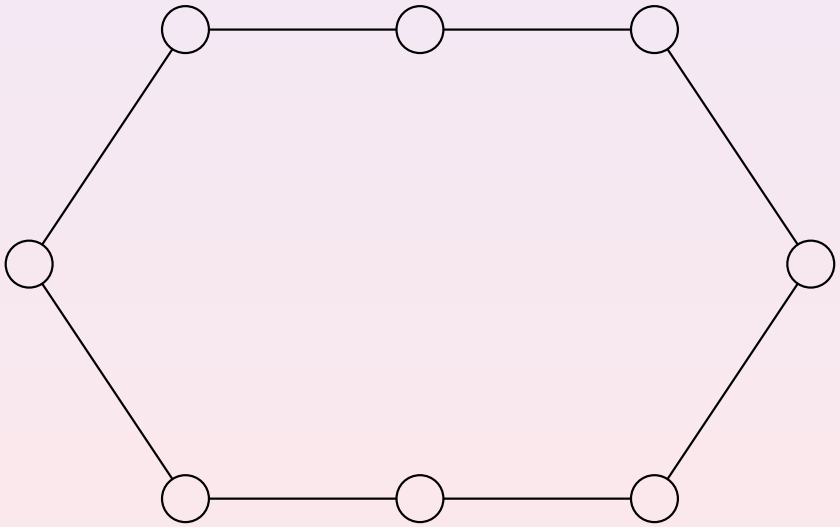
Exemples simples : le chemin et l'anneau



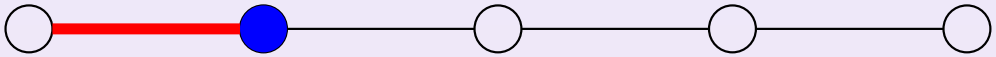


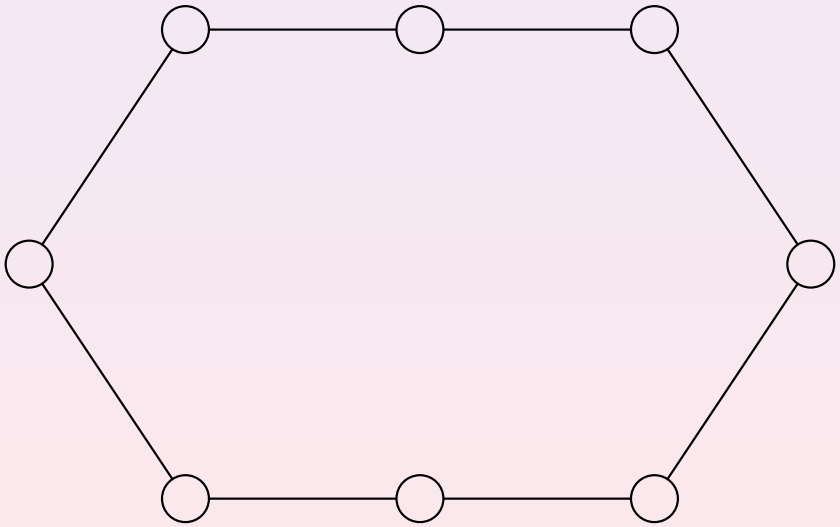
Exemples simples : le chemin et l'anneau



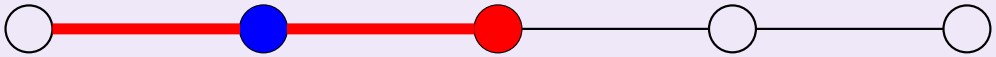


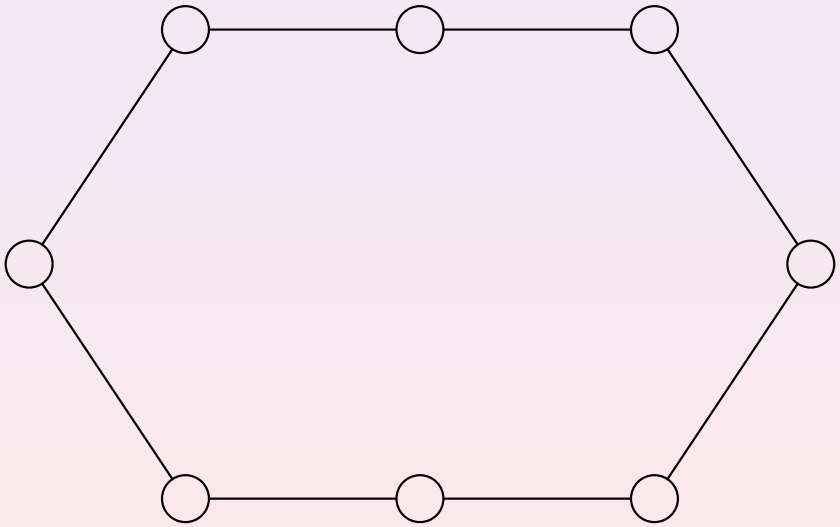
Exemples simples : le chemin et l'anneau



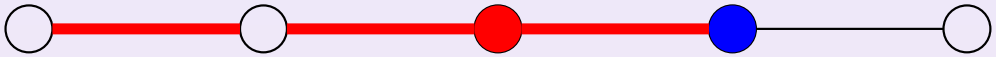


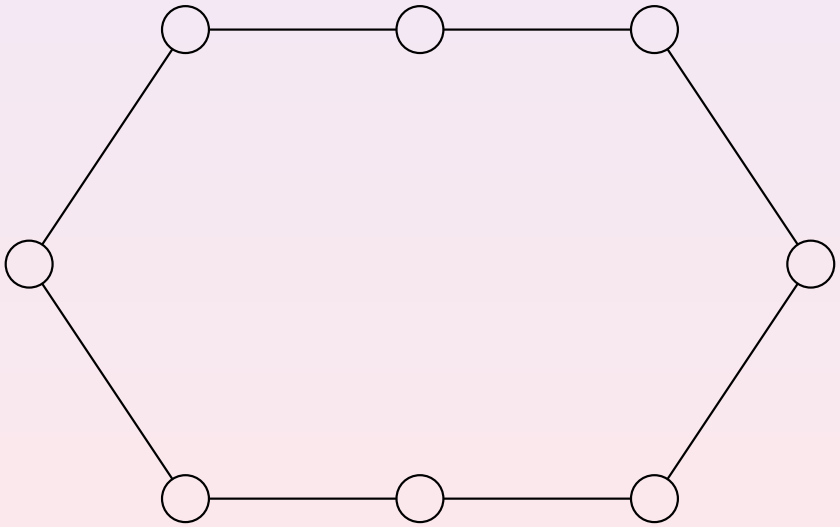
Exemples simples : le chemin et l'anneau



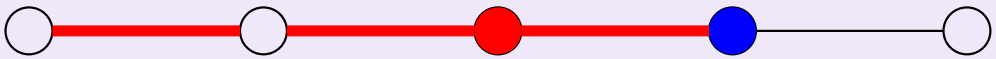


Exemples simples : le chemin et l'anneau

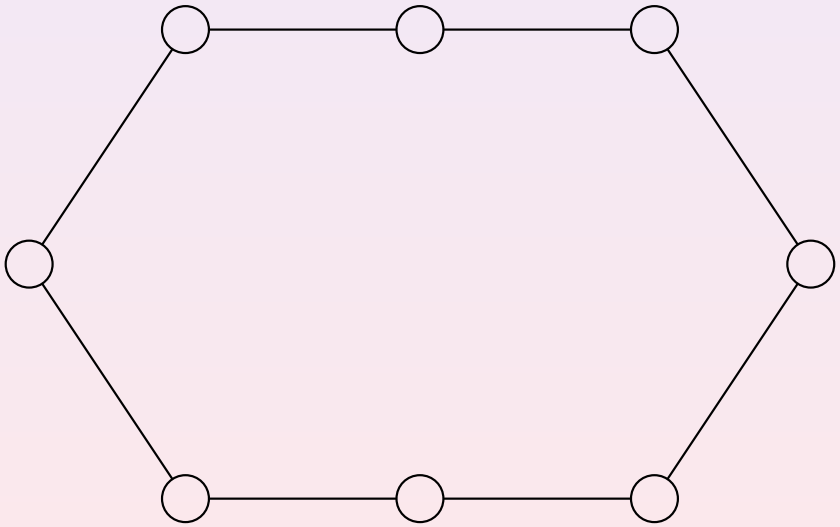




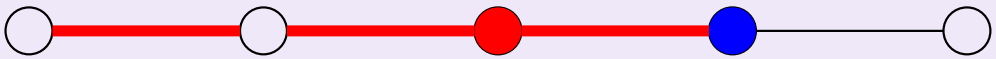
Exemples simples : le chemin et l'anneau



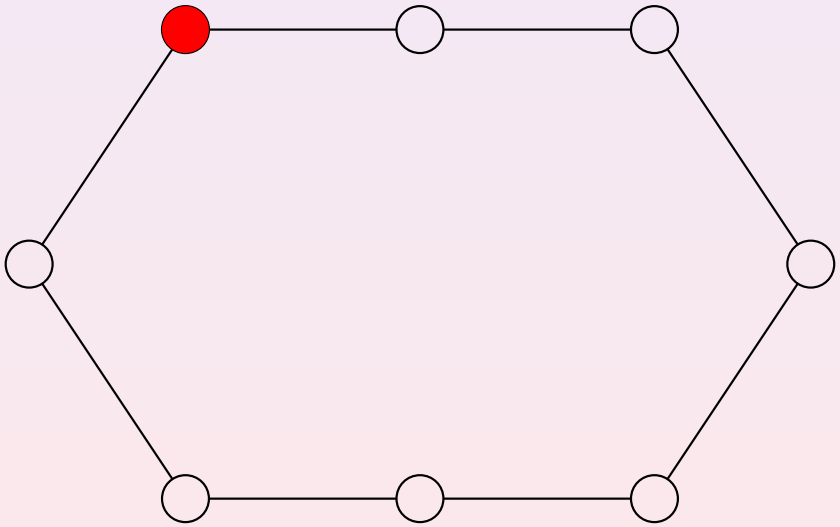
s(Chemin)=2



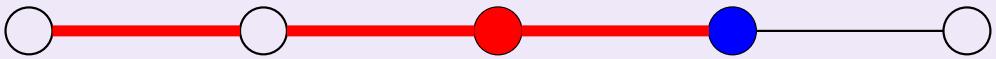
Exemples simples : le chemin et l'anneau



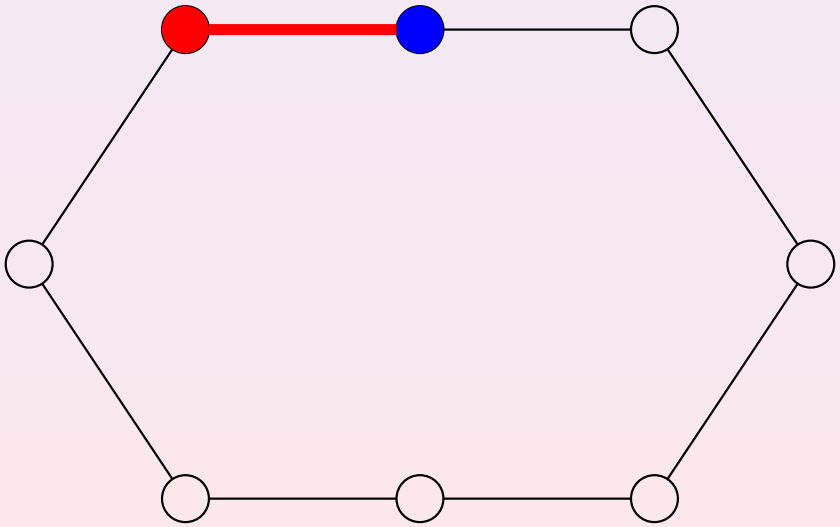
s(Chemin)=2



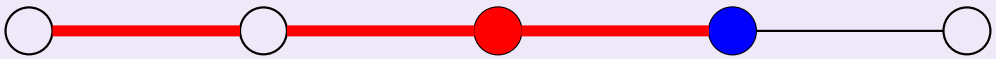
Exemples simples : le chemin et l'anneau



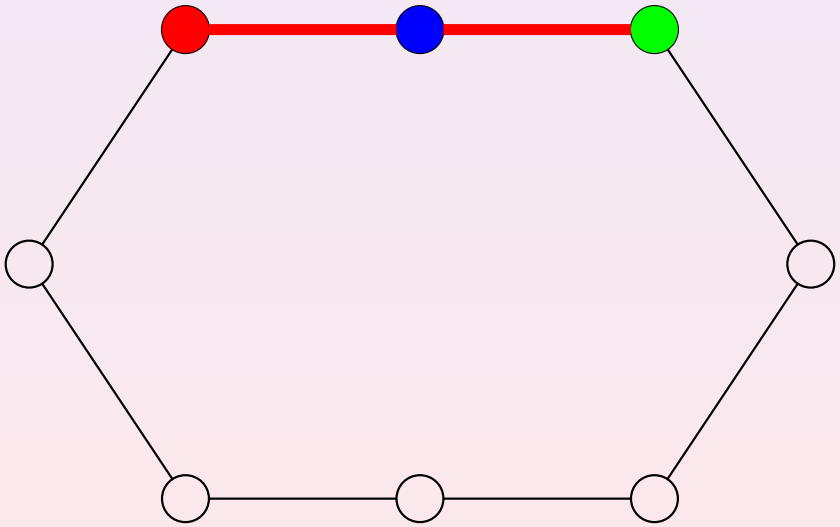
s(Chemin)=2



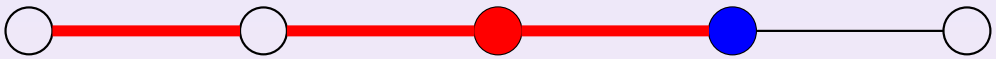
Exemples simples : le chemin et l'anneau



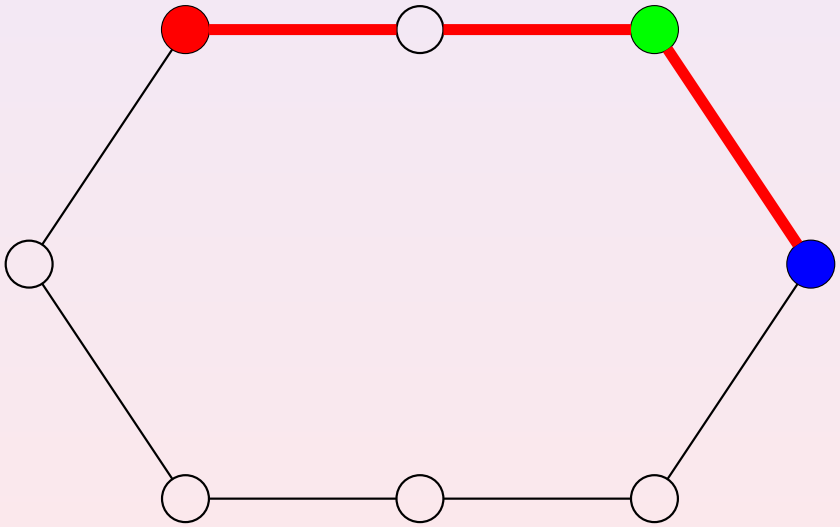
s(Chemin)=2



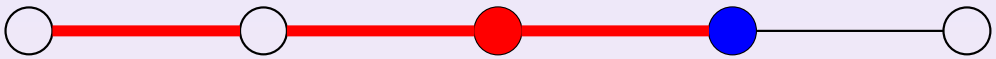
Exemples simples : le chemin et l'anneau



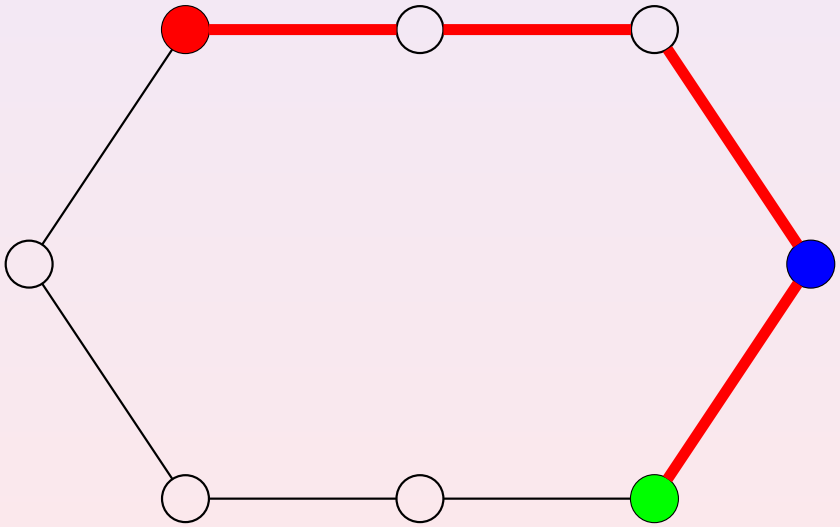
s(Chemin)=2



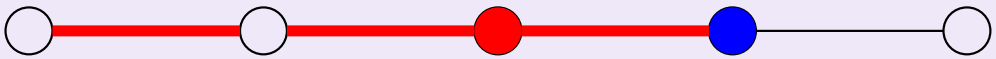
Exemples simples : le chemin et l'anneau



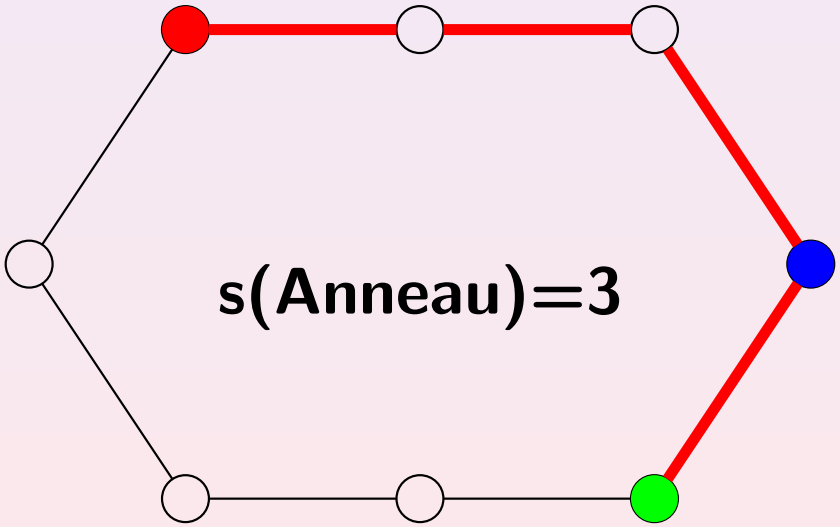
s(Chemin)=2



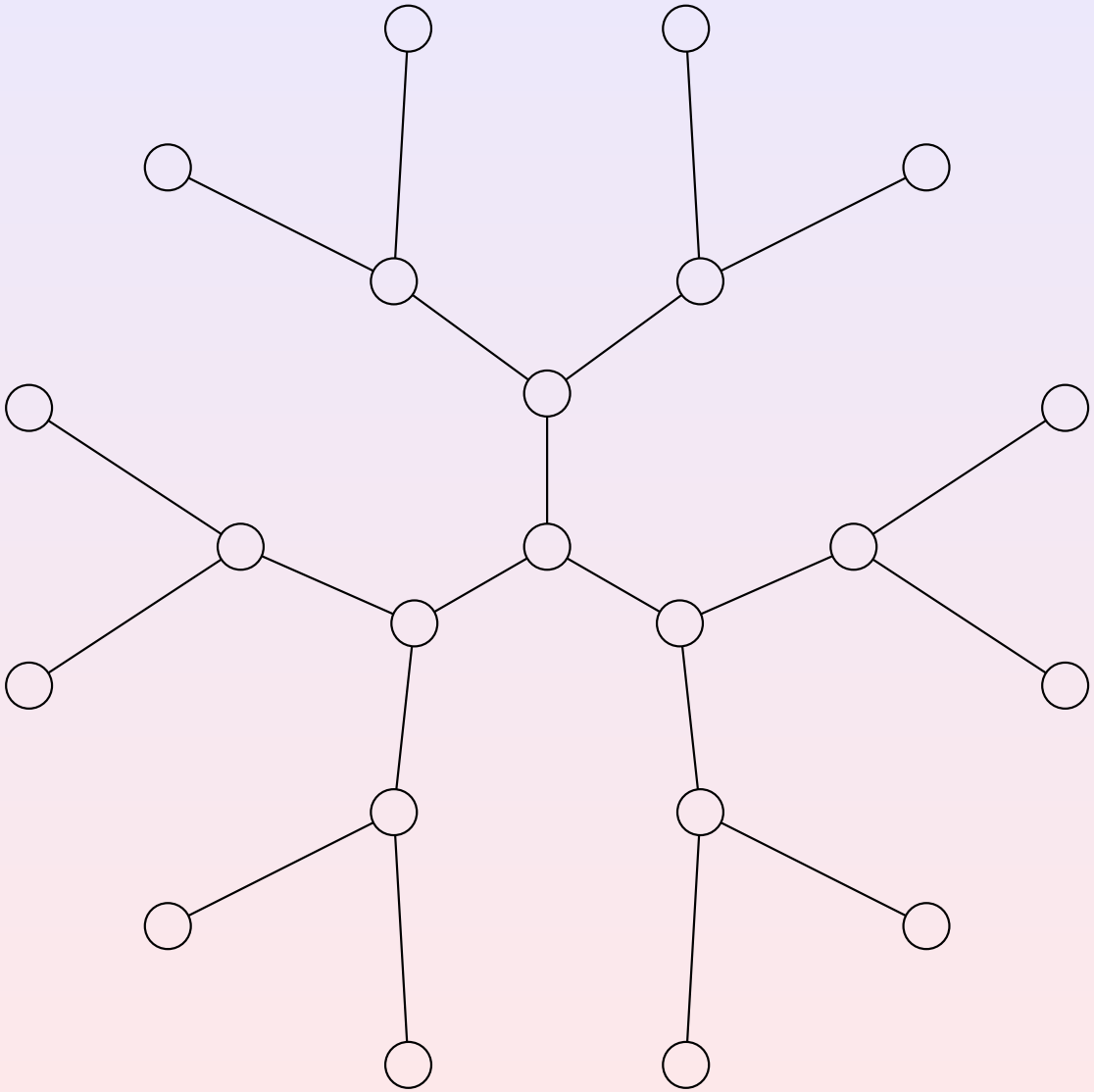
Exemples simples : le chemin et l'anneau



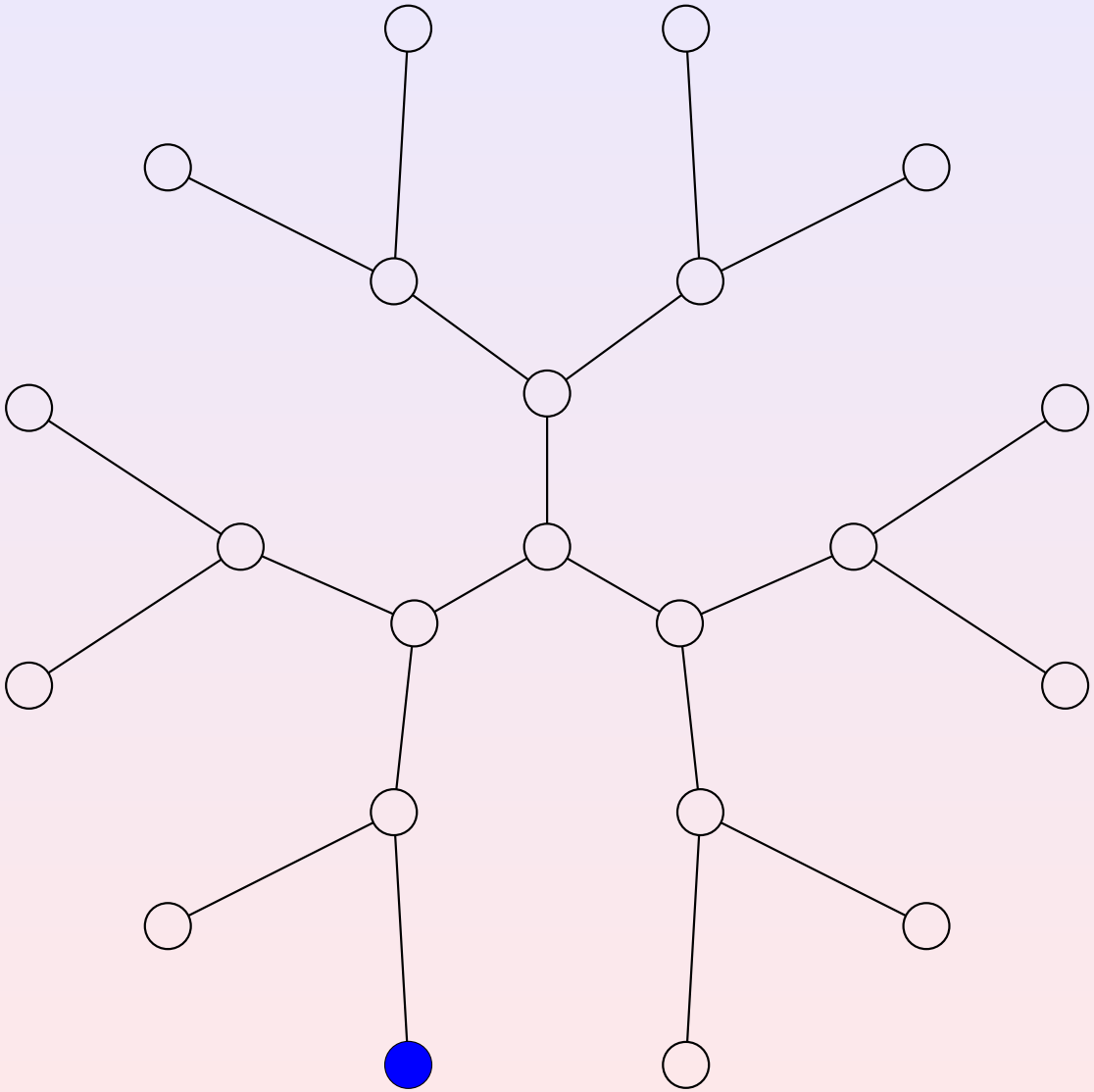
s(Chemin)=2



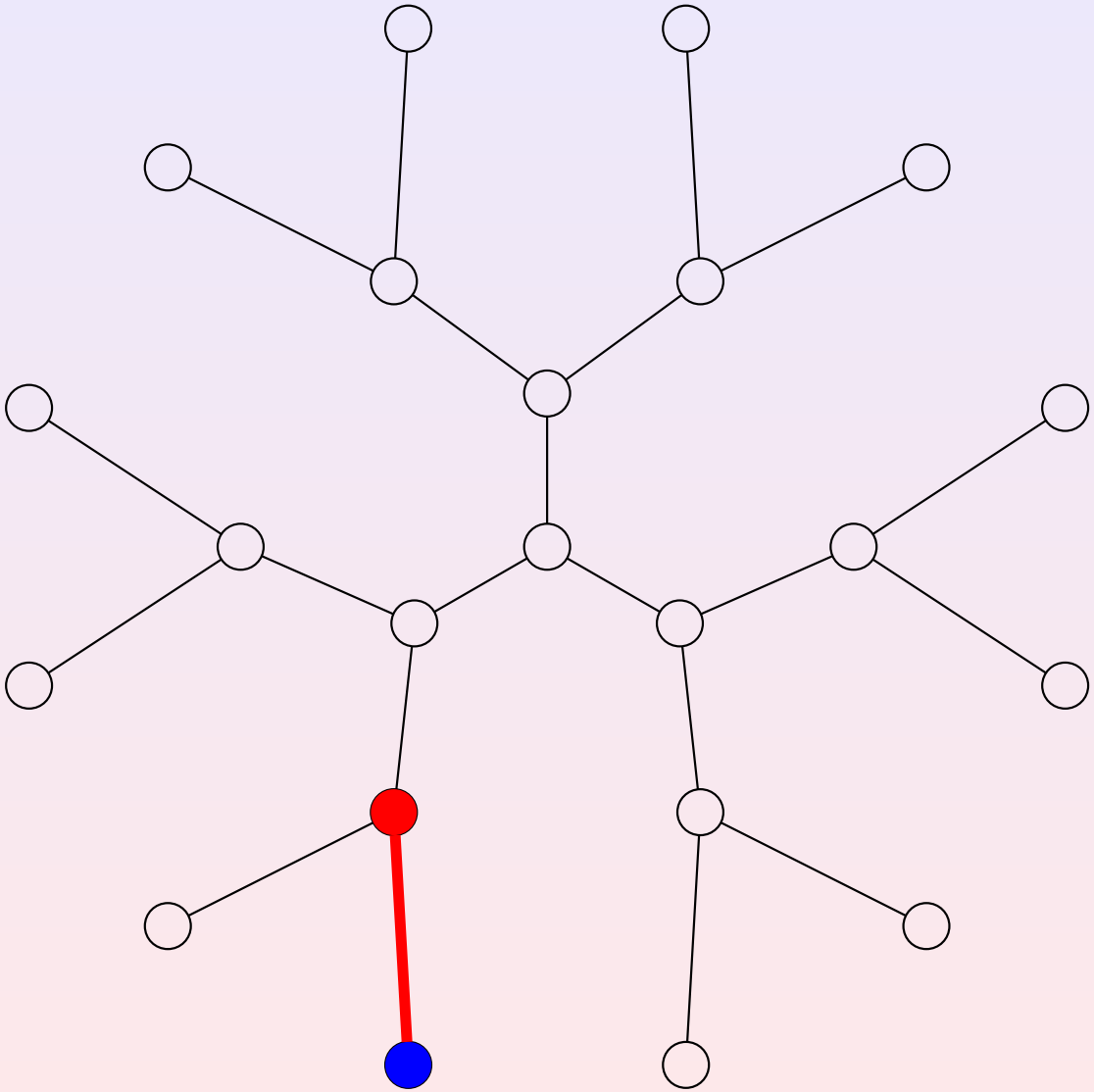
Encerclement dans un arbre



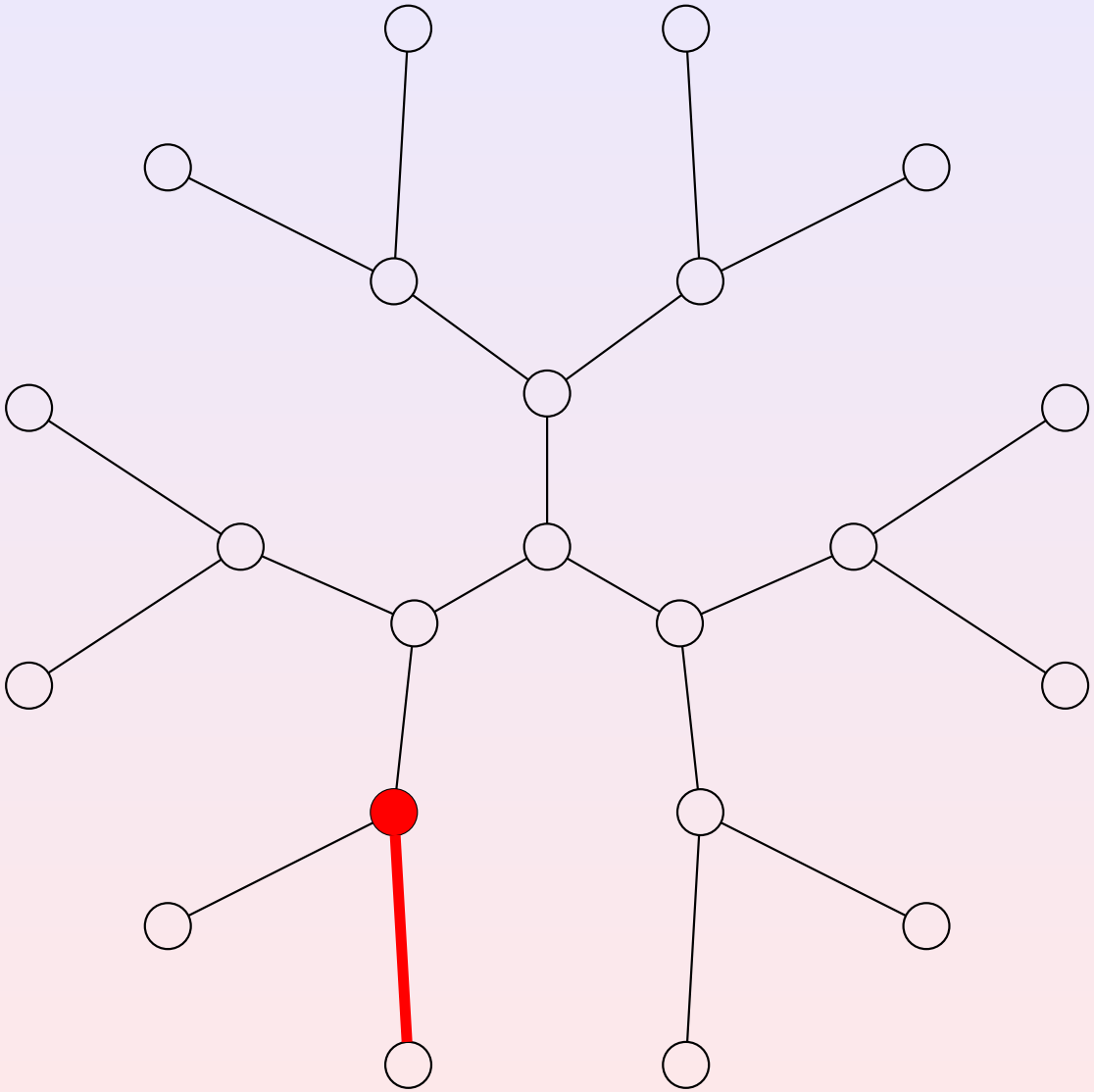
Encerclement dans un arbre



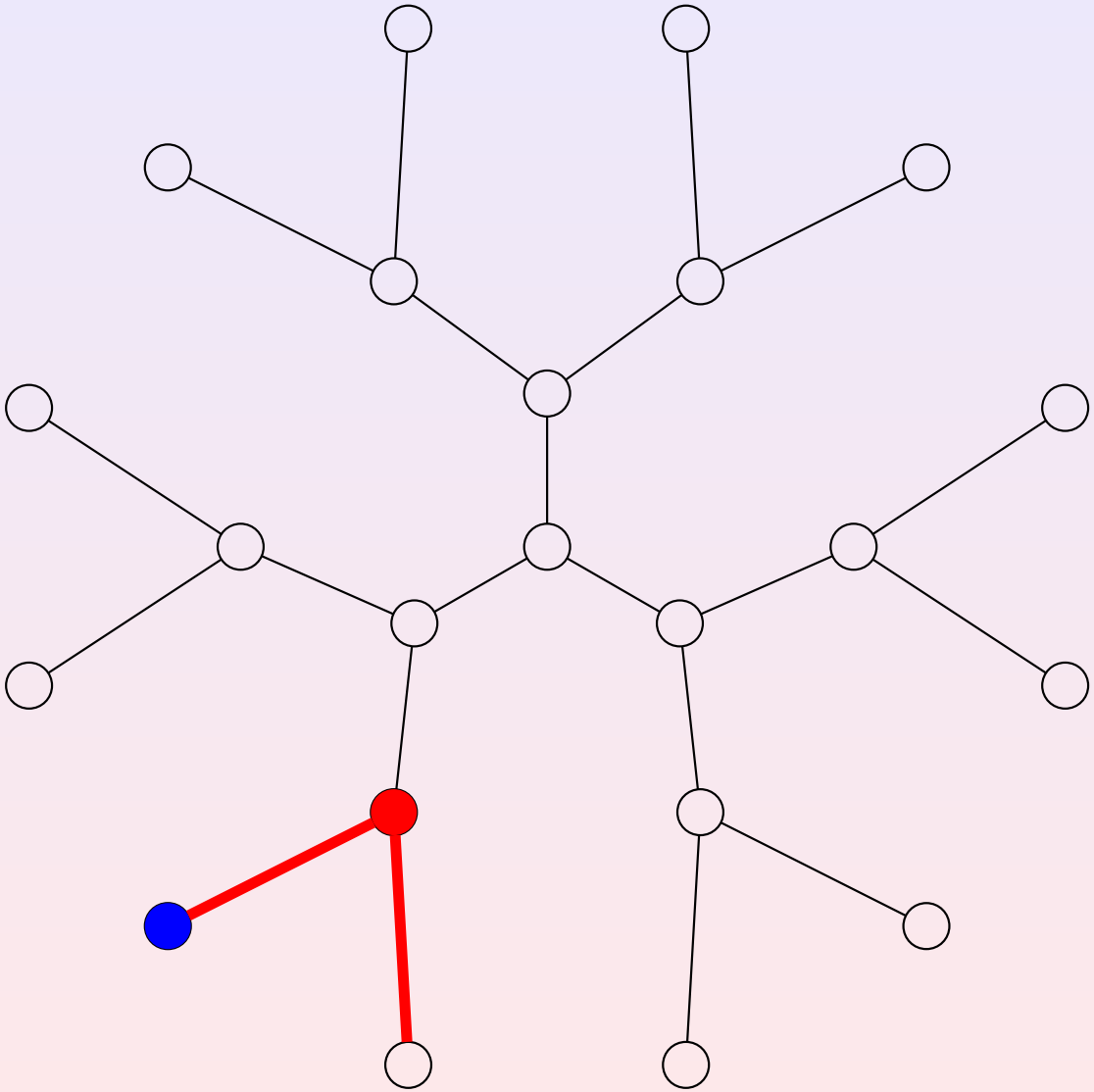
Encerclement dans un arbre



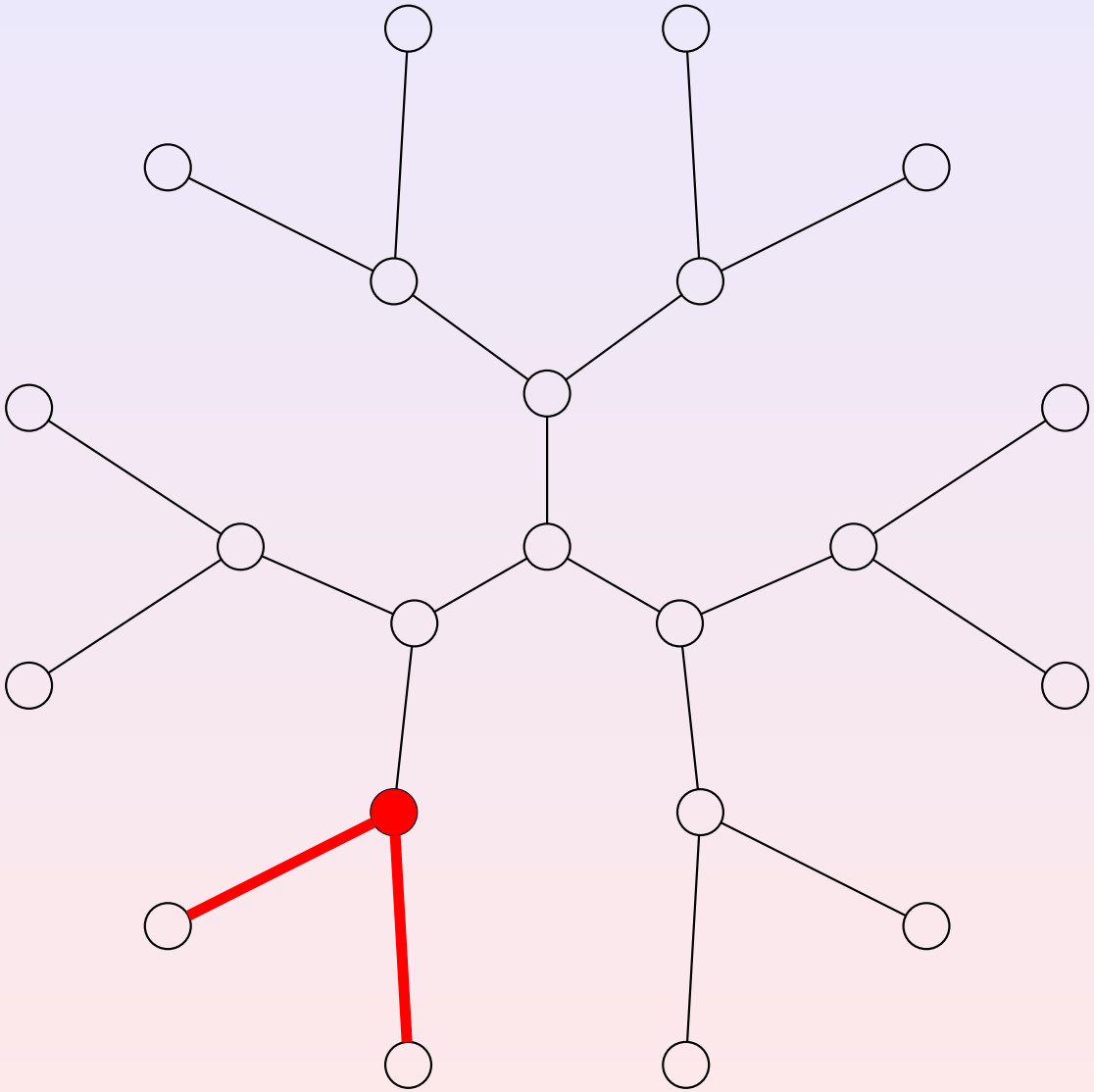
Encerclement dans un arbre



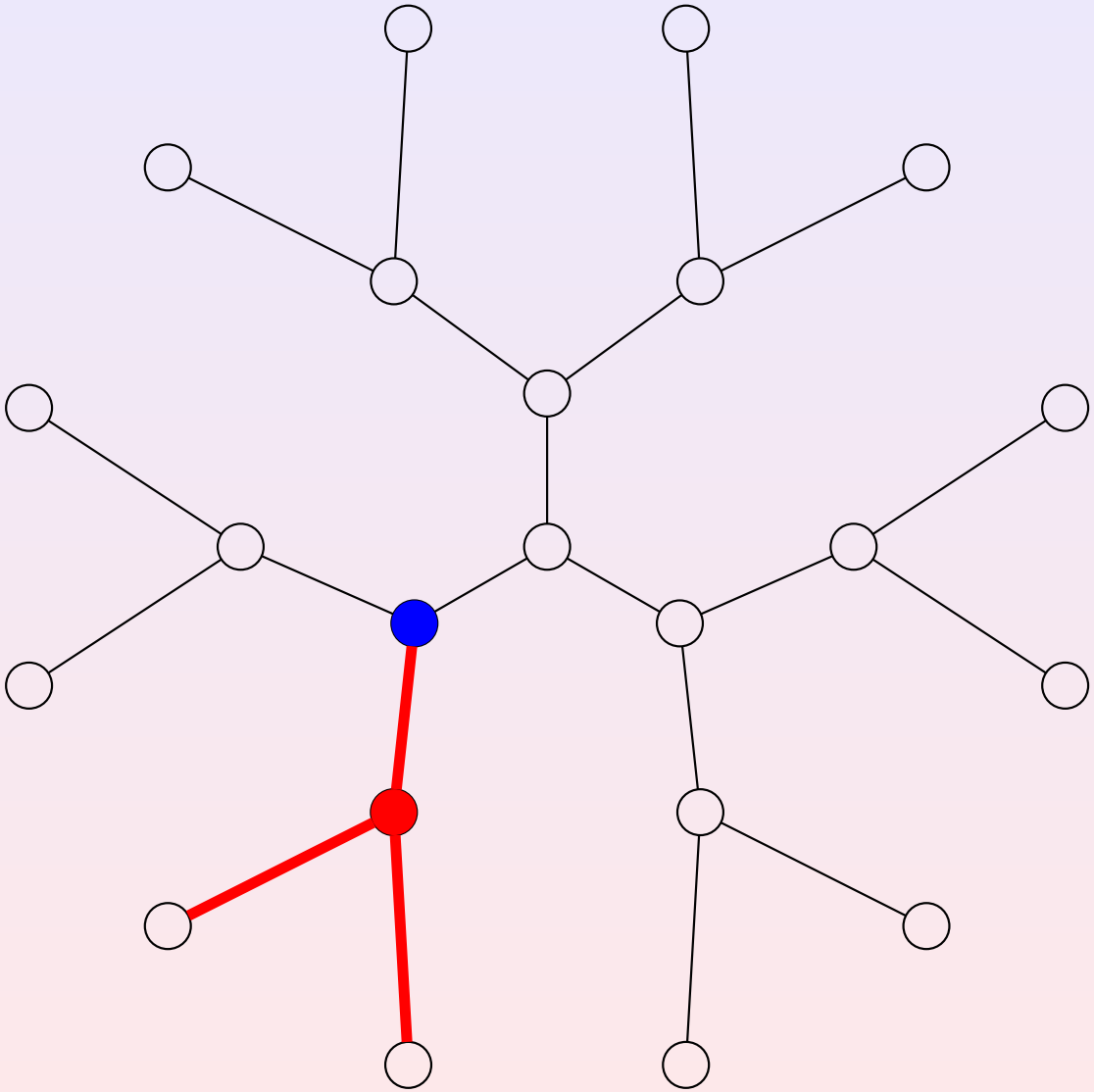
Encerclement dans un arbre



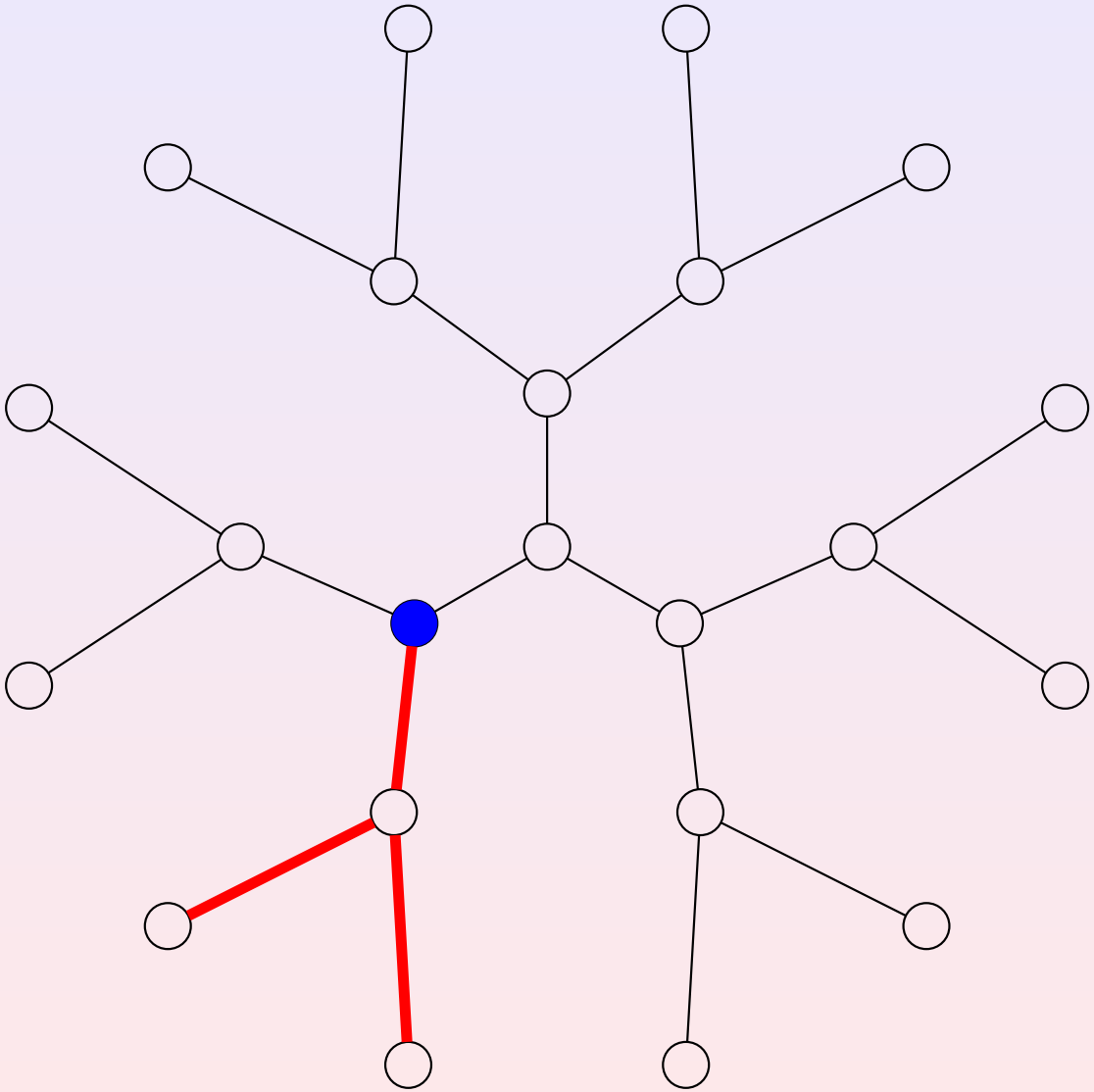
Encerclement dans un arbre



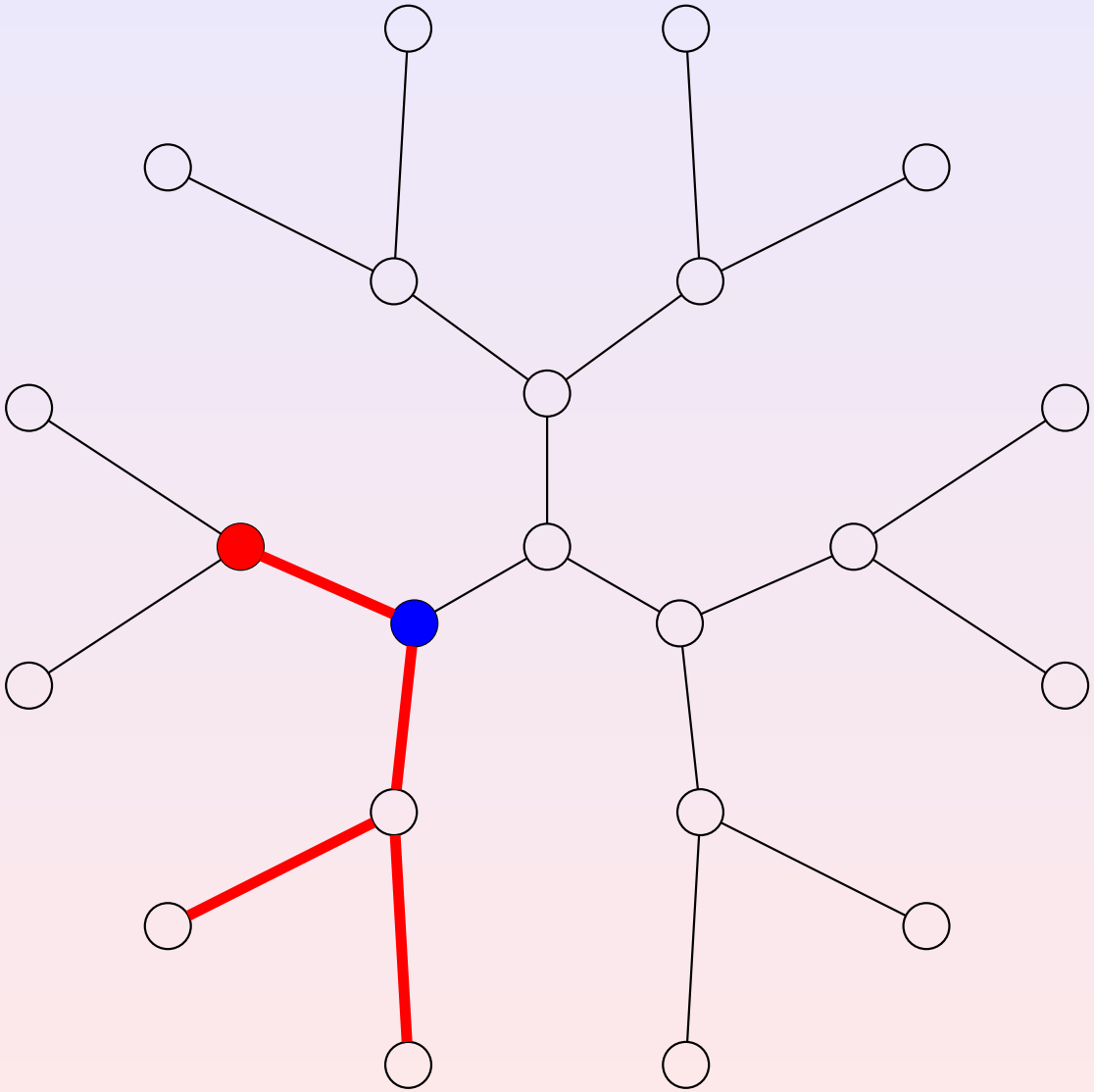
Encerclement dans un arbre



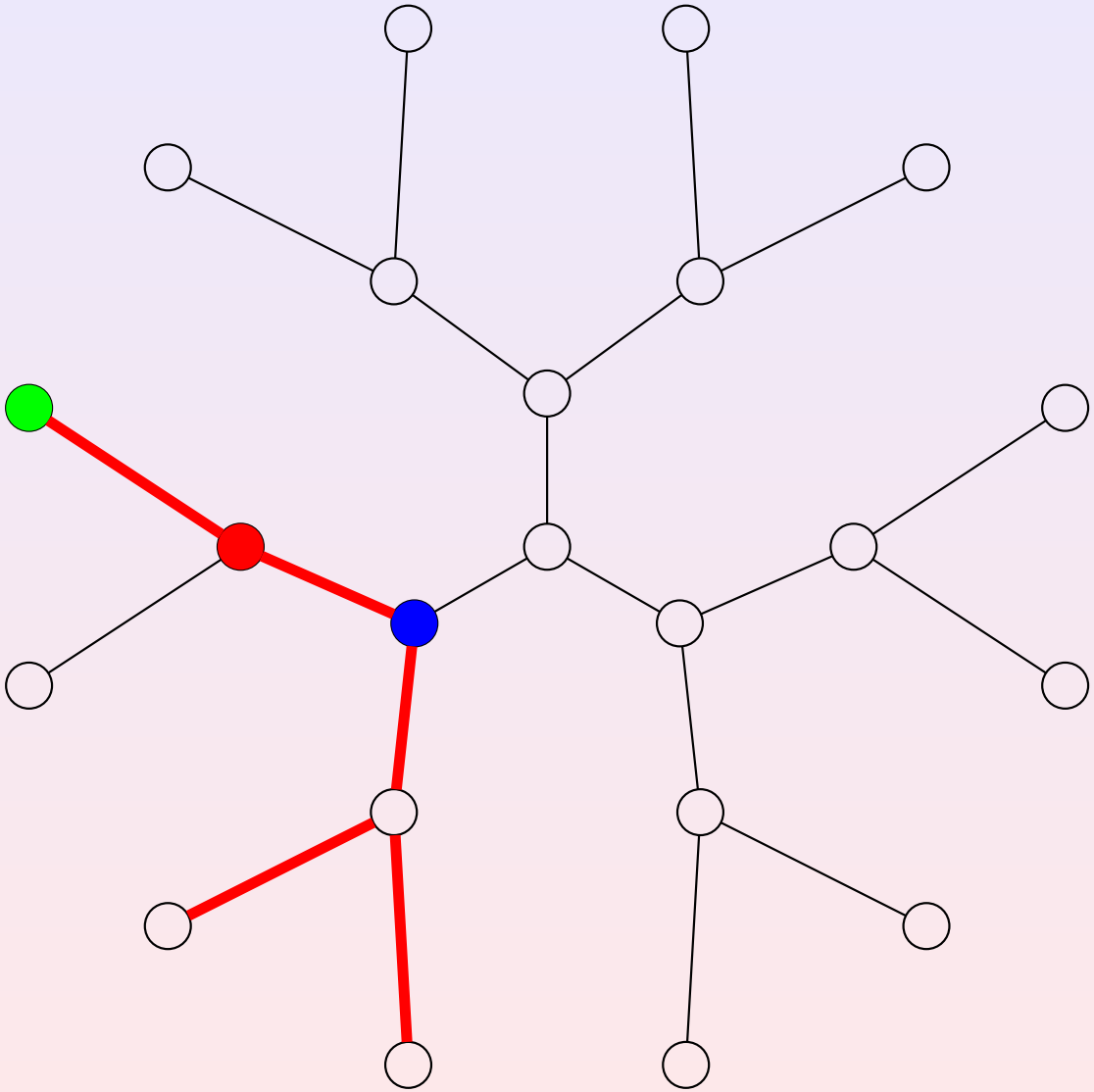
Encerclement dans un arbre



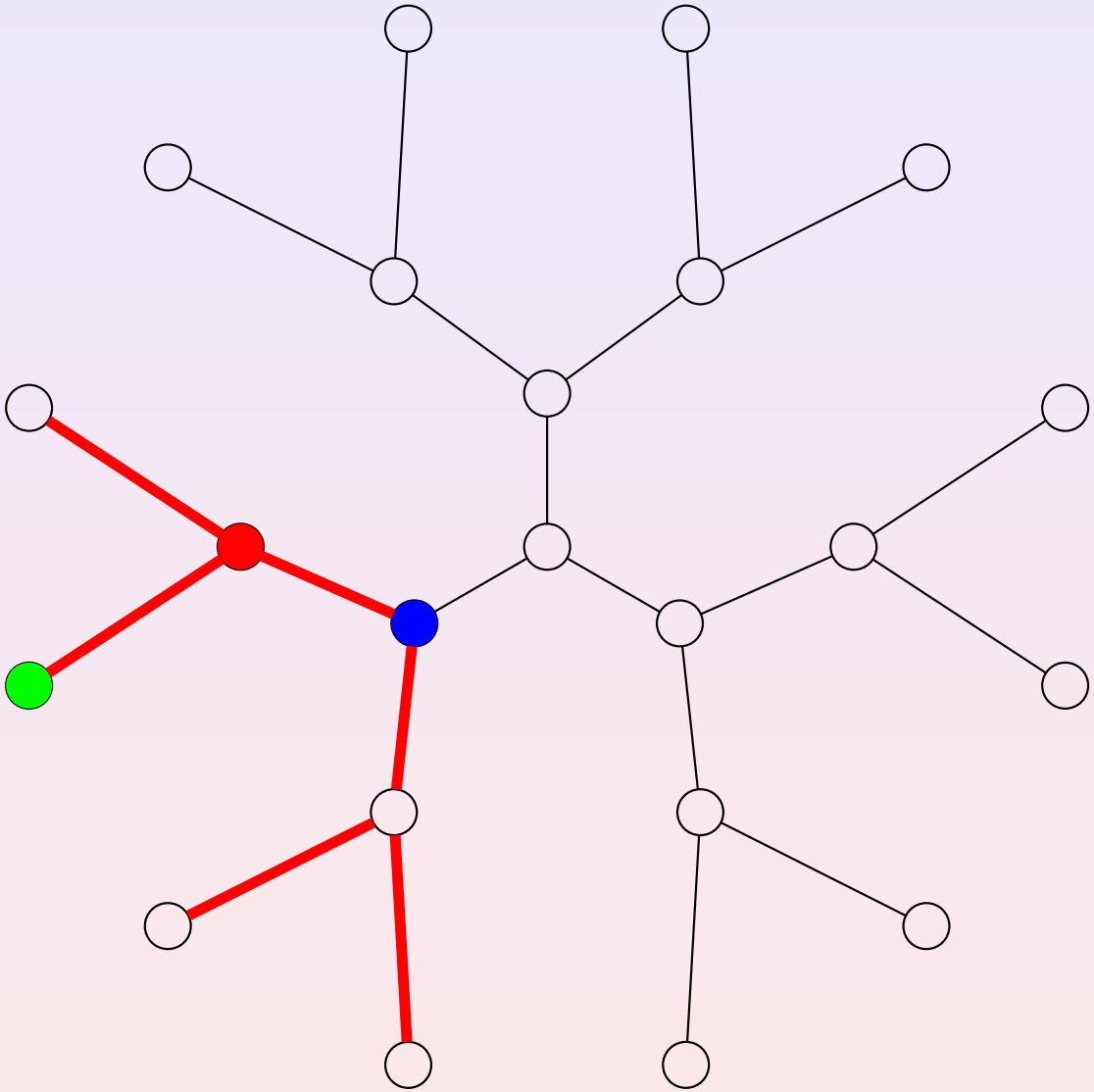
Encerclement dans un arbre



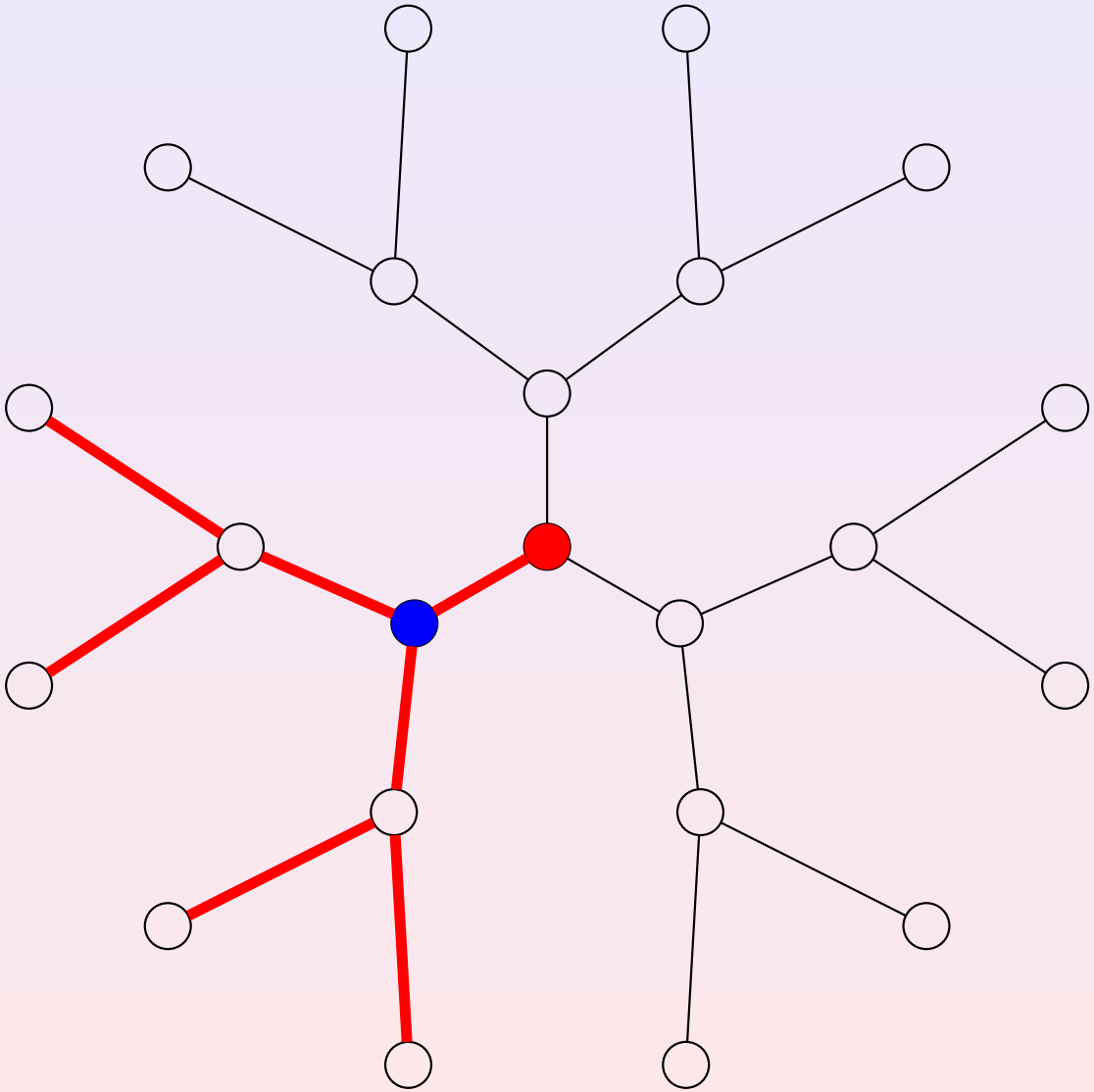
Encerclement dans un arbre



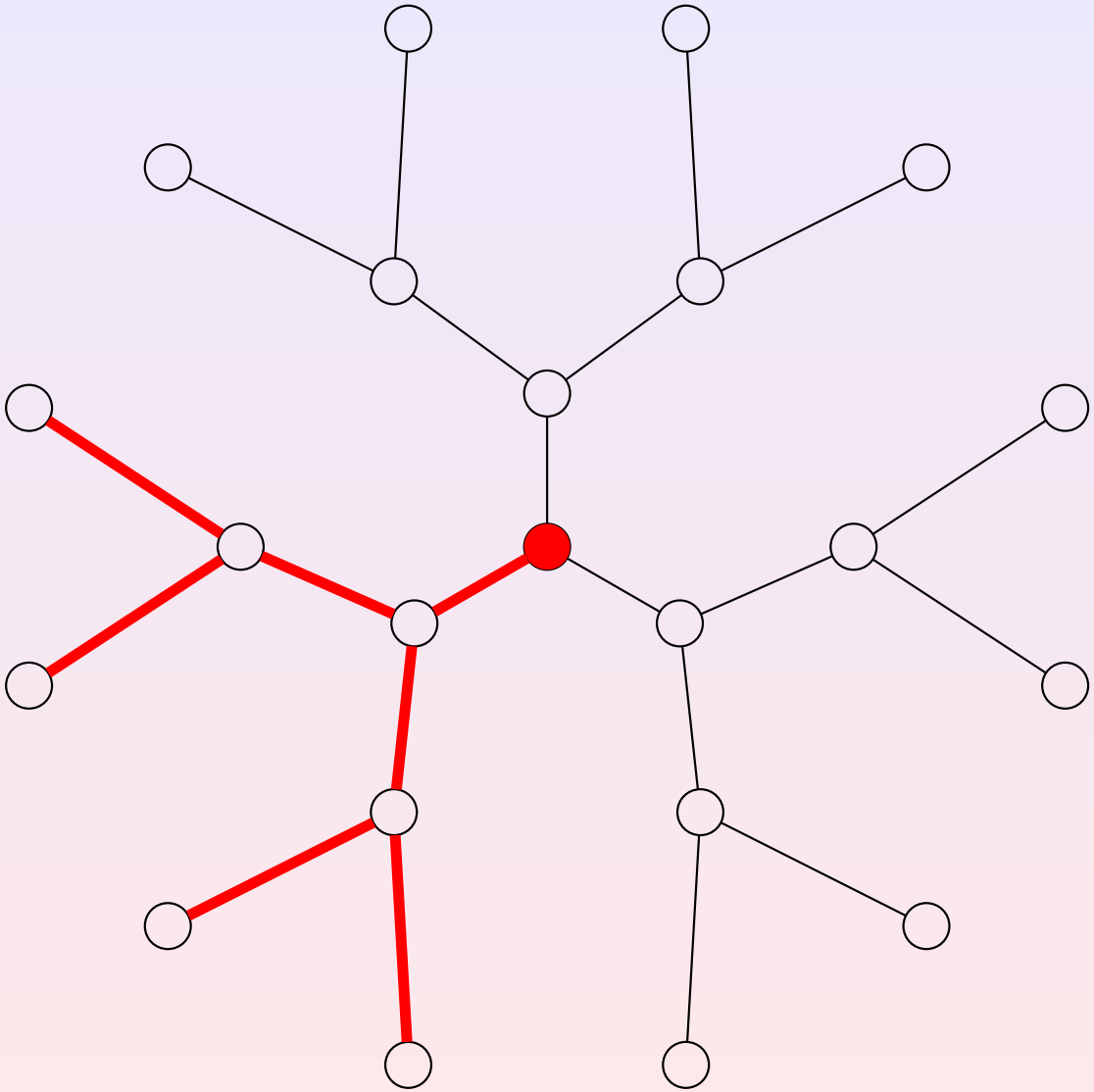
Encerclement dans un arbre



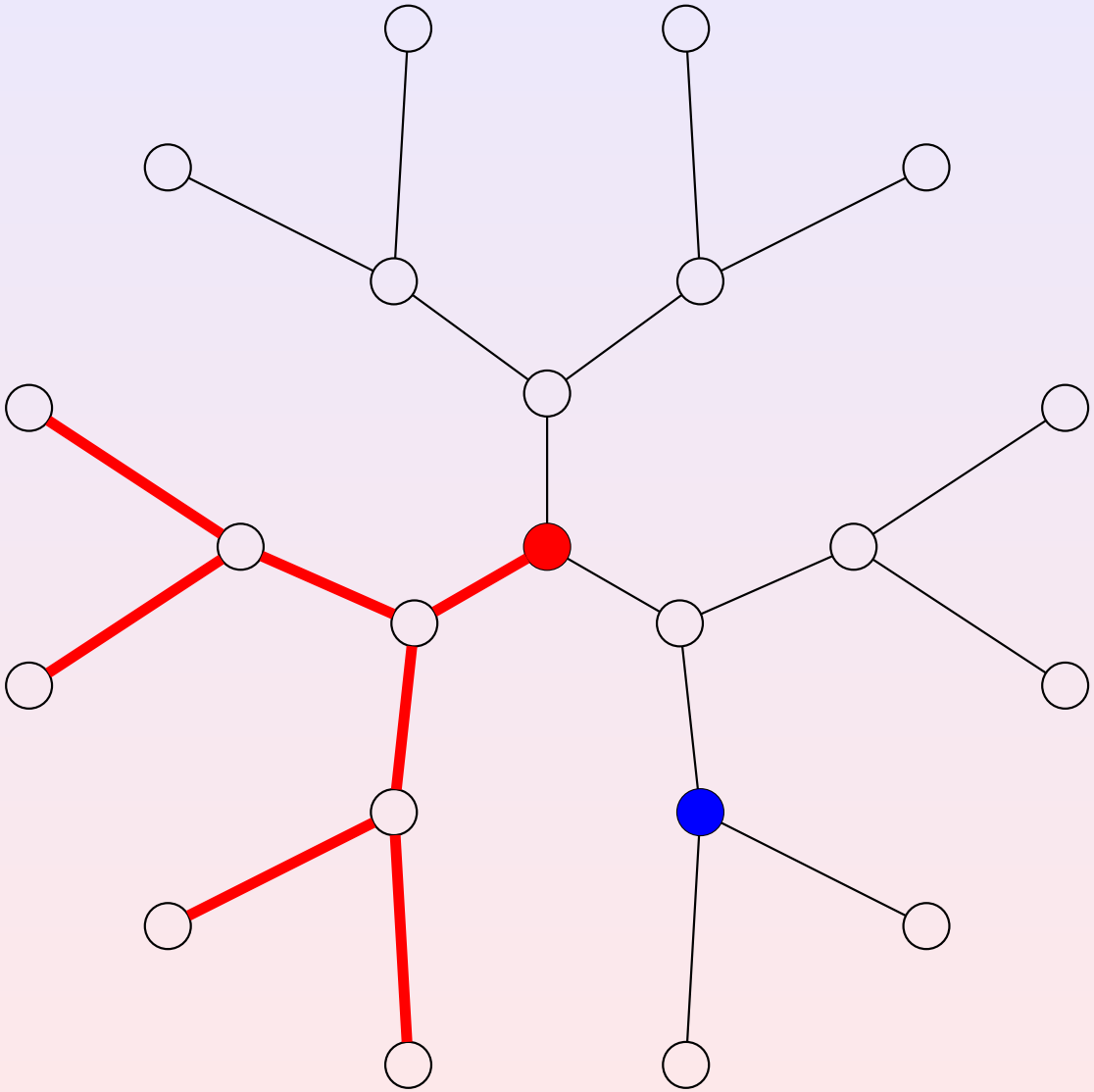
Encerclement dans un arbre



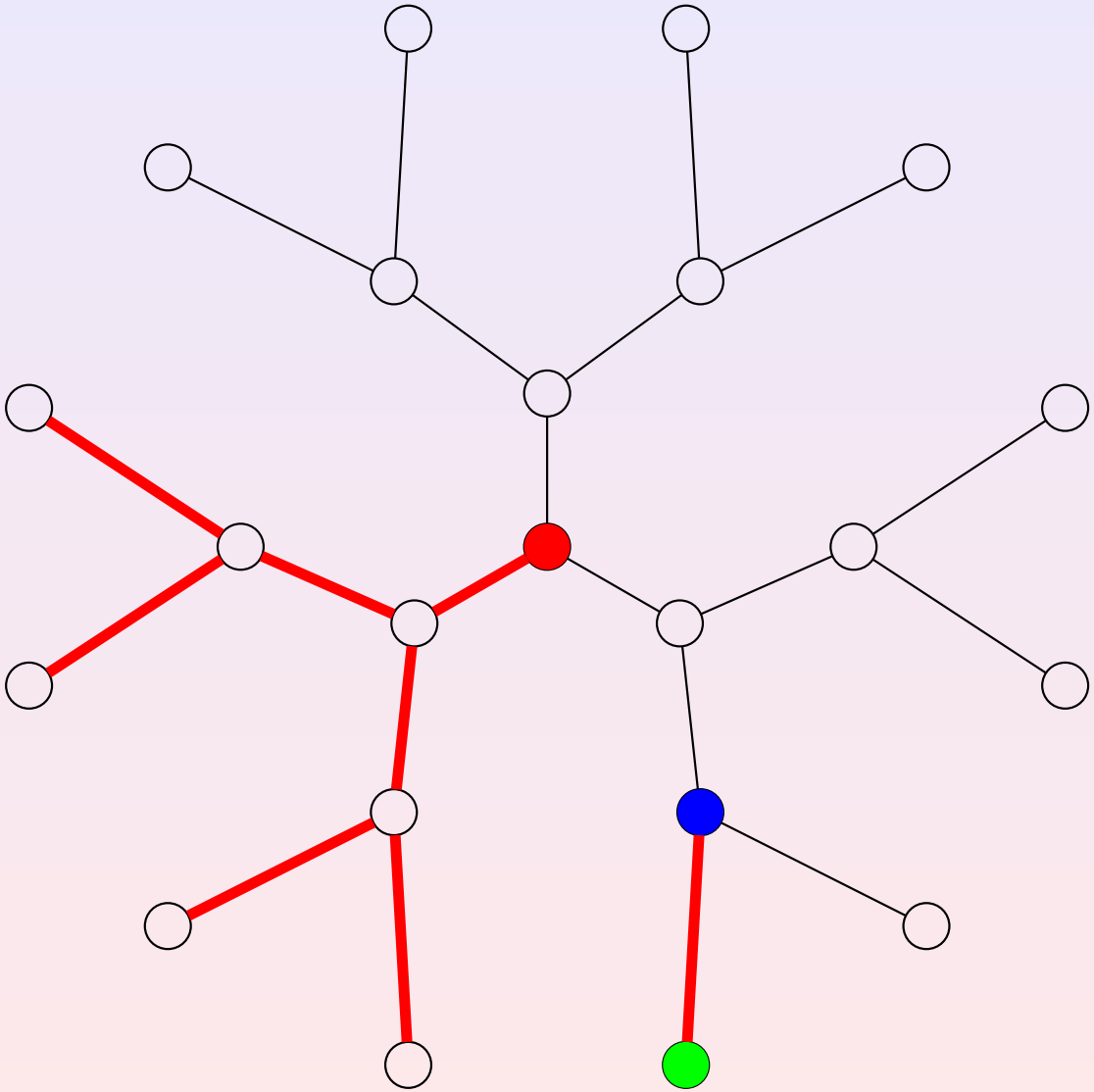
Encerclement dans un arbre



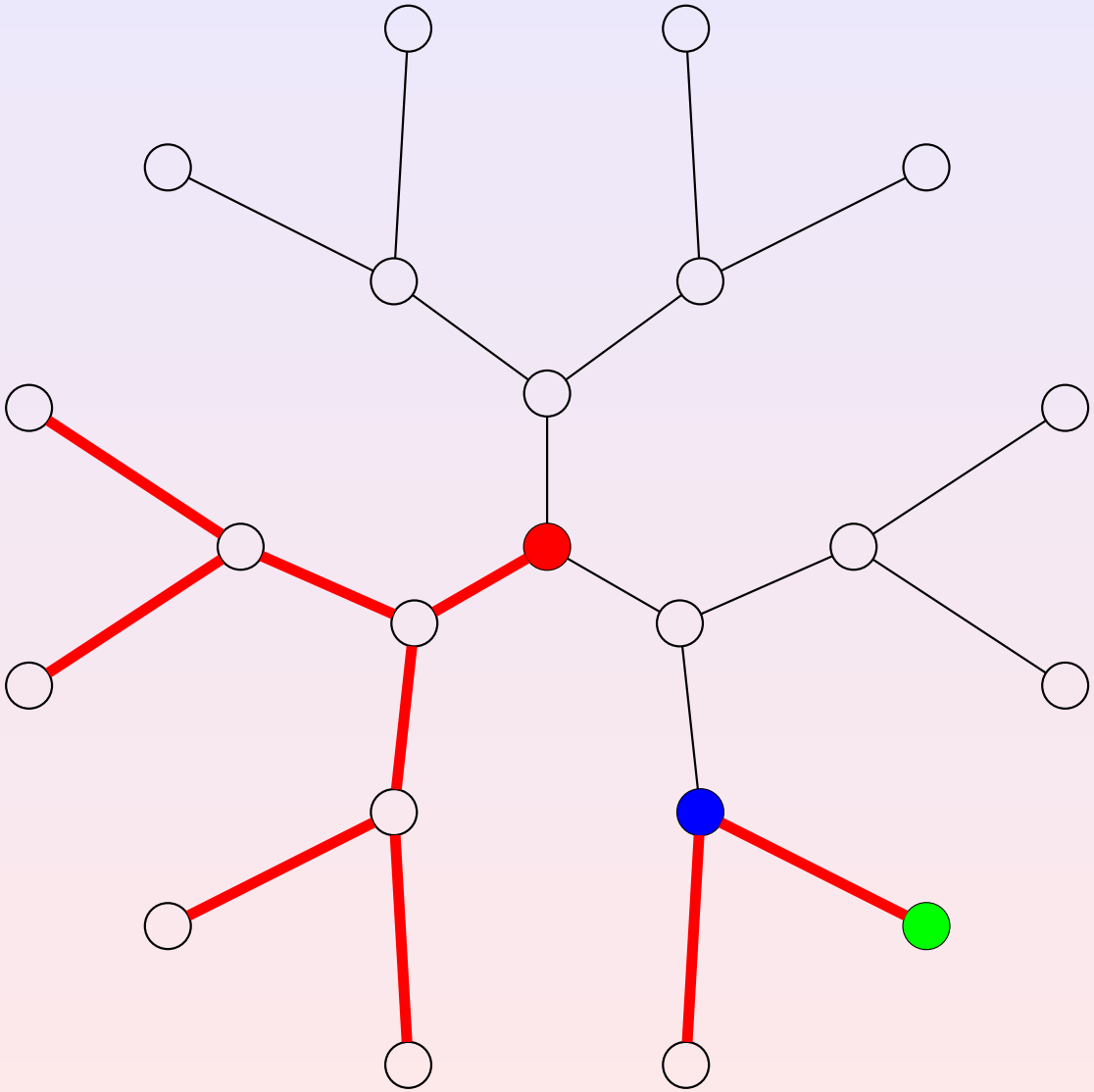
Encerclement dans un arbre



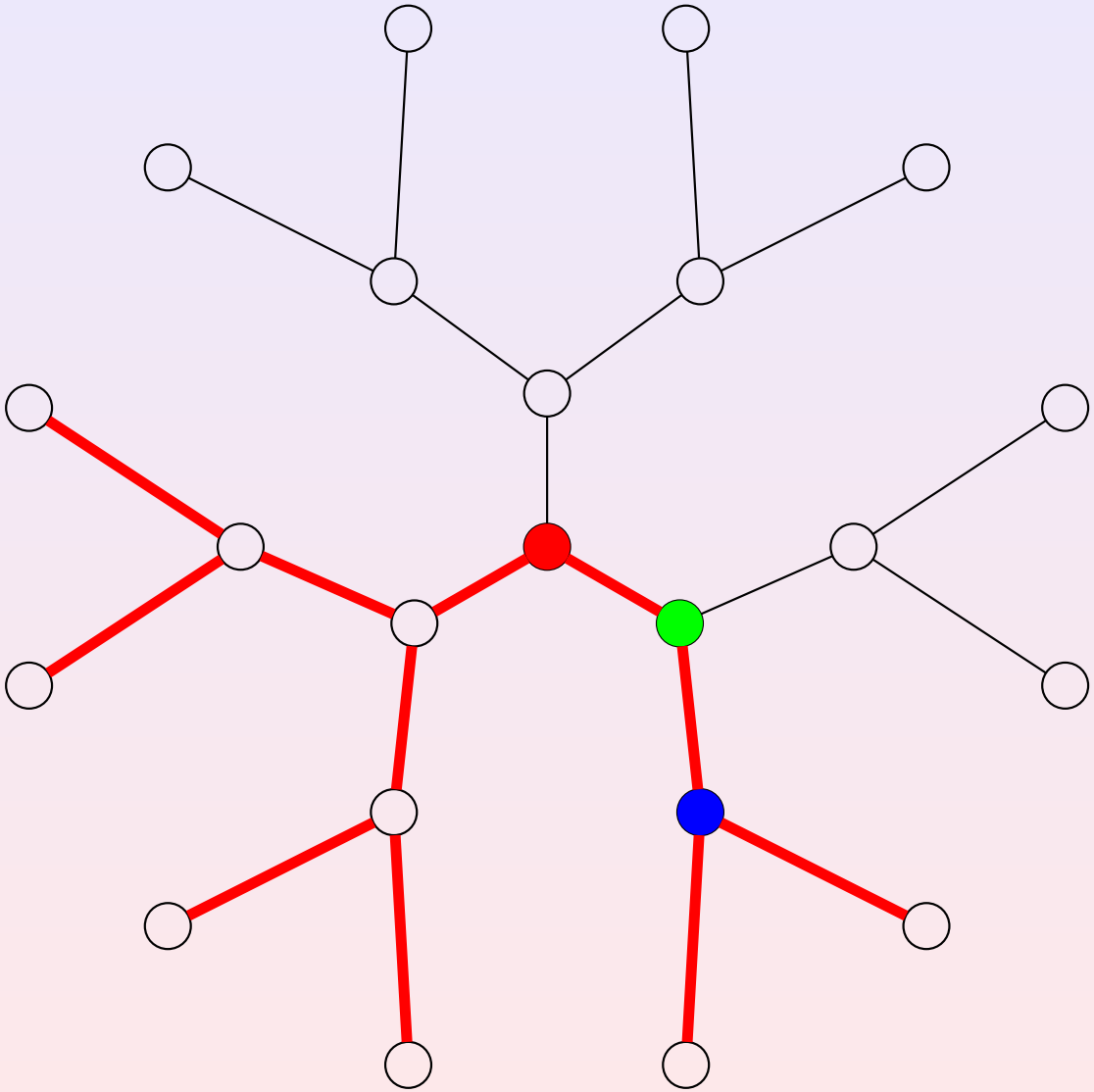
Encerclement dans un arbre



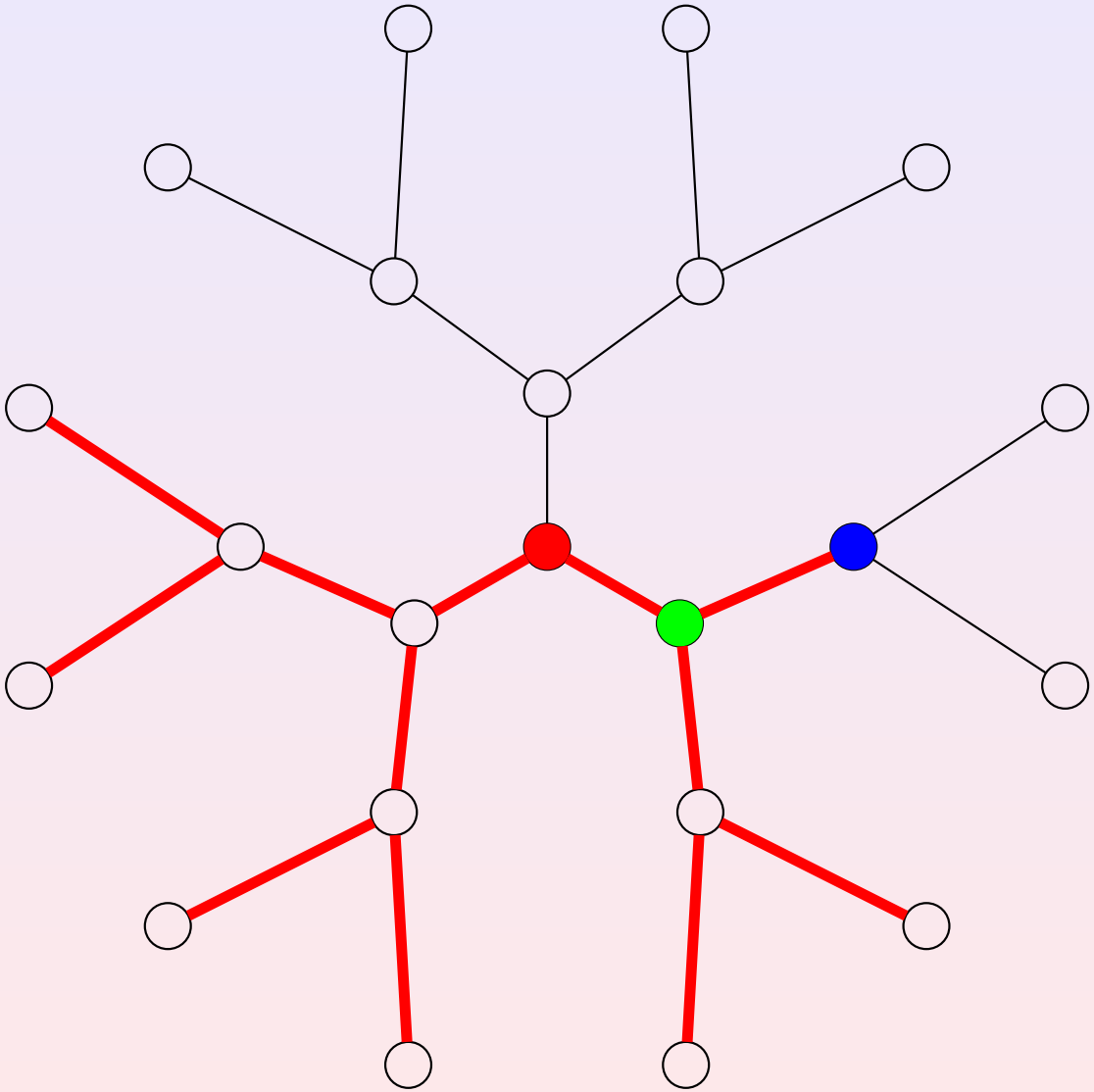
Encerclement dans un arbre



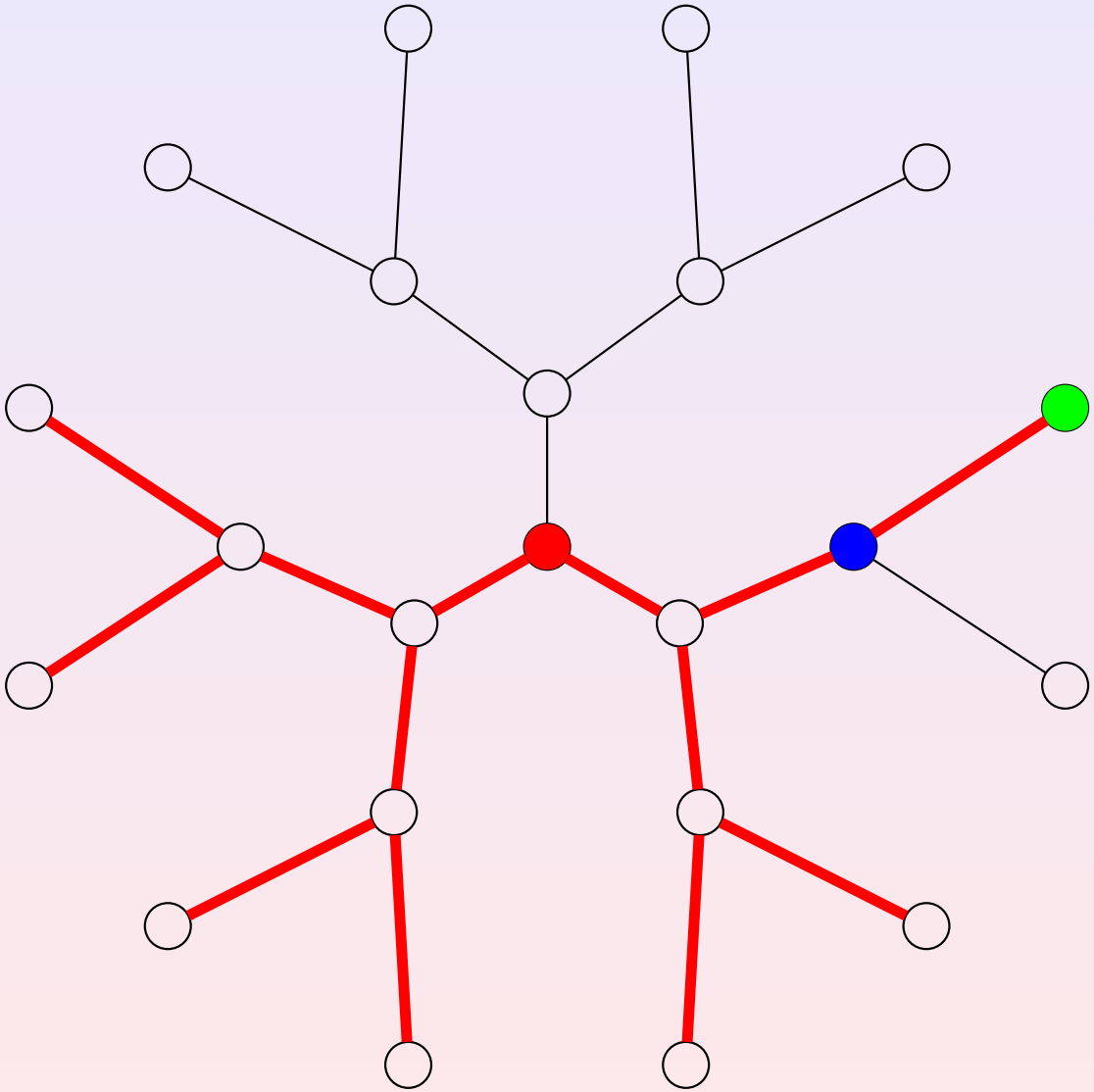
Encerclement dans un arbre



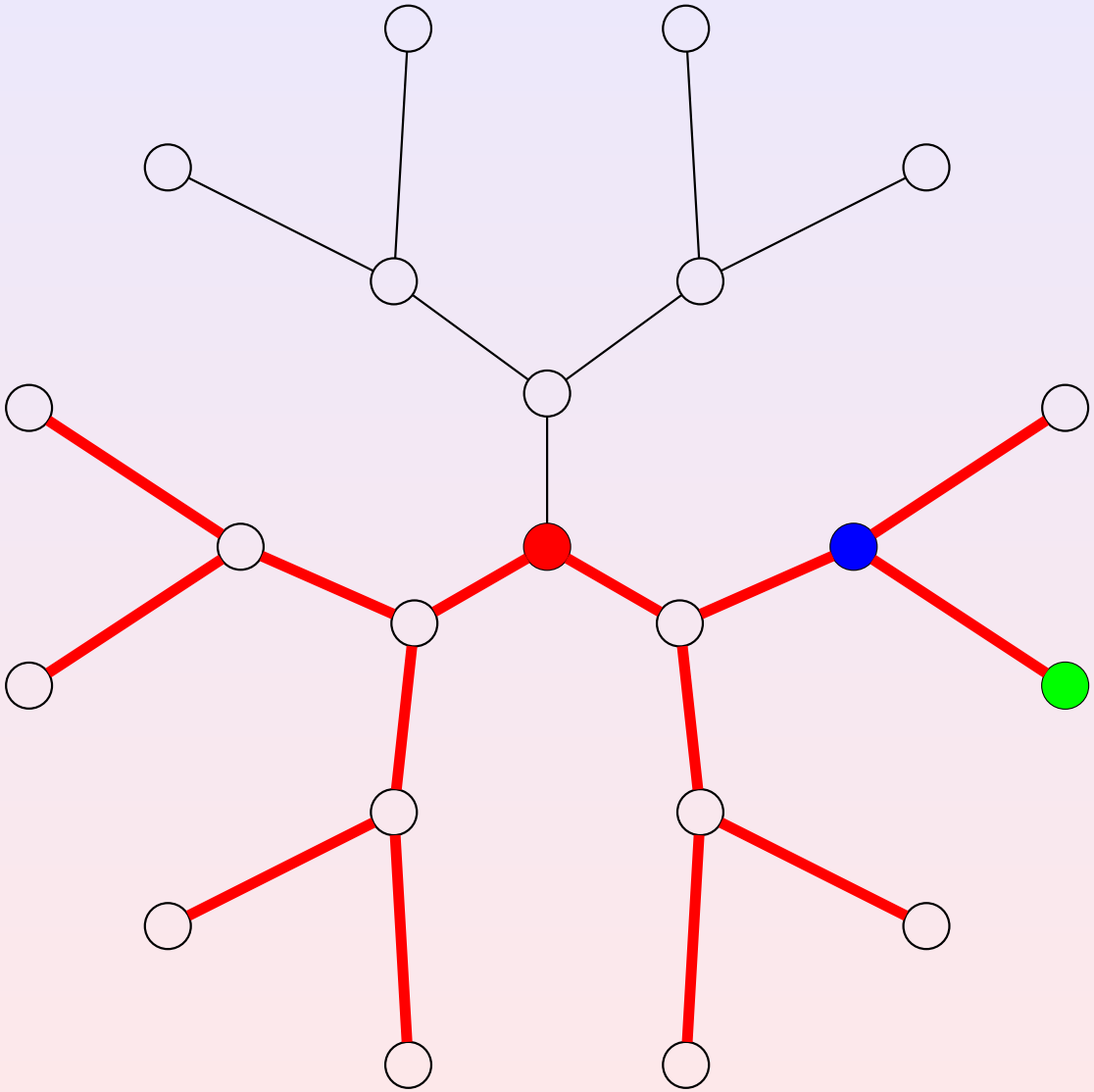
Encerclement dans un arbre



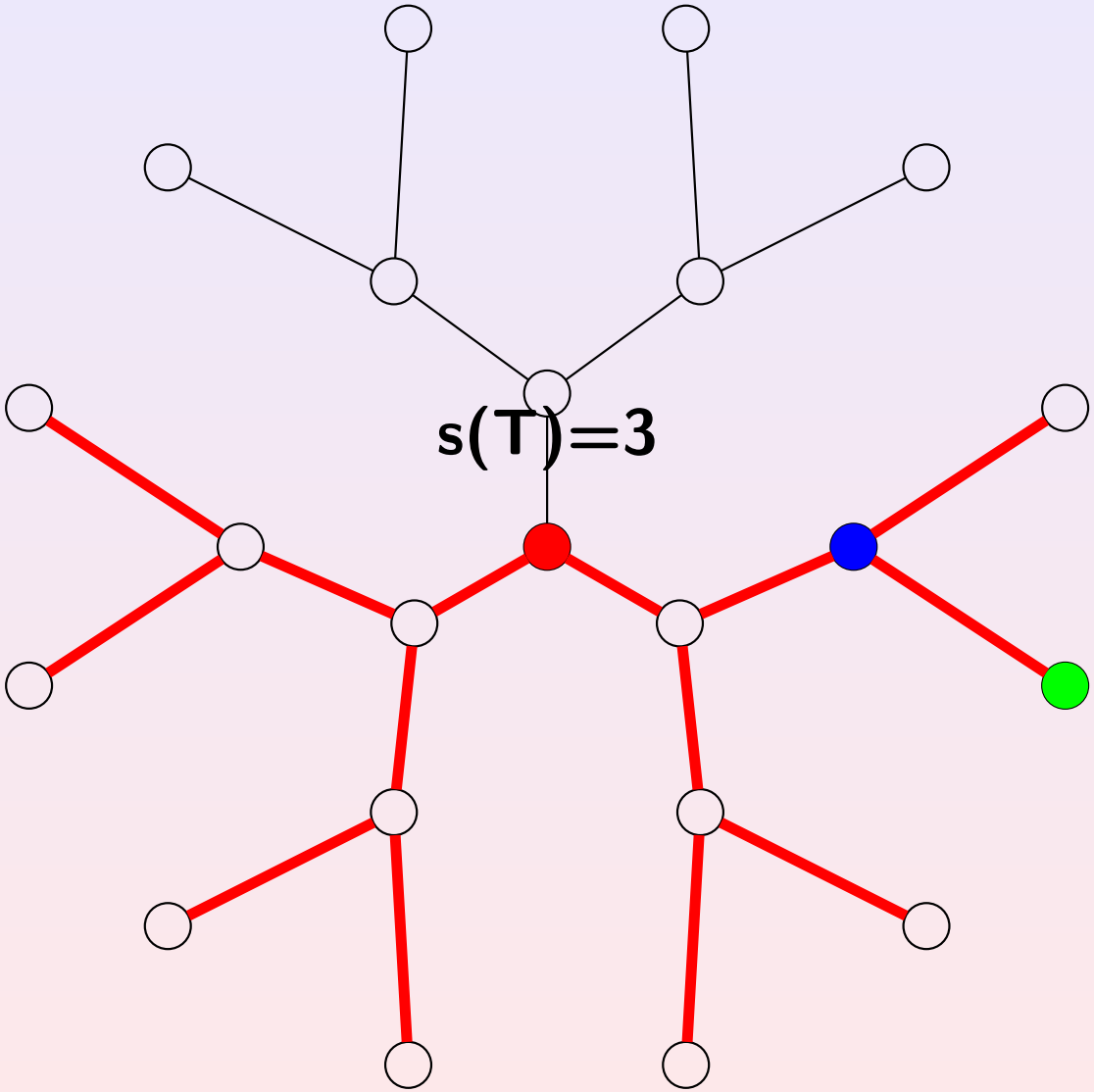
Encerclement dans un arbre



Encerclement dans un arbre



Encerclement dans un arbre



Encerclement visible

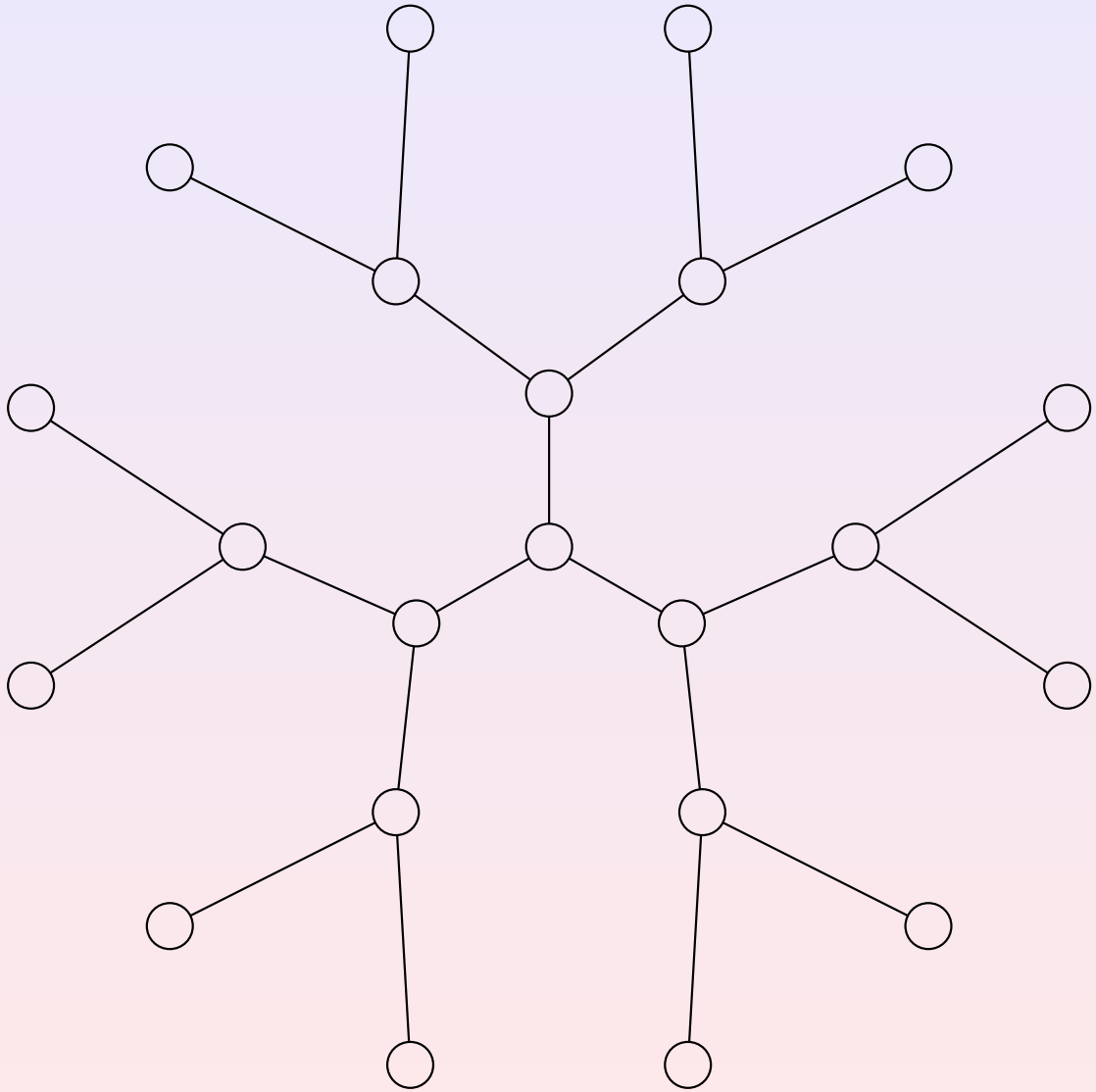
Visibilité du fugitif

Le fugitif est visible si, à chaque étape, les agents connaissent sa position (en fait la composante connexe où il se trouve). La stratégie peut donc être orientée d'après la position du fugitif.

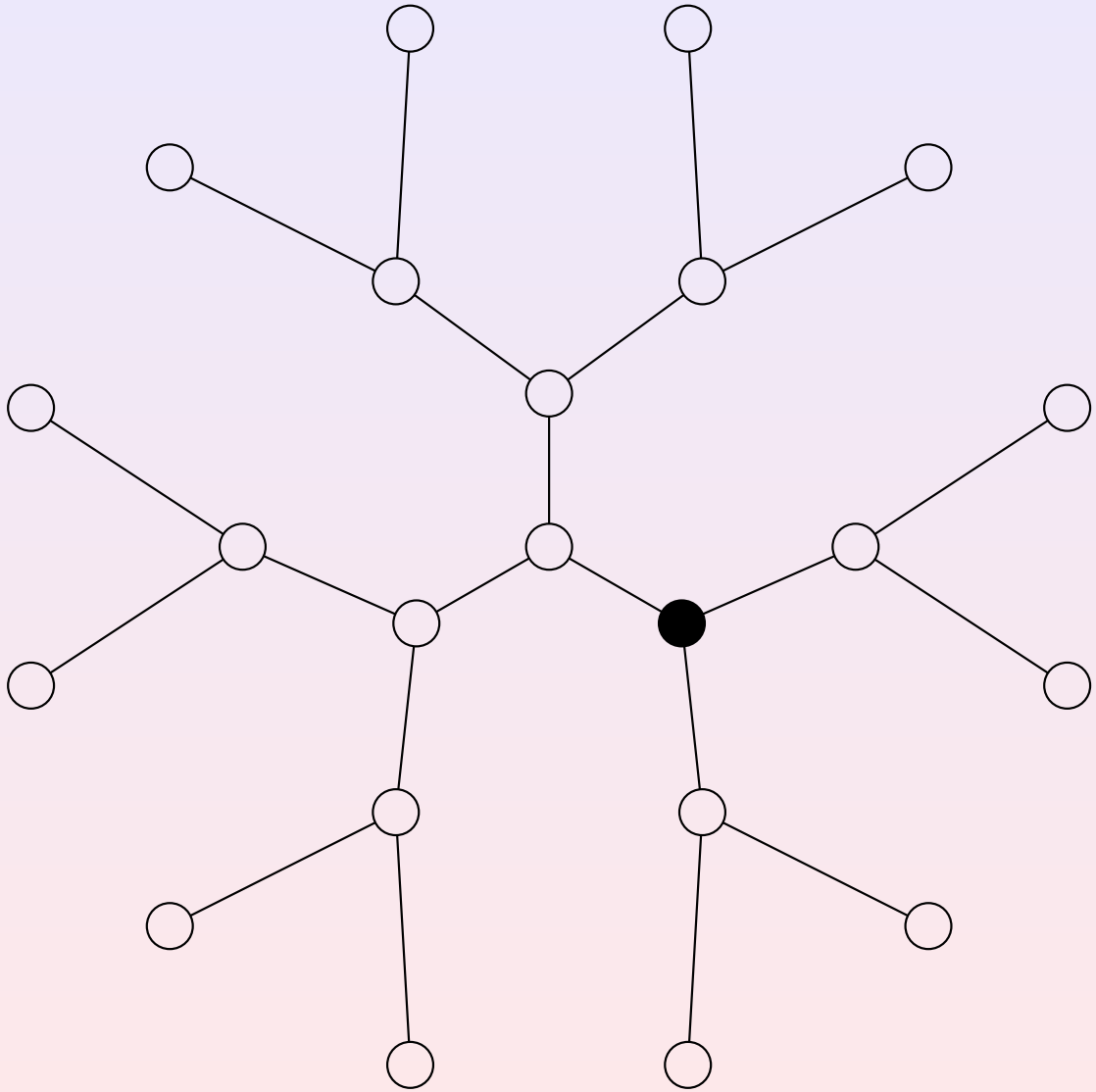
Paramètre associé

Soit **vs**(G) l'encerclement d'un fugitif visible dans le graphe G. De manière évidente, $vs(G) \leq s(G)$.

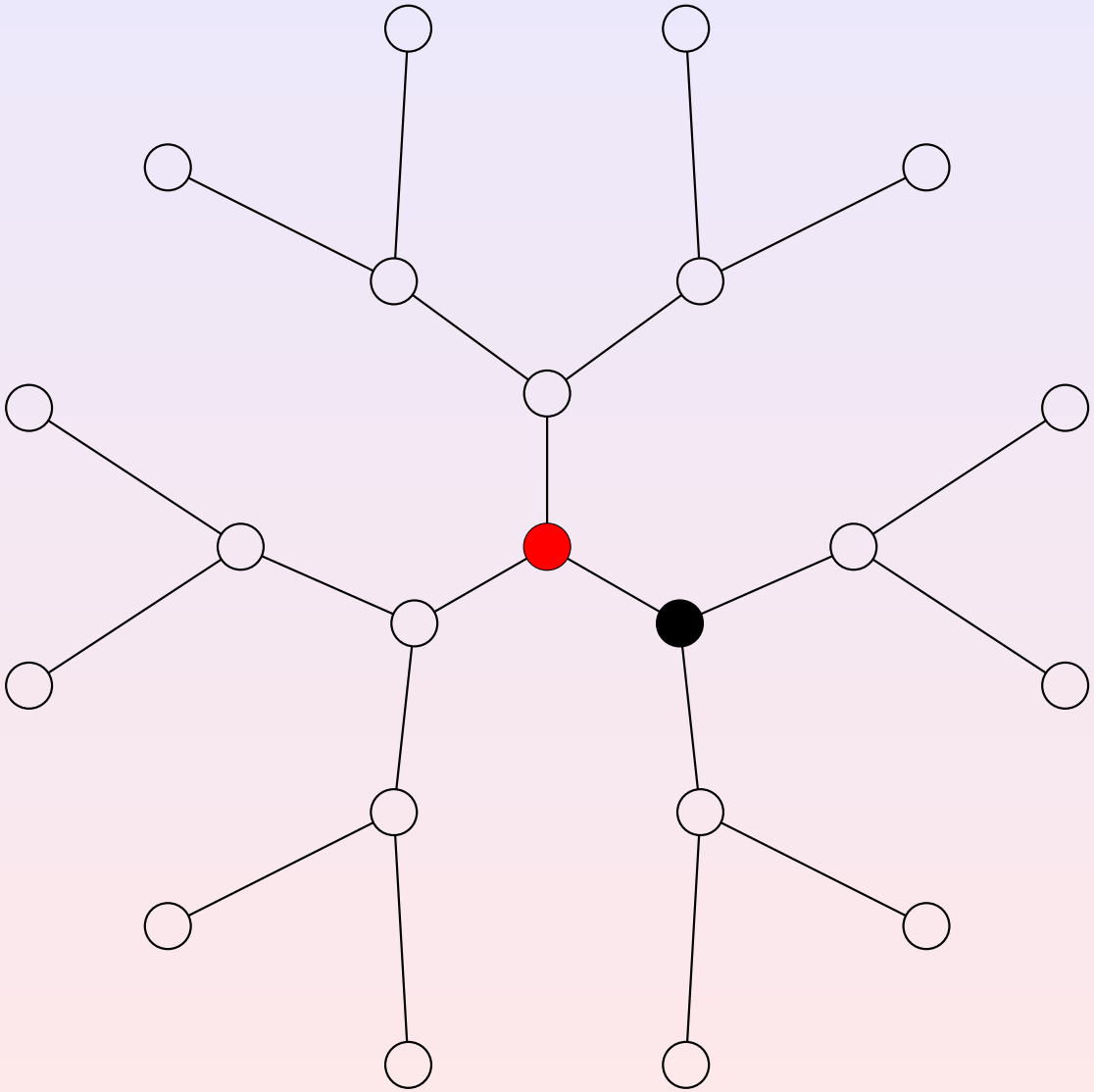
Encerclement d'un fugitif visible dans un arbre



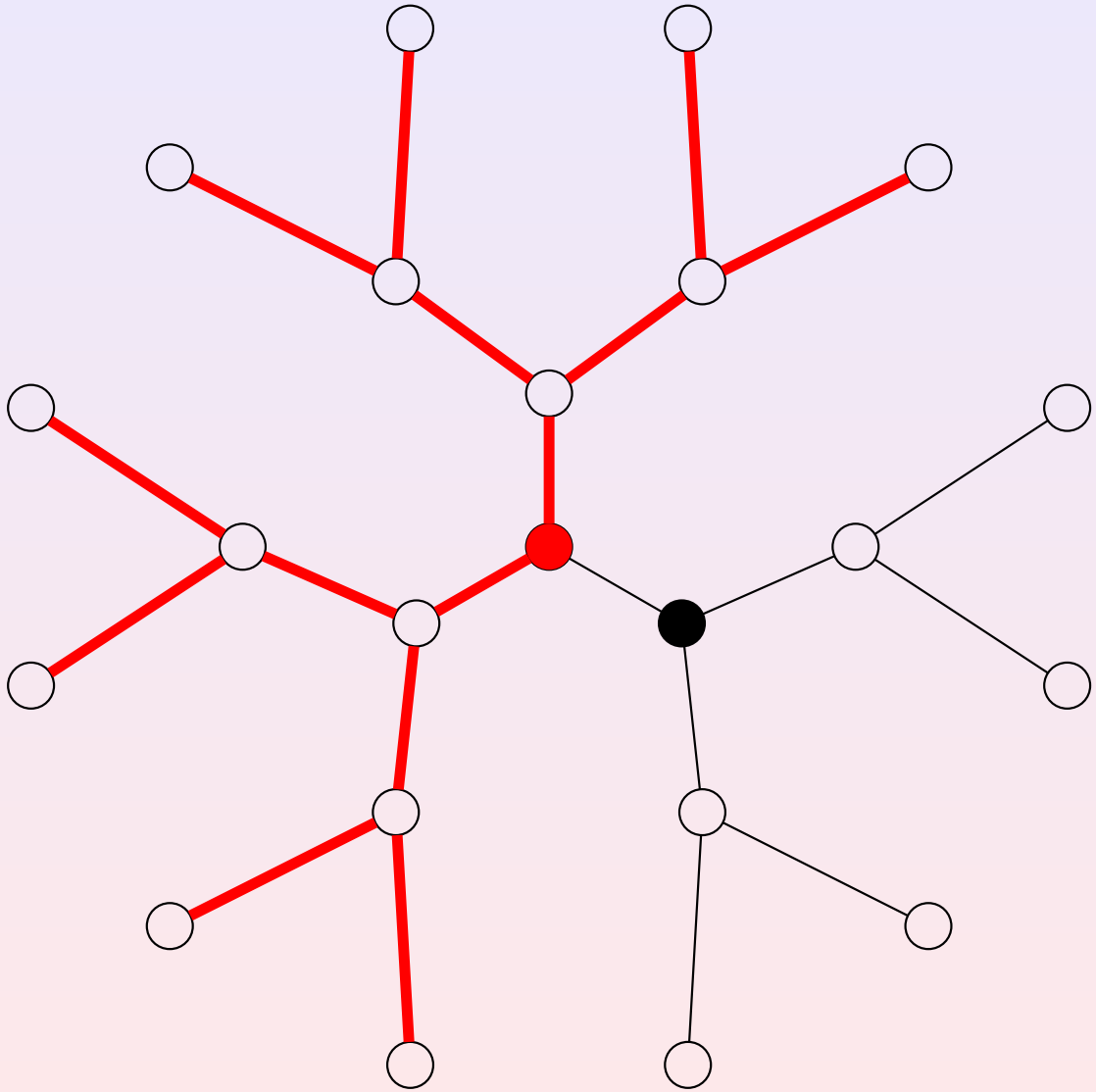
Encerclement d'un fugitif visible dans un arbre



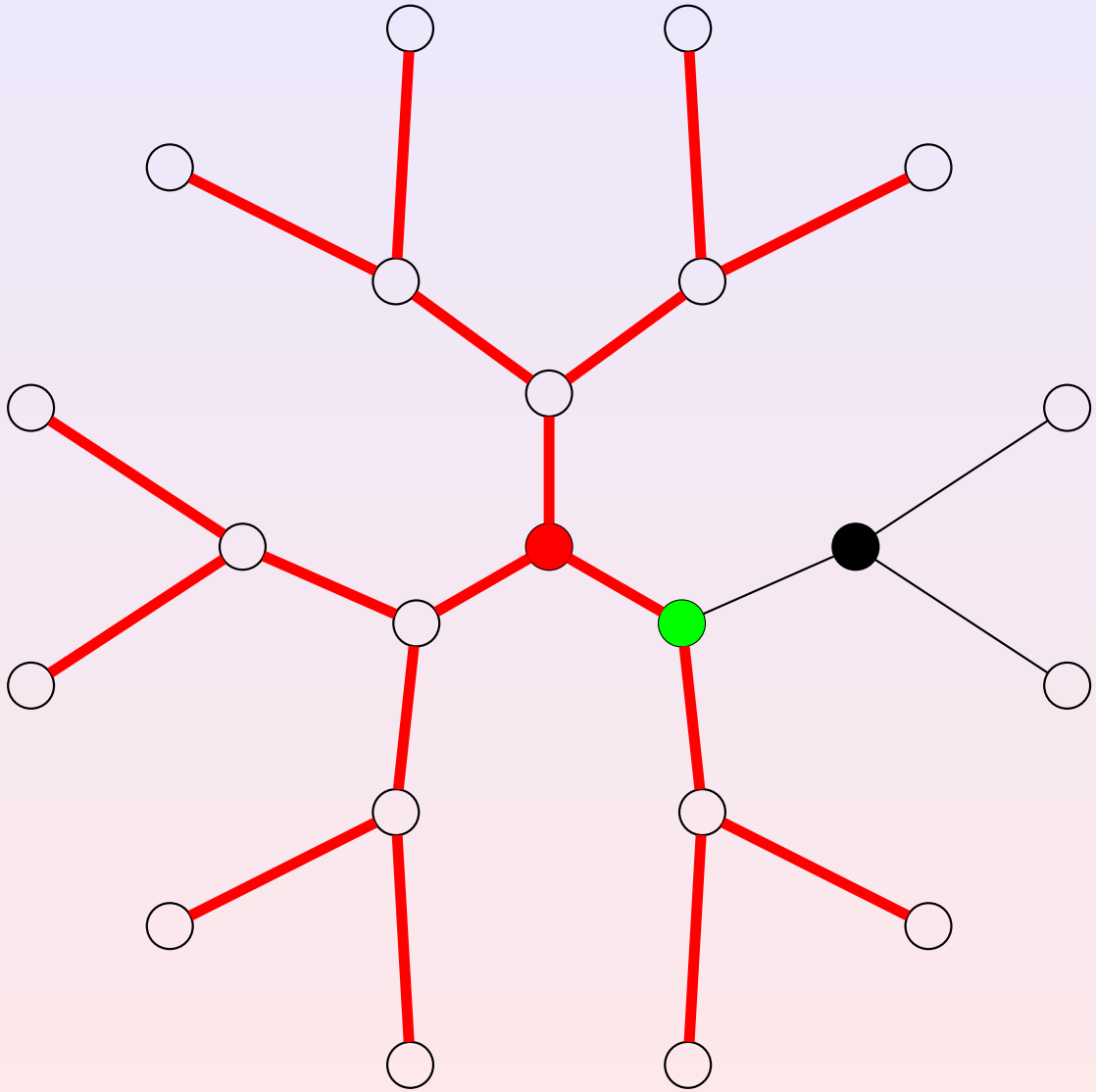
Encerclement d'un fugitif visible dans un arbre



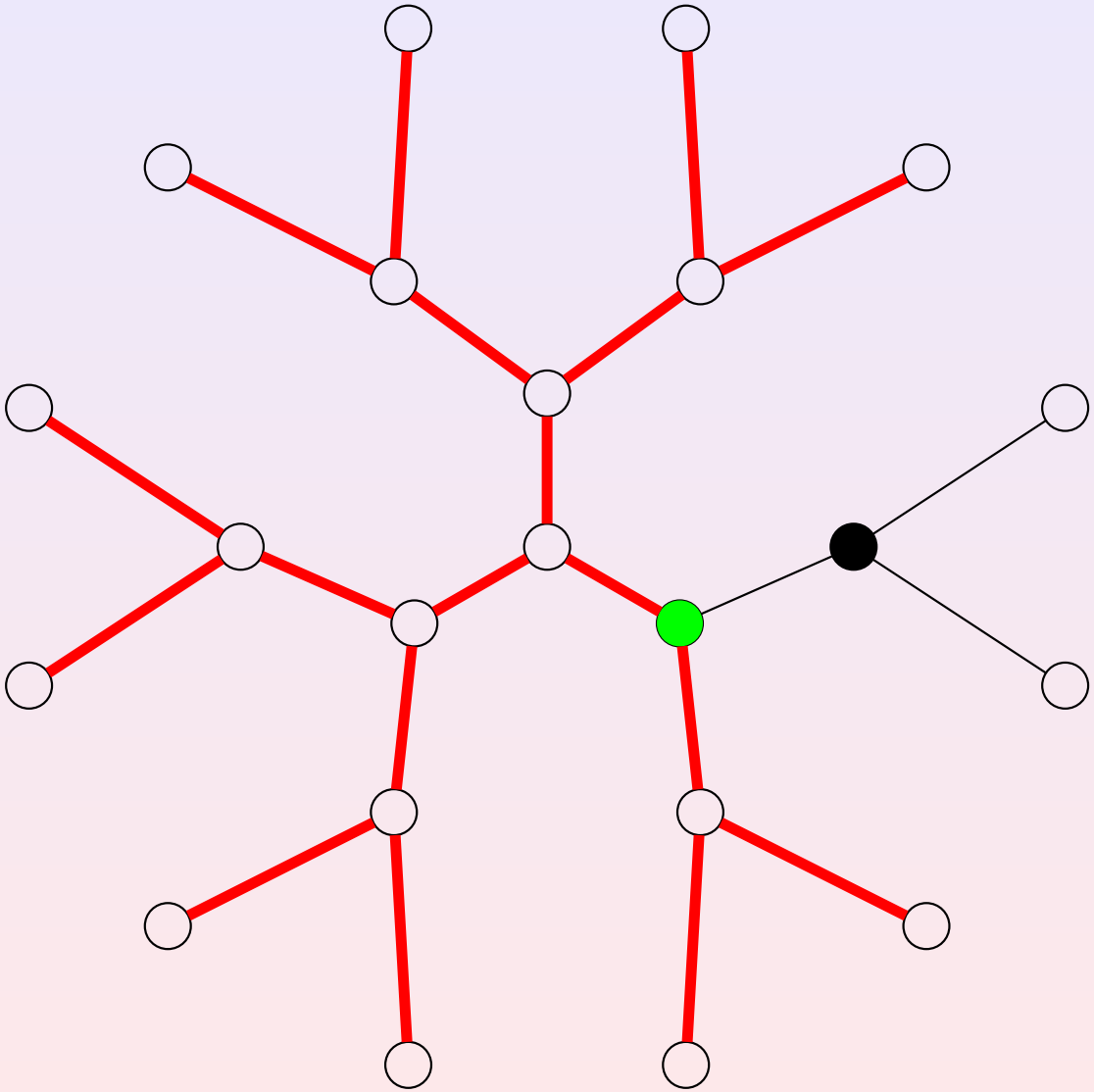
Encerclement d'un fugitif visible dans un arbre



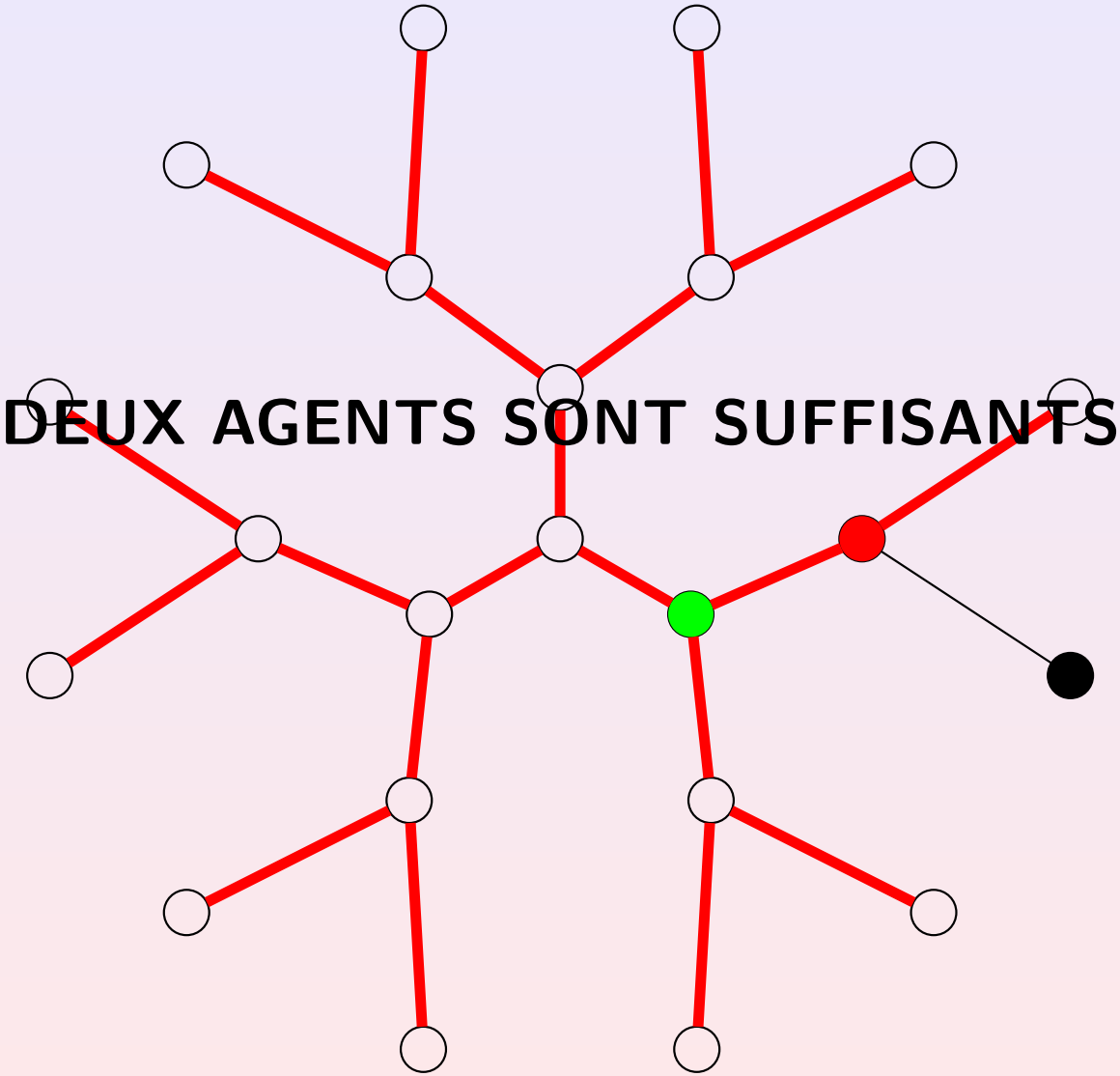
Encerclement d'un fugitif visible dans un arbre



Encerclement d'un fugitif visible dans un arbre



Encerclement d'un fugitif visible dans un arbre



Compléxité du problème

Déterminer $s(G)$ est NP-difficile

- **Megiddo** *et al*, J.of ACM, 1988
  The complexity of searching a graph.

Déterminer $vs(G)$ est NP-difficile

- **Seymour et Thomas**, J. of Comb. Th., 1993.
  Graph searching and a min-max theorem for tree-width

Remarque :

Ils sont linéaires dans le cas des arbres.

Question :

Ces problèmes sont-ils NP-complet ?

Compléxité du problème

Déterminer $s(G)$ est NP-difficile

- **Megiddo** *et al*, J.of ACM, 1988
  The complexity of searching a graph.

Déterminer $vs(G)$ est NP-difficile

- **Seymour et Thomas**, J. of Comb. Th., 1993.
  Graph searching and a min-max theorem for tree-width

Remarque :

Ils sont linéaires dans le cas des arbres.

Question :

Ces problèmes sont-ils NP-complet ?

Compléxité du problème

Déterminer $s(G)$ est NP-difficile

- **Megiddo** *et al*, J.of ACM, 1988
  The complexity of searching a graph.

Déterminer $vs(G)$ est NP-difficile

- **Seymour et Thomas**, J. of Comb. Th., 1993.
  Graph searching and a min-max theorem for tree-width

Remarque :

Ils sont linéaires dans le cas des arbres.

Question :

Ces problèmes sont-ils NP-complet ?

Monotonie et Recontamination

Recontamination

Un sommet est recontaminé au cours d'une stratégie S, si il a été nettoyé (occupé par un agent) à une étape, et que le fugitif à la possibilité d'occuper ce sommet lors d'une étape ultérieure.

Monotonie

Une stratégie est monotone si aucune recontamination n'est autorisée. Un sommet nettoyé reste propre jusqu'à la fin.
Soit $ms(G)$ l'encerclement monotone du graphe G.

Monotonie et Recontamination

Question : La recontamination aide-t-elle ?

Est-il plus difficile de nettoyer un graphe de manière monotone ? $\mathbf{ms}(G) > \mathbf{s}(G)$?

La recontamination n'aide pas : $s(G) = ms(G)$

- **Bienstock et Seymour**, J.of Alg., 1991
  Monotonicity in graph searching.
- **LaPaugh**, J.of ACM, 1993
  Recontamination does not help to search a graph.

$vs(G) = mvs(G)$ (Cas d'un fugitif visible)

- **Seymour et Thomas**, J. of Comb. Th., 1993.
  Graph searching and a min-max theorem for tree-width

Monotonie et Recontamination

Question : La recontamination aide-t-elle ?

Est-il plus difficile de nettoyer un graphe de manière monotone ? $\mathbf{ms}(G) > \mathbf{s}(G)$?

La recontamination n'aide pas : $s(G) = ms(G)$

- **Bienstock et Seymour**, J.of Alg., 1991
  Monotonicity in graph searching.
- **LaPaugh**, J.of ACM, 1993
  Recontamination does not help to search a graph.

$vs(G) = mvs(G)$ (Cas d'un fugitif visible)

- **Seymour et Thomas**, J. of Comb. Th., 1993.
  Graph searching and a min-max theorem for tree-width

La recontamination n'aide pas : conséquences

En d'autres termes...

Il existe toujours une stratégie S qui capture un fugitif invisible en utilisant le moins d'agents possible et telle que S est monotone.

Déterminer $s(G)$ (resp. $vs(G)$) est NP-complet

Une stratégie monotone est un certificat de taille polynomiale qui peut donc être vérifié en temps polynomial.

Pourquoi on est content que la recontamination n'aide pas

Il est (très) difficile de concevoir une stratégie non monotone. Maintenant, on peut se restreindre aux stratégies monotones.

La recontamination n'aide pas : conséquences

En d'autres termes...

Il existe toujours une stratégie S qui capture un fugitif invisible en utilisant le moins d'agents possible et telle que S est monotone.

Déterminer $s(G)$ (resp. $vs(G)$) est NP-complet

Une stratégie monotone est un certificat de taille polynomiale qui peut donc être vérifié en temps polynomial.

Pourquoi on est content que la recontamination n'aide pas

Il est (très) difficile de concevoir une stratégie non monotone.
Maintenant, on peut se restreindre aux stratégies monotones.

La recontamination n'aide pas : conséquences

En d'autres termes...

Il existe toujours une stratégie S qui capture un fugitif invisible en utilisant le moins d'agents possible et telle que S est monotone.

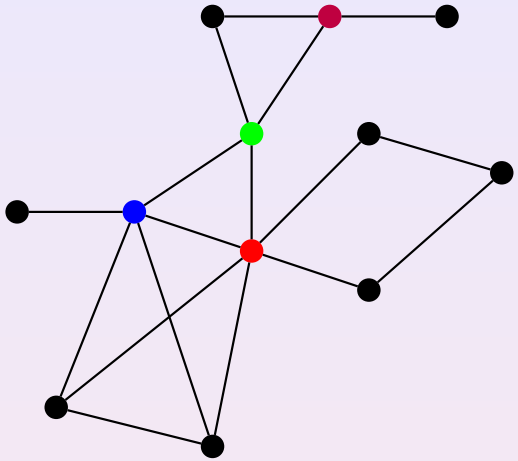
Déterminer $s(G)$ (resp. $vs(G)$) est NP-complet

Une stratégie monotone est un certificat de taille polynomiale qui peut donc être vérifié en temps polynomial.

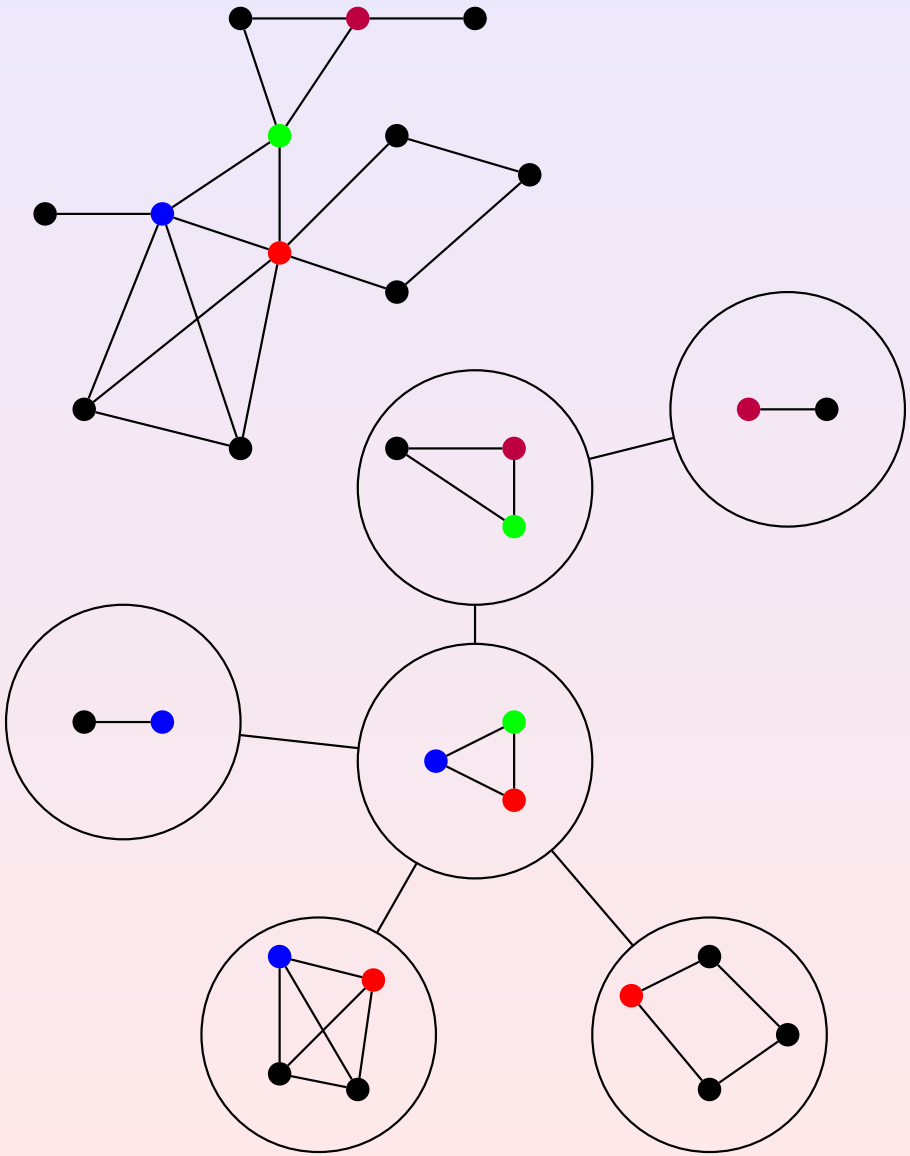
Pourquoi on est content que la recontamination n'aide pas

Il est (très) difficile de concevoir une stratégie non monotone. Maintenant, on peut se restreindre aux stratégies monotones.

Décomposition arborescente et linéaire

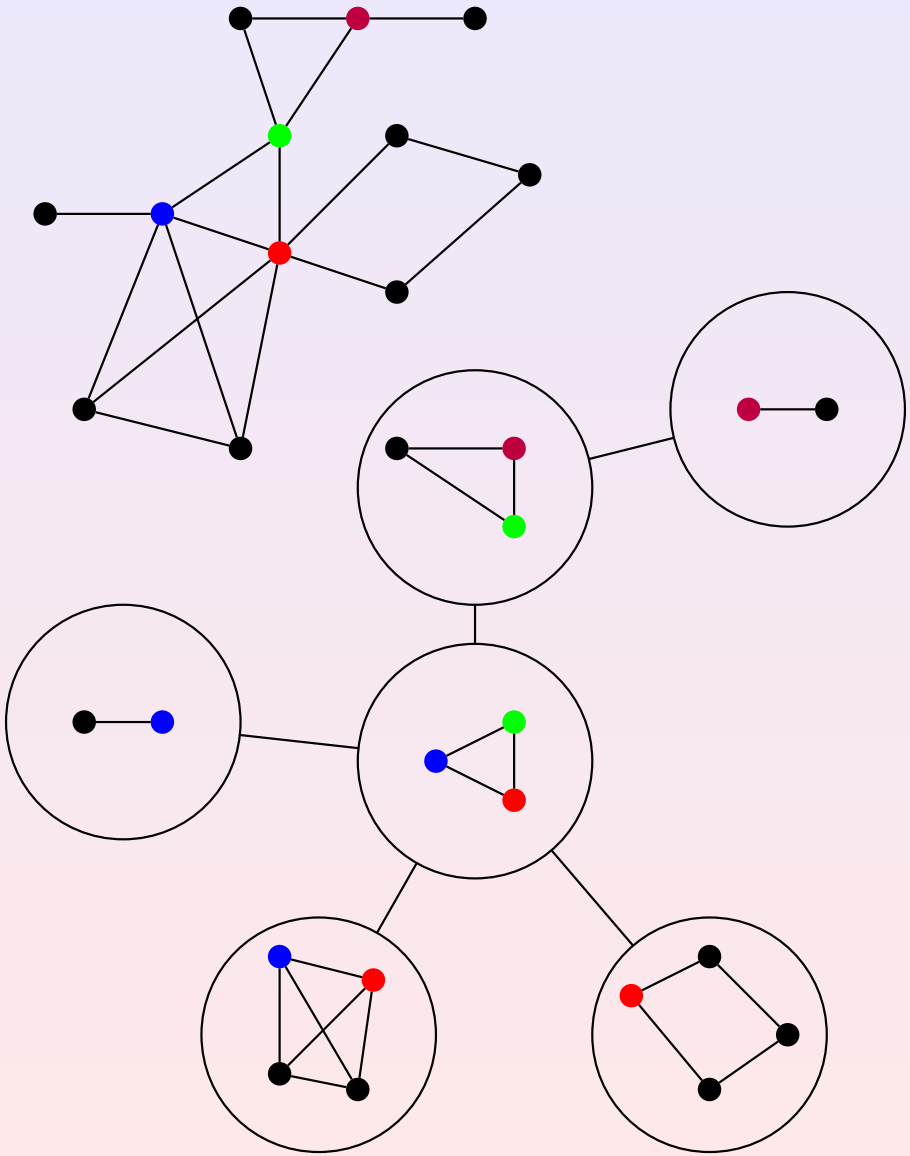


Décomposition arborescente et linéaire



arbre T, des sacs $(X_t)_{t \in V(T)}$

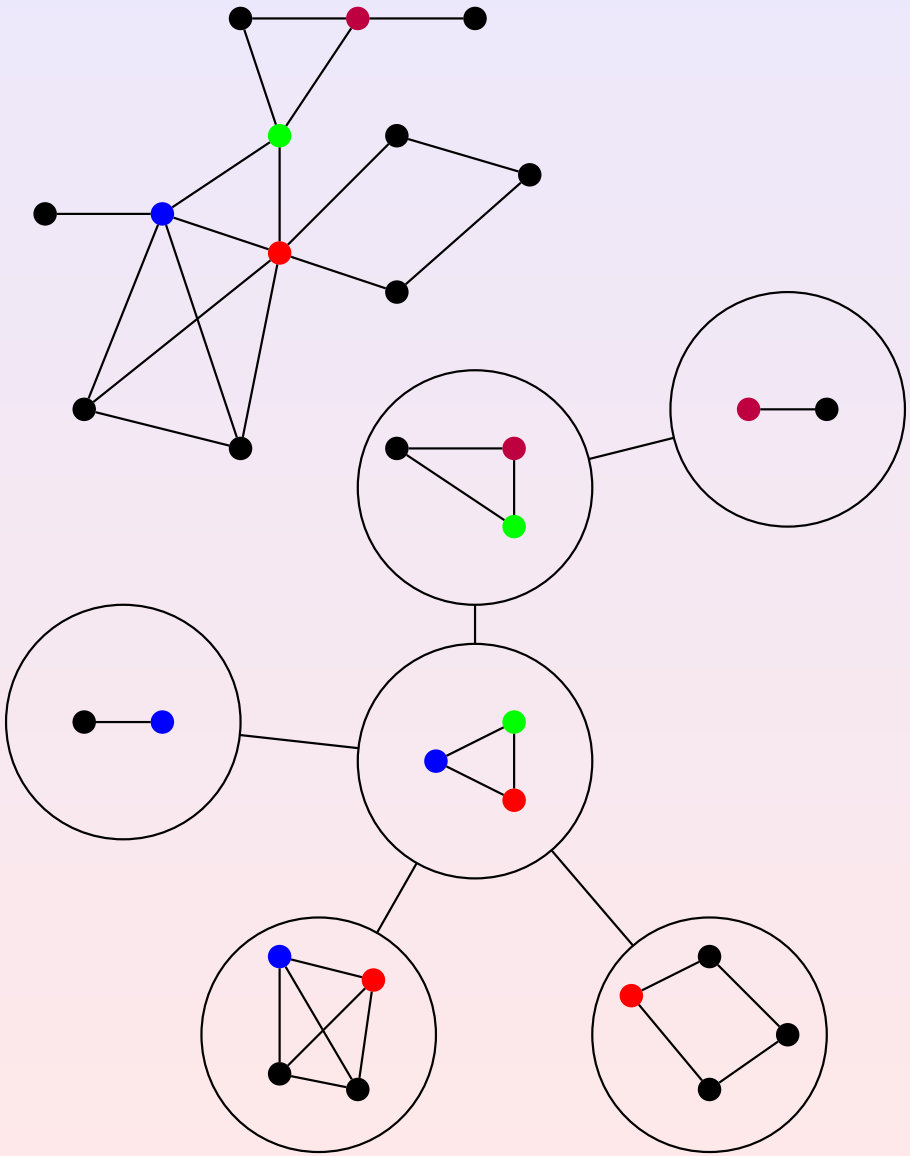
Décomposition arborescente et linéaire



arbre T, des sacs $(X_t)_{t \in V(T)}$

- chaque sommet de G est dans au moins un sac ;

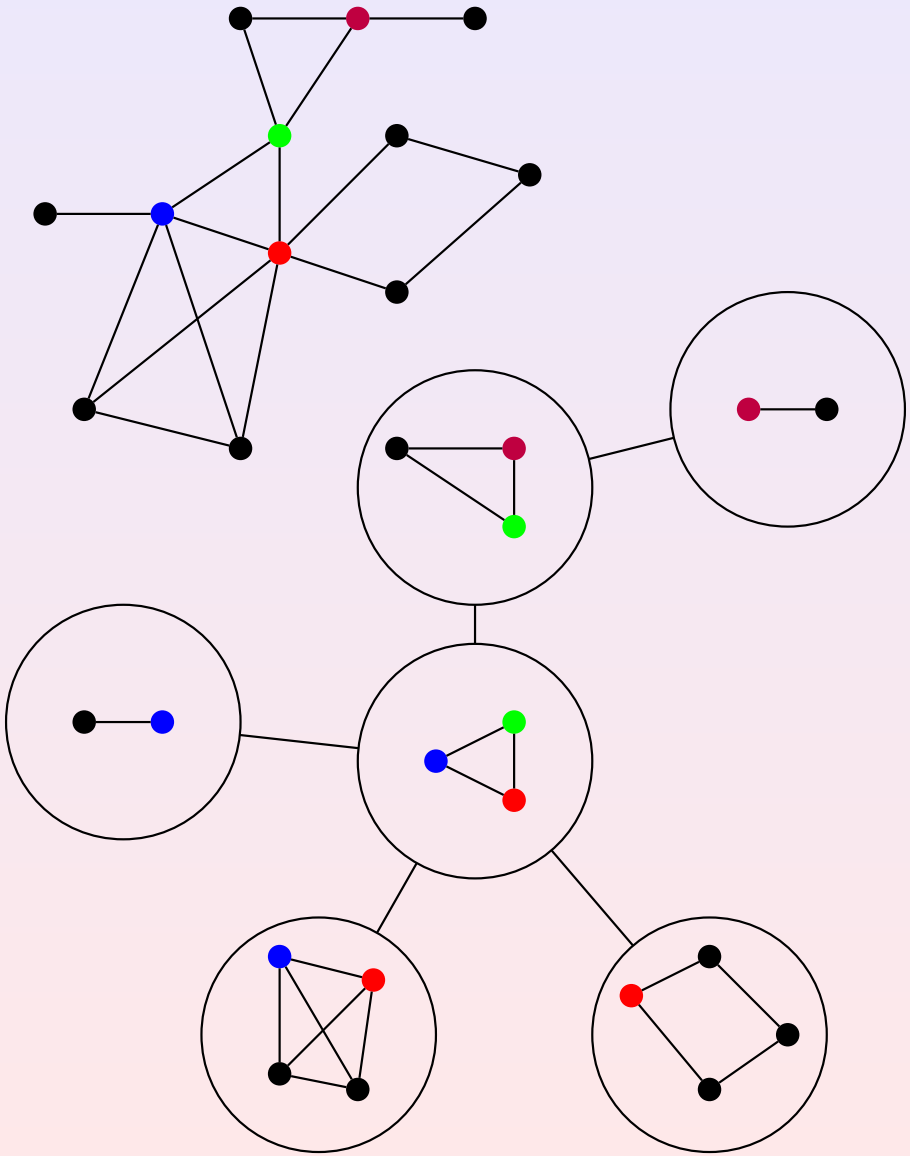
Décomposition arborescente et linéaire



arbre T, des sacs $(X_t)_{t \in V(T)}$

- chaque **sommet** de G est dans au moins un sac ;
- les 2 extrémités d'une arête de G sont dans au moins un sac ;

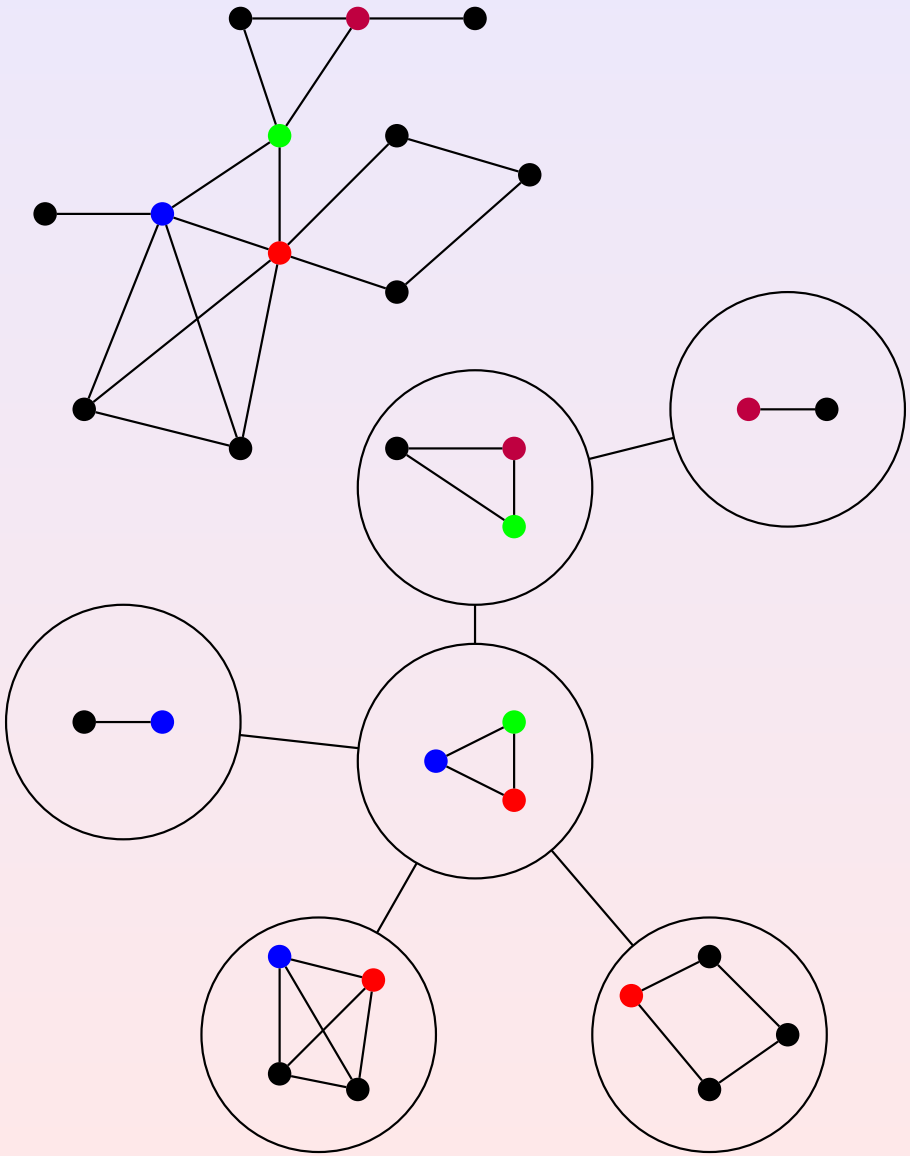
Décomposition arborescente et linéaire



arbre T, des sacs $(X_t)_{t \in V(T)}$

- chaque **sommet** de G est dans au moins un sac ;
- les 2 extrémités d'une **arête** de G sont dans au moins un sac ;
- pour tout sommet de G, les sacs qui le contiennent forment un sous-arbre.

Décomposition arborescente et linéaire

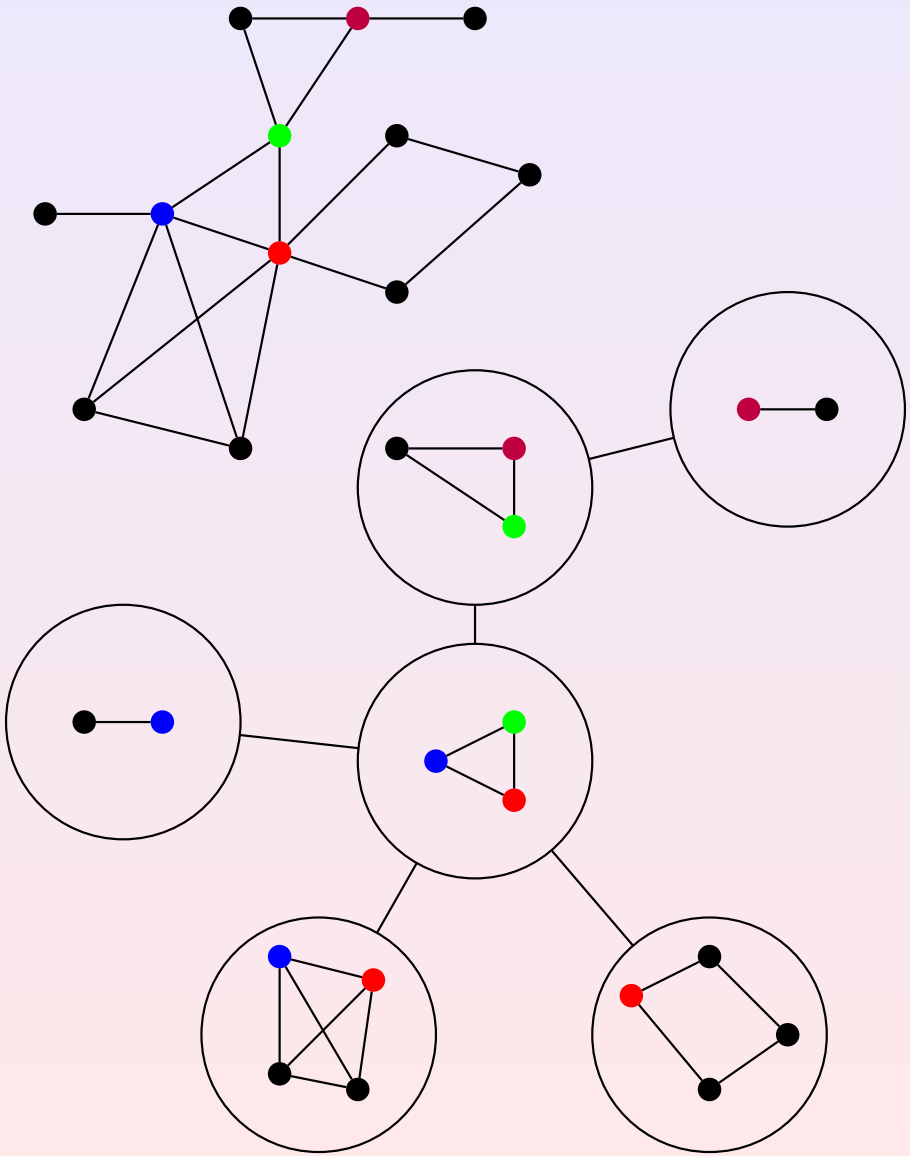


arbre T, des sacs $(X_t)_{t \in V(T)}$

- chaque **sommet** de G est dans au moins un sac ;
- les 2 extrémités d'une **arête** de G sont dans au moins un sac ;
- pour tout sommet de G, les sacs qui le contiennent forment un **sous-arbre**.

Largeur = |plus grand sac| -1

Décomposition arborescente et linéaire



arbre T, des sacs $(X_t)_{t \in V(T)}$

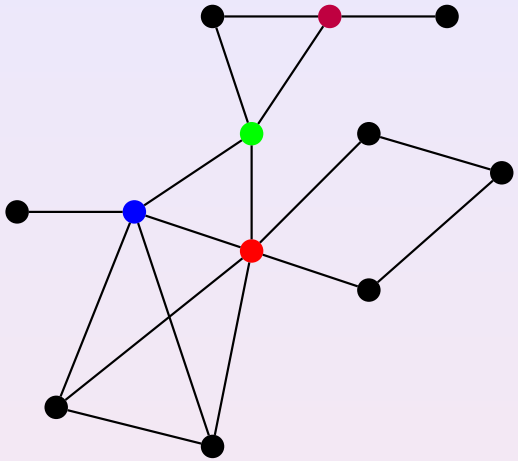
- chaque **sommet** de G est dans au moins un sac ;
- les 2 extrémités d'une **arête** de G sont dans au moins un sac ;
- pour tout sommet de G, les sacs qui le contiennent forment un **sous-arbre**.

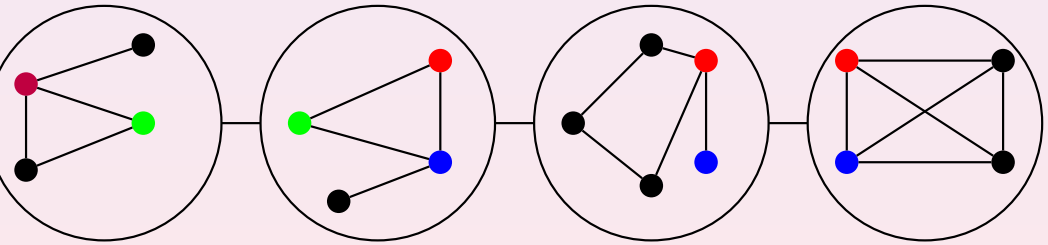
Largeur = |plus grand sac| -1

treewidth de G

tw(G), largeur minimum parmi les décompositions arborescentes

Décomposition arborescente et linéaire





chemin P, des sacs $(X_t)_{t \in V(T)}$

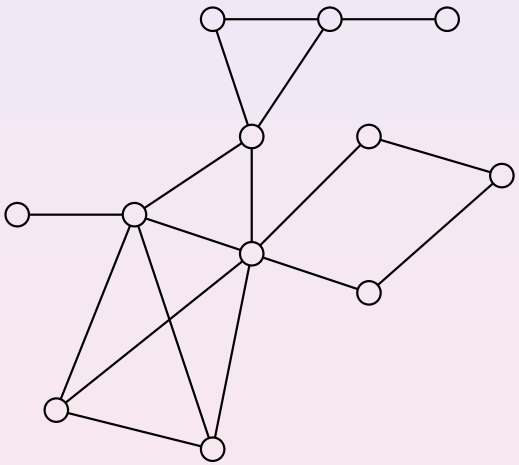
- chaque **sommet** de G est dans au moins un sac ;
- les 2 extrémités d'une **arête** de G sont dans au moins un sac ;
- pour tout sommet de G, les sacs qui le contiennent forment un **sous-chemin**.

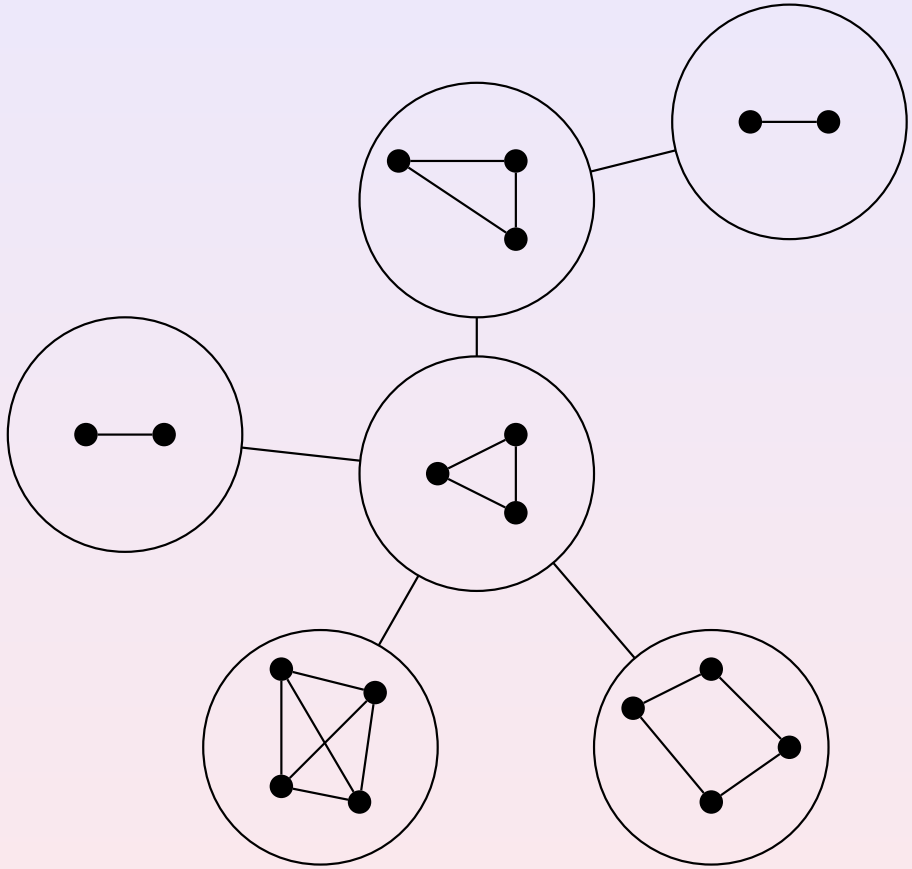
Largeur = |plus grand sac| -1

pathwidth de G

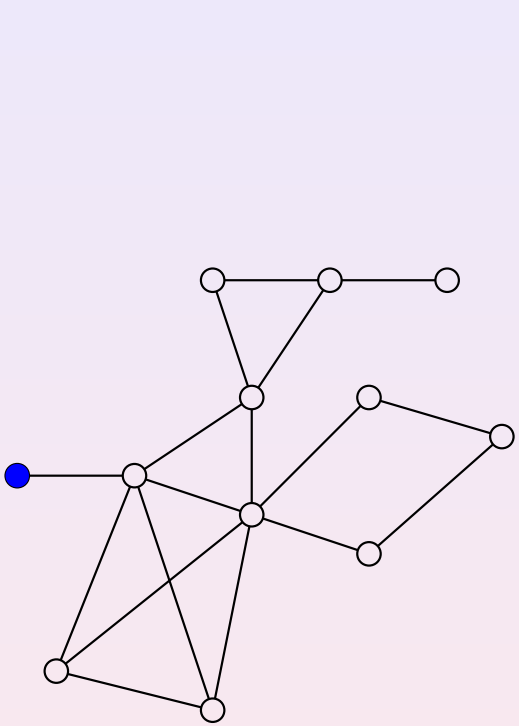
pw(G), largeur minimum parmi les décompositions linéaires

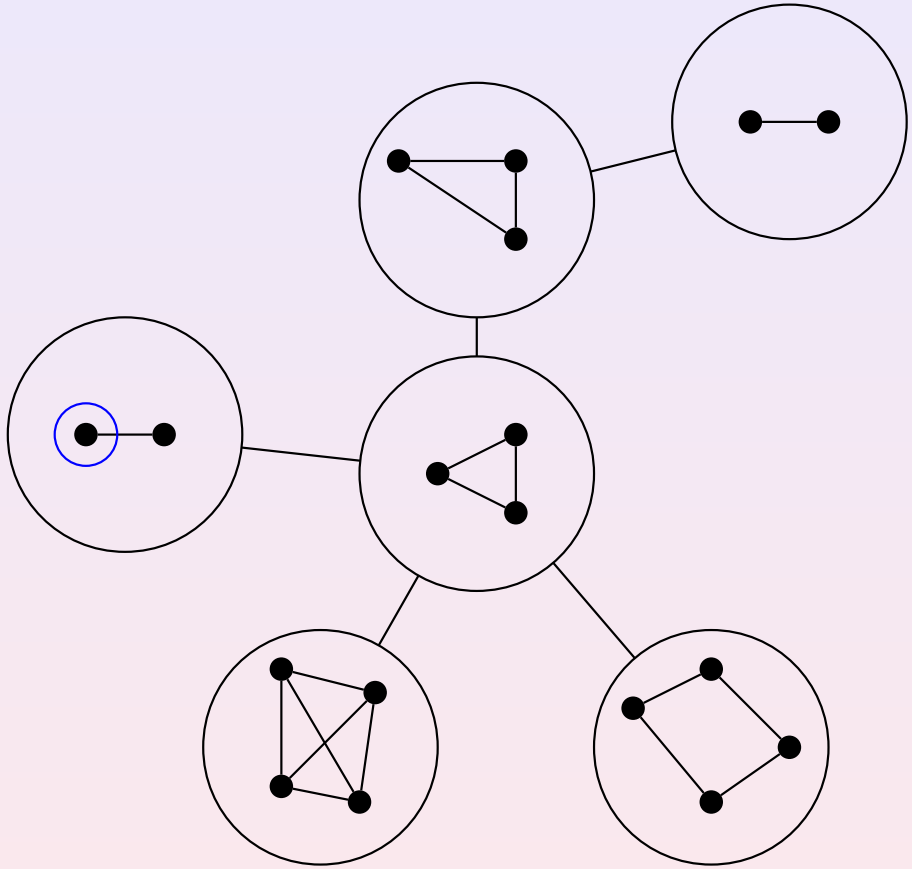
Décomposition arborescente et encerclement visible



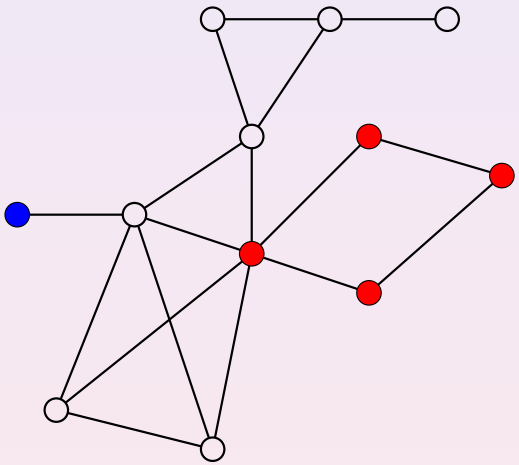


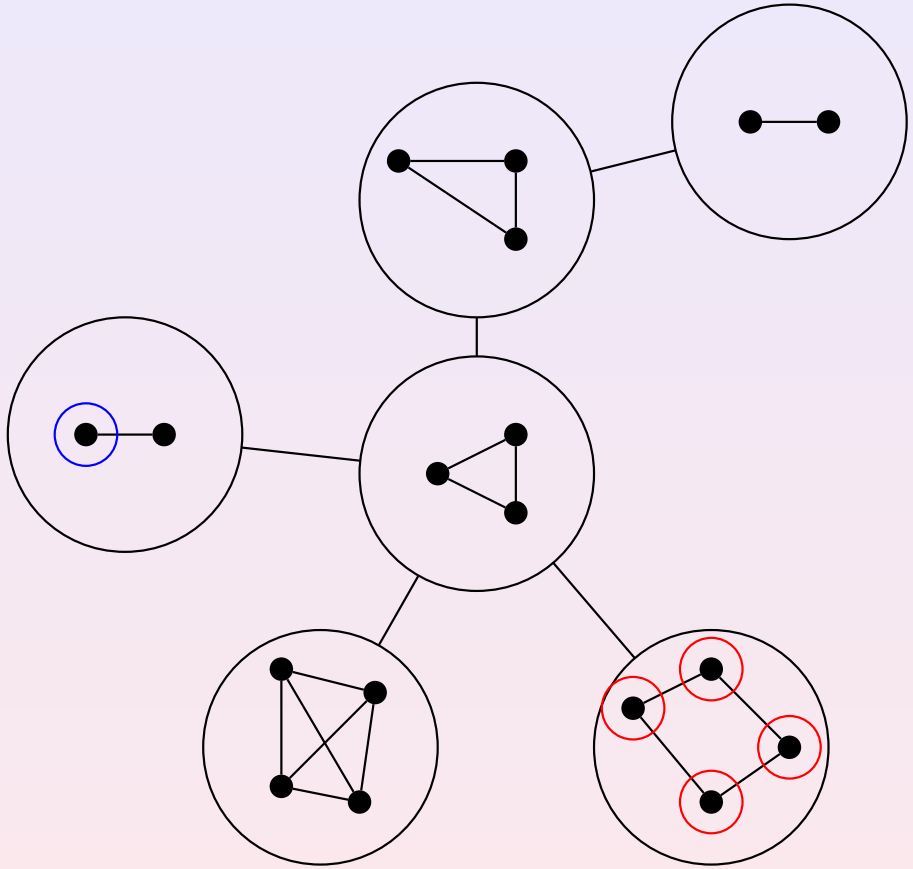
Décomposition arborescente et encerclement visible



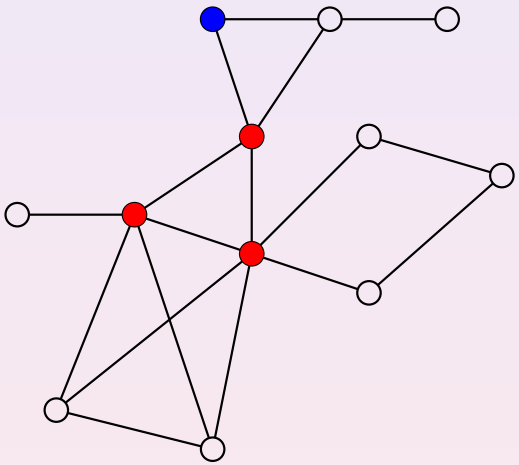


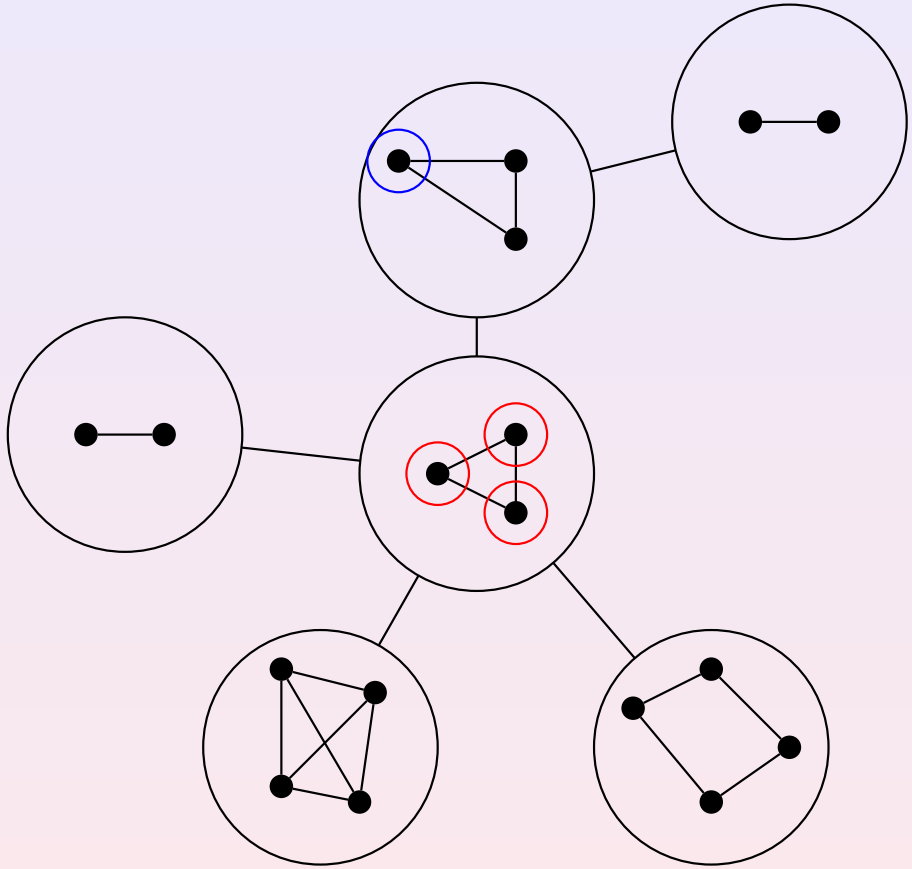
Décomposition arborescente et encerclement visible



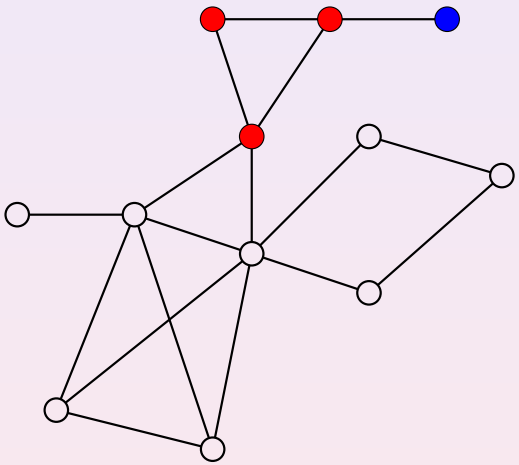


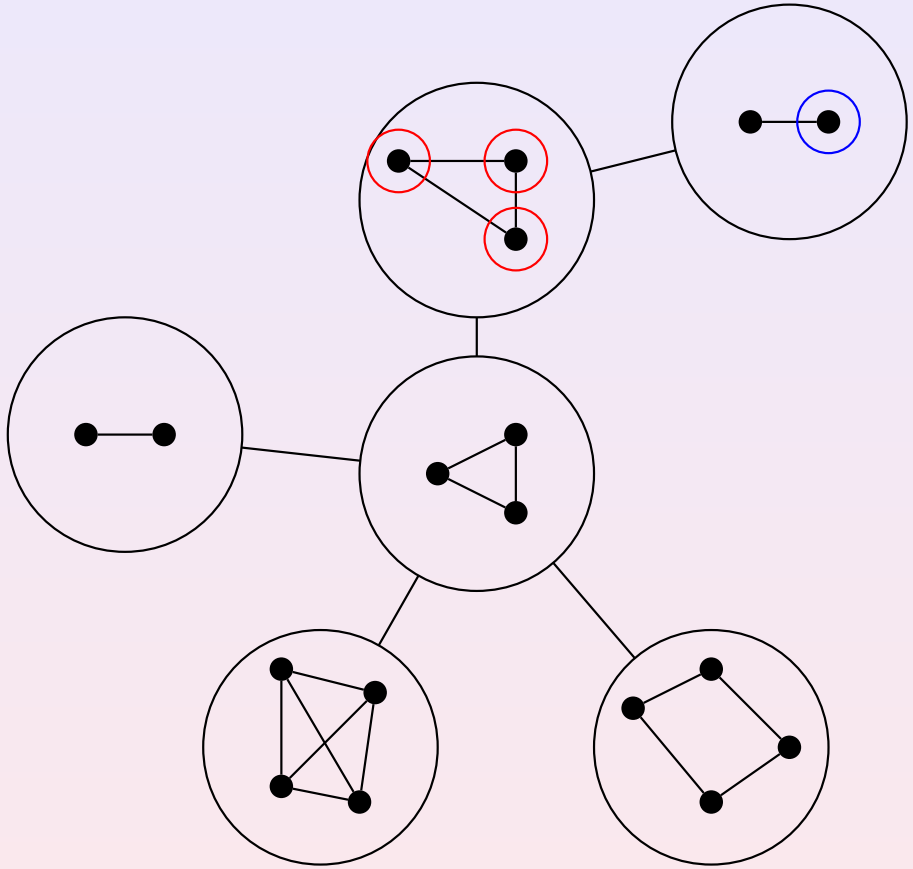
Décomposition arborescente et encerclement visible



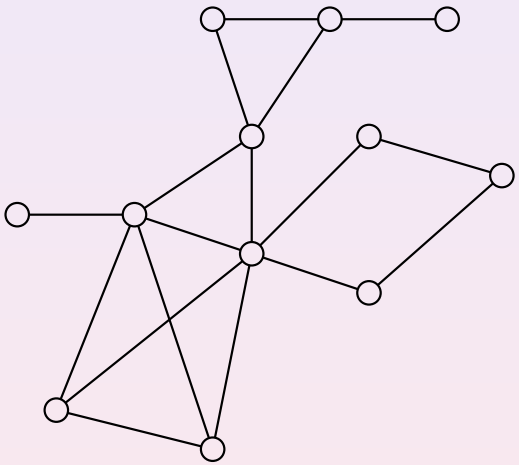


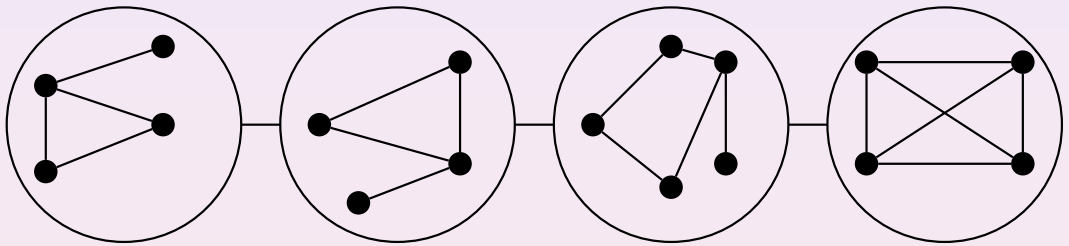
Décomposition arborescente et encerclement visible



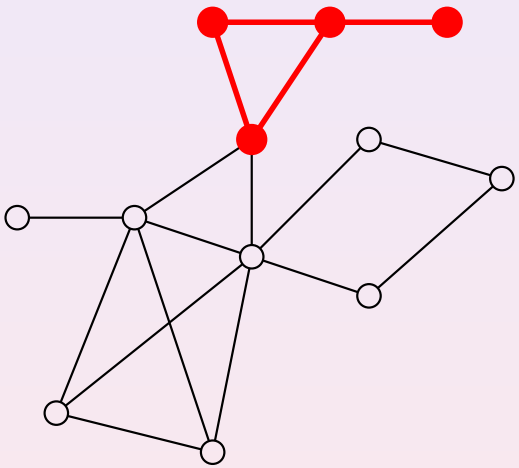


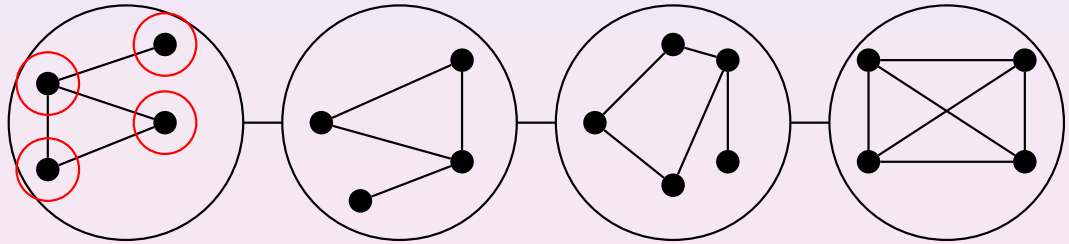
Décomposition linéaire et encerclement



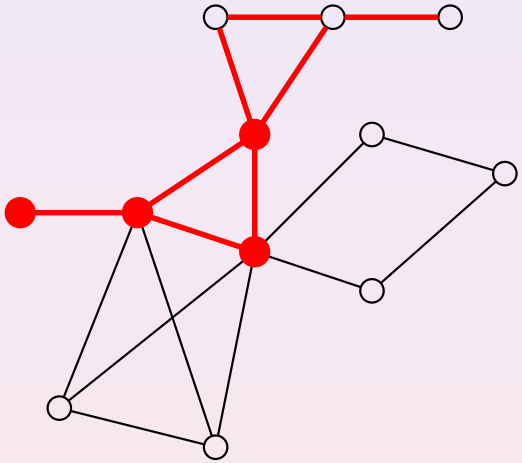


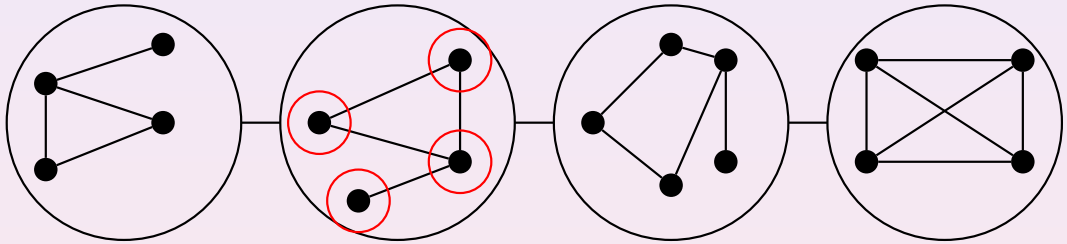
Décomposition linéaire et encerclement



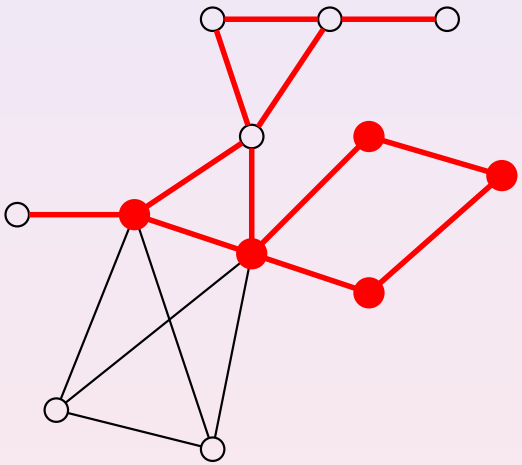


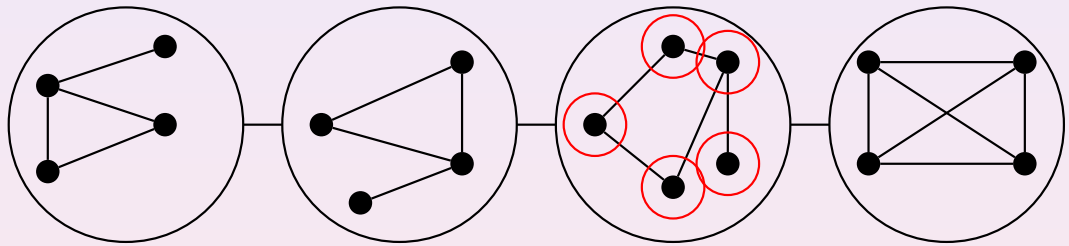
Décomposition linéaire et encerclement





Décomposition linéaire et encerclement





Lien entre encerclement et largeur linéaire

Sur ce transparent, vs = vertex separation

Pour tout graphe G, $s(G) = vs(G) + 1$

J.A. Ellis, I.H. Sudborough et J.S. Turner. Vertex Separation and Search Number of a graph. [Inf. Comput.,1994]

Pour tout graphe G, $vs(G) = pw(G)$

N.G. Kinnersley. The Vertex Separation number of a Graph equals its Pathwidth. [IPL.,1992]

Pour tout graphe G :

$\mathbf{s}(G) = \mathbf{pw}(G) + 1$

Pour tout graphe G : $\mathbf{vs}(G) = \mathbf{tw}(G) + 1$

Seymour et Thomas. Graph searching and min-max theorem for treewidth. [J.Combin. Theory, 1993.]

Lien entre encerclement et largeur linéaire

Pour tout graphe G, $s(G) = vs(G) + 1$

J.A. Ellis, I.H. Sudborough et J.S. Turner. Vertex Separation and Search Number of a graph. [Inf. Comput.,1994]

Pour tout graphe G, $vs(G) = pw(G)$

N.G. Kinnersley. The Vertex Separation number of a Graph equals its Pathwidth. [IPL.,1992]

Pour tout graphe G :

$\mathbf{s}(G) = \mathbf{pw}(G) + 1$

Pour tout graphe G : $\mathbf{vs}(G) = \mathbf{tw}(G) + 1$

Seymour et Thomas. Graph searching and min-max theorem for treewidth. [J.Combin. Theory, 1993.]

Plan

Encerclement connexe

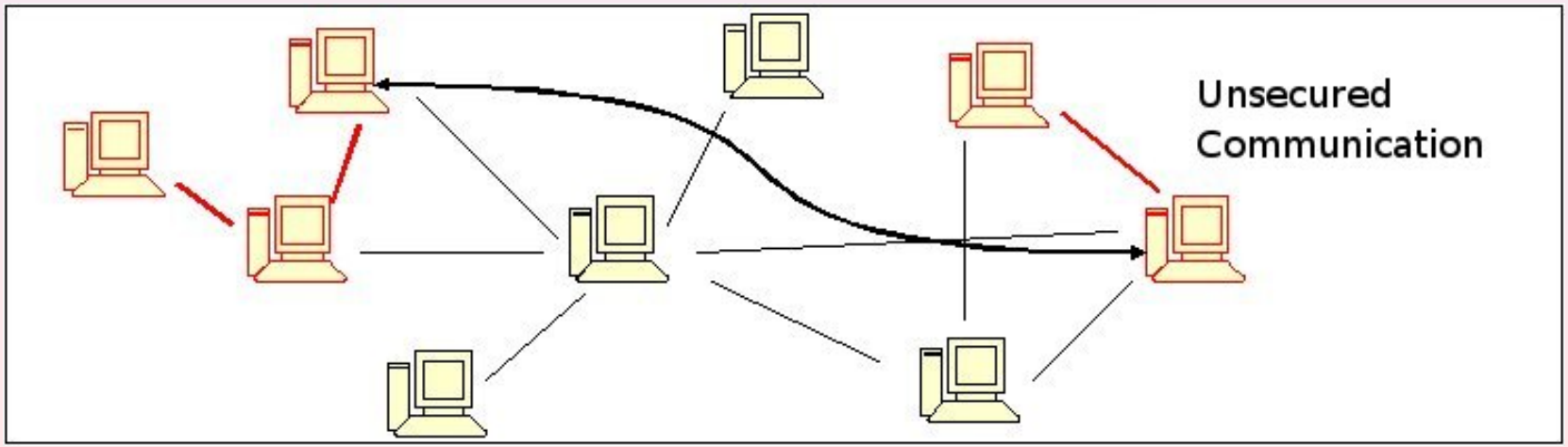
Limites du modèle de Parson

- Impossibilité de se déplacer à volonté dans la réalité
- Il est préférable que les agents restent *groupés.*

Encerclement connexe

Limites du modèle de Parson

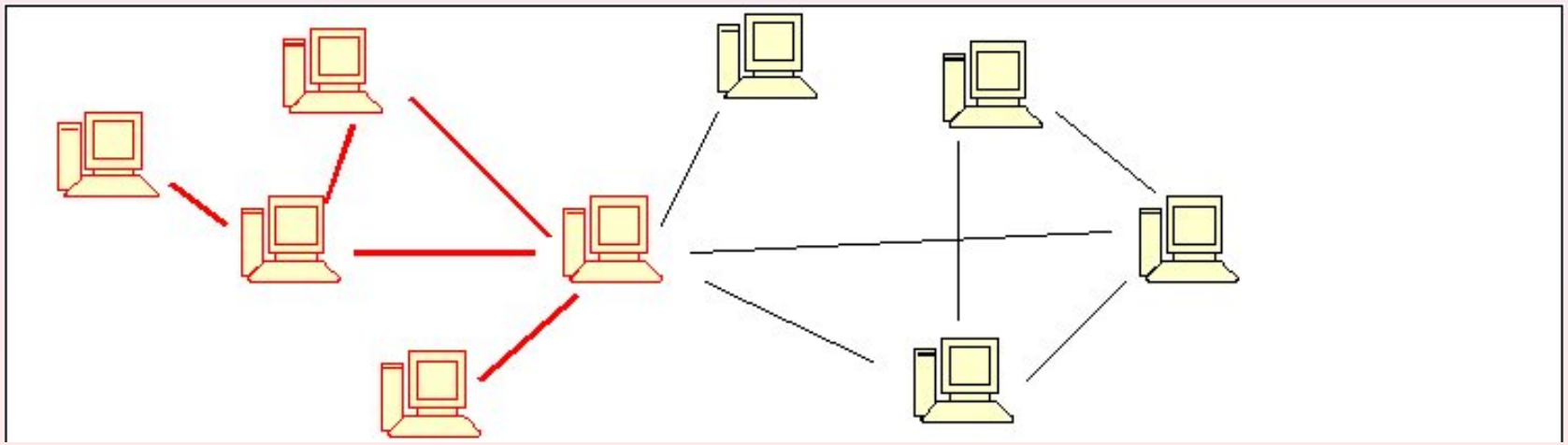
- Impossibilité de se déplacer à volonté dans la réalité
- Il est préférable que les agents restent *groupés.*



Encerclement connexe

Limites du modèle de Parson

- Impossibilité de se déplacer à volonté dans la réalité
- Il est préférable que les agents restent *groupés.*



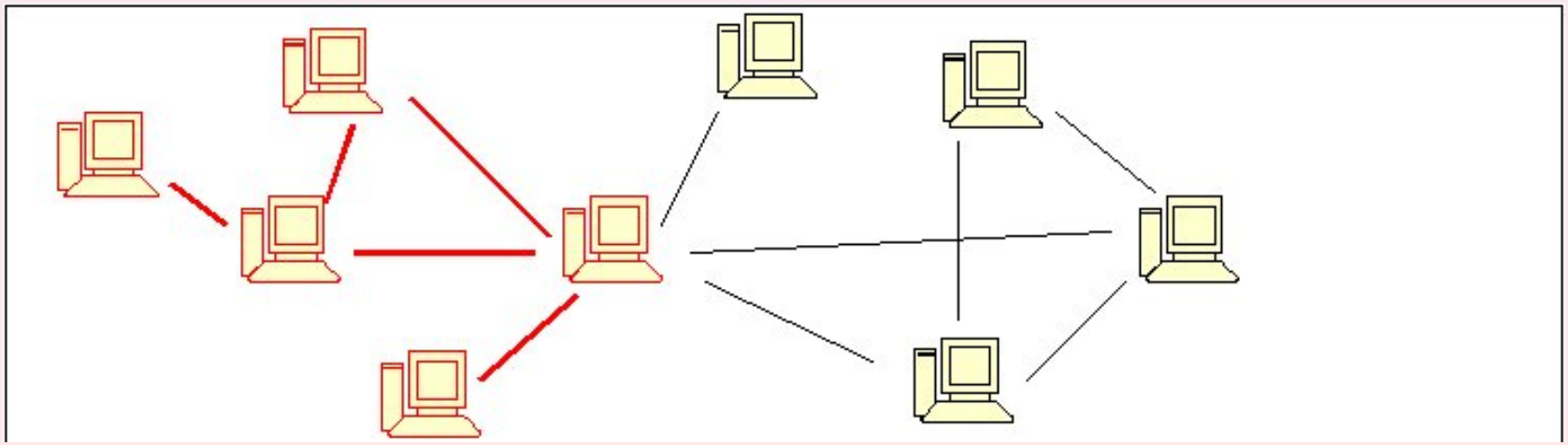
Encerclement connexe

Limites du modèle de Parson

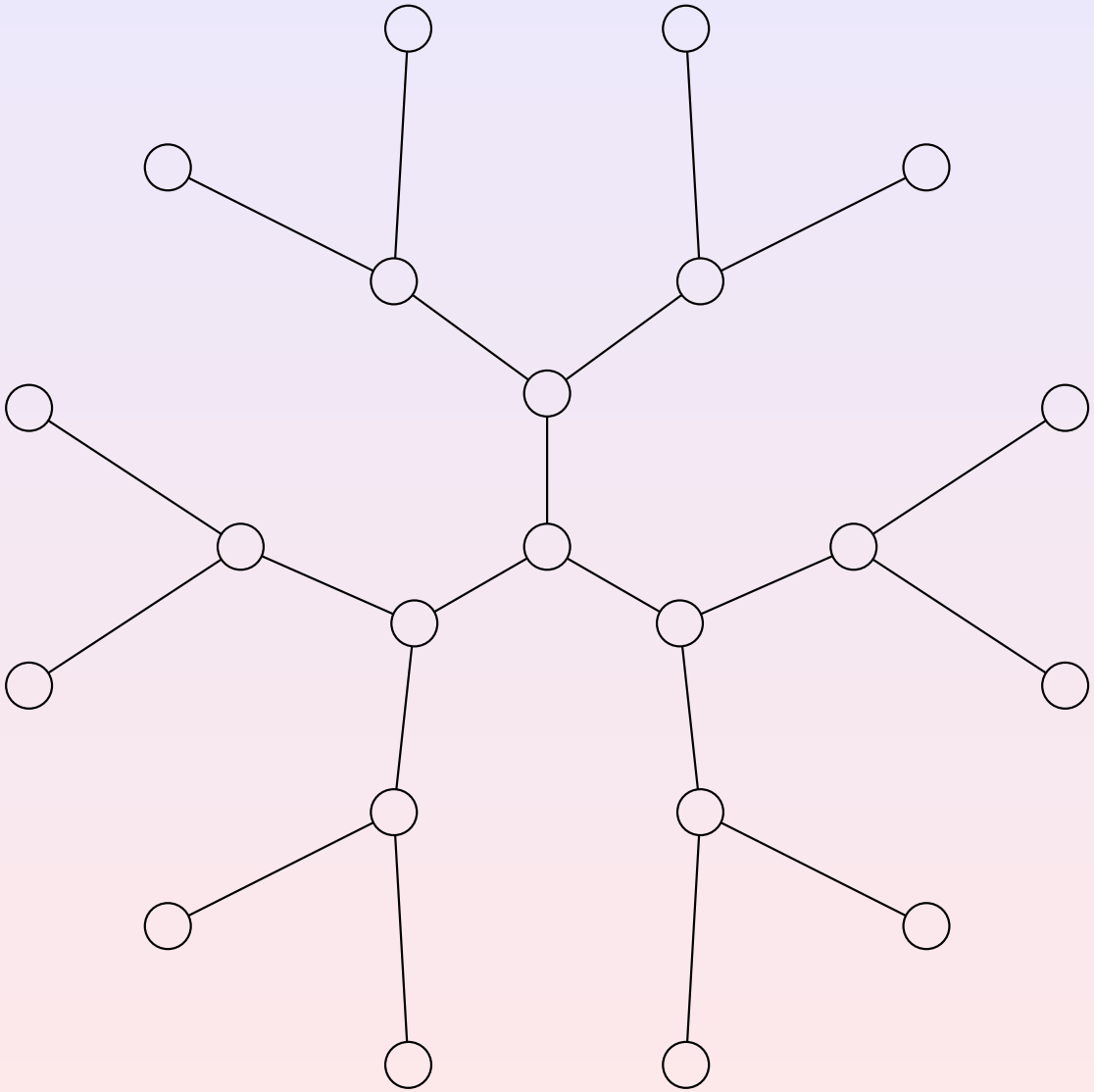
- Impossibilité de se déplacer à volonté dans la réalité
- Il est préférable que les agents restent *groupés.*

Stratégie d'encerclement connexe

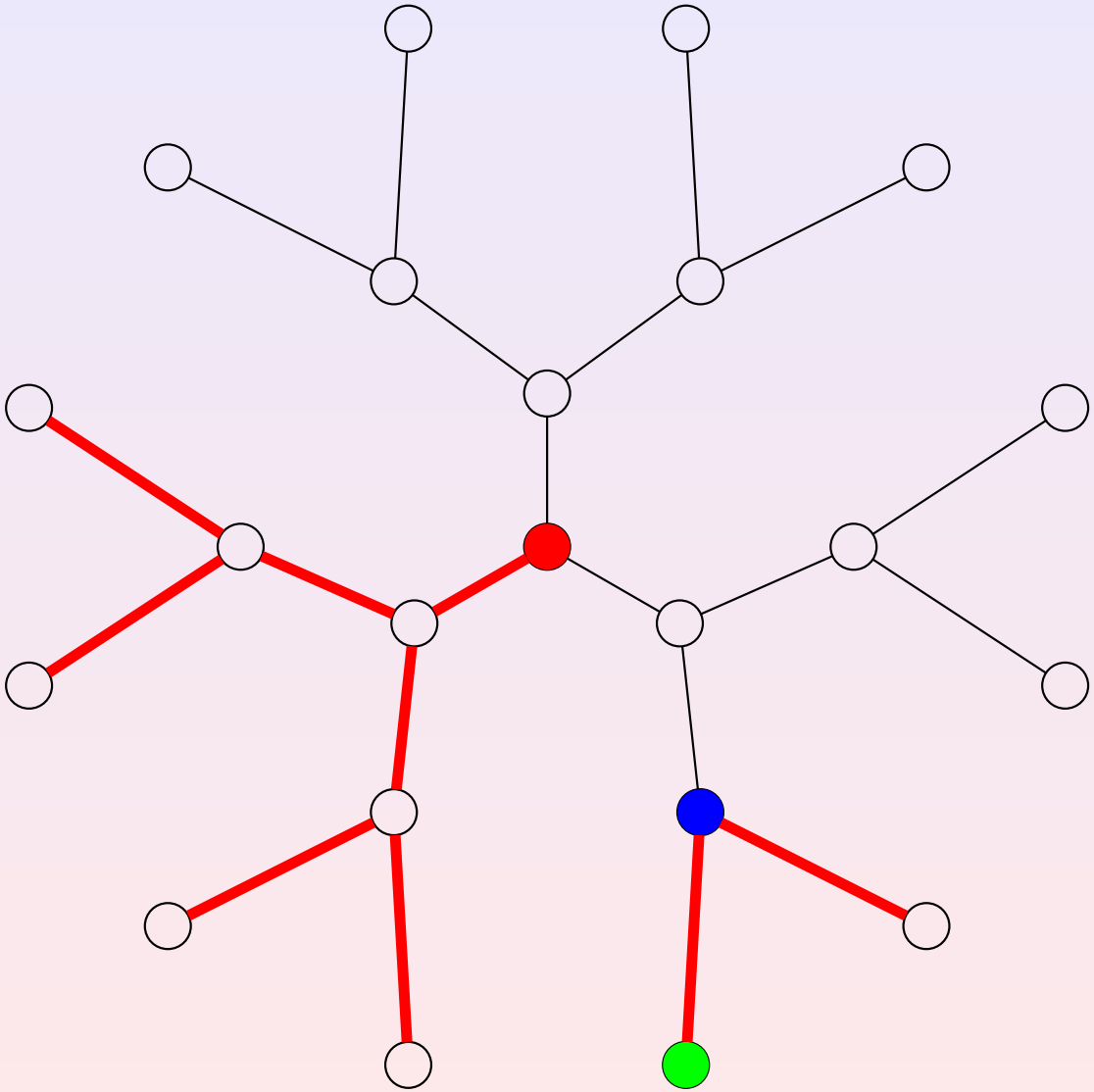
A chaque étape, la partie nettoyée doit être connexe.
cs(G) est le plus petit nombre d'agents nécessaires à une stratégie d'encerclement connexe dans le graphe.



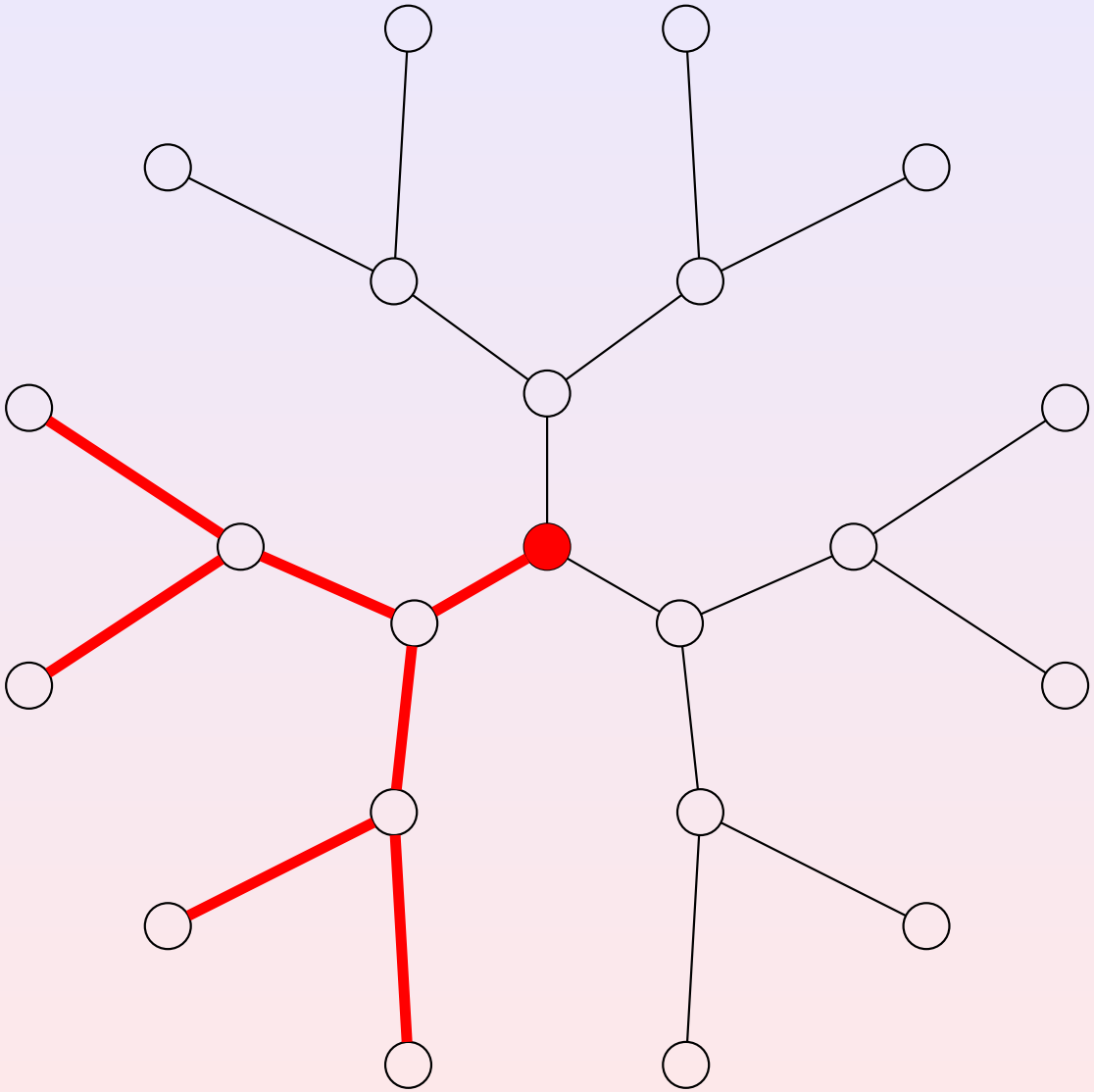
Connexe vs. non connexe



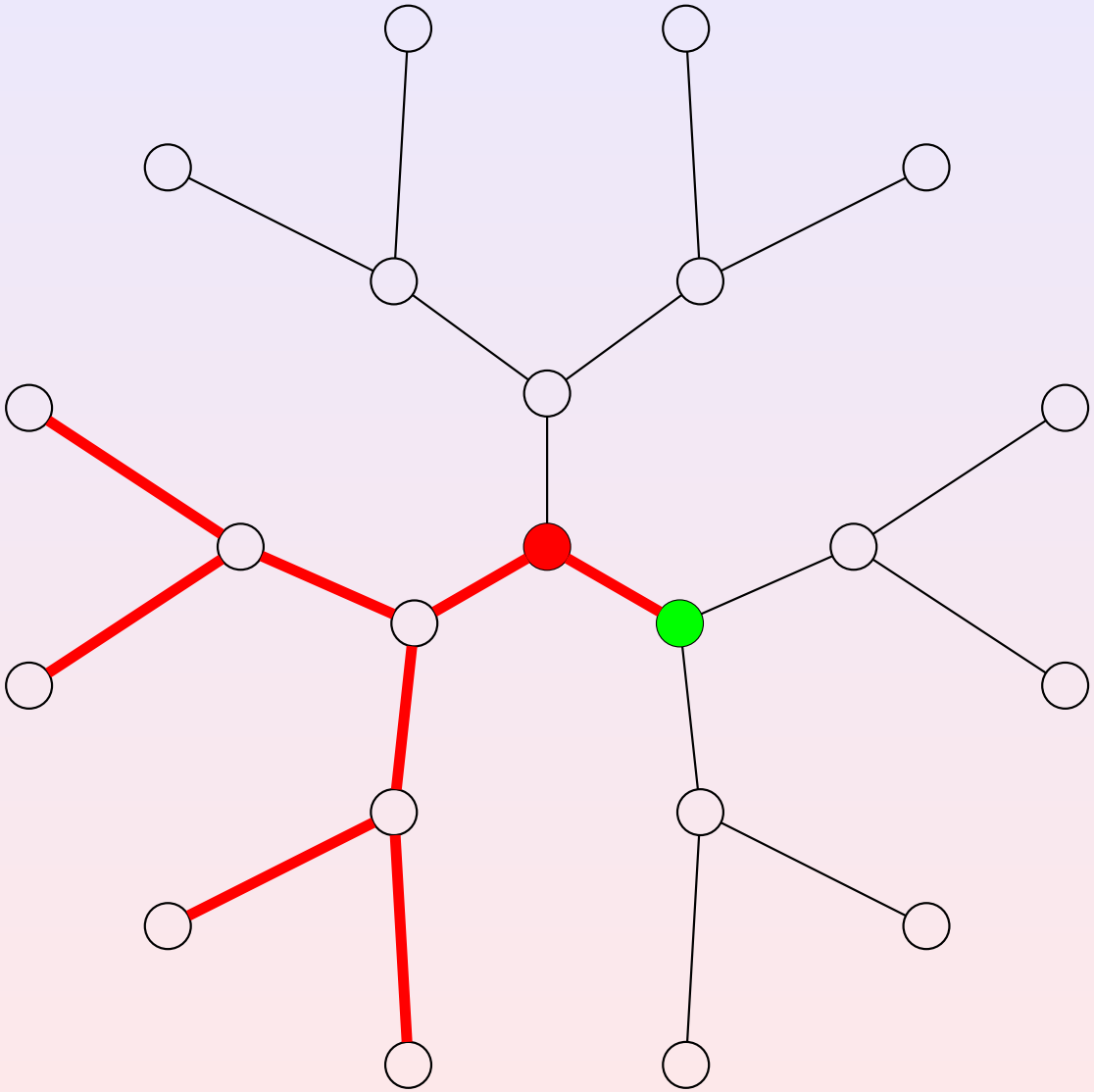
Connexe vs. non connexe



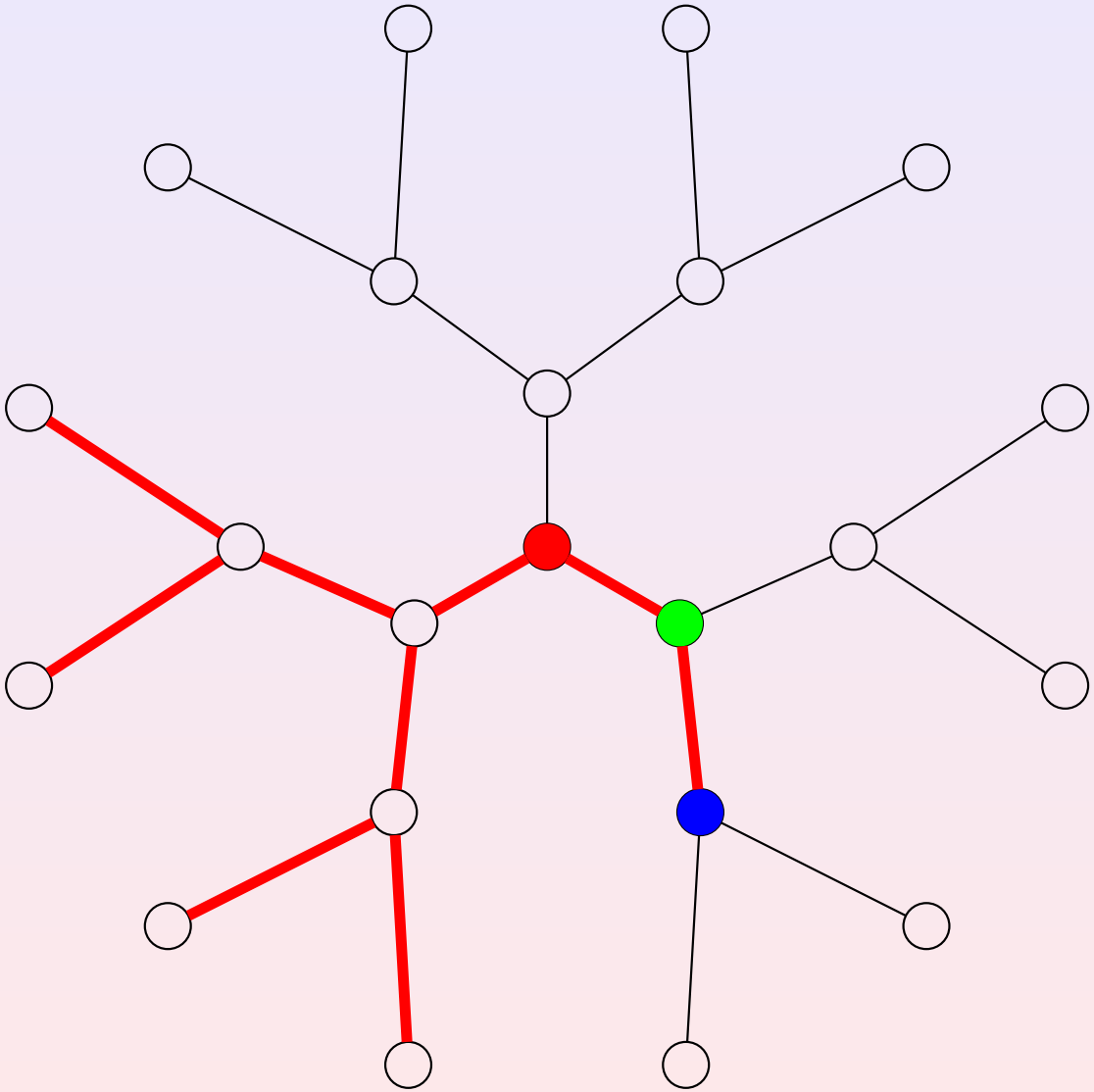
Connexe vs. non connexe



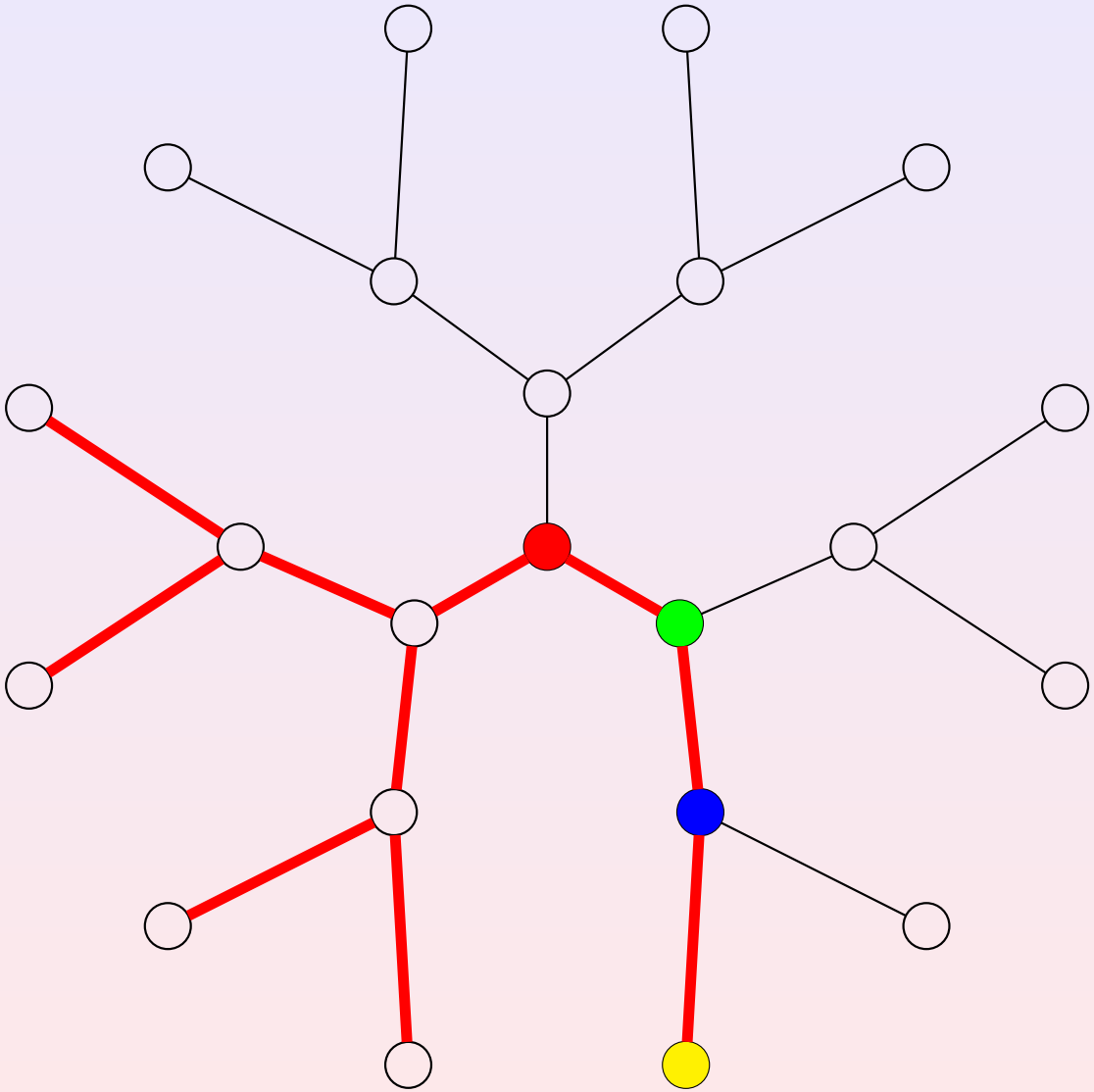
Connexe vs. non connexe



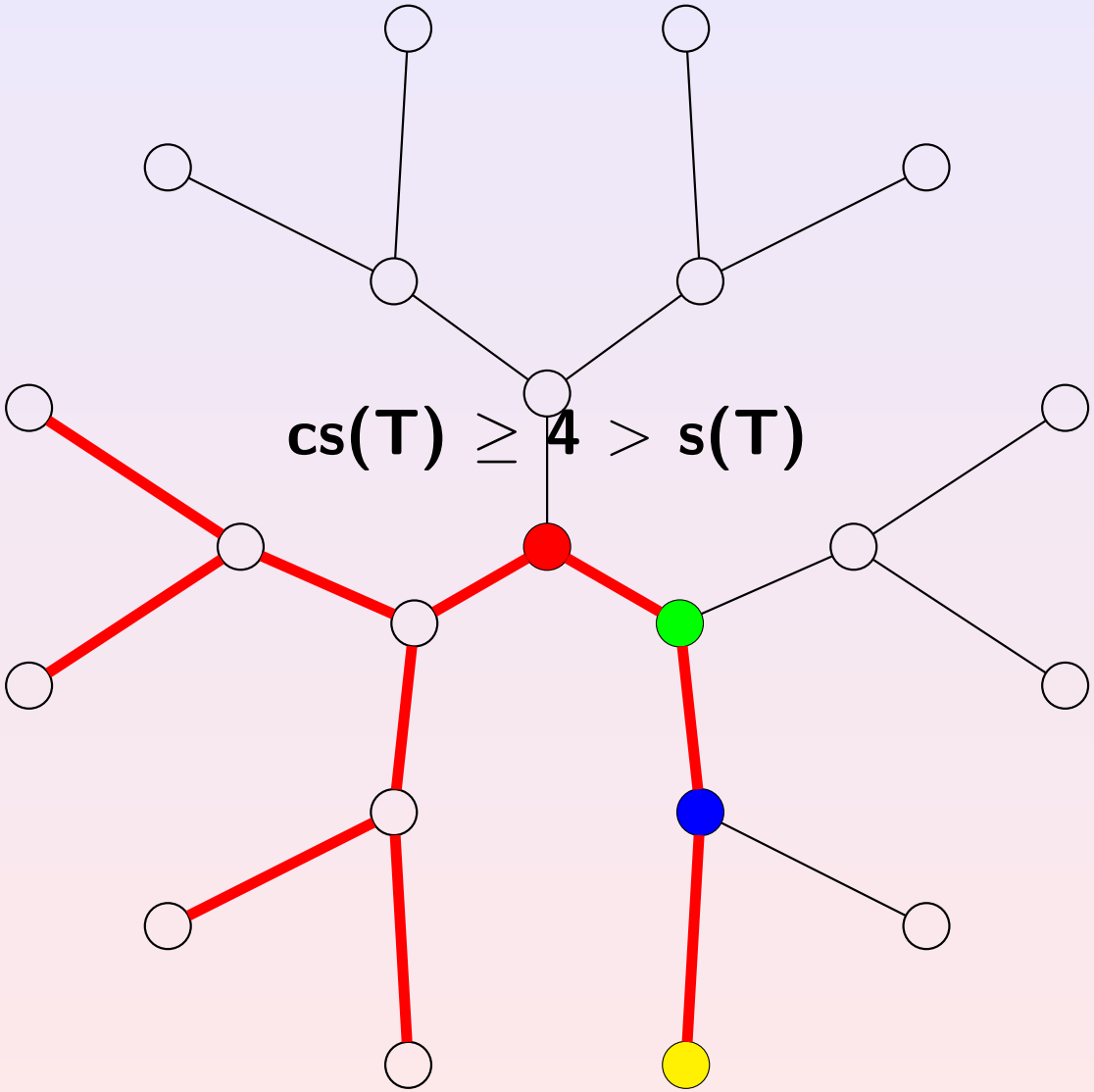
Connexe vs. non connexe



Connexe vs. non connexe



Connexe vs. non connexe



Prix de la connexité : cas des arbres

Barrière, Flocchini, Fraigniaud et Santoro. [SPAA, 2002]

Algorithme linéaire

Barrière, Fraigniaud, Santoro et Thilikos. [WG, 2003]

Pour tout arbre T, $s(T) \leq cs(T) \leq 2\ s(T) - 2$.
Ces bornes sont optimales.

Prix de la connexité : cas des graphes arbitraires

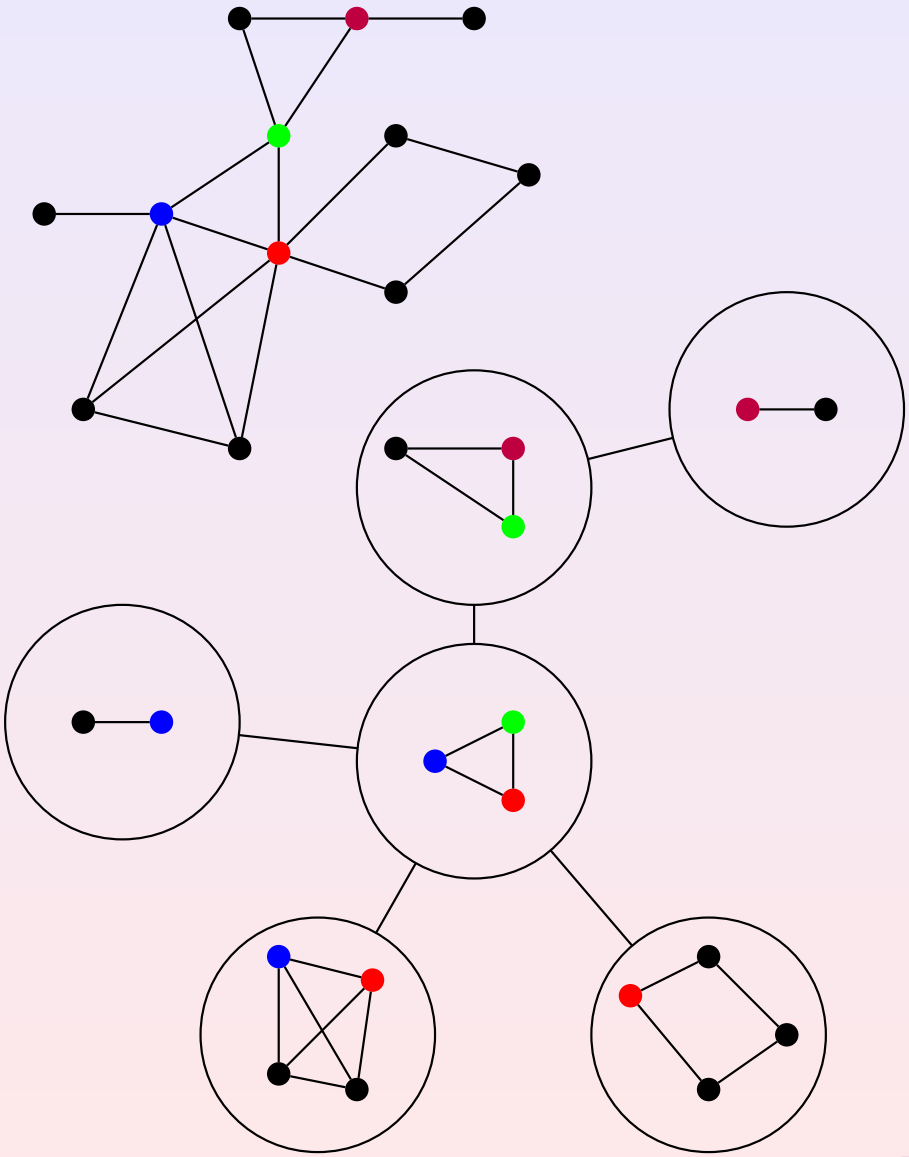
Seymour et Thomas. [Combinatorica, 1994]

Bond Carving (*Carving connexe*)

Fomin, Fraigniaud et Thilikos. [Technical repport, 2004]

- Algorithme polynomial constructif, étant donnée une décomposition en branche.
- Pour tout graphe connexe G, $cs(G) \leq s(G)\ (2 + \log |E(G)|)$.

Décomposition arborescente, rappel



Treewidth connexe

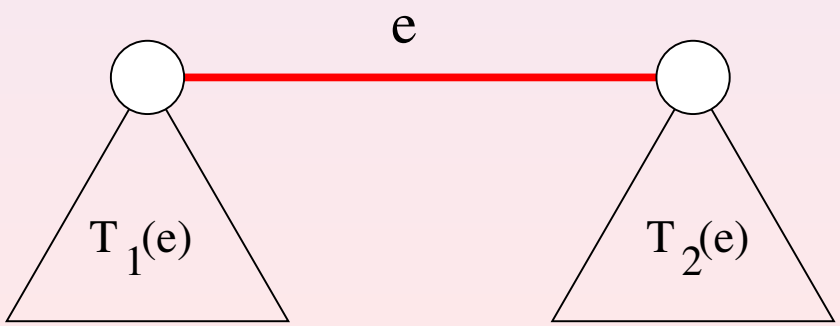
Arête connexe

e est dite connexe si $G[T_1(e)]$ et $G[T_2(e)]$ sont des sous-graphes connexes de G.

Décomposition arborescente connexe (T,X)

Toute arête de $E(T)$ est connexe.

Largeur arborescente connexe, ctw(G)



Treewidth connexe (2)

Pour tout graphe connexe G, $ctw(G) = tw(G)$

Arbre de cliques d'une triangulation minimale

M. Golumbic.
Algorithmic graph theory and perfect graphs.

Une telle décomposition est connexe

A. Parra et P. Scheffler. [DAM 1997]
Characterizations and algorithmic applications of Chordal Graphs embedding.

Resultats (1)

Théorème 1 : nouvelle preuve

Pour tout graphe connexe G, $ctw(G) = tw(G)$

Preuve constructive

Nous proposons un algorithme polynomial qui, étant donnée une décomposition arborescente de largeur k de G, retourne une décomposition arborescente connexe de largeur $\leq k$ de G, en temps $O(n.k^3)$.

Resultats (1)

Théorème 1 : nouvelle preuve

Pour tout graphe connexe G, $ctw(G) = tw(G)$

Preuve constructive

Nous proposons un algorithme polynomial qui, étant donnée une décomposition arborescente de largeur k de G, retourne une décomposition arborescente connexe de largeur $\leq k$ de G, en temps $O(n.k^3)$.

Resultats (2)

Théorèm 2

Pour tout graphe connexe G, $\mathbf{cs}(G) \leq \mathbf{s}(G)\ (1 + \log_2 |V(G)|)$.
Fraigniaud et Nisse. [LATIN06]

Preuve constructive

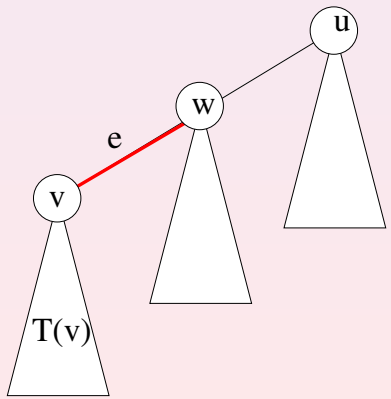
Etant donnée une décomposition arborescente du graphe G, notre algorithme calcule une strtégie d'encerclement connexe de G, avec au plus $\mathbf{tw}(G) \log |V(G)|$ agents et en temps polynomial.

Idée de la preuve du Théorème 1

Décomposition arborescente enracinée en u

Sous-connexité

- Une arête $e = (w, v)$ est sous-connexe si $G[T(v)]$ induit un sous graphe connexe
- (T, X) sous-connexe en v si $G[T(v)]$ induit un sous graphe connexe et toute arête de $T(v)$ est sous-connexe



Idée de la preuve du Théorème 1

Théorème 1 :

Pour tout graphe connexe G, $ctw(G) = tw(G)$

Entrée de l'algorithme

(T_u, X) une décomposition arborescente enracinée en u et de largeur k d' un graphe G.

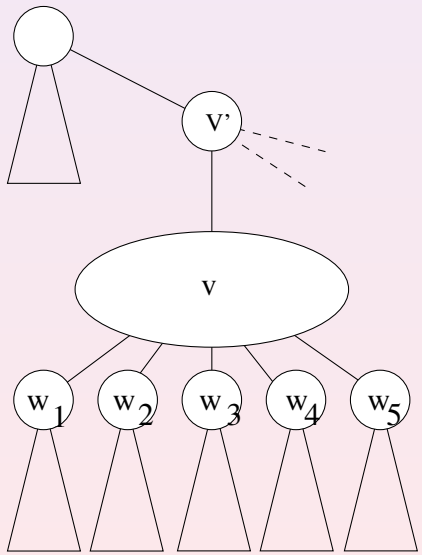
2 phases

- Montée : rend la décomposition sous-connexe en u
- Descente : rend la décomposition connexe.

Idée de la preuve du Théorème 1

Sous procédure

appliquée à un sommet v tel que (T, X) est sous-connexe en ses fils $w_1, \cdots, w_r$.

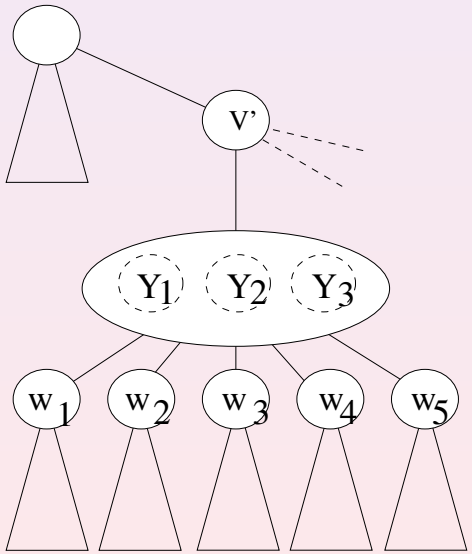


Idée de la preuve du Théorème 1

Sous procédure

appliquée à un sommet v tel que (T, X) est sous-connexe en ses fils $w_1, \cdots, w_r$.

détermine les composantes connexes de X_v : $Y_1, \ldots, Y_p$

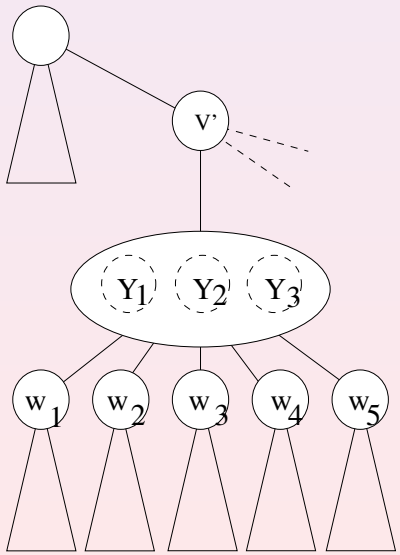


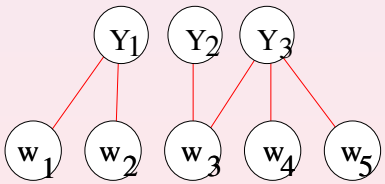
Idée de la preuve du Théorème 1

Sous procédure

appliquée à un sommet v tel que (T, X) est sous-connexe en ses fils $w_1, \cdots, w_r$.

crée un graphe biparti dont une partition est formée des composantes Y_i et l'autre des sommets w_j. Avec une arête entre Y_i et w_j ssi $Y_i \cap w_j \neq \emptyset$.



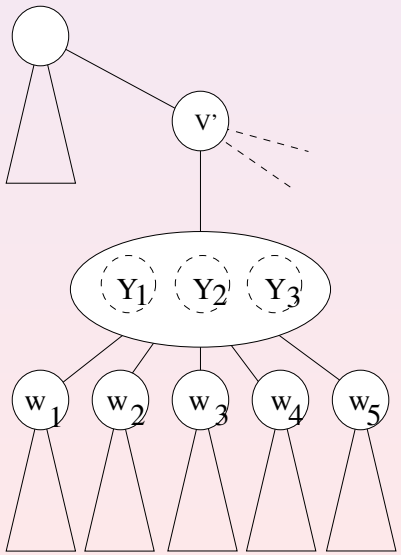


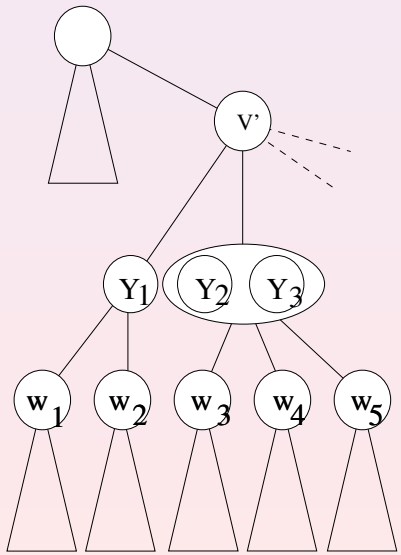
Idée de la preuve du Théorème 1

Sous procédure

appliquée à un sommet v tel que (T, X) est sous-connexe en ses fils $w_1, \cdots, w_r$.

La décomposition arborescente est modifiée en fonction des composantes connexes du graphe bipartie.



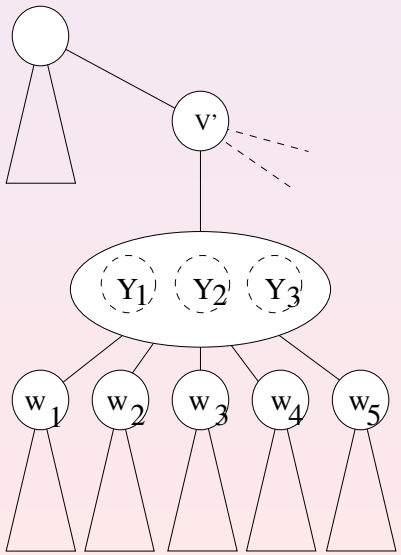


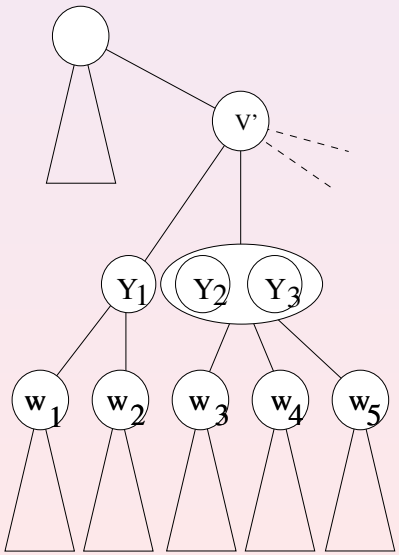
Idée de la preuve du Théorème 1

Sous procédure

appliquée à un sommet v tel que (T, X) est sous-connexe en ses fils $w_1, \cdots, w_r$.

La décomposition arborescente résultante est sous-connexe en les nouveaux descendants de v'.



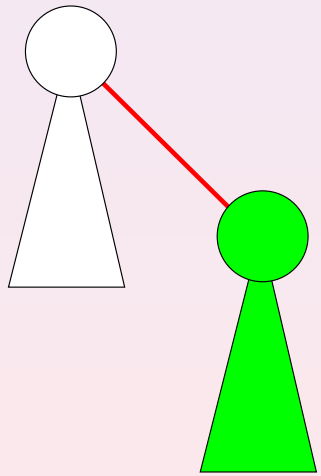


Idée de la preuve du Théorème 1

Phase 2

Entrée : décomposition arborescente sous-connexe

Il reste des arêtes qui ne sont pas connexes.

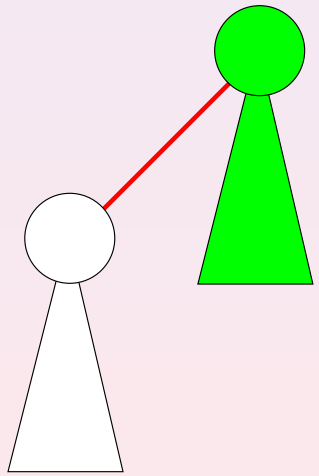


Idée de la preuve du Théorème 1

Phase 2

Entrée : décomposition arborescente sous-connexe
Il reste des arêtes qui ne sont pas connexes.

rotation de la décomposition.

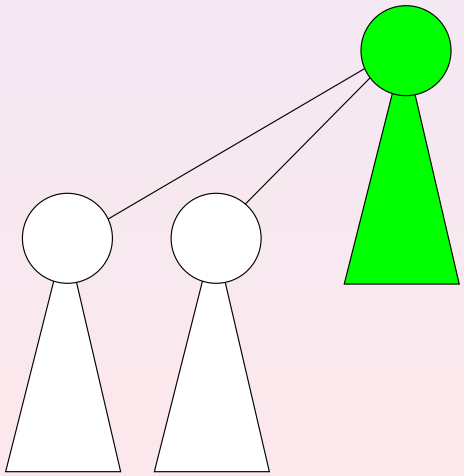


Idée de la preuve du Théorème 1

Phase 2

Entrée : décomposition arborescente sous-connexe
Il reste des arêtes qui ne sont pas connexes.

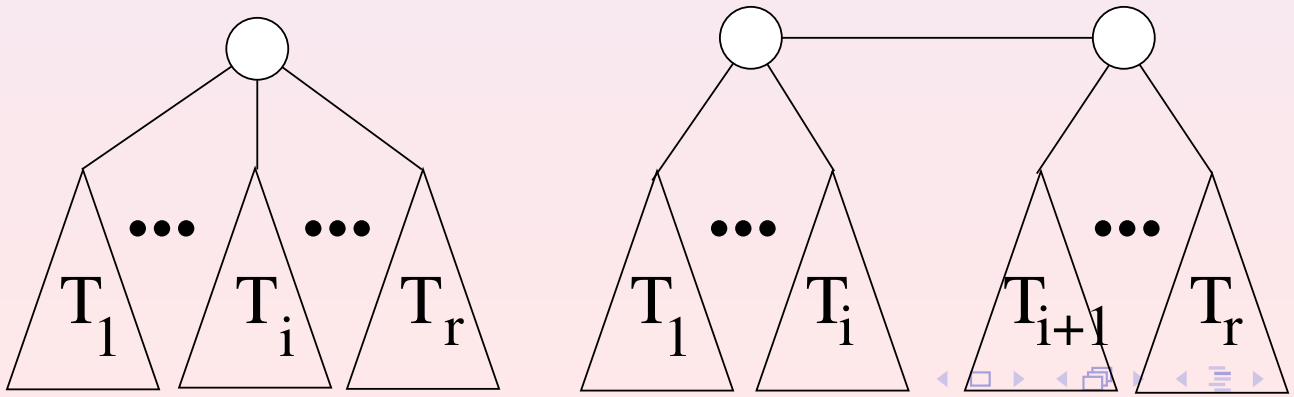
Application de la sous-procédure



Idée de la preuve du Théorème 2

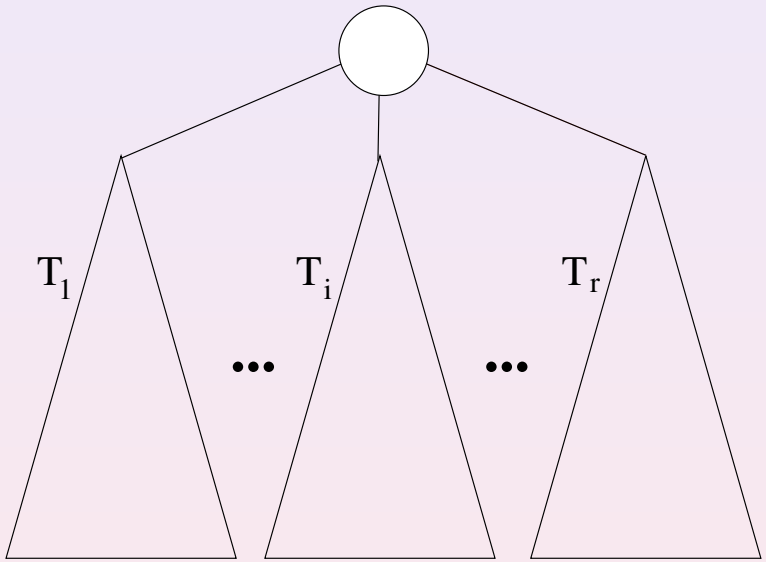
Pour tout graphe connexe G avec n sommets,
$\mathbf{cs}(G) \leq \mathbf{s}(G)\ (1 + \log_2 n)$

- preuve par induction sur n
- Robertson et Seymour. Graph Minors II. Algorithmic Aspects of Tree-Width. J. of Alg 7, 1986.
  - Pour toute décomposition (T, X) d'un graphe G avec n sommets, il existe un (ou deux sommets adjacents) de T tel(s) que :
  - pour tout $1 \leq j \leq r$, $|G[T_j]| \leq n/2$



Idée de la preuve du Théorème 2

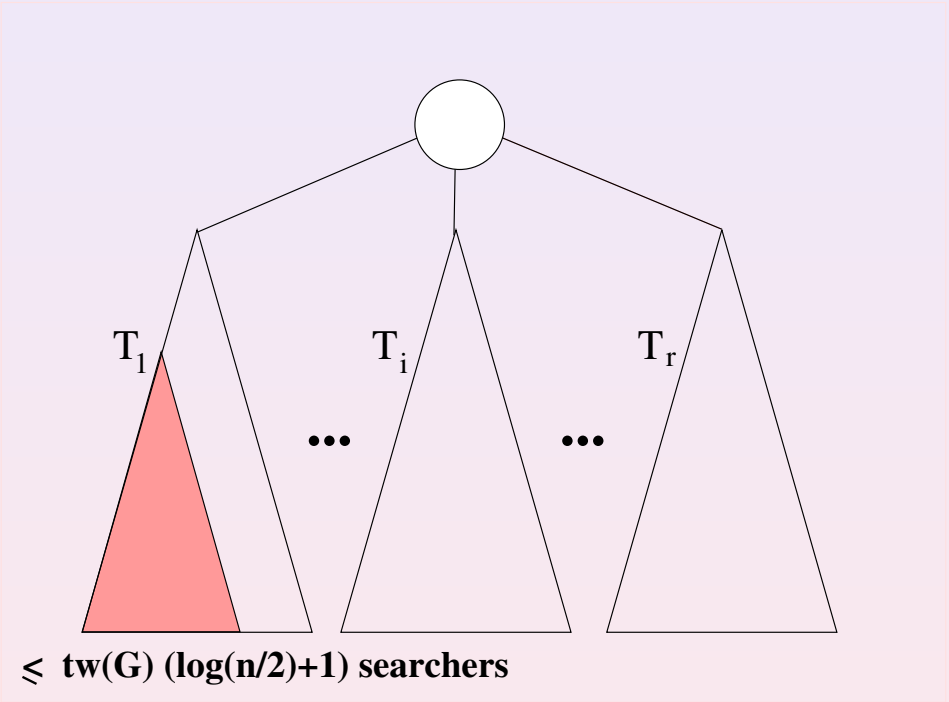
A partir d'une décomposition arborescente connexe de G



Pour tout $1 \leq i \leq r$, $G[T_i]$ est un sous-graphe connexe avec au plus $n/2$ sommets.

Idée de la preuve du Théorème 2

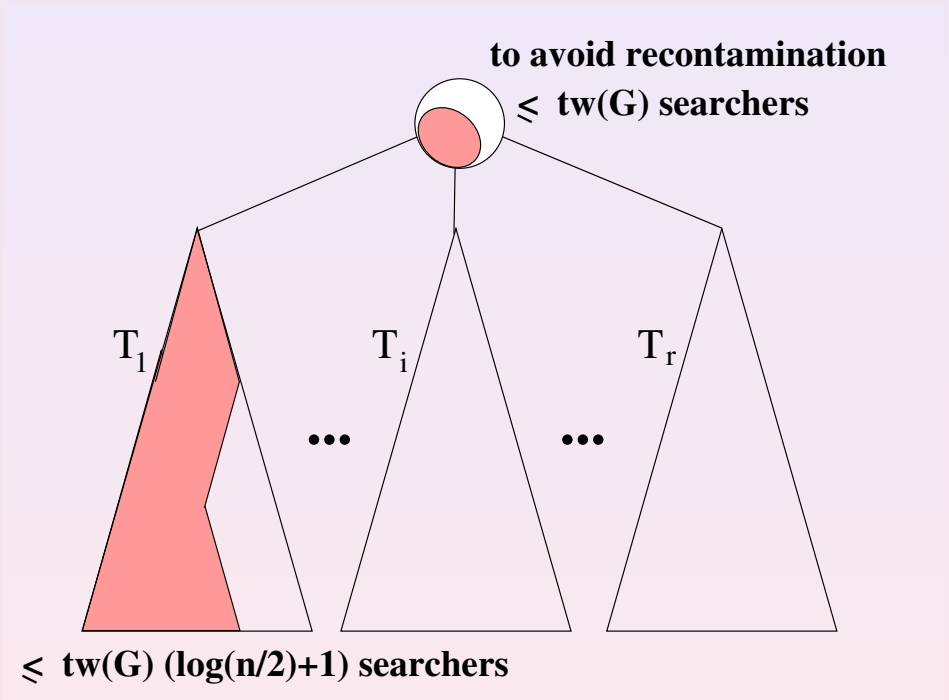
A partir d'une décomposition arborescente connexe de G



il existe une stratégie connexe de $G[T_1]$, avec au plus $\mathbf{tw}(G)(\log(n/2)+1)$ agents.

Idée de la preuve du Théorème 2

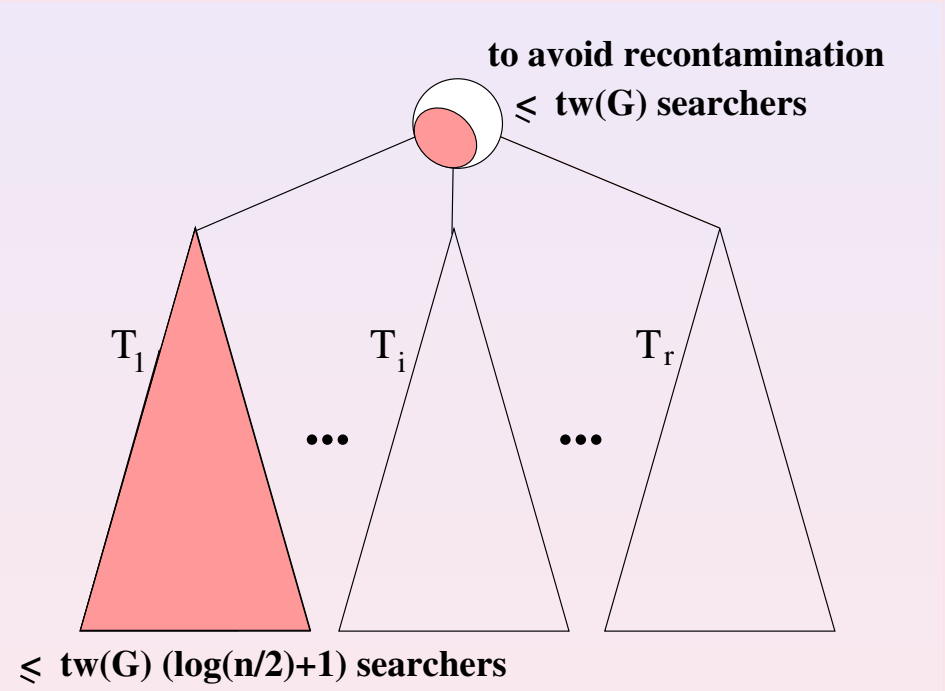
A partir d'une décomposition arborescente connexe de G



Au plus **tw**(G) agents sont suffisants pour protéger $G[T_1]$ de la recontamination du reste de G.

Idée de la preuve du Théorème 2

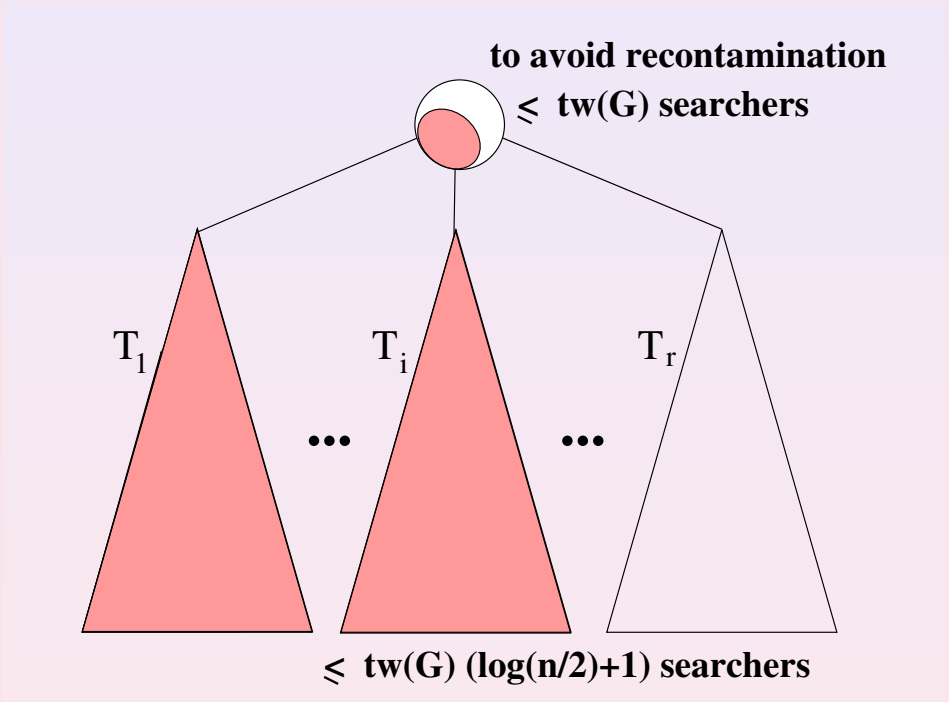
A partir d'une décomposition arborescente connexe de G



On peut terminer le nettoyage de $G[T_1]$.

Idée de la preuve du Théorème 2

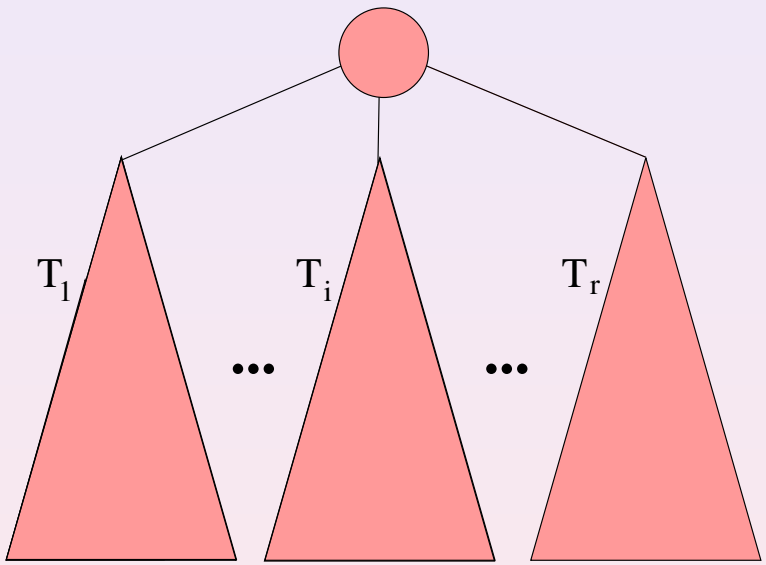
A partir d'une décomposition arborescente connexe de G



Alors, les **tw**$(G)(\log(n/2)+1)$ agents peuvent être utilisés pour nettoyer un autre sous-graphe $G[T_i]$, etc...

Idée de la preuve du Théorème 2

A partir d'une décomposition arborescente connexe de G



Stratégie connexe utilisant au plus $\mathbf{tw}(G)(\log n + 1)$ agents.
D'où, $\mathbf{cs}(G) \leq \mathbf{s}(G)(\log n + 1)$

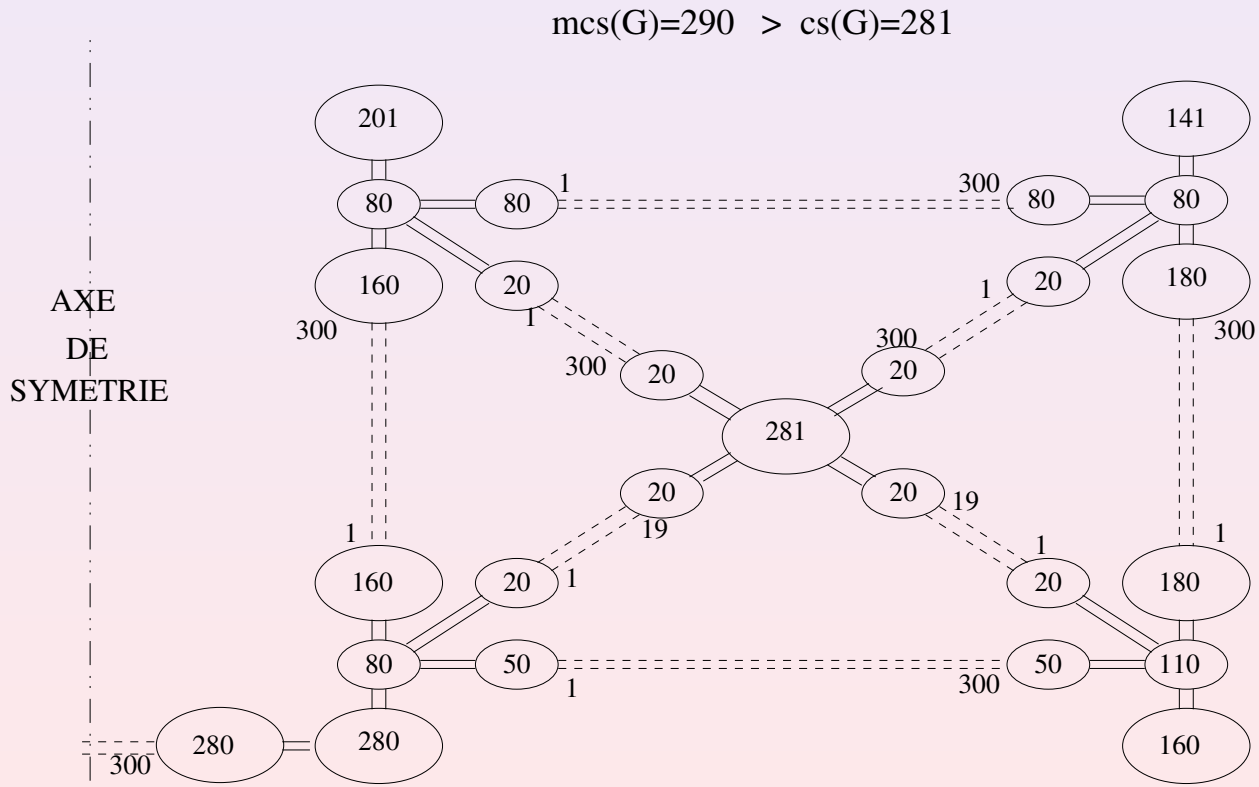
Connexe vs. non connexe

Déterminer $\mathbf{cs}(G)$ (resp. $cvs(G)$) est NP-difficile.
Question : est-ce NP-complet ?

Connexe vs. non connexe

Dans le cas connexe, la recontamination aide

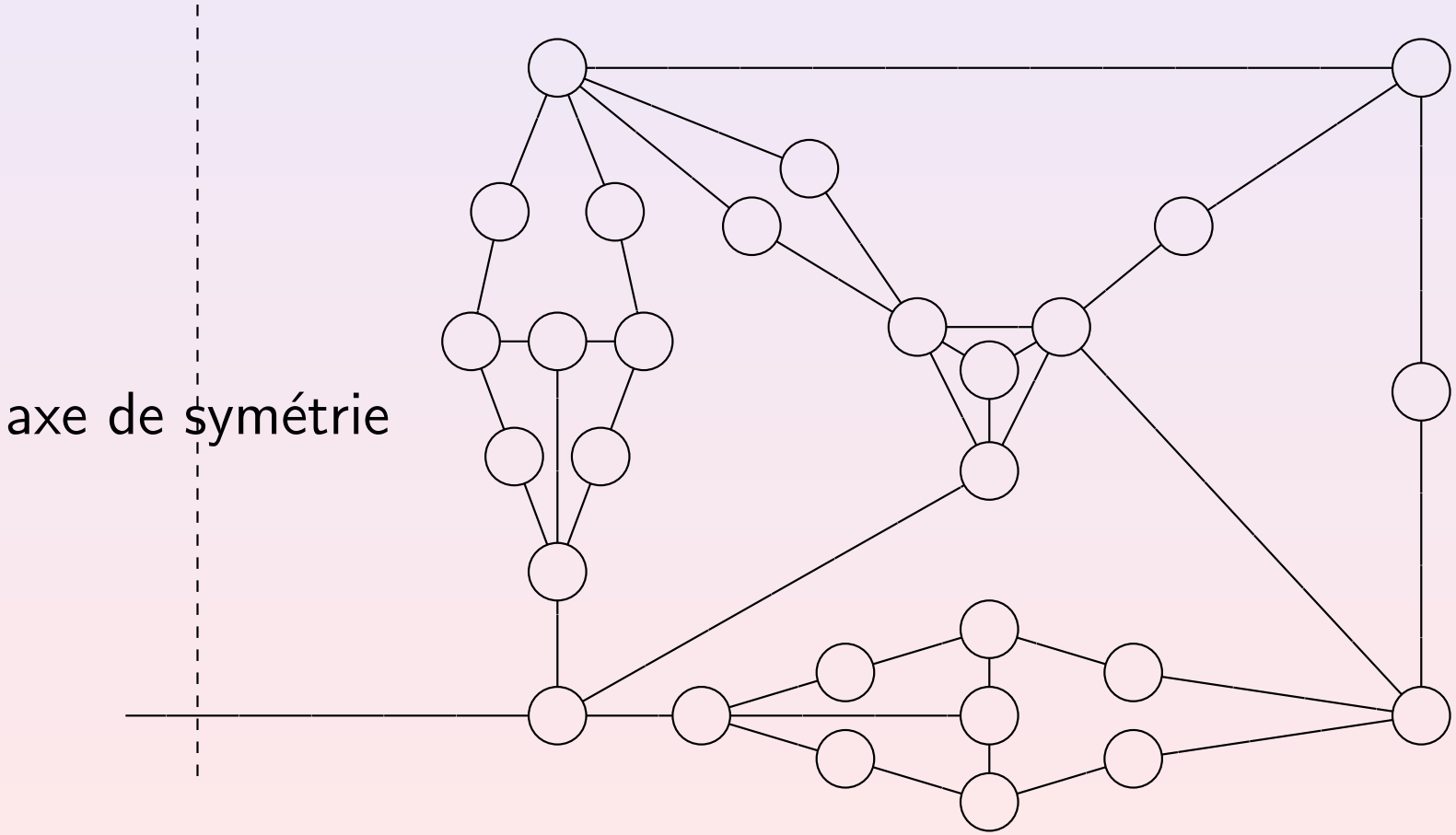
Yang, Dyer, Alspach. Sweeping Graphs and Digraphs. [ISAAC04] (cas d'un fugitif invisible)



Connexe vs non connexe

Dans le cas connexe visible, la recontamination aide

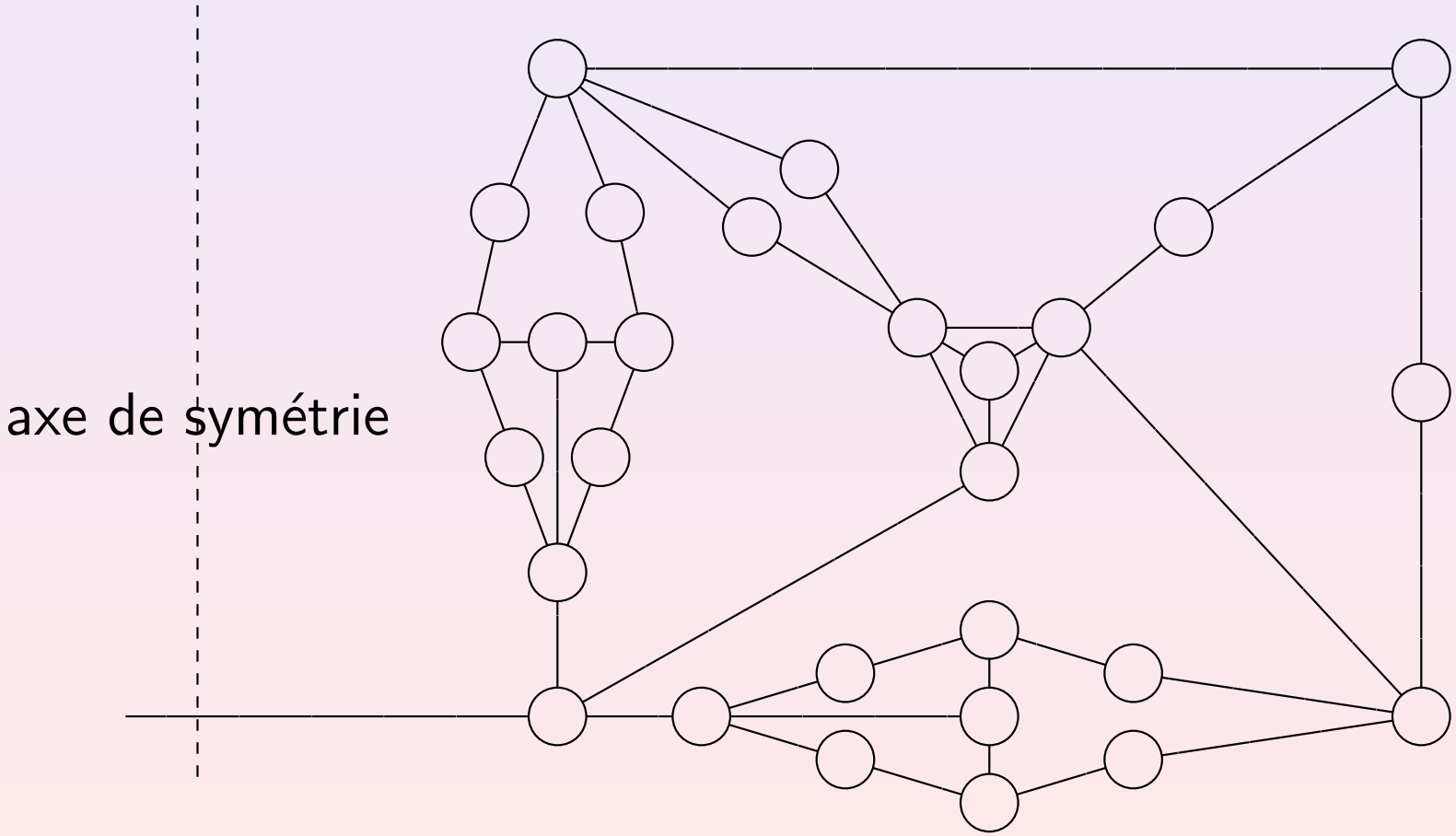
$\exists$ des graphes G tel que $mcvs(G) > cvs(G)$. Fraigniaud, N.



Connexe vs non connexe

Cas d'un fugitif visible

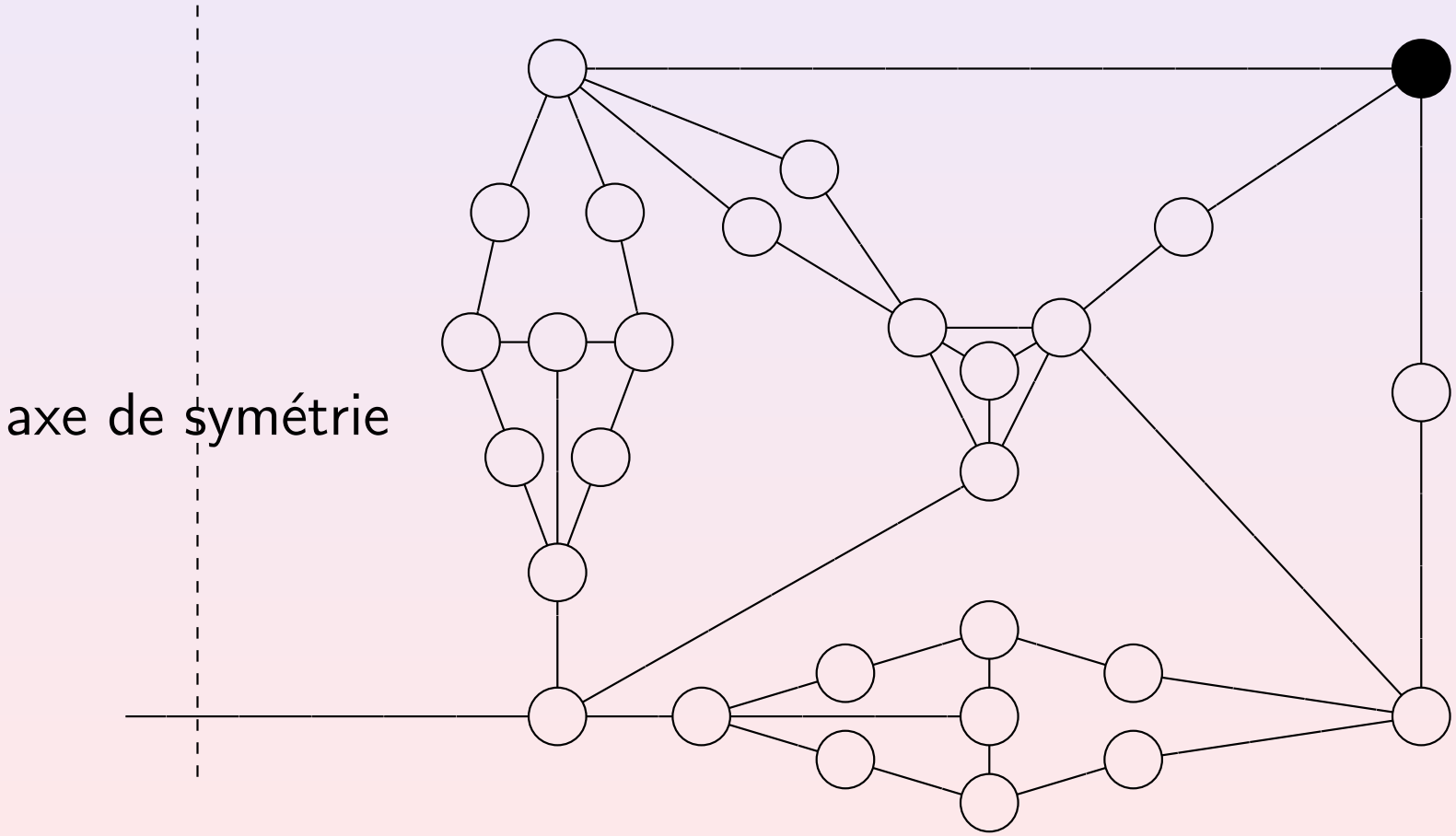
Soit G le graphe ci-dessous : $mcvs(G) > cvs(G) = 4$.



Connexe vs non connexe

Cas d'un fugitif visible

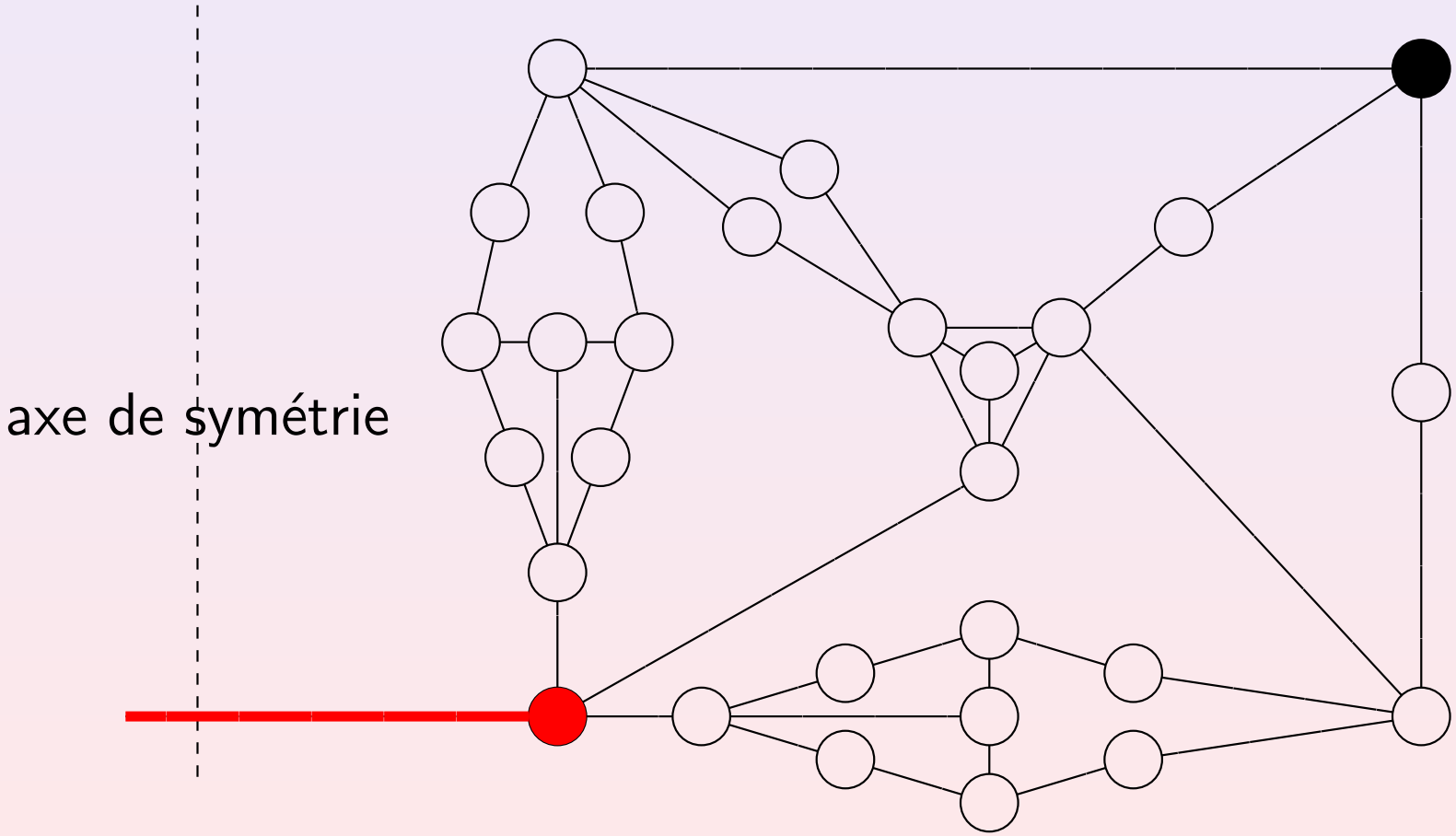
Soit G le graphe ci-dessous : $mcvs(G) > cvs(G) = 4$.



Connexe vs non connexe

Cas d'un fugitif visible

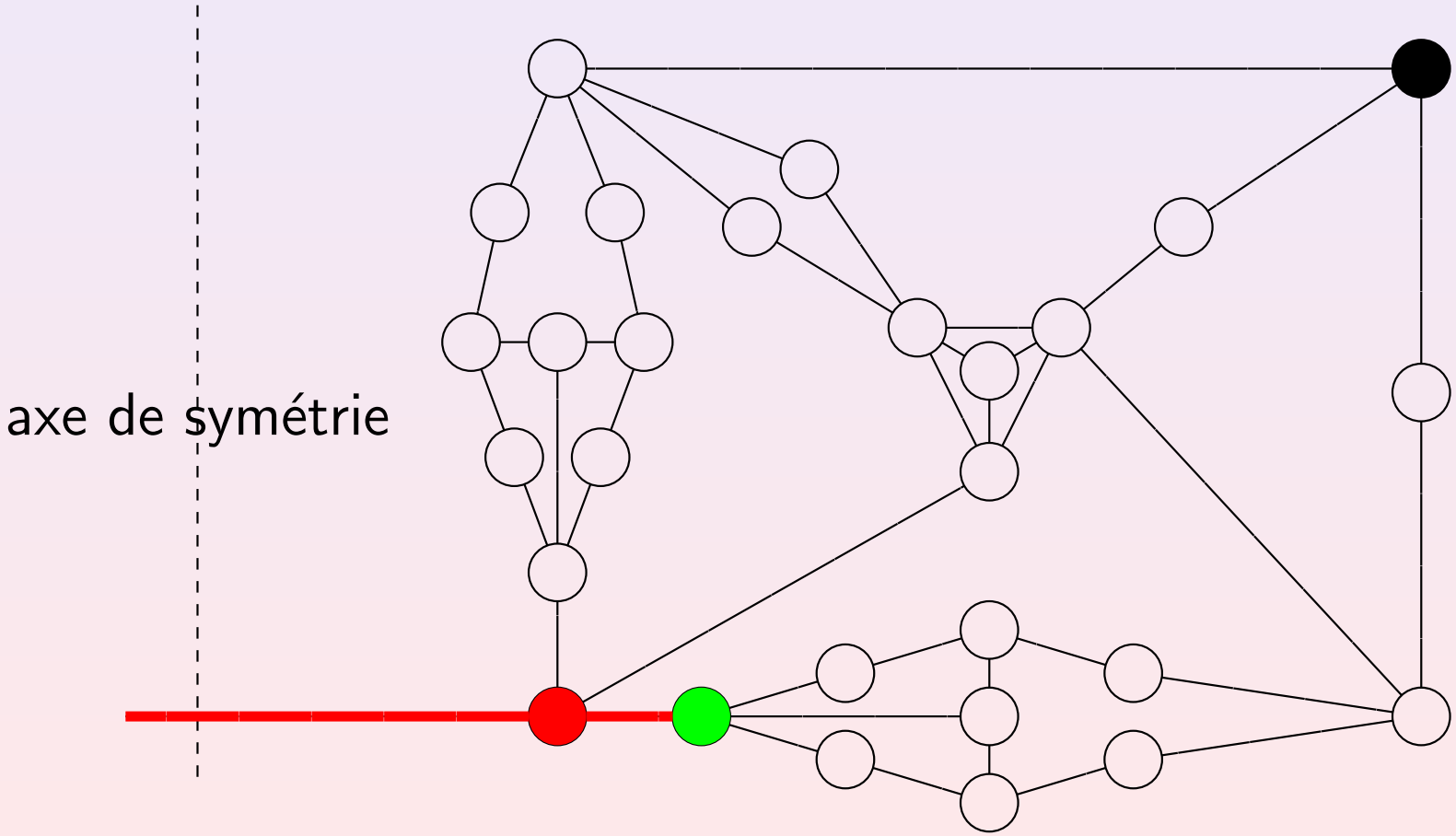
Soit G le graphe ci-dessous : $mcvs(G) > cvs(G) = 4$.



Connexe vs non connexe

Cas d'un fugitif visible

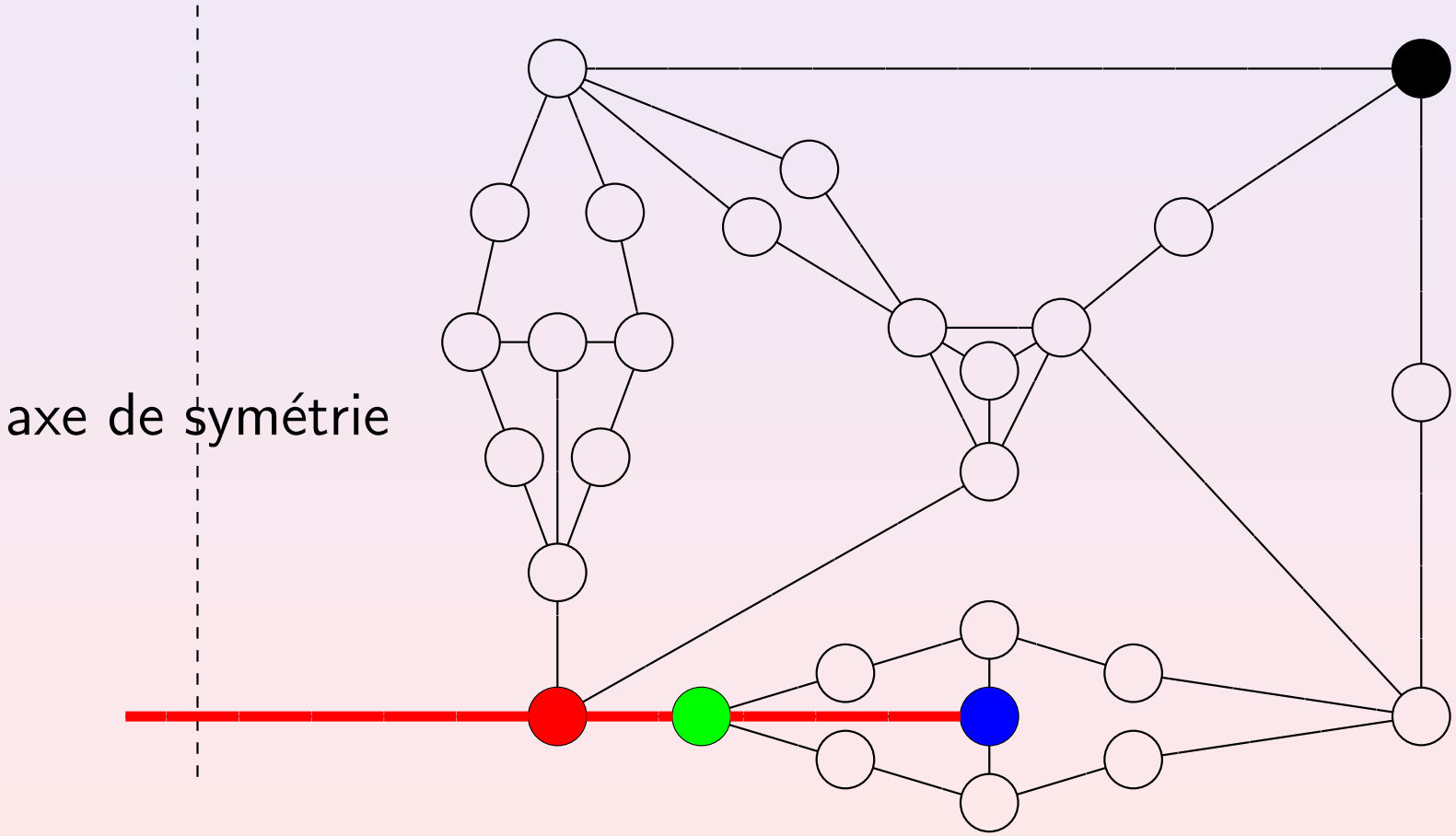
Soit G le graphe ci-dessous : $mcvs(G) > cvs(G) = 4$.



Connexe vs non connexe

Cas d'un fugitif visible

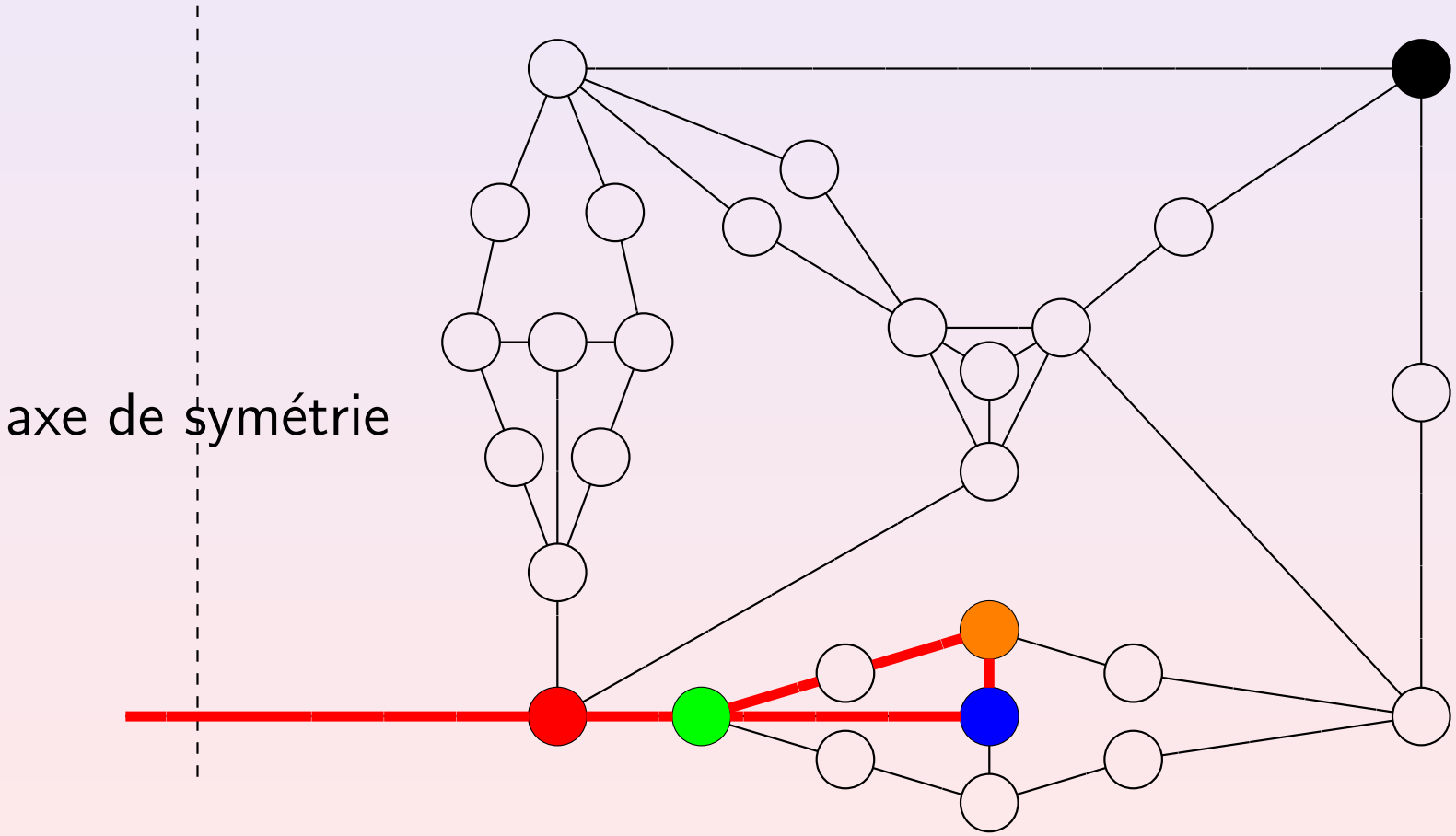
Soit G le graphe ci-dessous : $mcvs(G) > cvs(G) = 4$.



Connexe vs non connexe

Cas d'un fugitif visible

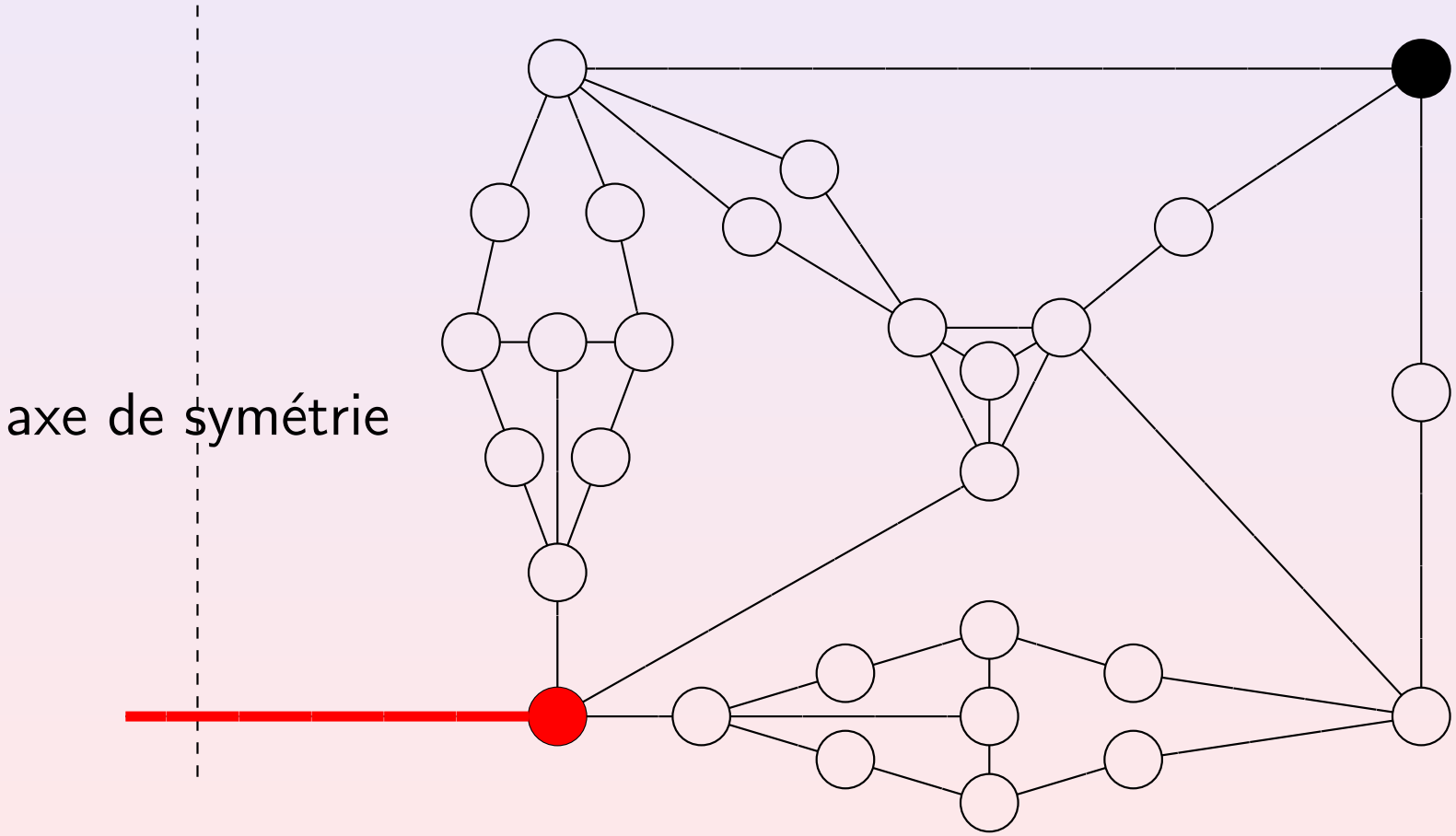
Soit G le graphe ci-dessous : $mcvs(G) > cvs(G) = 4$.



Connexe vs non connexe

Cas d'un fugitif visible

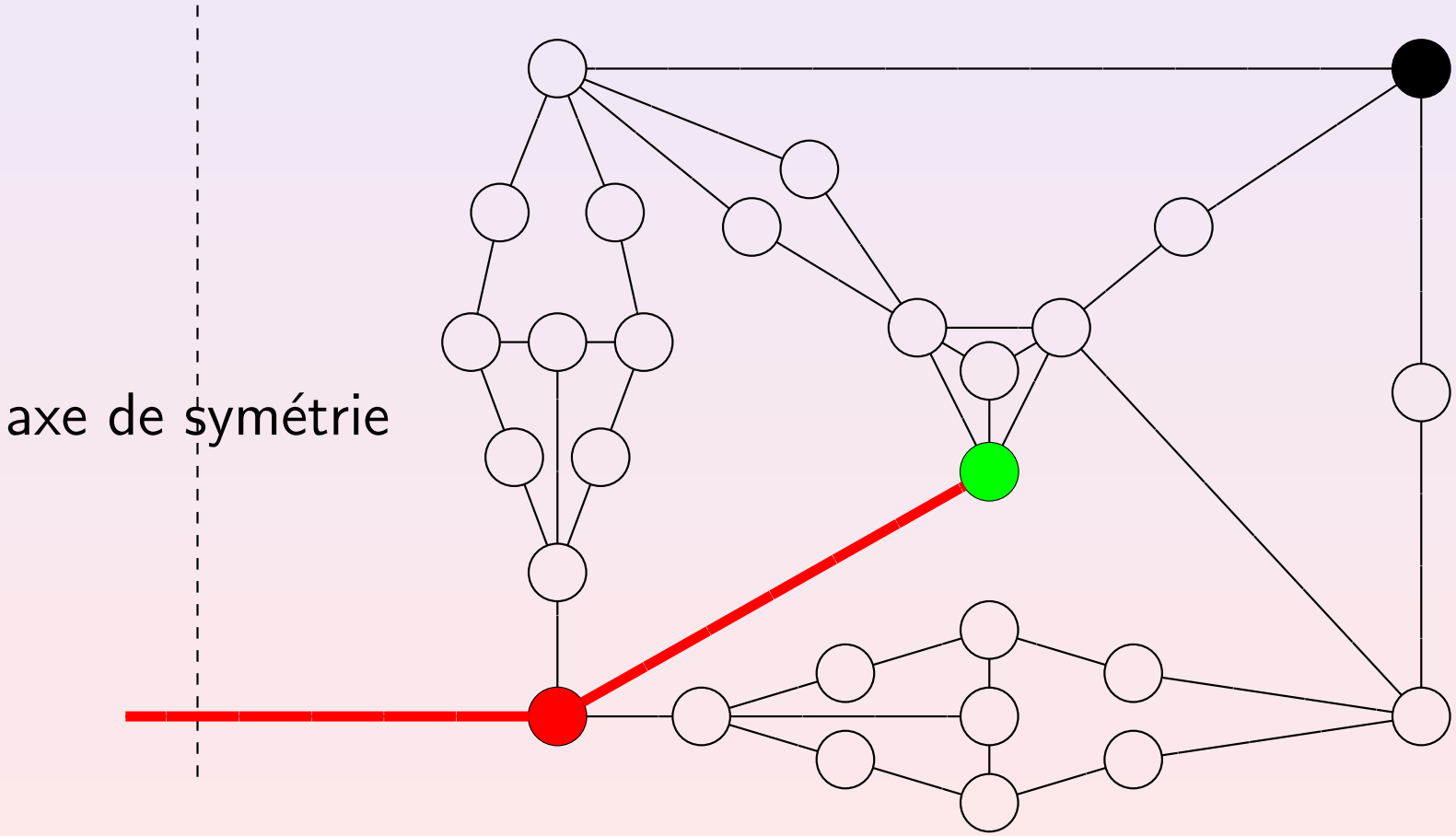
Soit G le graphe ci-dessous : $mcvs(G) > cvs(G) = 4$.



Connexe vs non connexe

Cas d'un fugitif visible

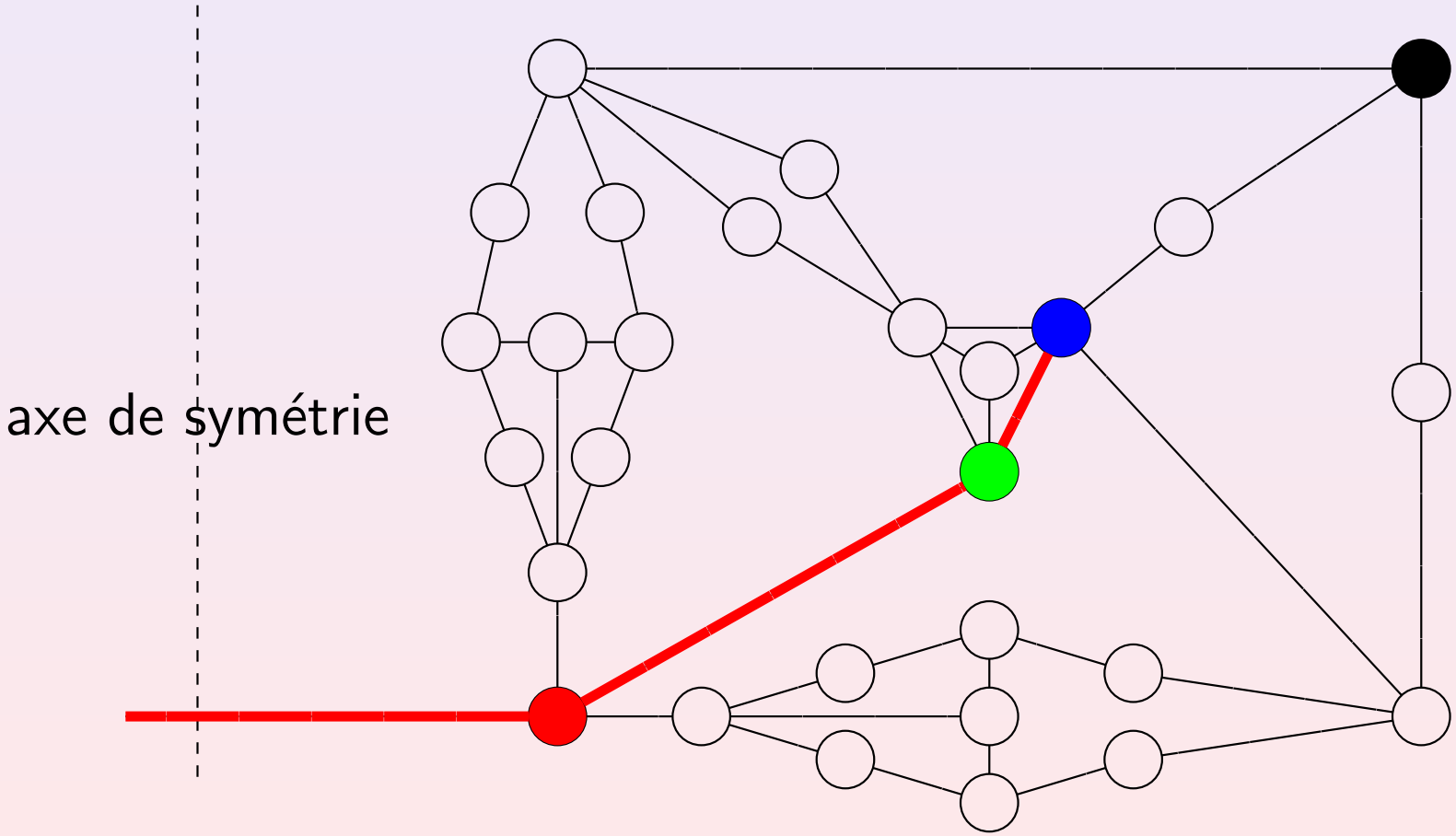
Soit G le graphe ci-dessous : $mcvs(G) > cvs(G) = 4$.



Connexe vs non connexe

Cas d'un fugitif visible

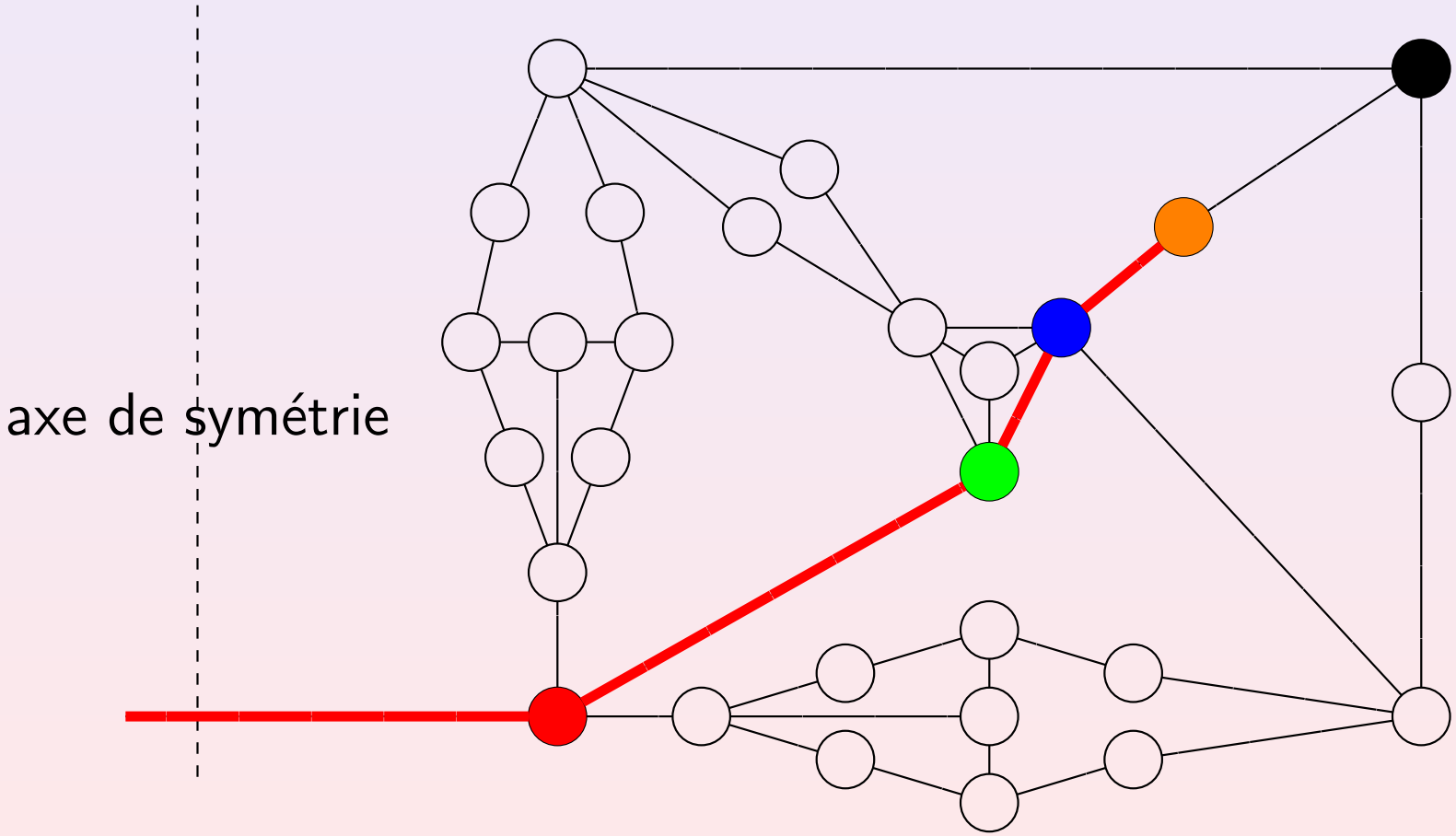
Soit G le graphe ci-dessous : $mcvs(G) > cvs(G) = 4$.



Connexe vs non connexe

Cas d'un fugitif visible

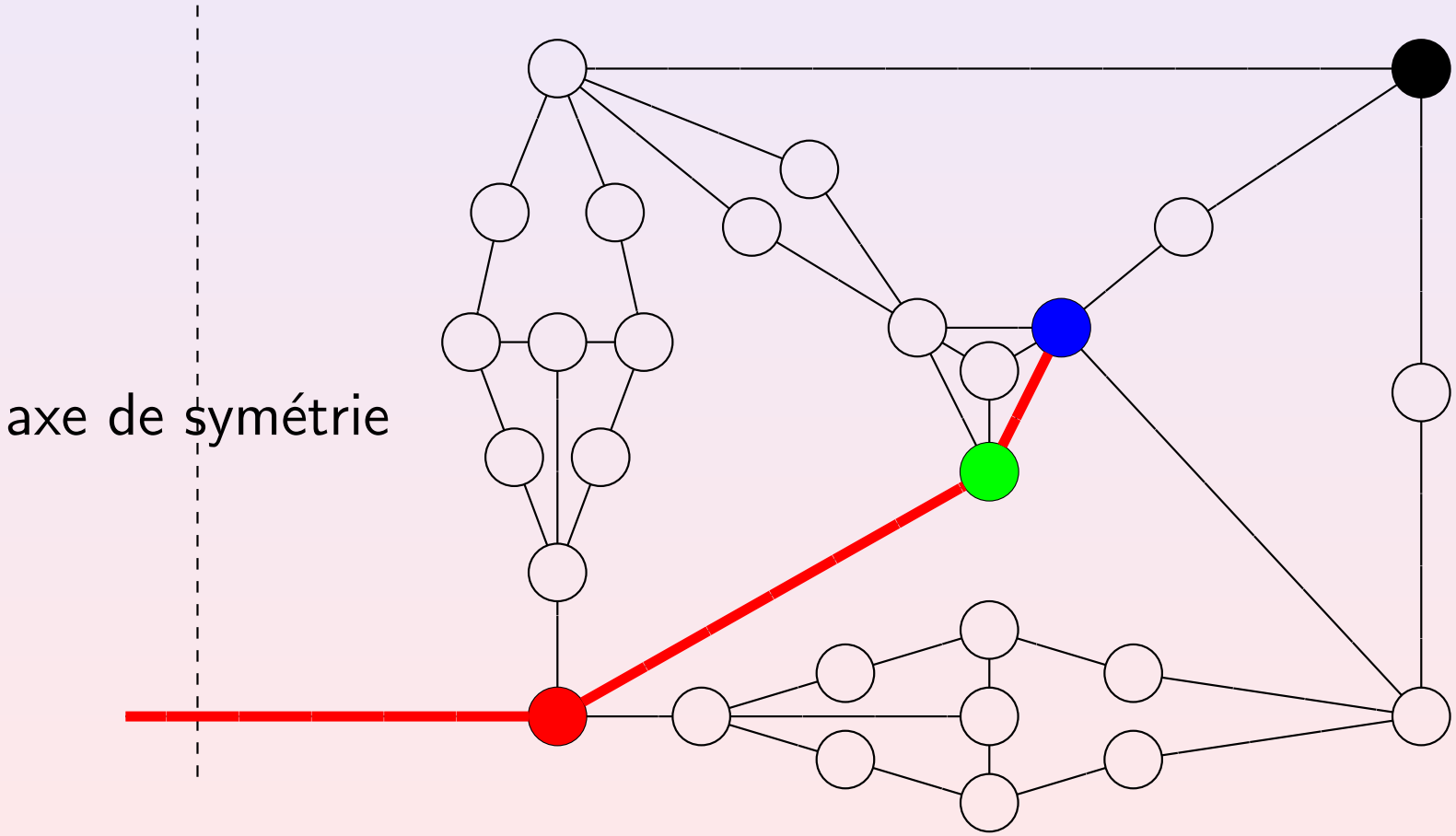
Soit G le graphe ci-dessous : $mcvs(G) > cvs(G) = 4$.



Connexe vs non connexe

Cas d'un fugitif visible

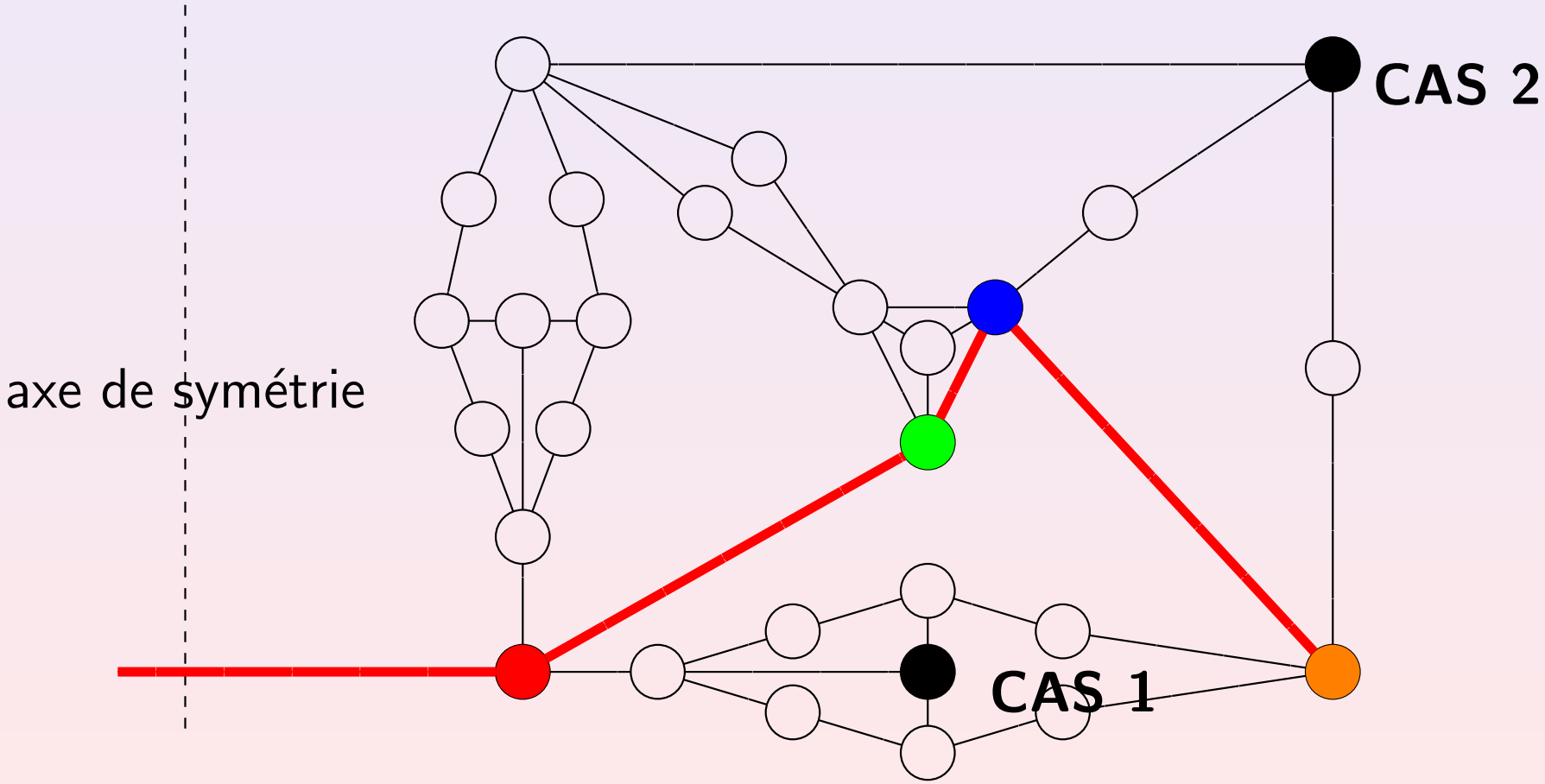
Soit G le graphe ci-dessous : $mcvs(G) > cvs(G) = 4$.



Connexe vs non connexe

Cas d'un fugitif visible

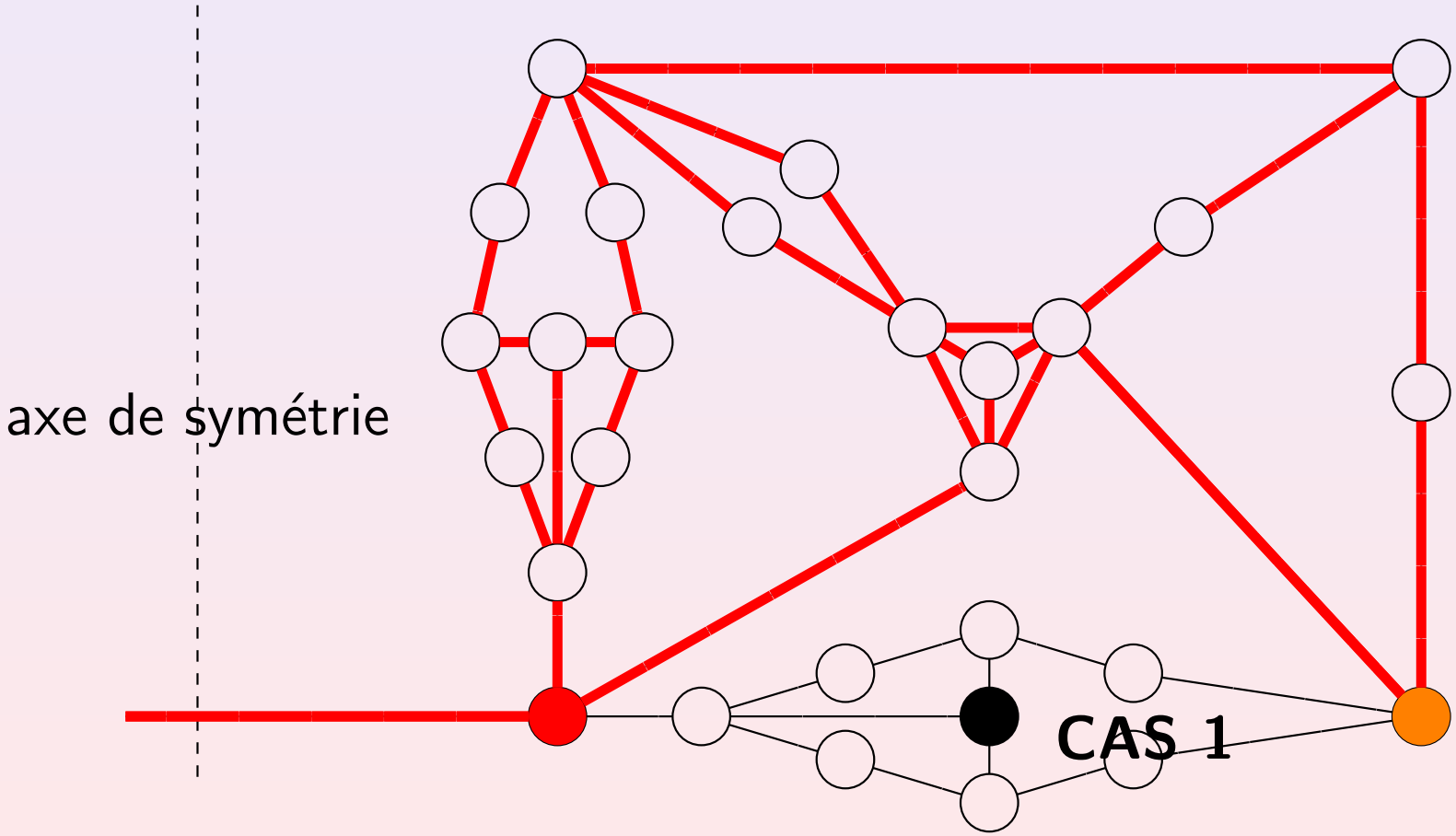
Soit G le graphe ci-dessous : $mcvs(G) > cvs(G) = 4$.



Connexe vs non connexe

Cas d'un fugitif visible

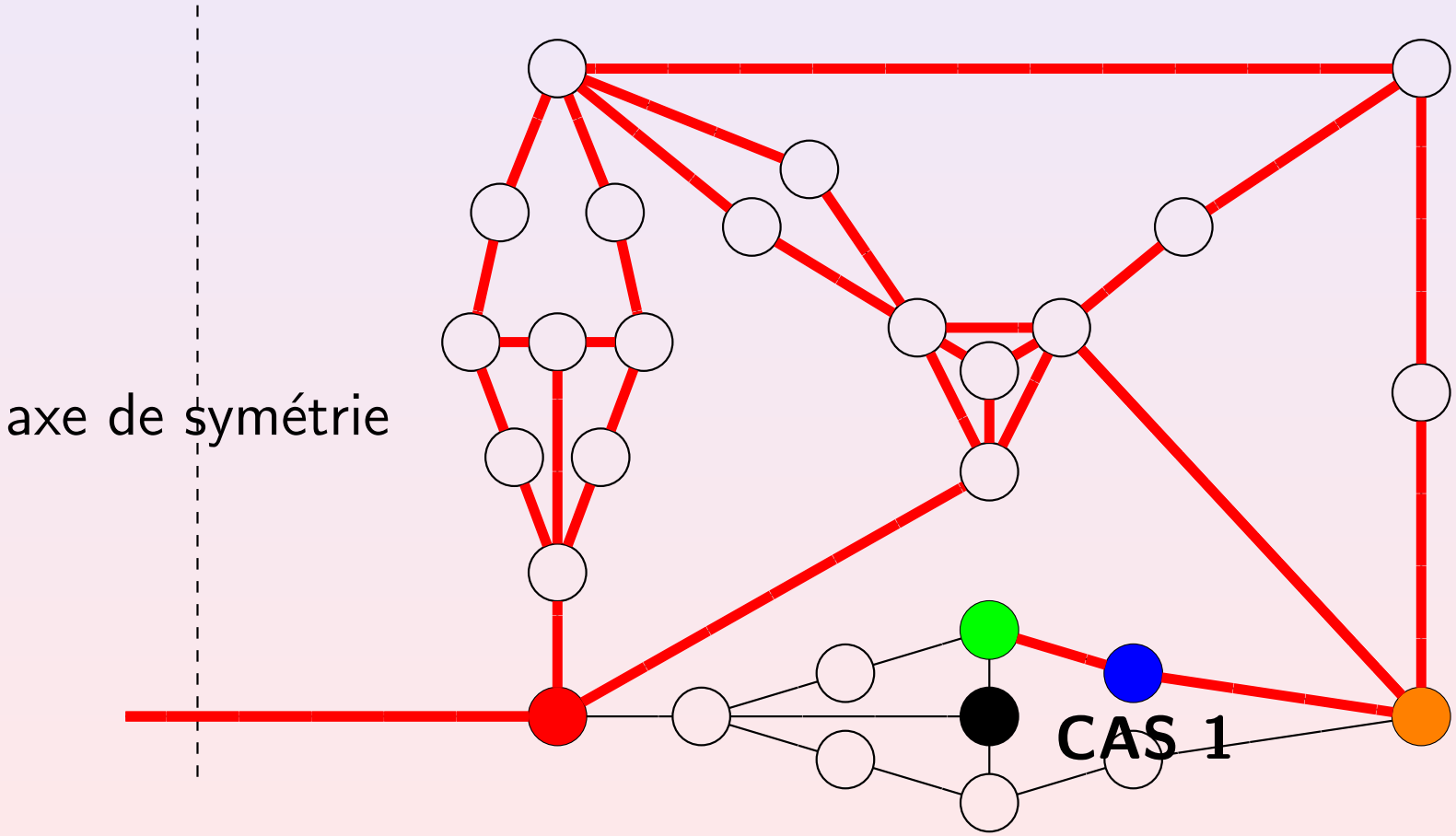
Soit G le graphe ci-dessous : $mcvs(G) > cvs(G) = 4$.



Connexe vs non connexe

Cas d'un fugitif visible

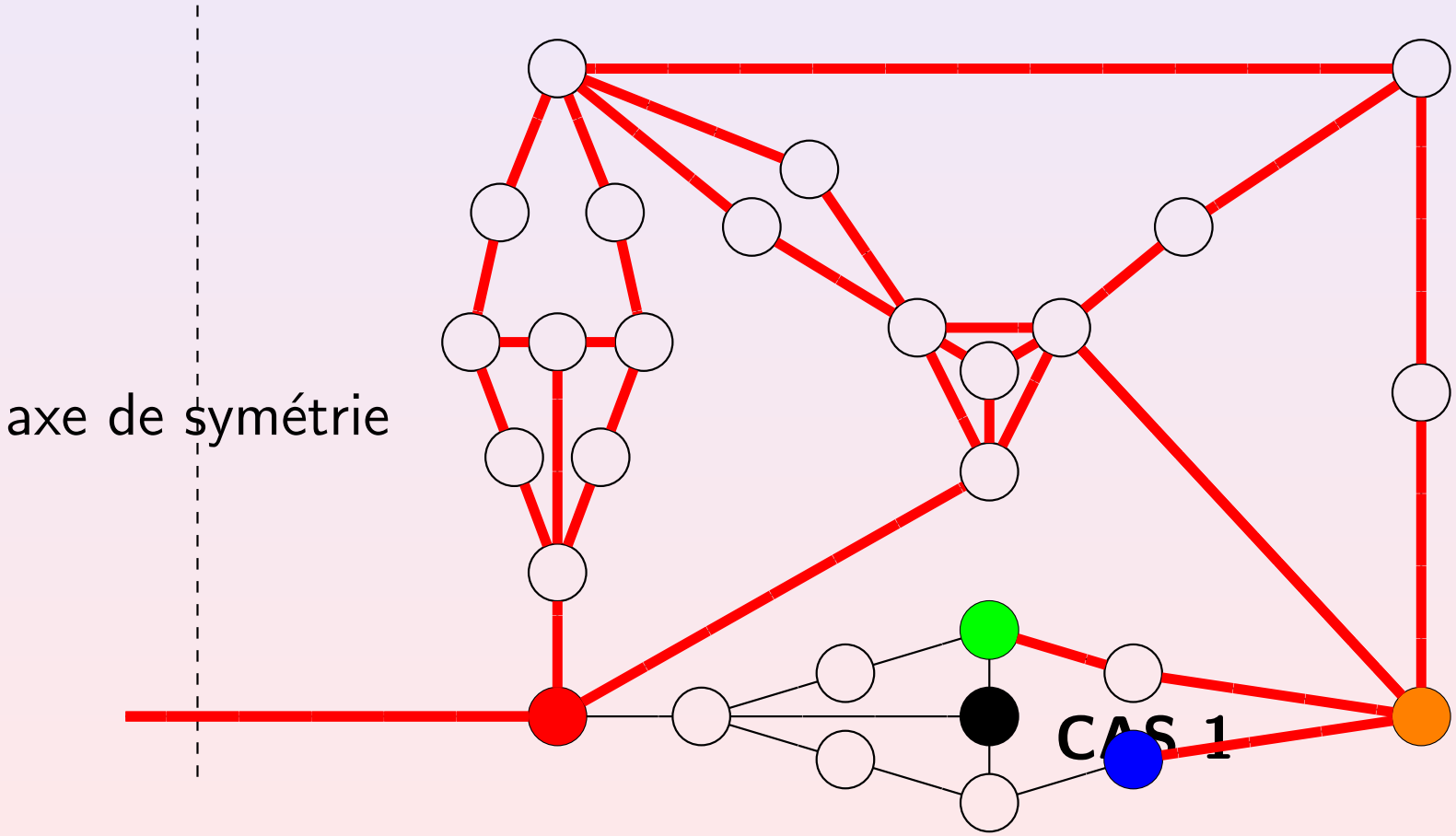
Soit G le graphe ci-dessous : $mcvs(G) > cvs(G) = 4$.



Connexe vs non connexe

Cas d'un fugitif visible

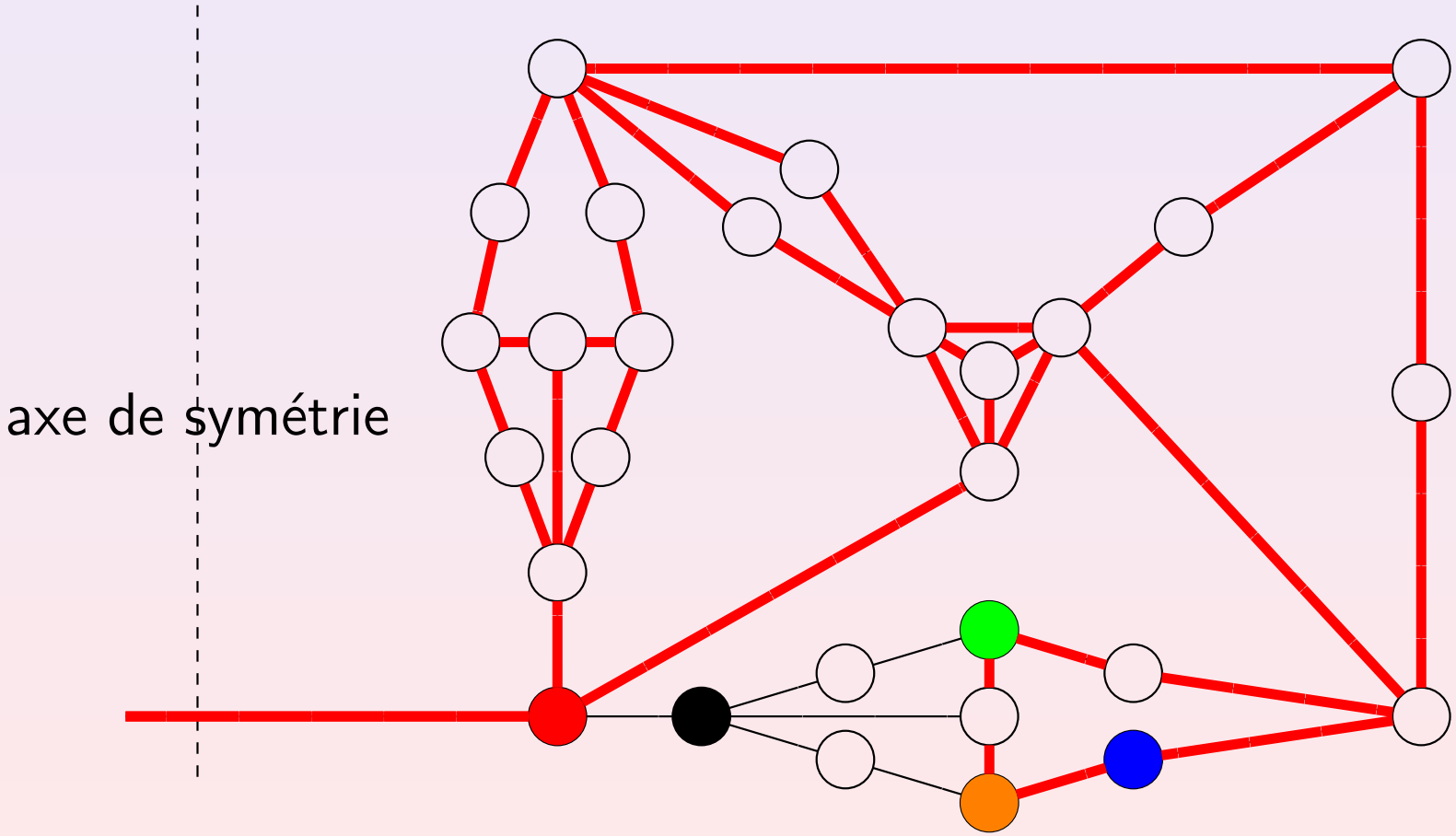
Soit G le graphe ci-dessous : $mcvs(G) > cvs(G) = 4$.



Connexe vs non connexe

Cas d'un fugitif visible

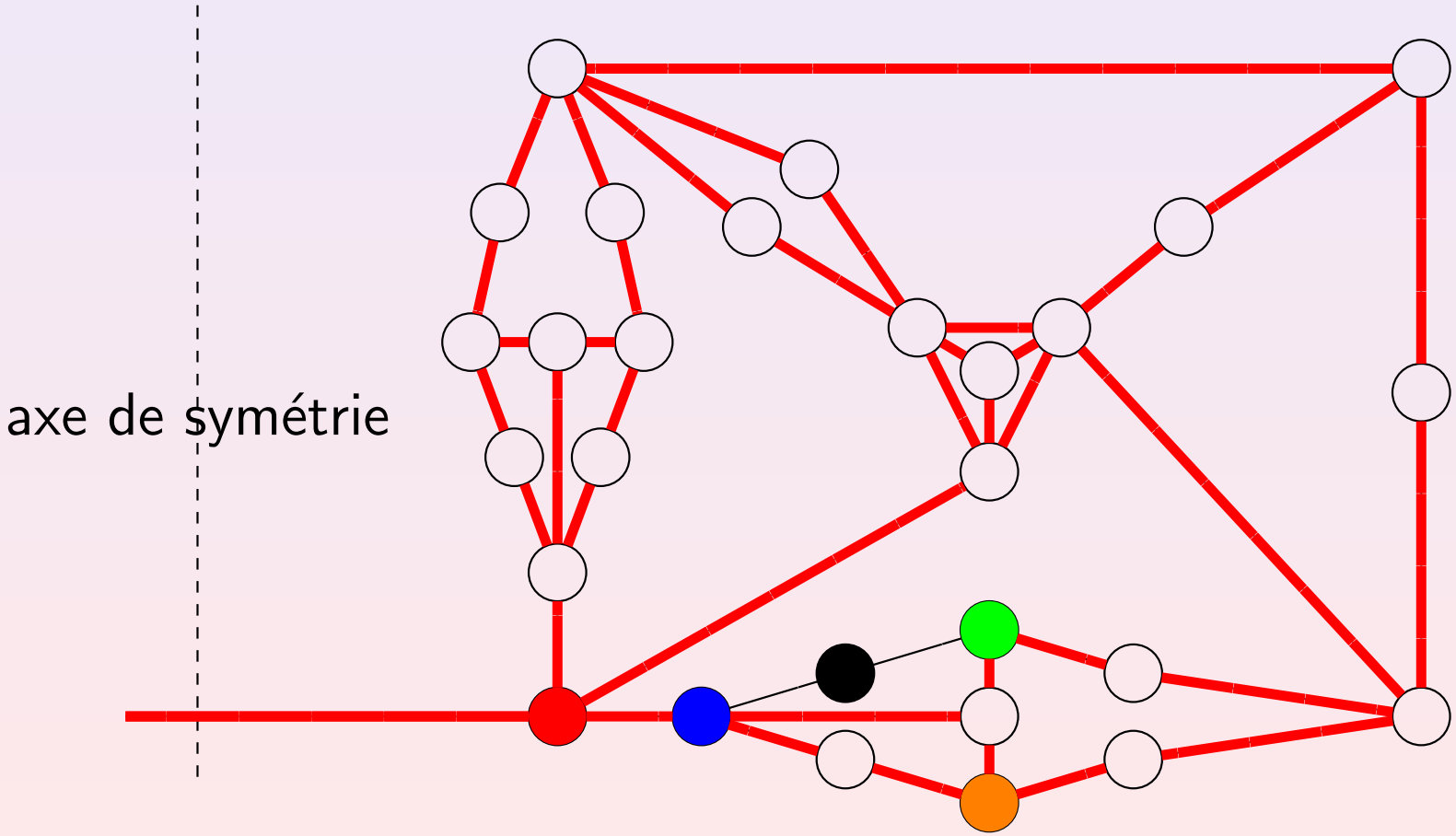
Soit G le graphe ci-dessous : $mcvs(G) > cvs(G) = 4$.



Connexe vs non connexe

Cas d'un fugitif visible

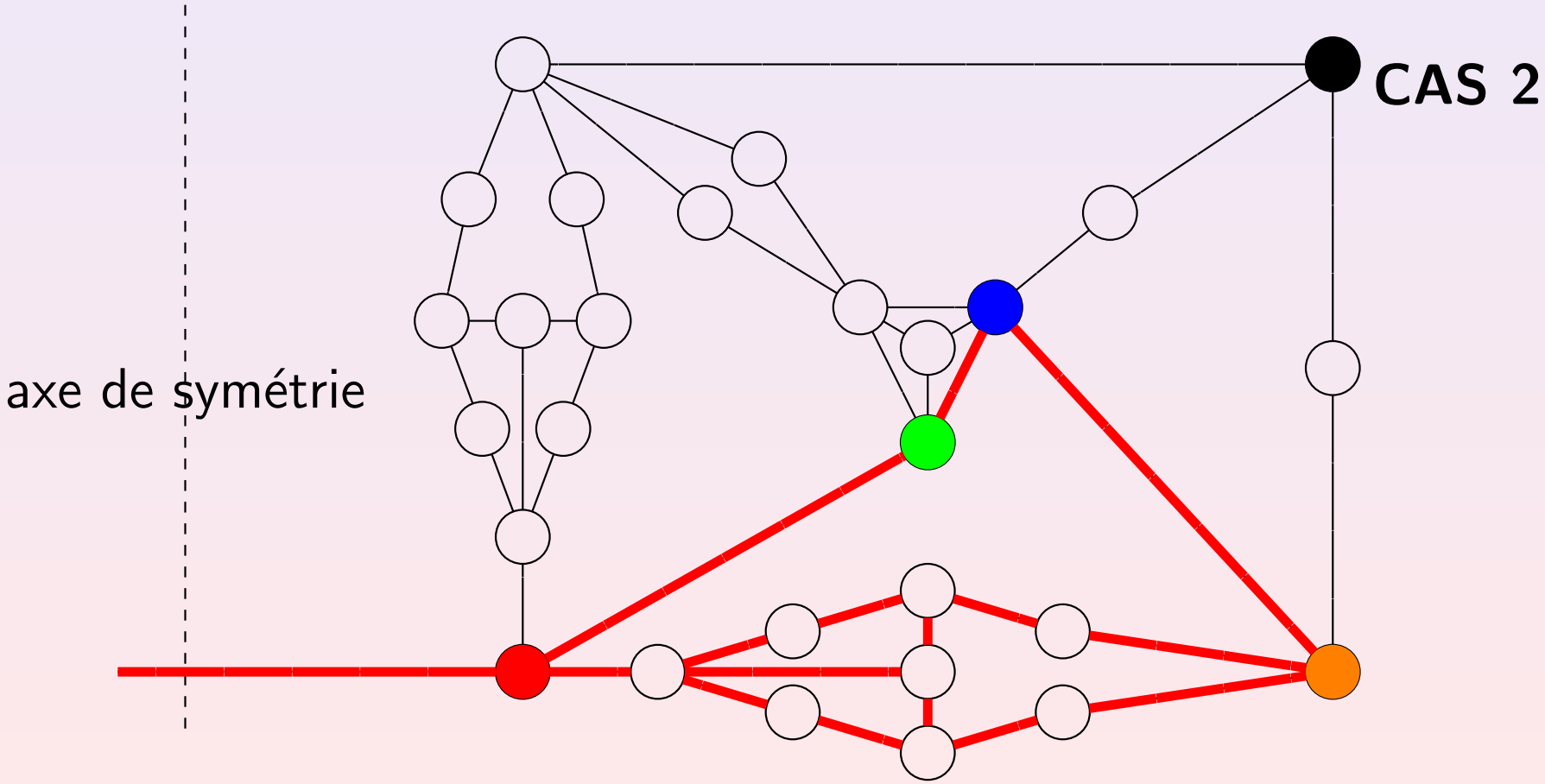
Soit G le graphe ci-dessous : $mcvs(G) > cvs(G) = 4$.



Connexe vs non connexe

Cas d'un fugitif visible

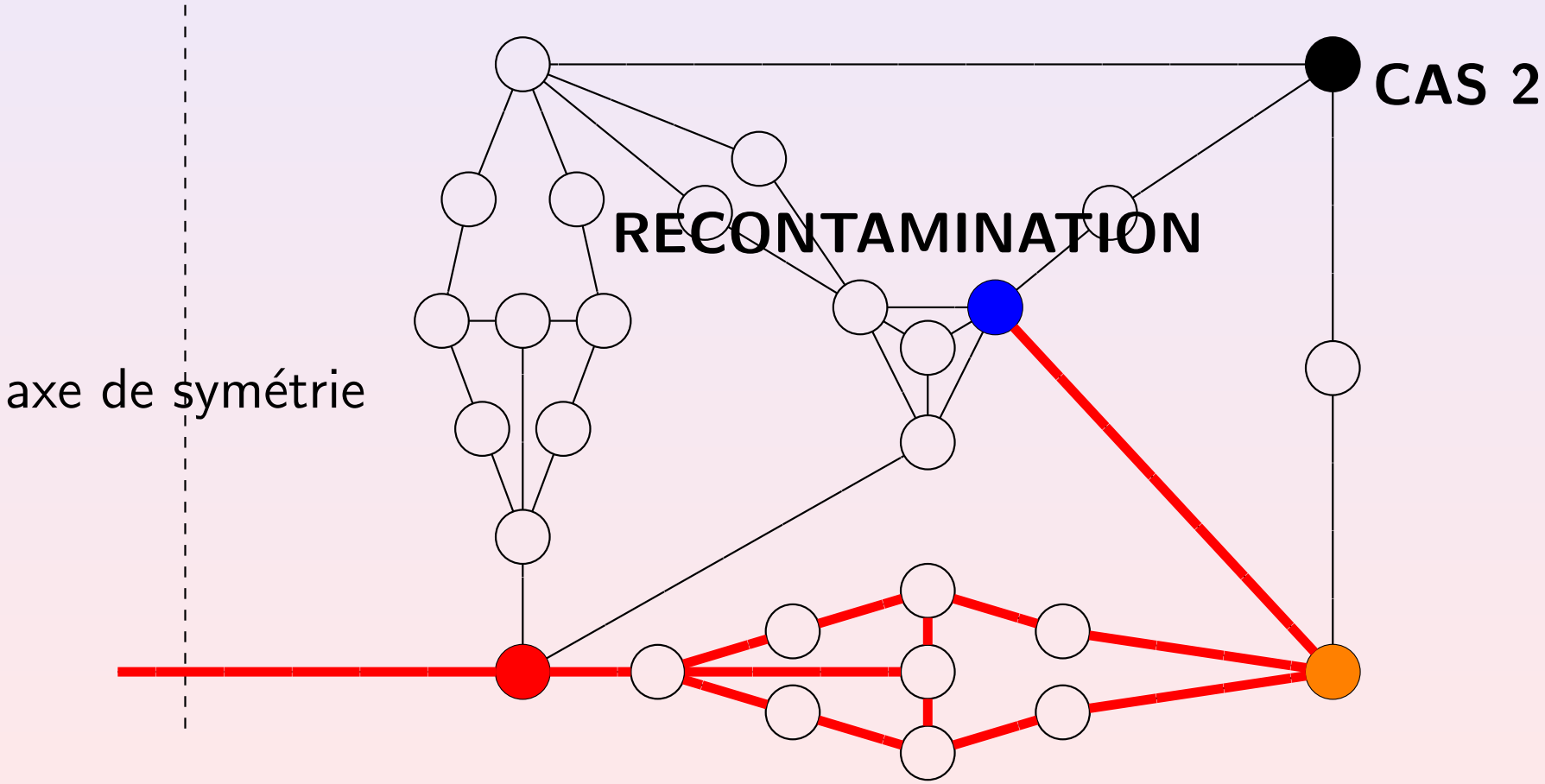
Soit G le graphe ci-dessous : $mcvs(G) > cvs(G) = 4$.



Connexe vs non connexe

Cas d'un fugitif visible

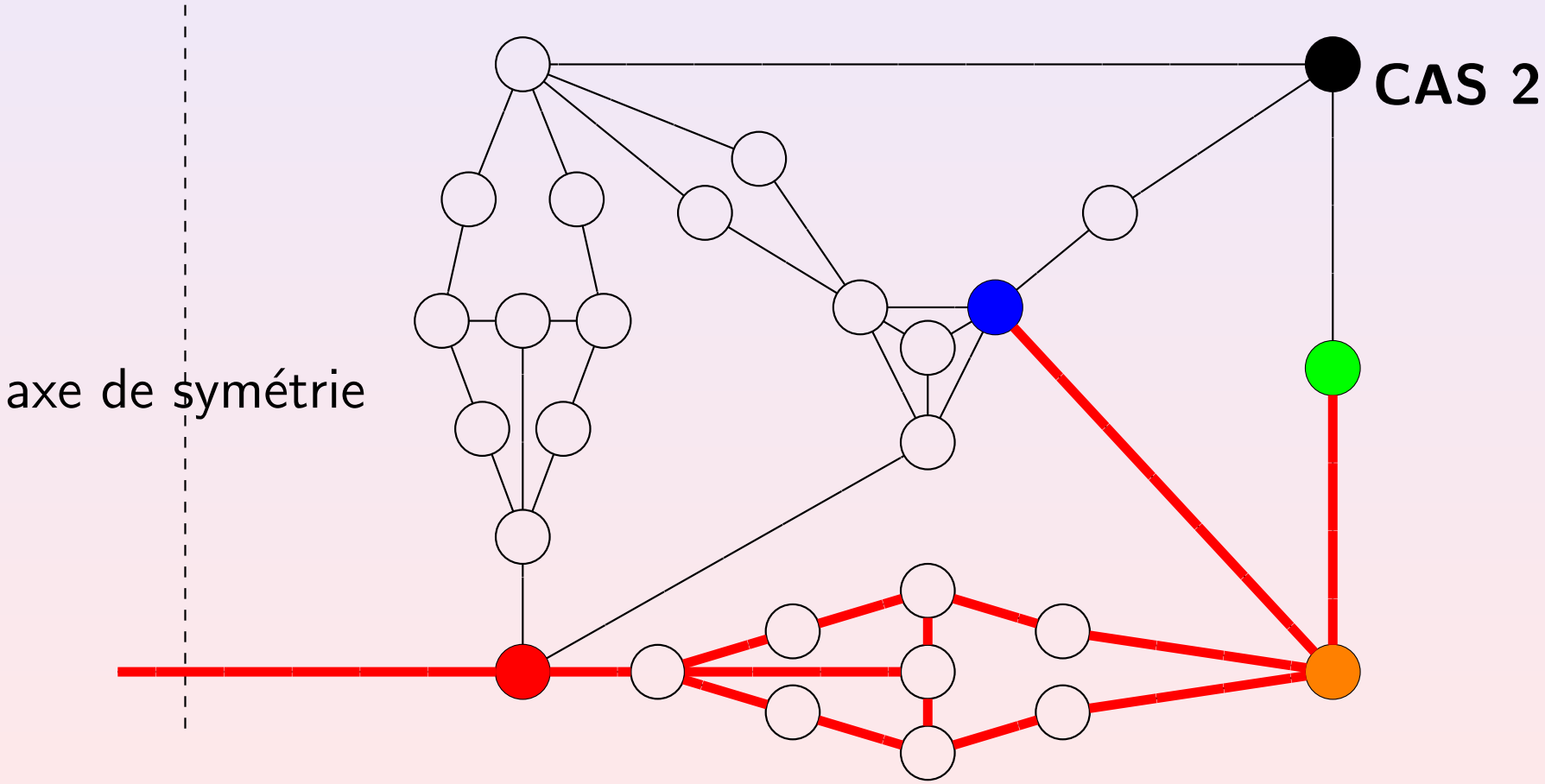
Soit G le graphe ci-dessous : $mcvs(G) > cvs(G) = 4$.



Connexe vs non connexe

Cas d'un fugitif visible

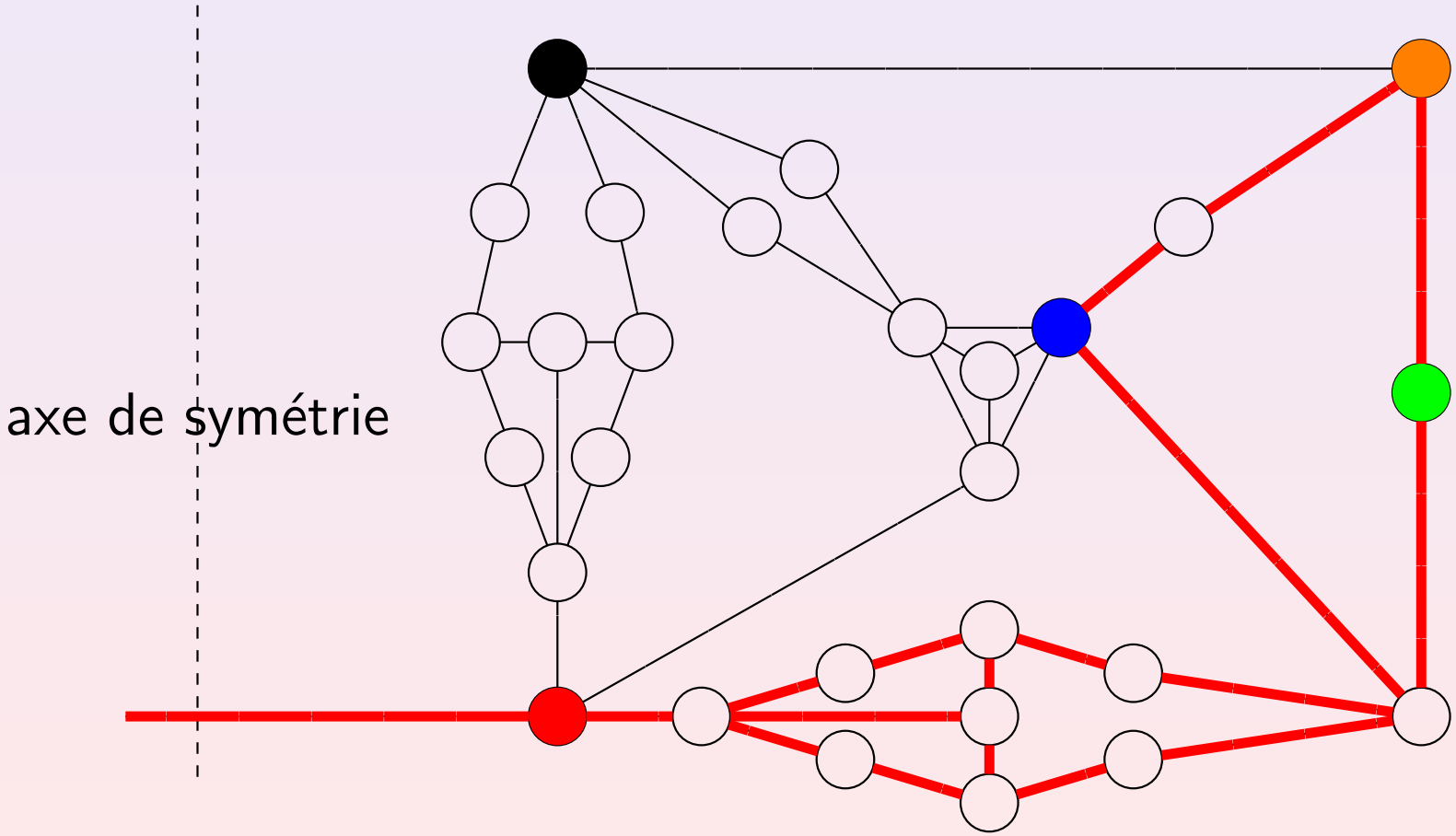
Soit G le graphe ci-dessous : $mcvs(G) > cvs(G) = 4$.



Connexe vs non connexe

Cas d'un fugitif visible

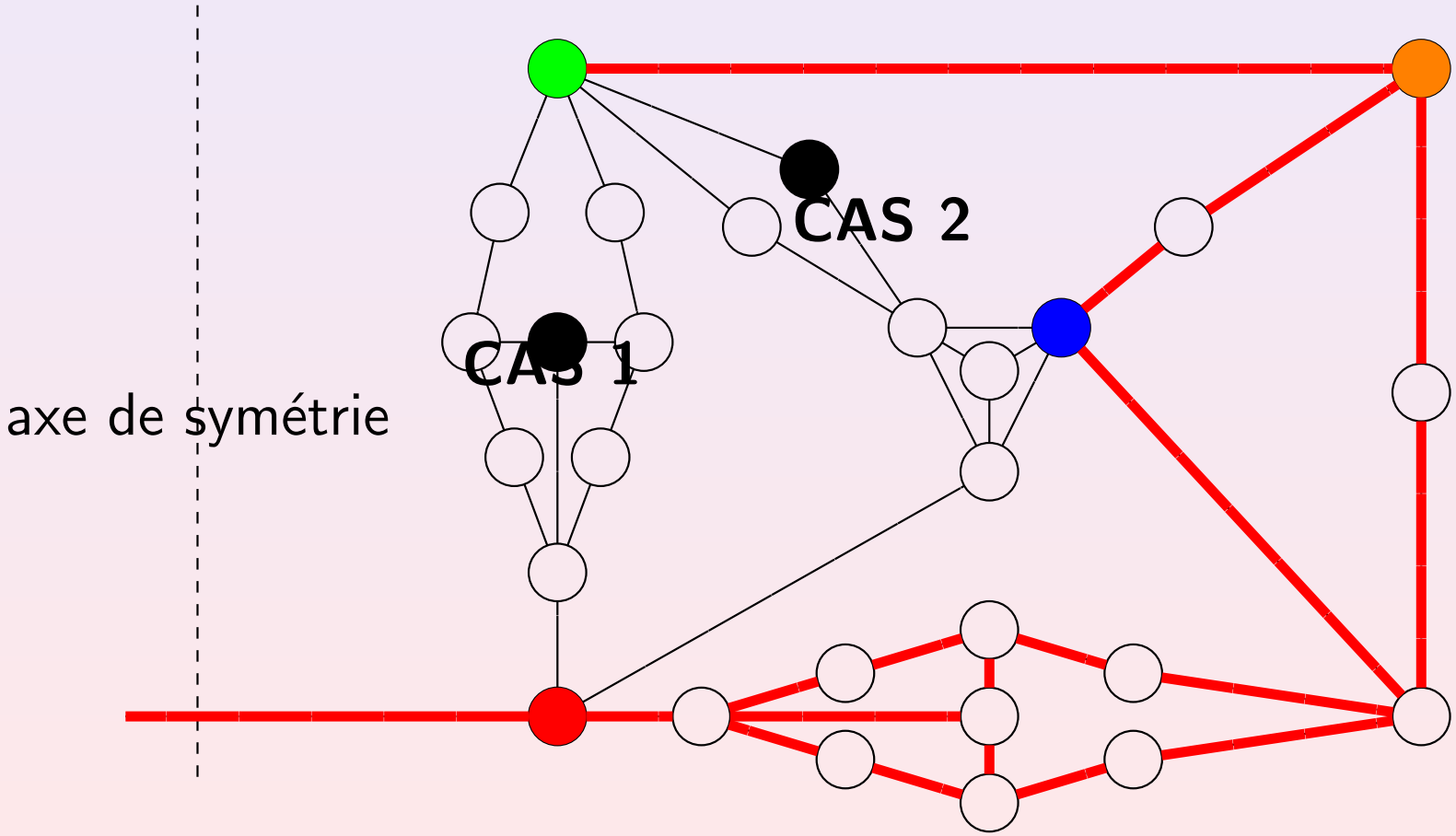
Soit G le graphe ci-dessous : $mcvs(G) > cvs(G) = 4$.



Connexe vs non connexe

Cas d'un fugitif visible

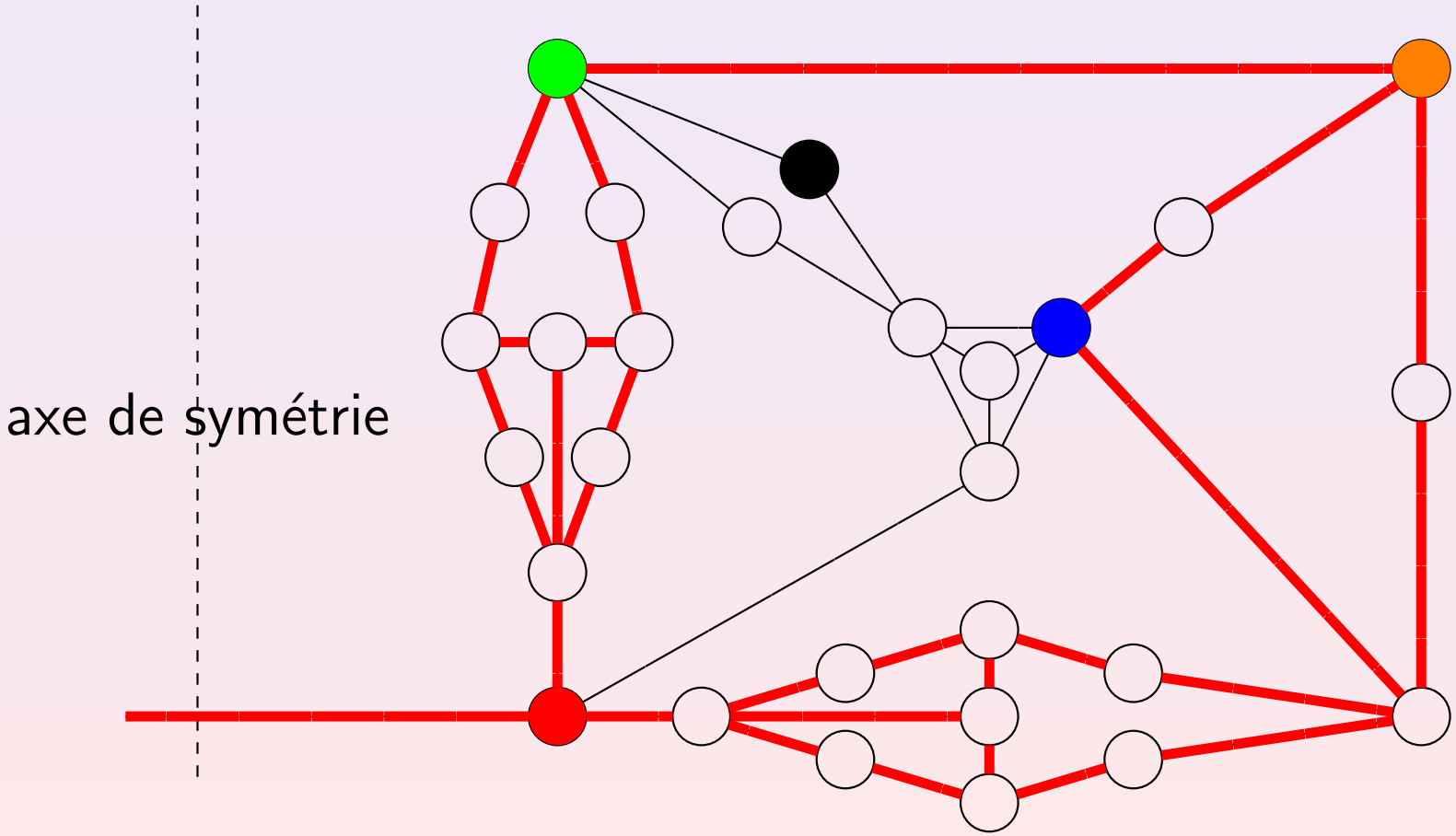
Soit G le graphe ci-dessous : $mcvs(G) > cvs(G) = 4$.



Connexe vs non connexe

Cas d'un fugitif visible

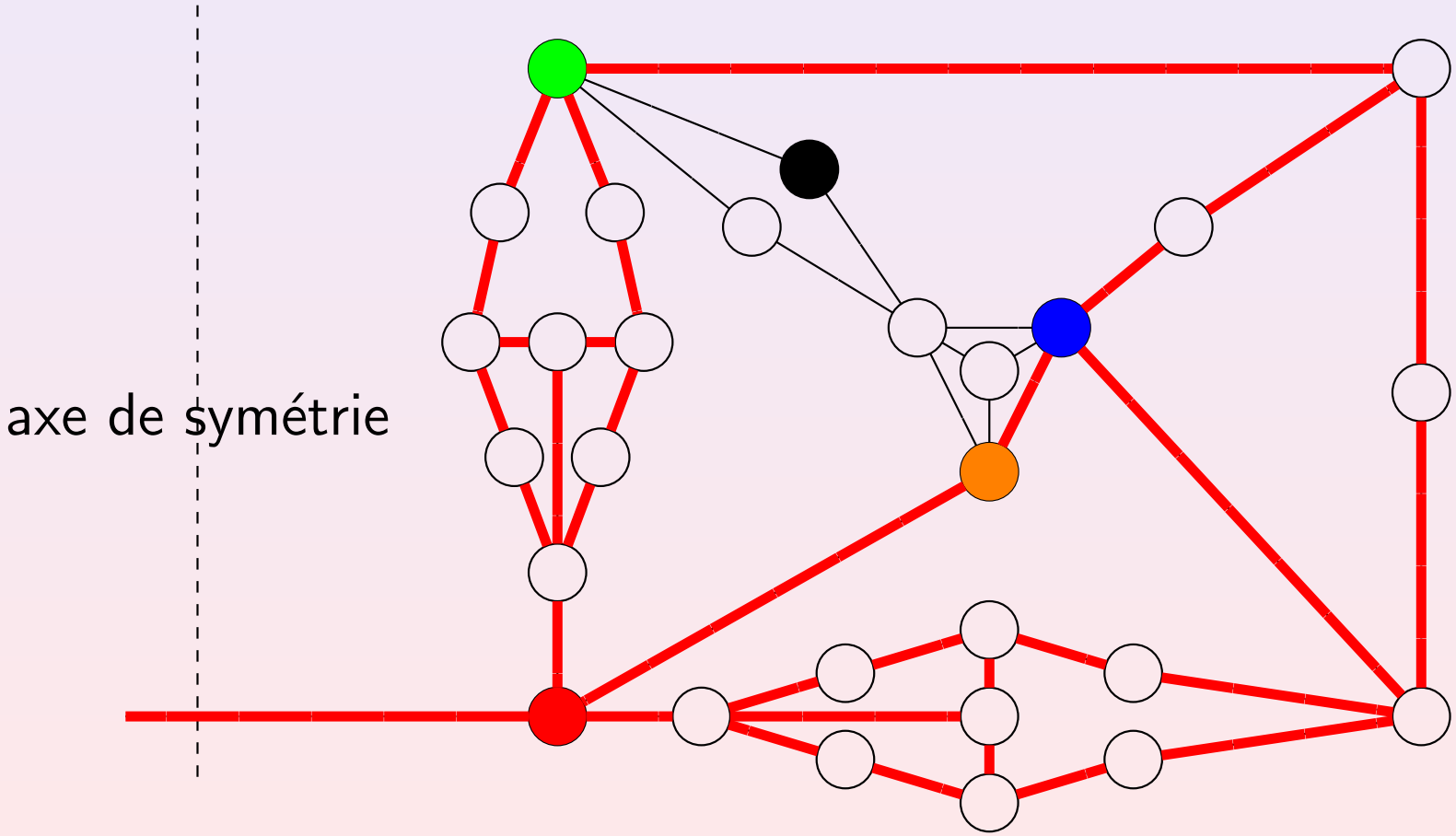
Soit G le graphe ci-dessous : $mcvs(G) > cvs(G) = 4$.



Connexe vs non connexe

Cas d'un fugitif visible

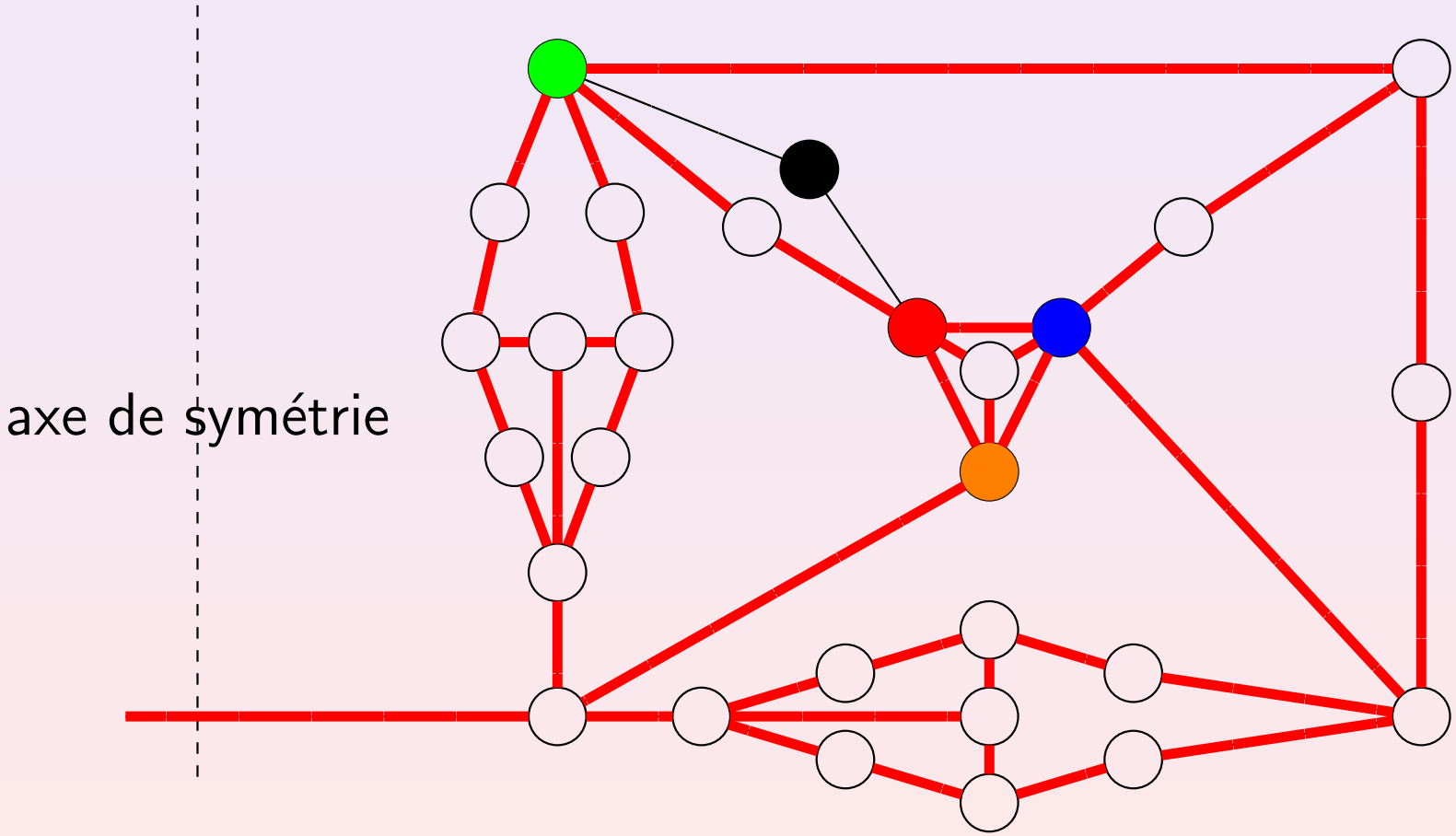
Soit G le graphe ci-dessous : $mcvs(G) > cvs(G) = 4$.



Connexe vs non connexe

Cas d'un fugitif visible

Soit G le graphe ci-dessous : $mcvs(G) > cvs(G) = 4$.



Conclusion et Perspectives

Le prix de la connexité

- nouvelle borne pour $\mathbf{cs}(G)/\mathbf{s}(G)$
- algorithme constructif

Problèmes ouverts

- Quelle est la borne optimale ?
  Dans les arbres : $\mathbf{cs}(T)/\mathbf{s}(T) \leq 2$, borne optimale [Barrière et al.].
  Si le fugitif est visible : $\mathbf{cs}(G)/\mathbf{s}(G) \leq \log n$, borne optimale [Fraigniaud, Nisse]
- Le problème de calculer $\mathbf{cs}(G)$ est-il NP-complet ?
  problème NP-dur.