Stratégies d'encerclement non déterministes dans les graphes

Fedor V.Fomin[1] Pierre Fraigniaud[2] Nicolas Nisse[2]

Department of Informatics, University of Bergen, Norway.

Lab. de Recherche en Informatique, Université Paris-Sud, Orsay, France.

AlgoTel 2006, jeudi 11 mai 2006

Encerclement dans les graphes

But

Dans un graphe simple, non-orienté,

- un fugitif omniscient et arbitrairement rapide ;
- une équipe d'agents ;

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

Motivation

- sécurité dans les réseaux de type internet ;
- notion en étroite relation avec :
  la largeur arborescente et la largeur linéaire.

Encerclement dans les graphes

But

Dans un graphe simple, non-orienté,

- un fugitif omniscient et arbitrairement rapide ;
- une équipe d'agents ;

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

Motivation

- sécurité dans les réseaux de type internet ;
- notion en étroite relation avec :
  la largeur arborescente et la largeur linéaire.

Stratégies d' Encerclement, Parson. [GTC,1978]

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,1978]

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,1978]

Séquence de deux opérations élémentaires,. . .

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

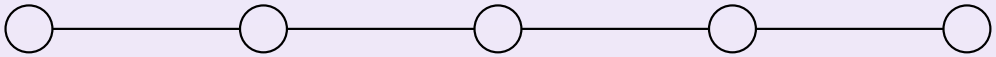
. . . qui doit aboutir à la capture du fugitif

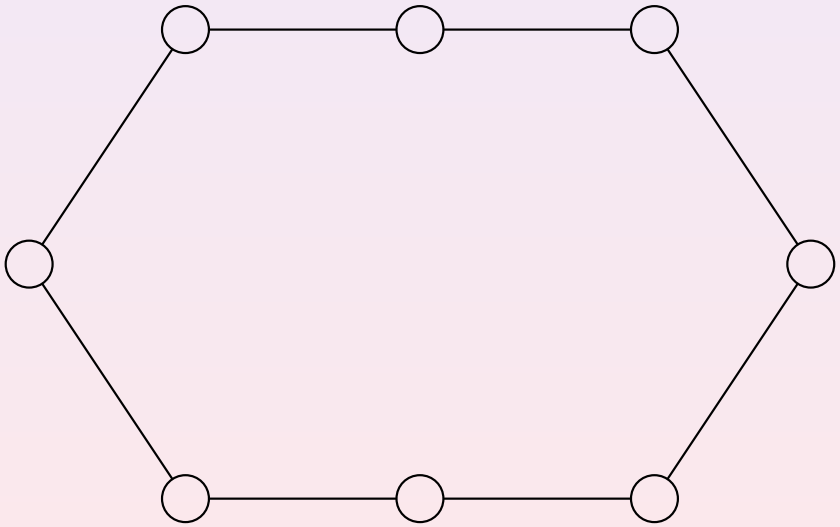
Le fugitif est capturé lorsqu'il occupe (ou croise) un sommet occupé par un agent. Une arête est dite nettoyée, lorsque ses extrémités sont occupées par des agents.

Il faut minimiser le nombre d'agents.

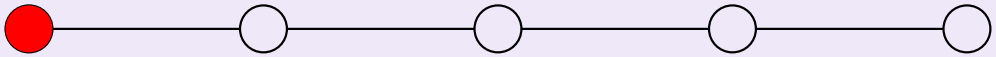
Soit $s(G)$ le nombre minimum d'agents nécessaires pour capturer un fugitif invisible dans le graphe G.

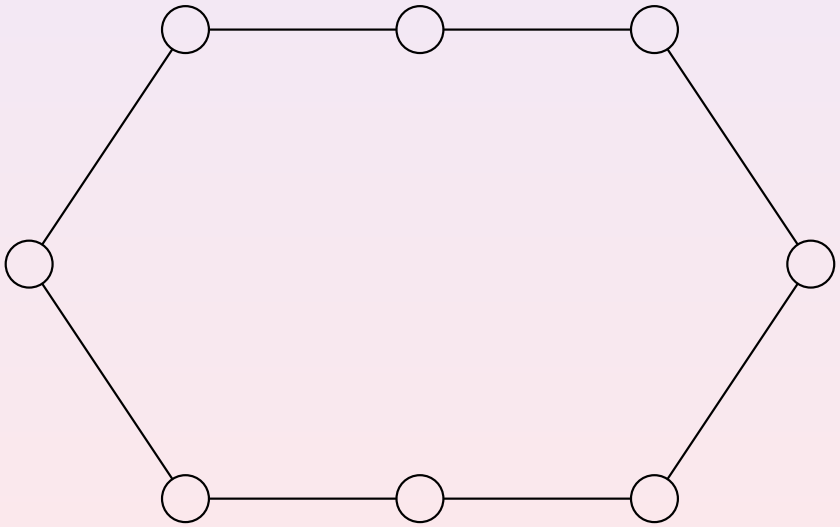
Exemples simples : le chemin et l'anneau



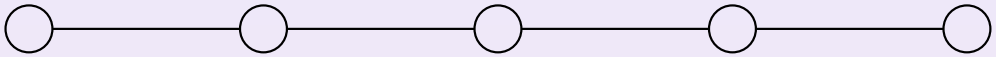


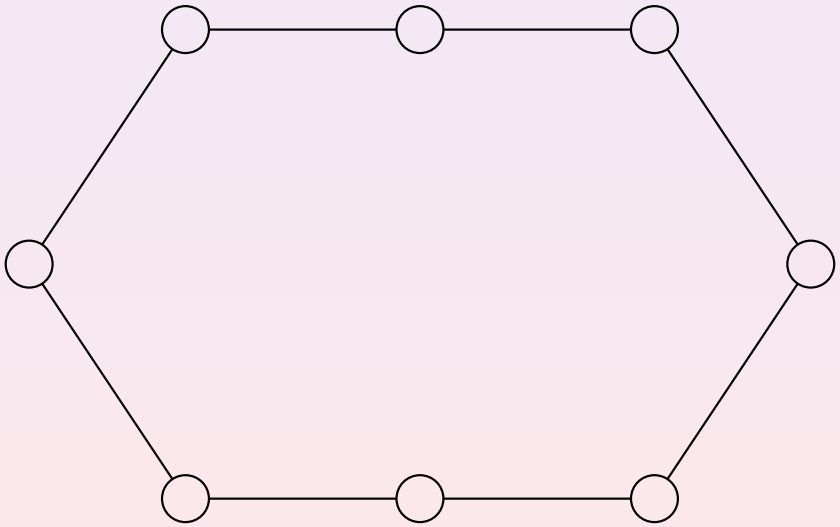
Exemples simples : le chemin et l'anneau



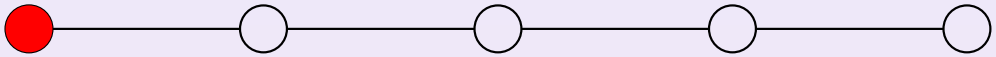


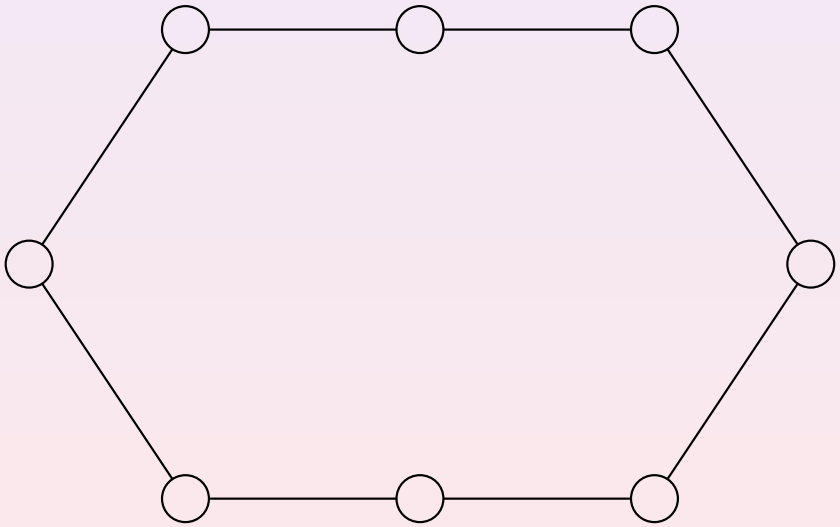
Exemples simples : le chemin et l'anneau



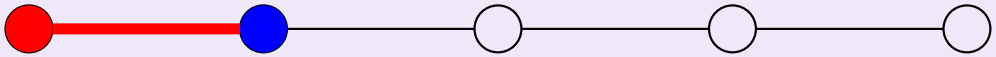


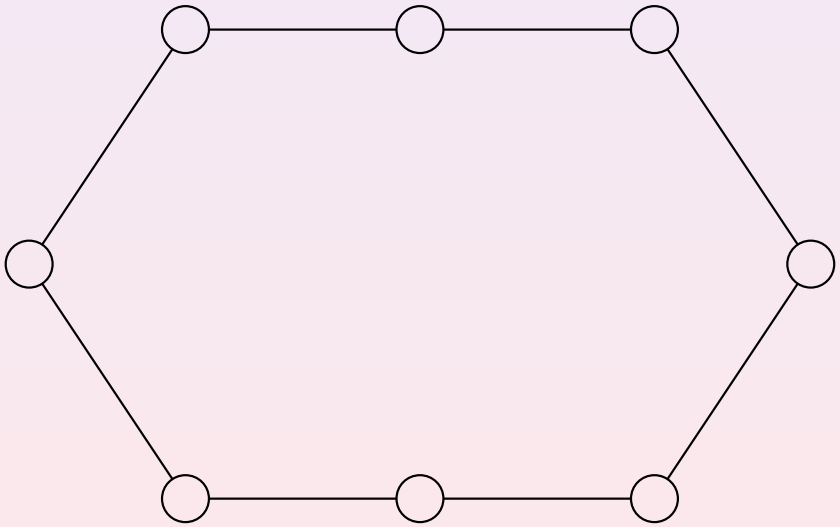
Exemples simples : le chemin et l'anneau



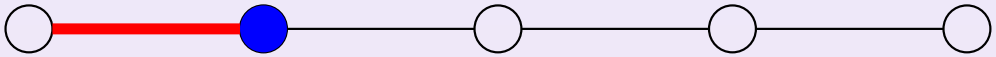


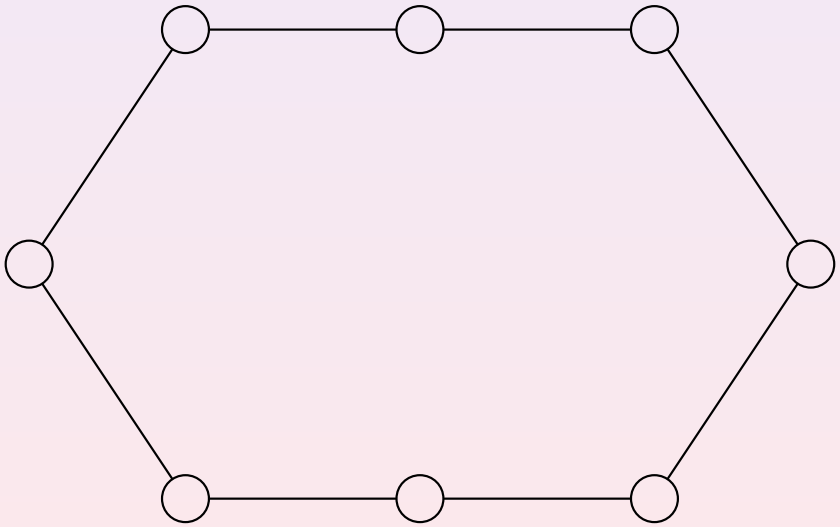
Exemples simples : le chemin et l'anneau



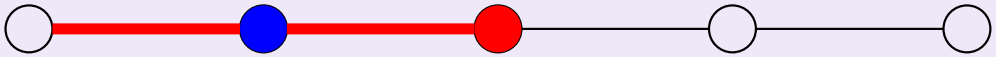


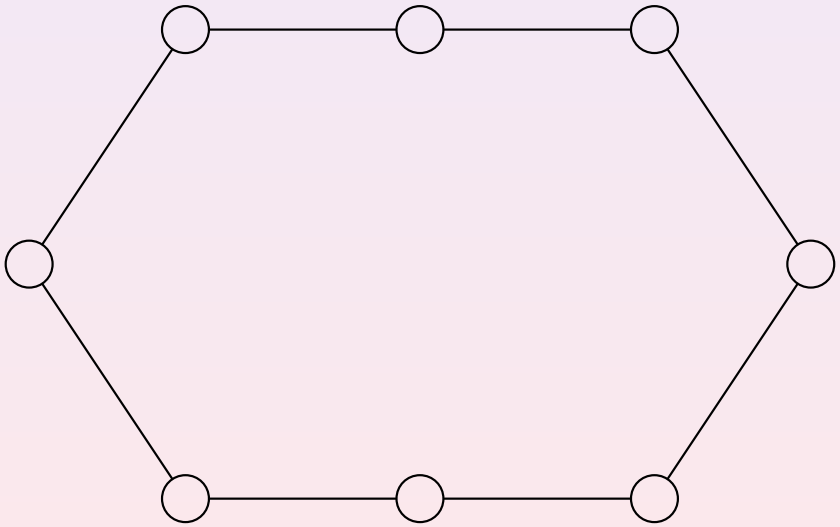
Exemples simples : le chemin et l'anneau



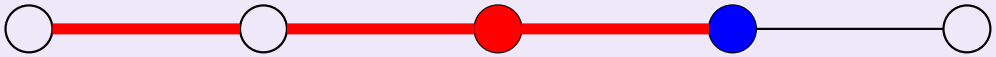


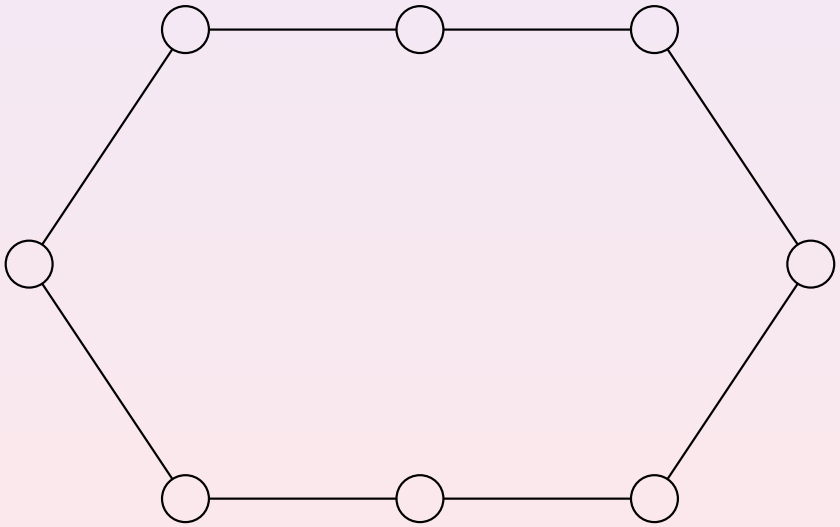
Exemples simples : le chemin et l'anneau



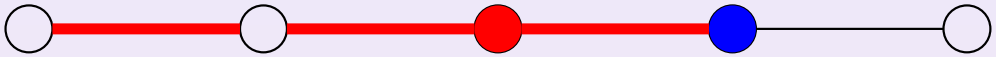


Exemples simples : le chemin et l'anneau

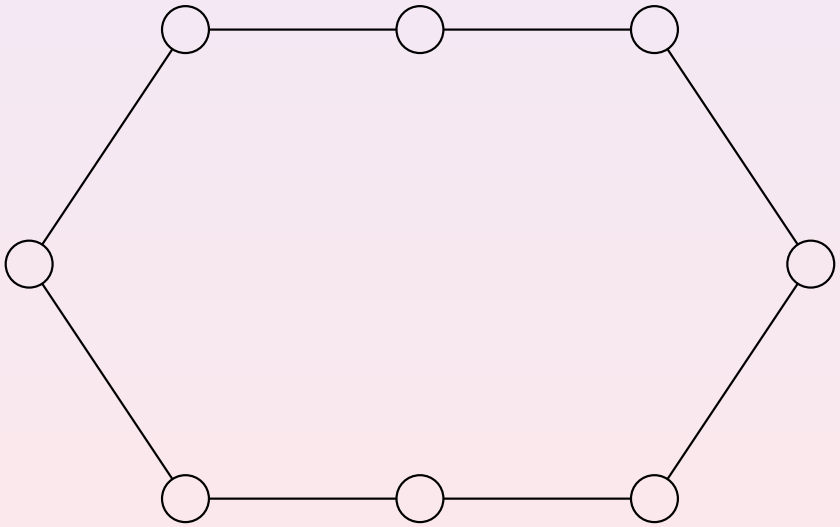




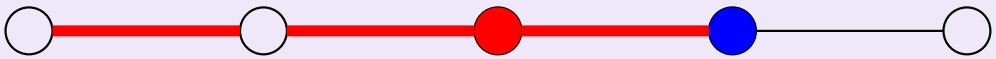
Exemples simples : le chemin et l'anneau



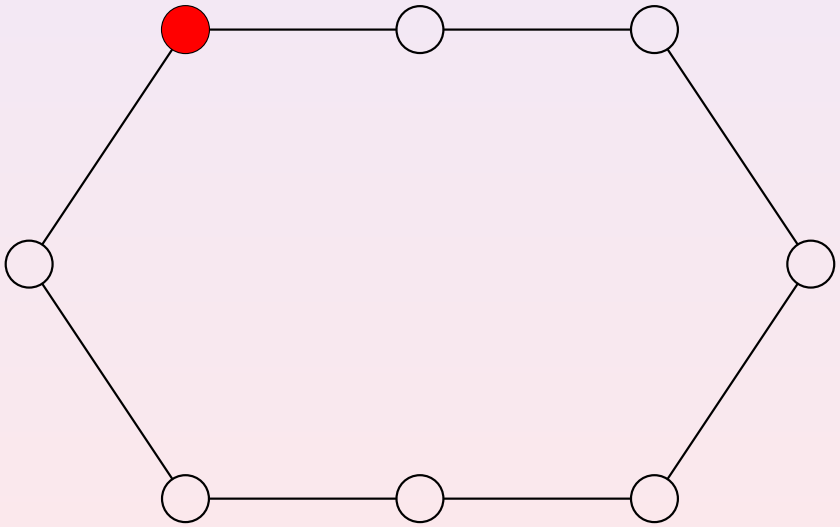
s(Chemin)=2



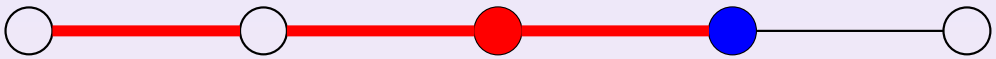
Exemples simples : le chemin et l'anneau



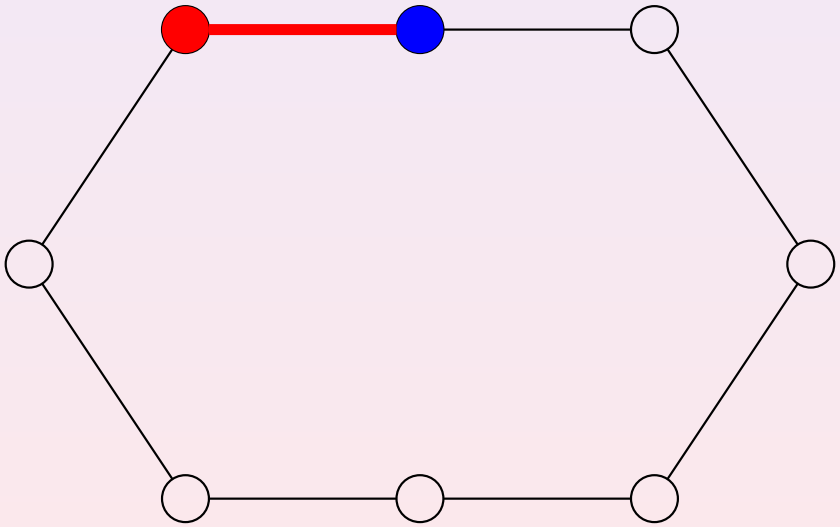
s(Chemin)=2



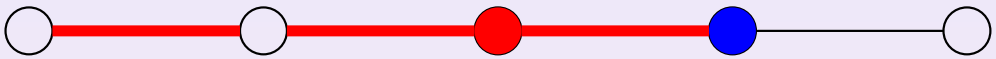
Exemples simples : le chemin et l'anneau



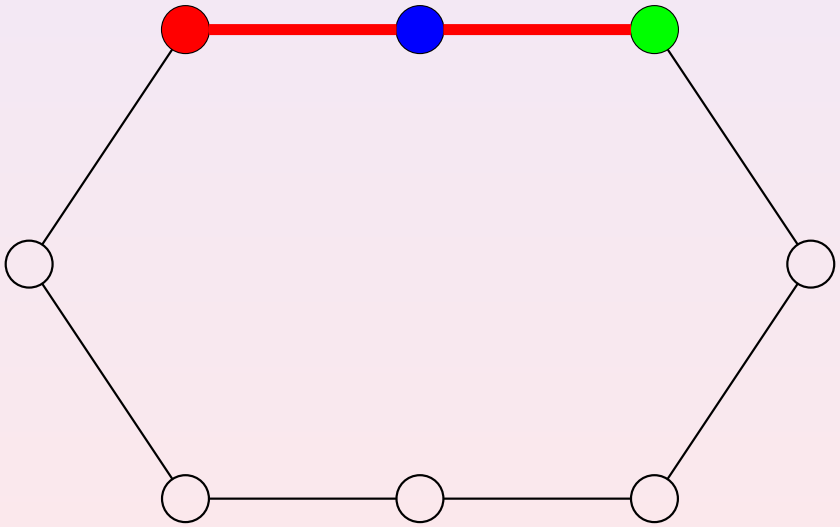
s(Chemin)=2



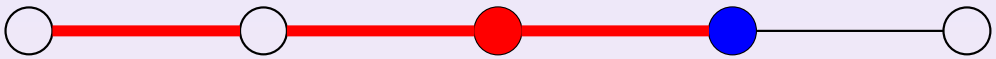
Exemples simples : le chemin et l'anneau



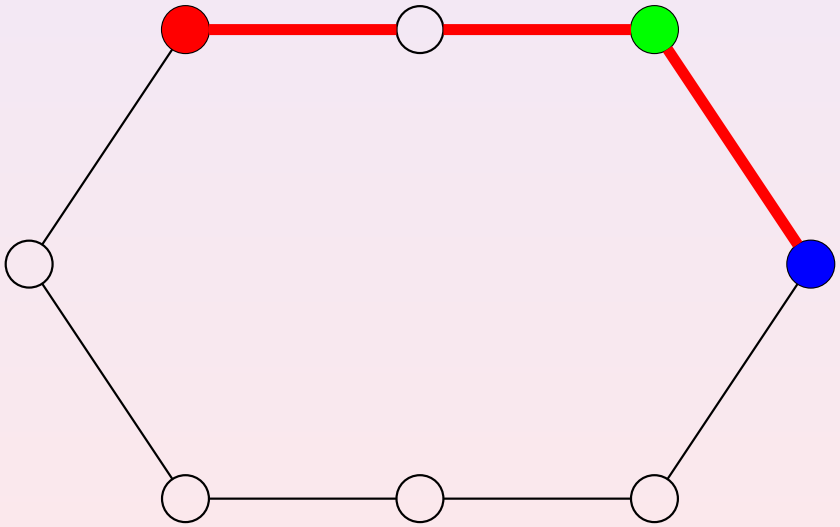
s(Chemin)=2



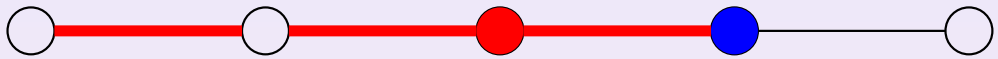
Exemples simples : le chemin et l'anneau



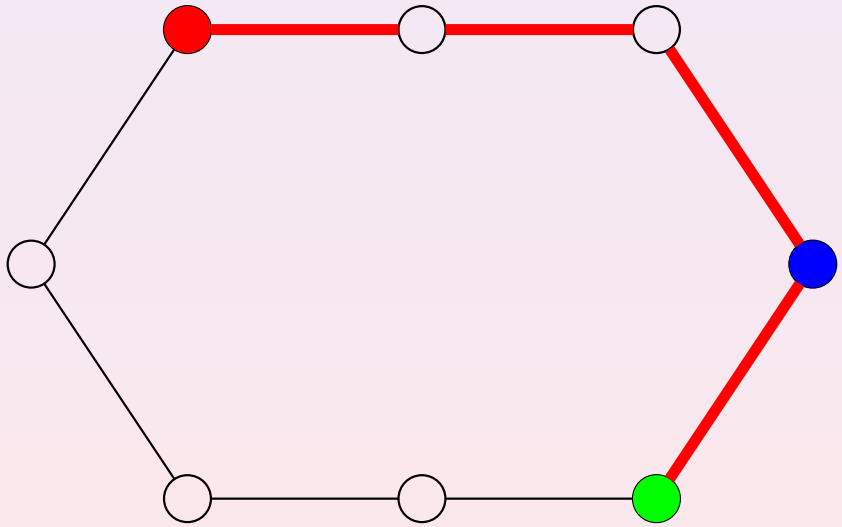
s(Chemin)=2



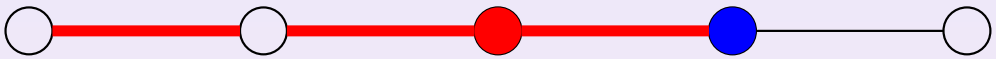
Exemples simples : le chemin et l'anneau



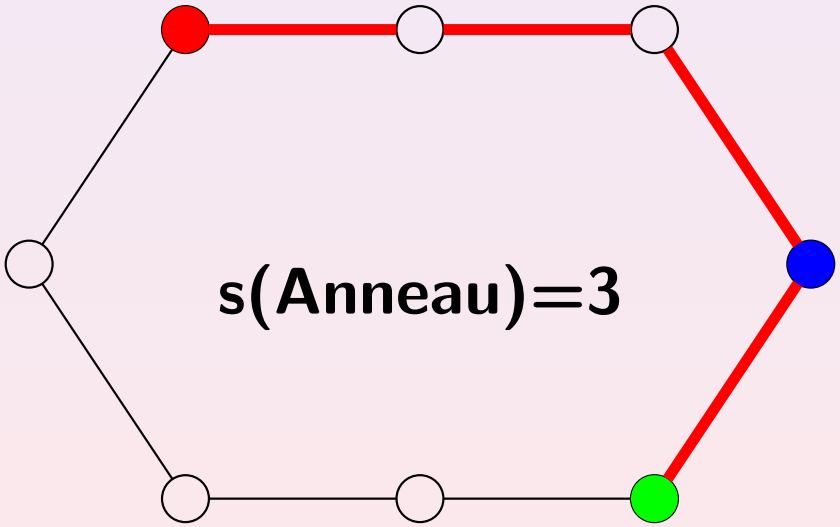
s(Chemin)=2



Exemples simples : le chemin et l'anneau

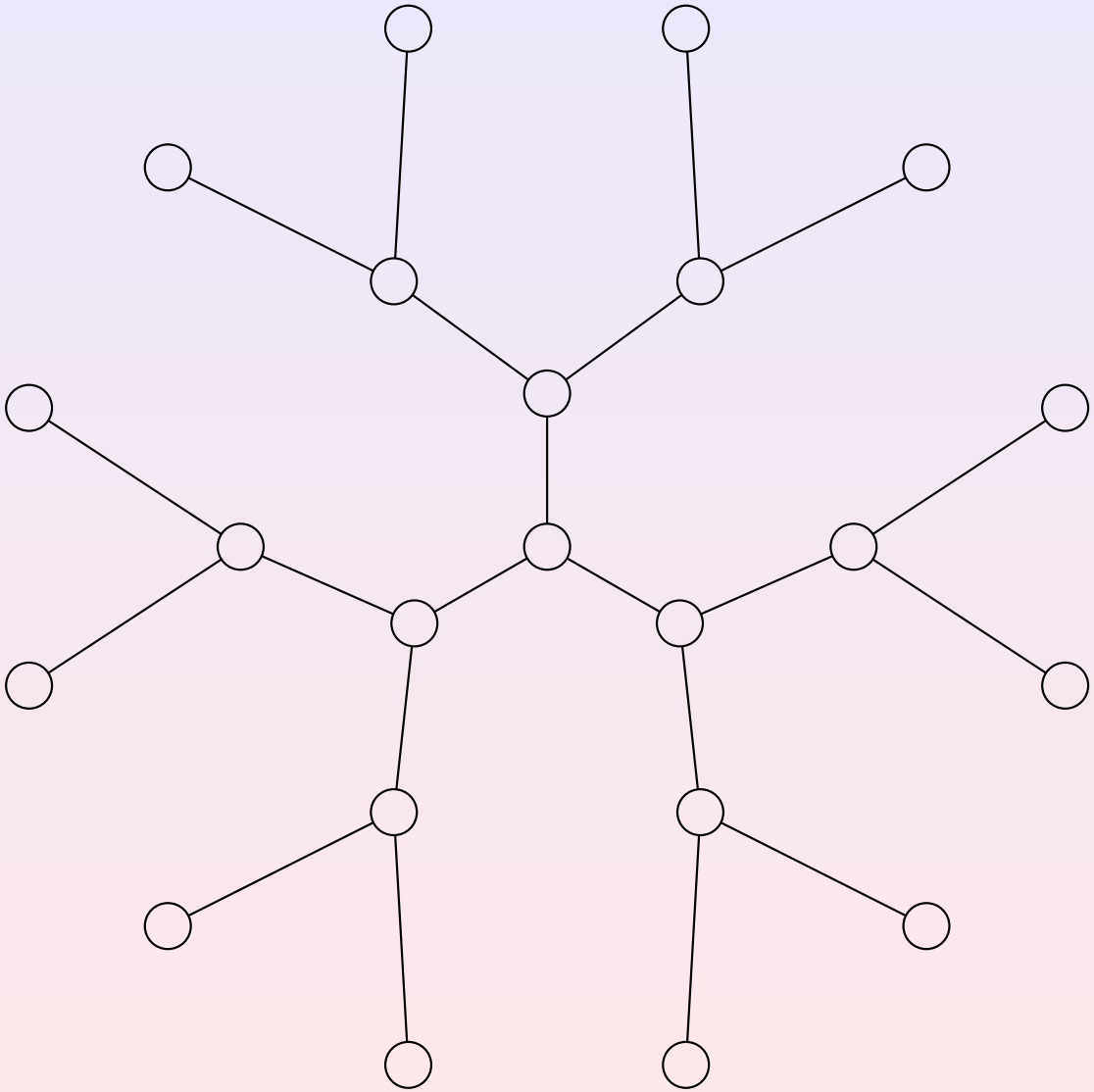


s(Chemin)=2

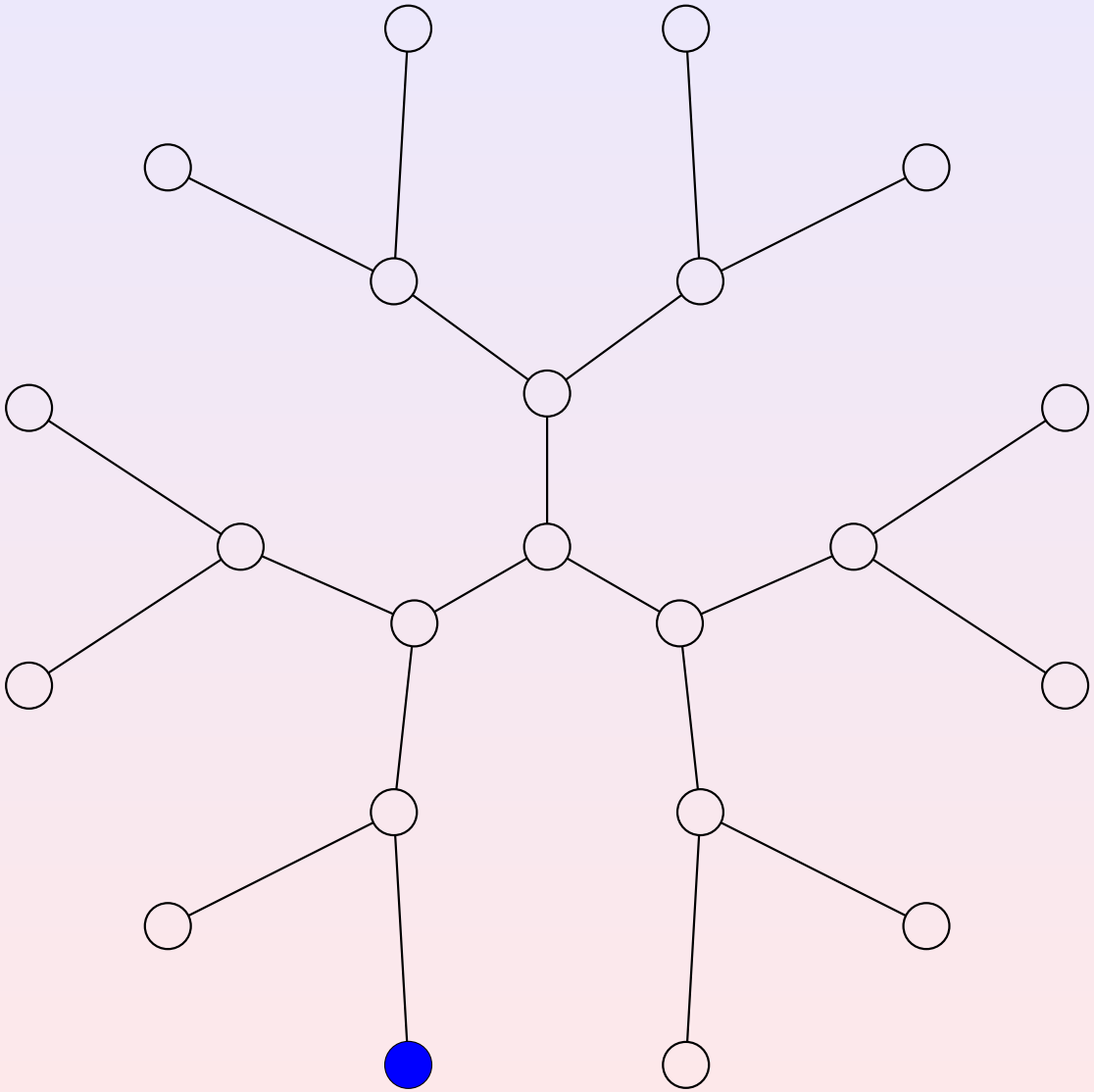


s(Anneau)=3

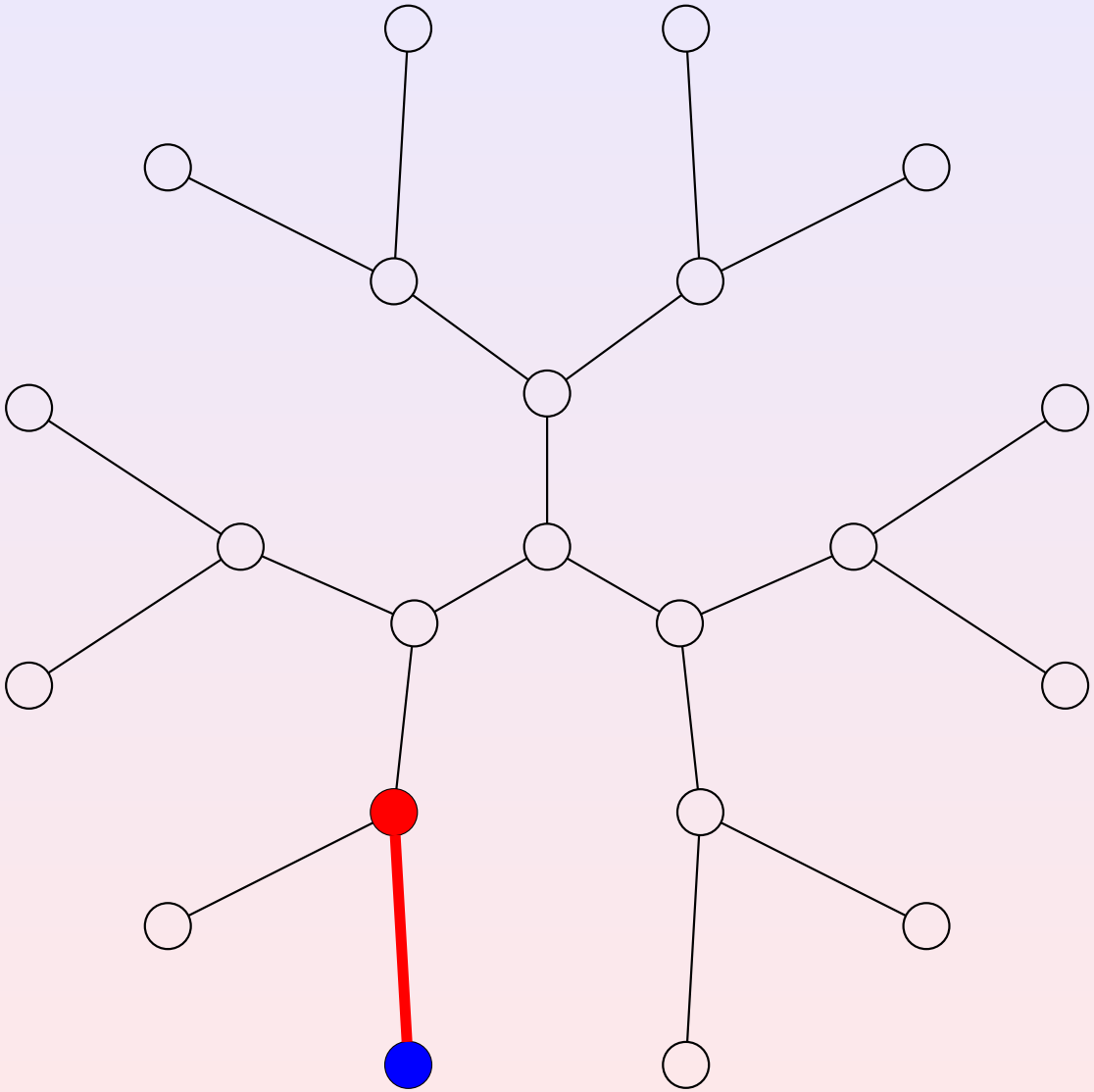
Encerclement dans un arbre



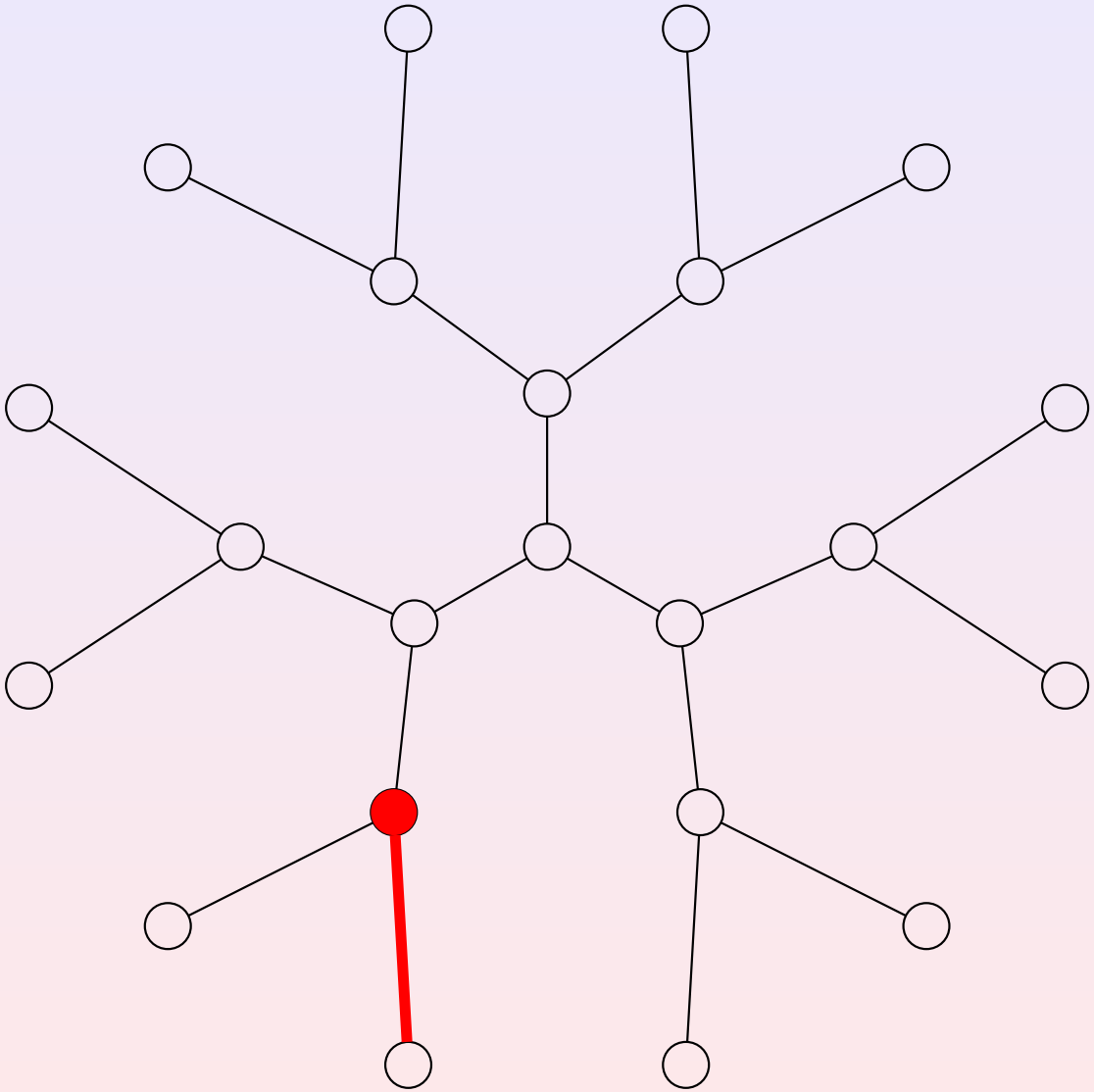
Encerclement dans un arbre



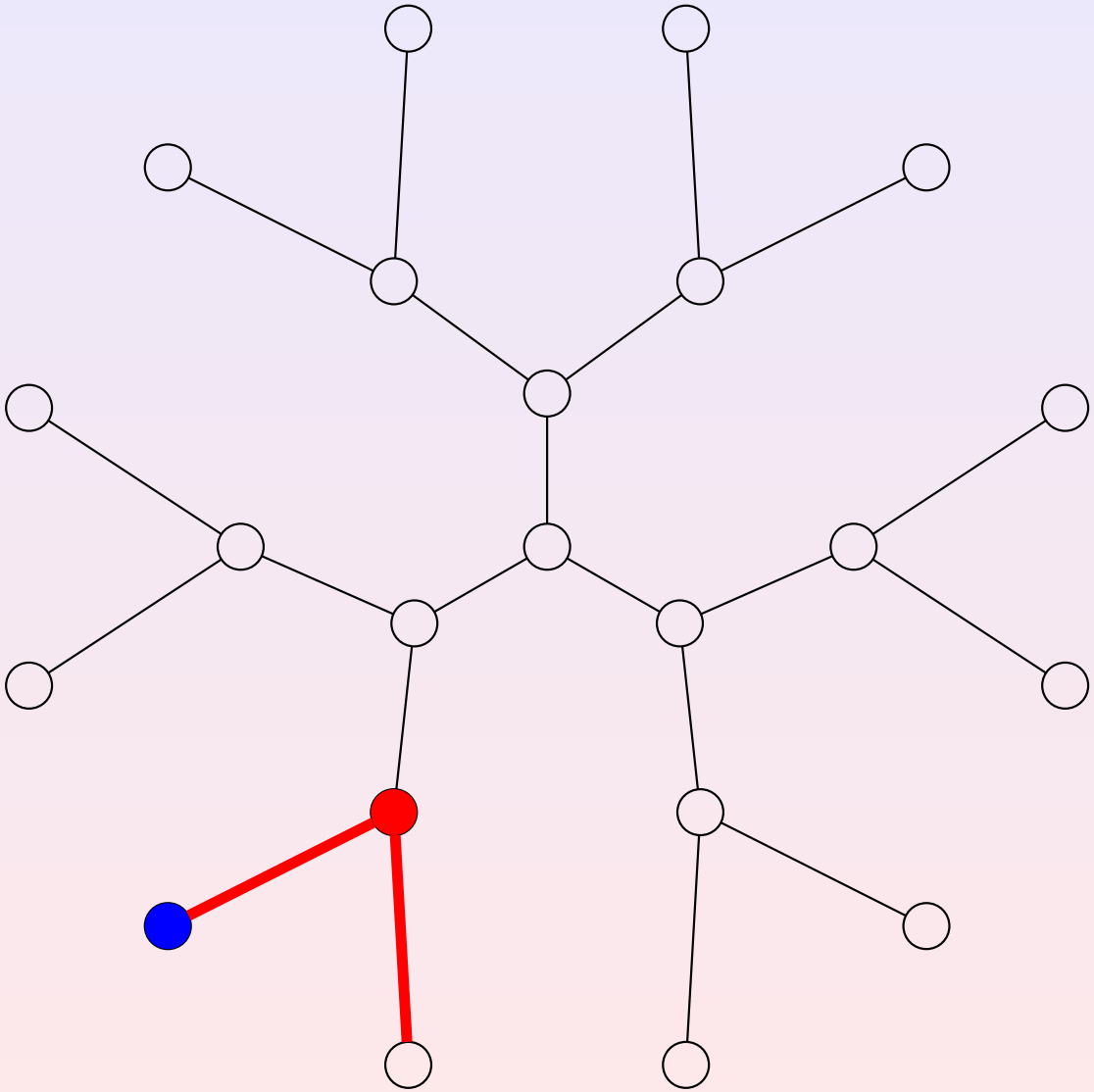
Encerclement dans un arbre



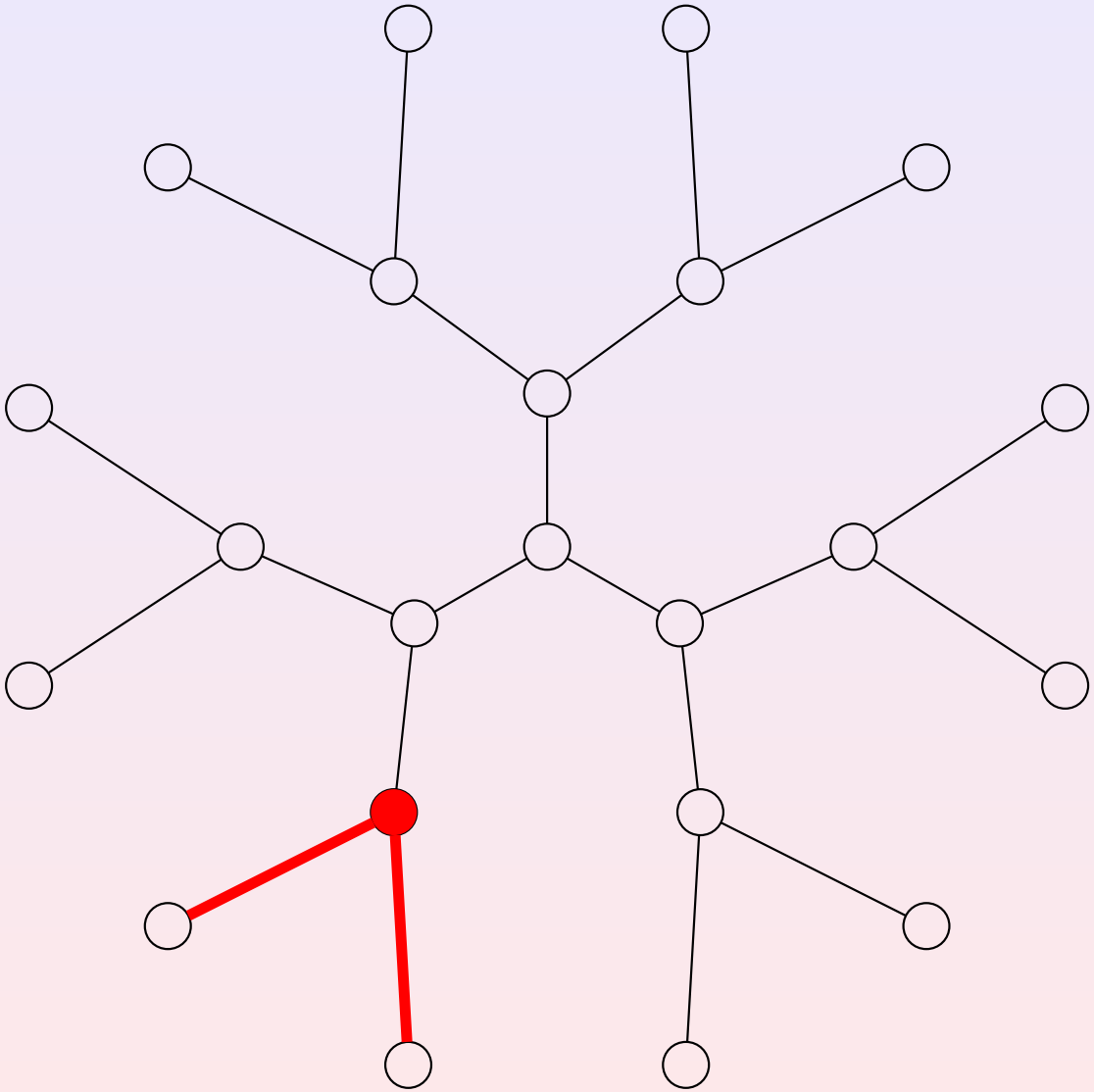
Encerclement dans un arbre



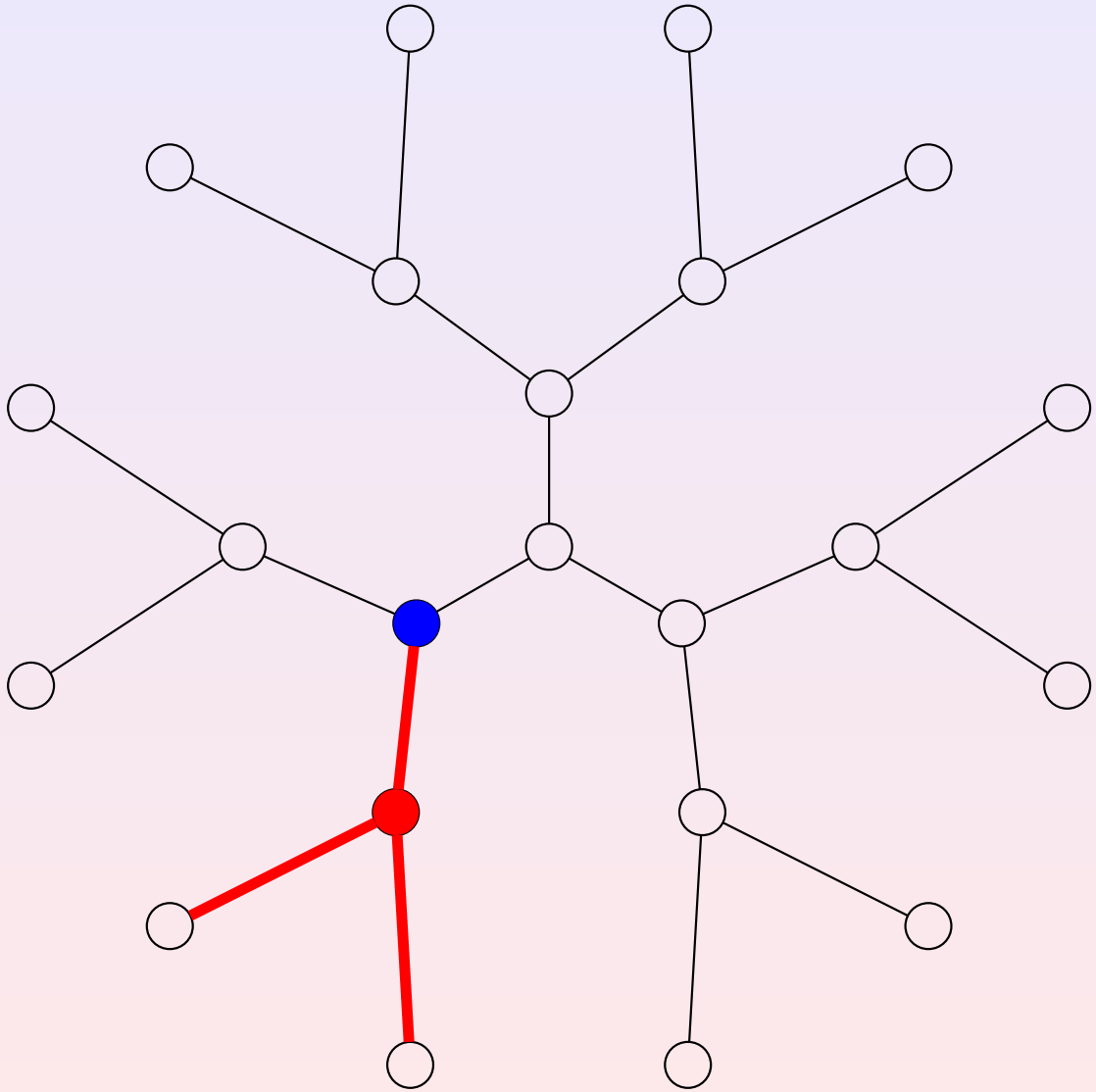
Encerclement dans un arbre



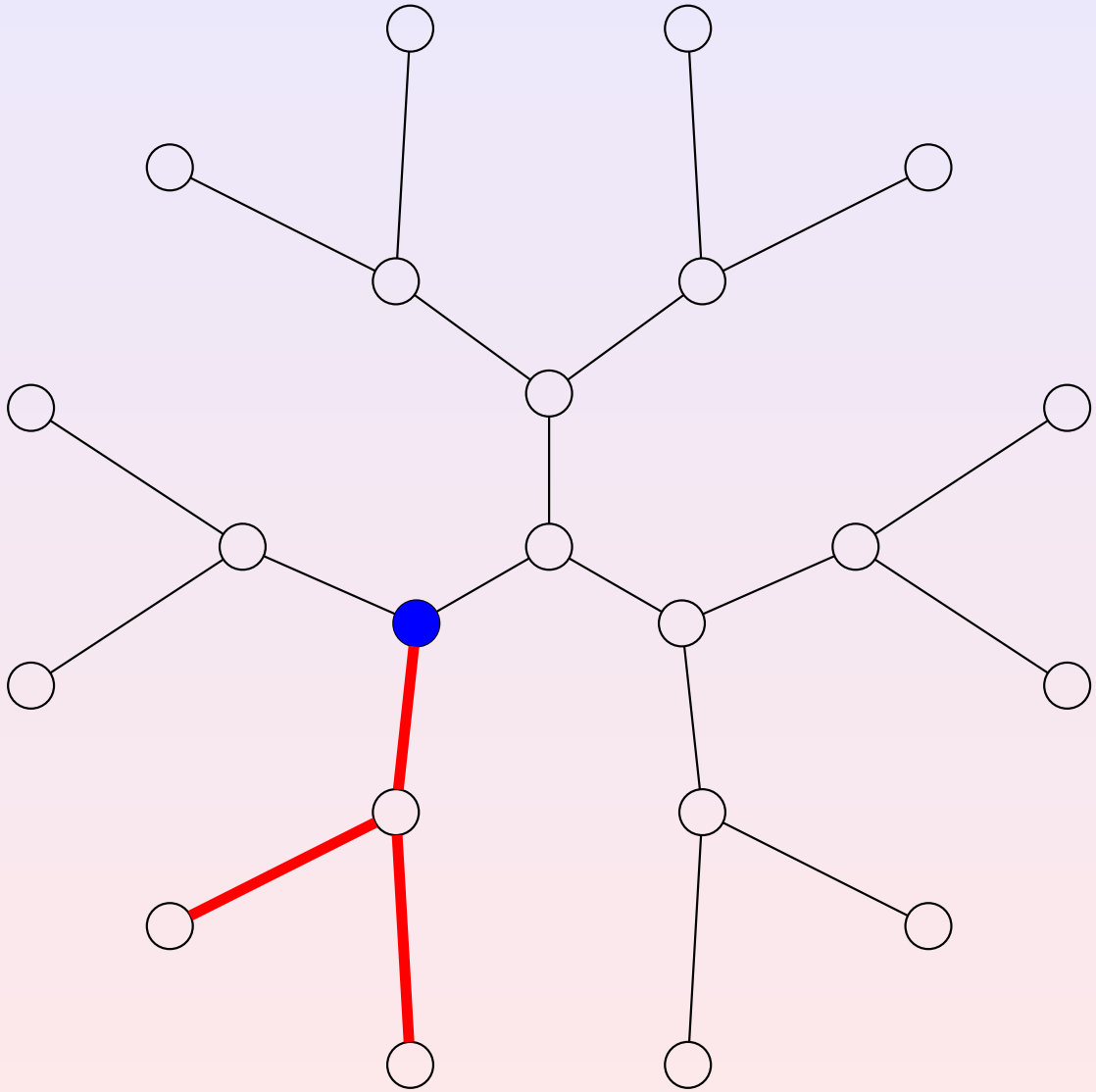
Encerclement dans un arbre



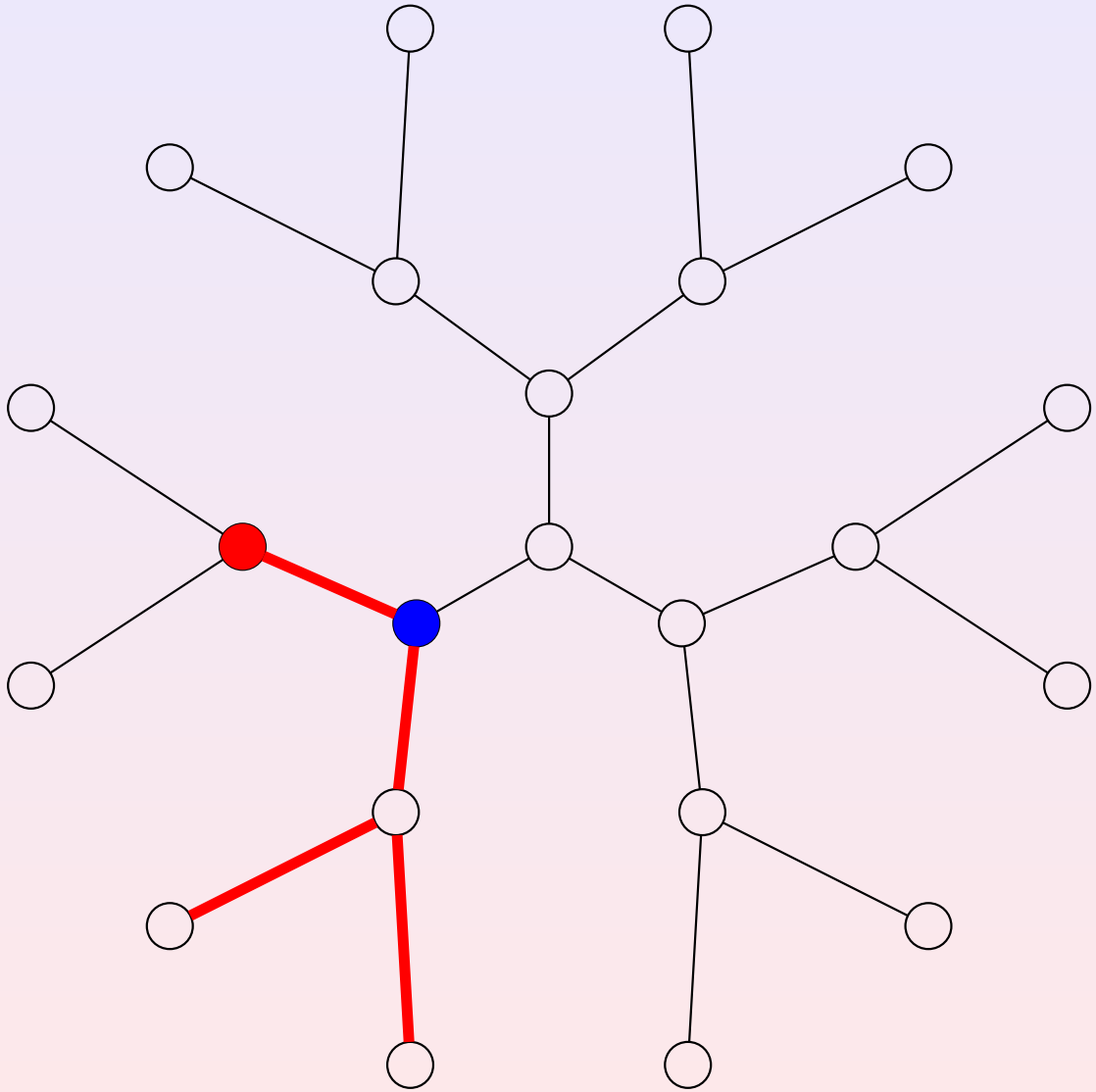
Encerclement dans un arbre



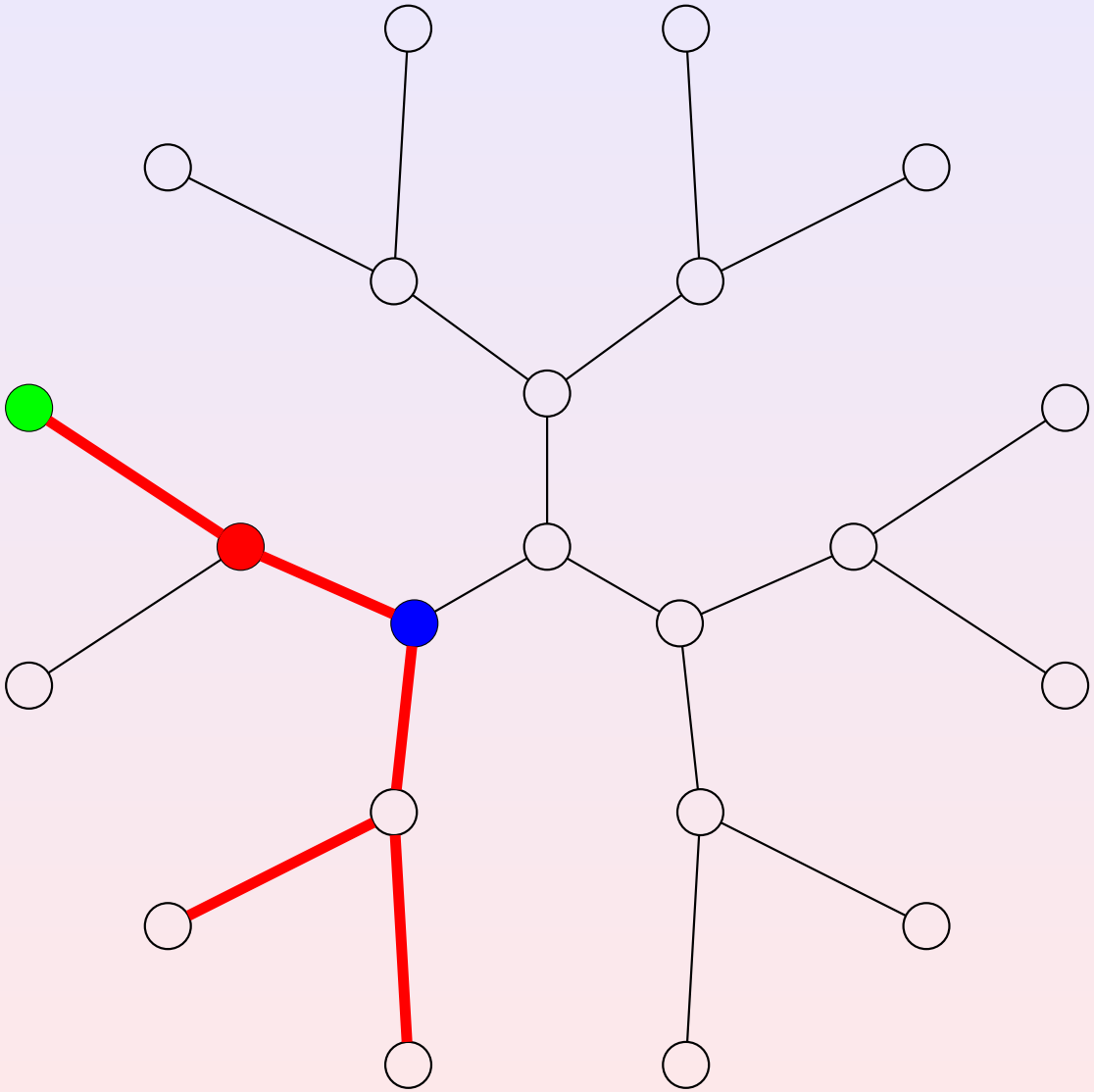
Encerclement dans un arbre



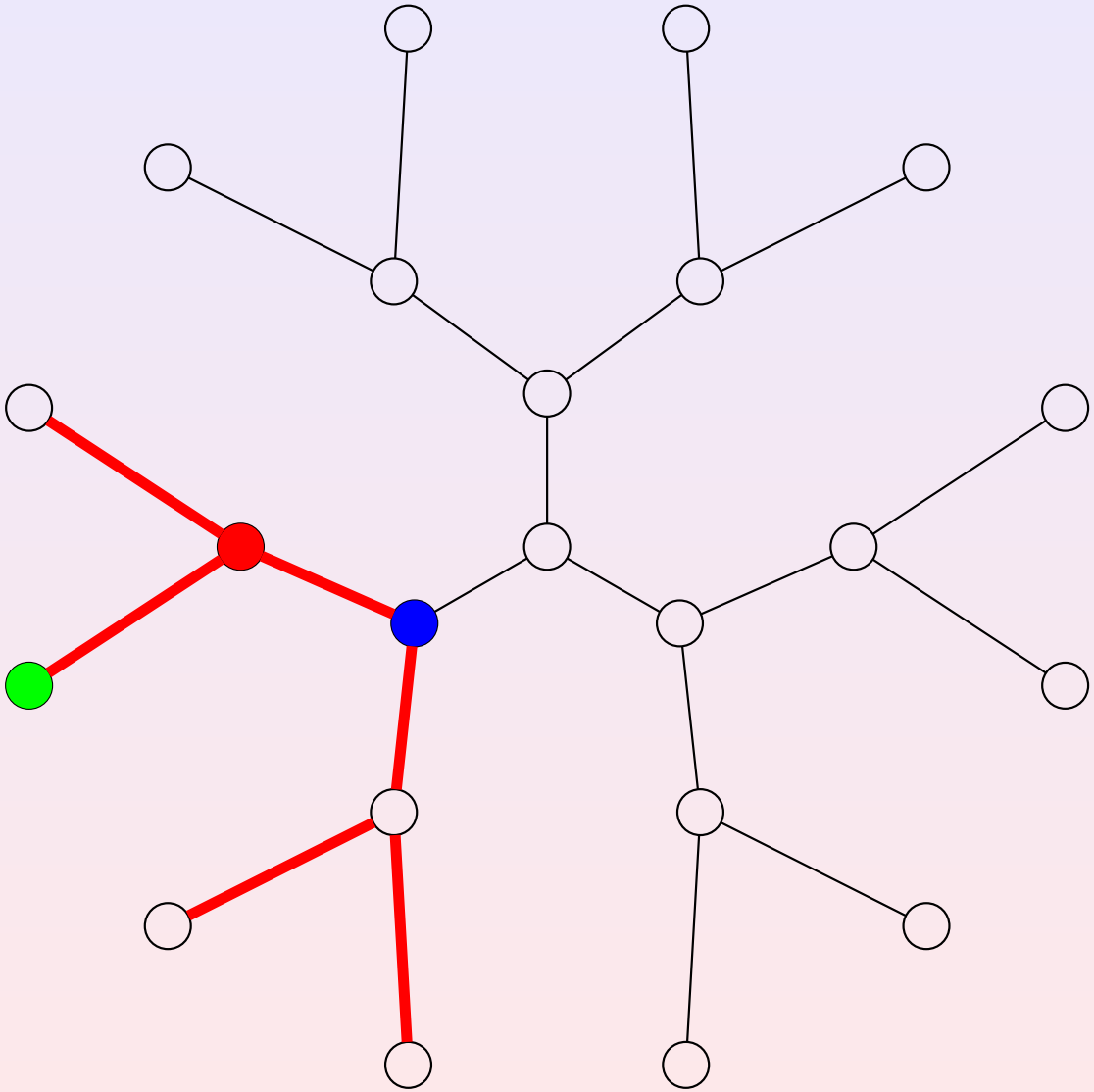
Encerclement dans un arbre



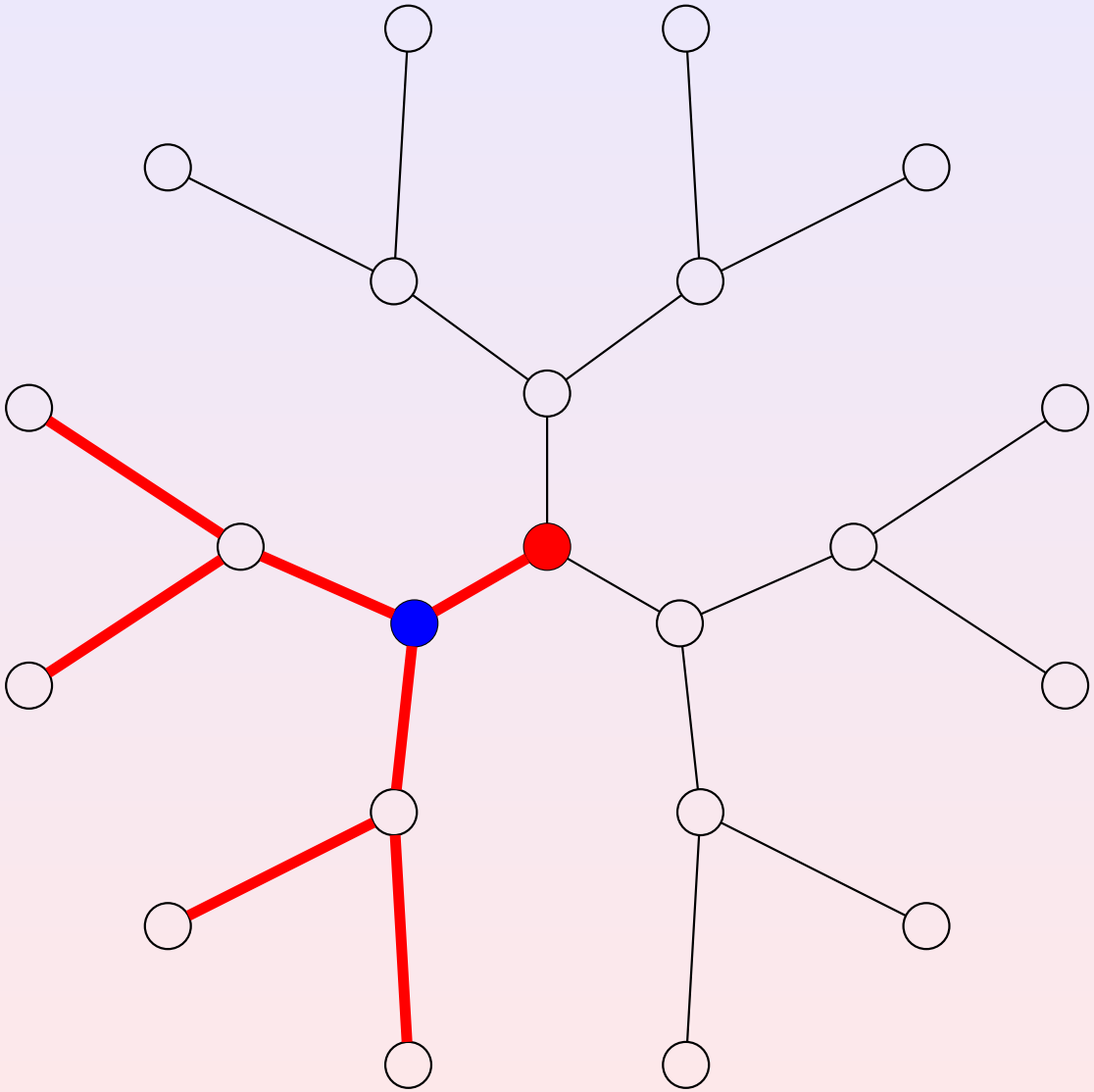
Encerclement dans un arbre



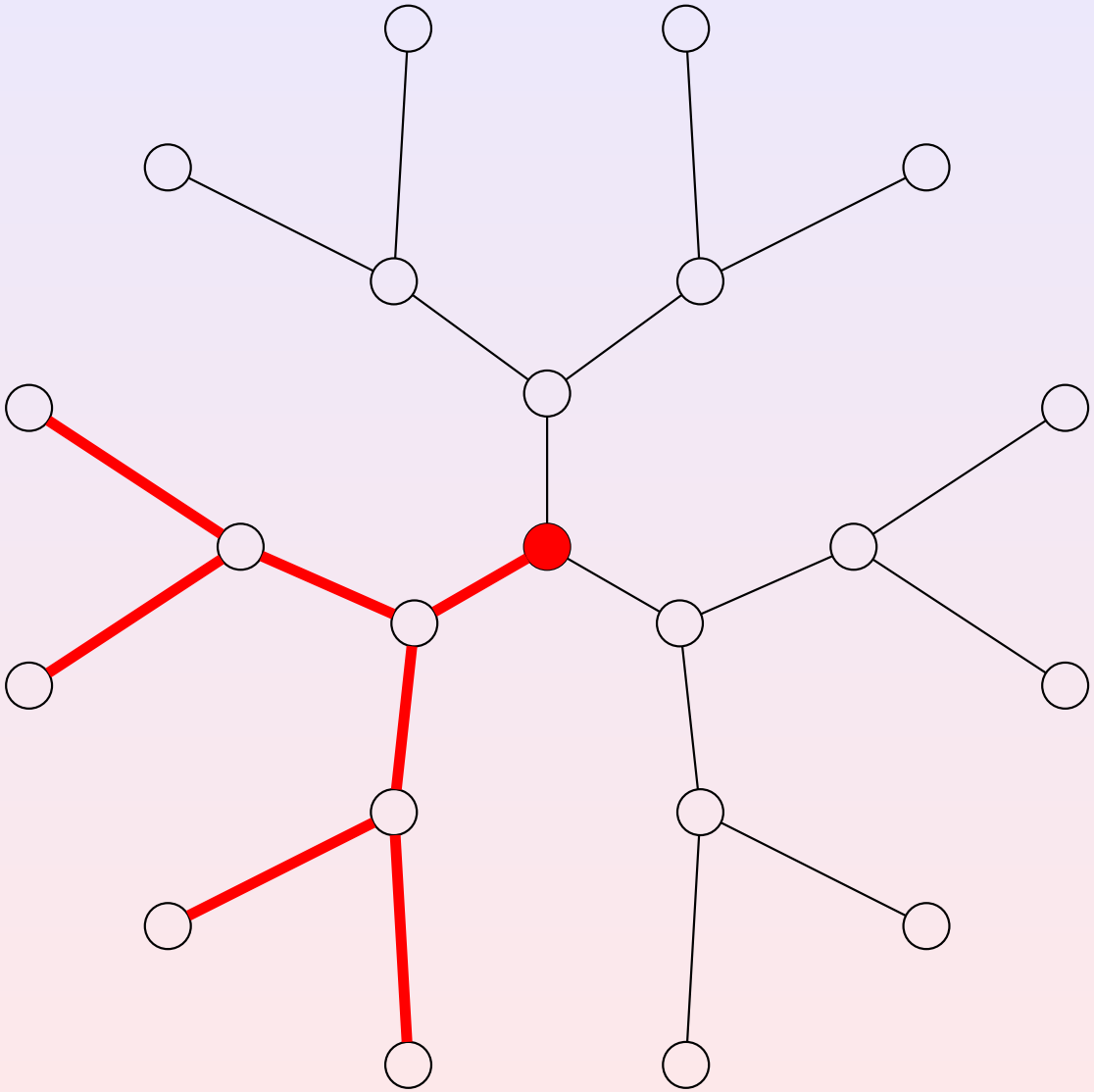
Encerclement dans un arbre



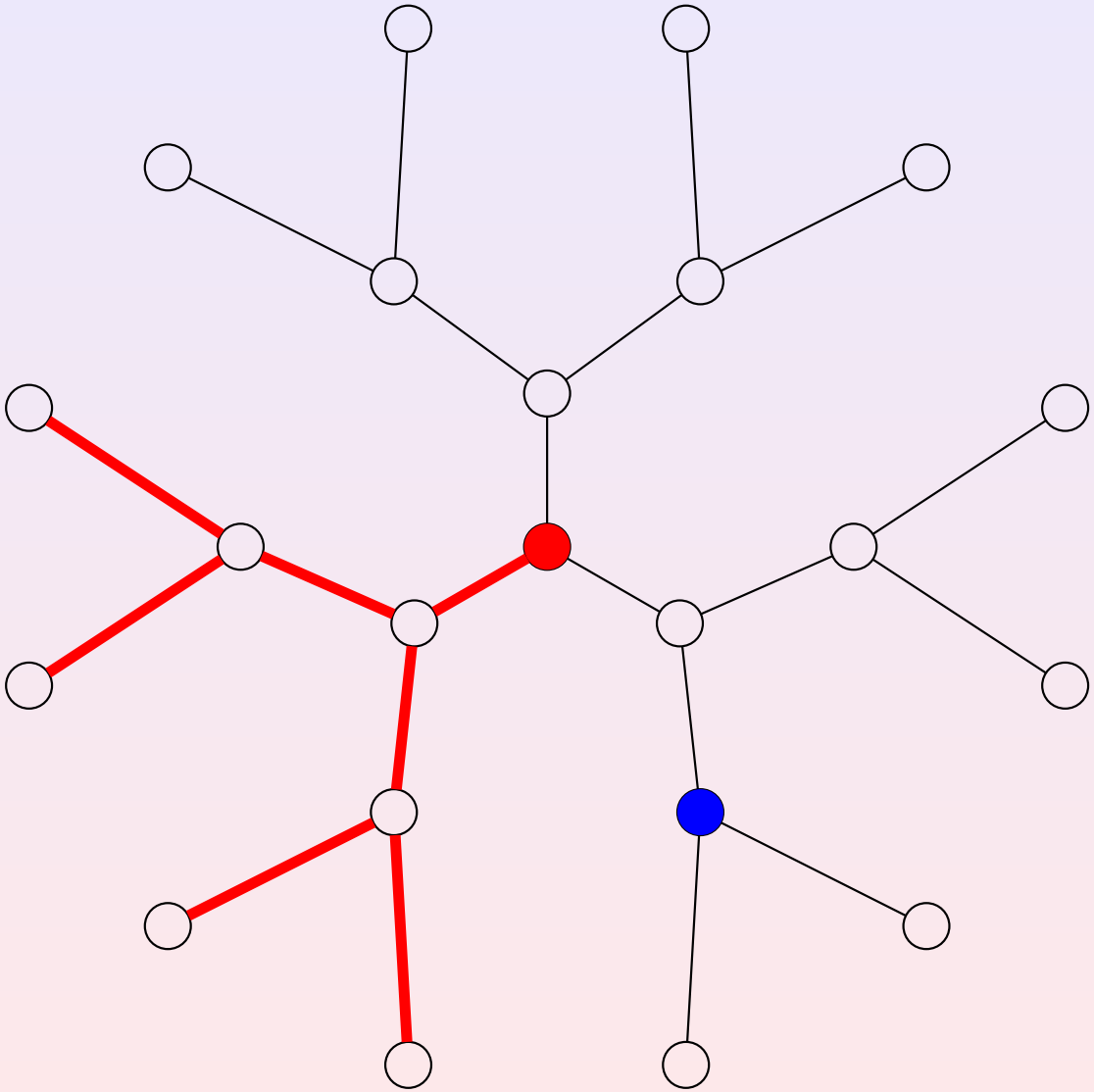
Encerclement dans un arbre



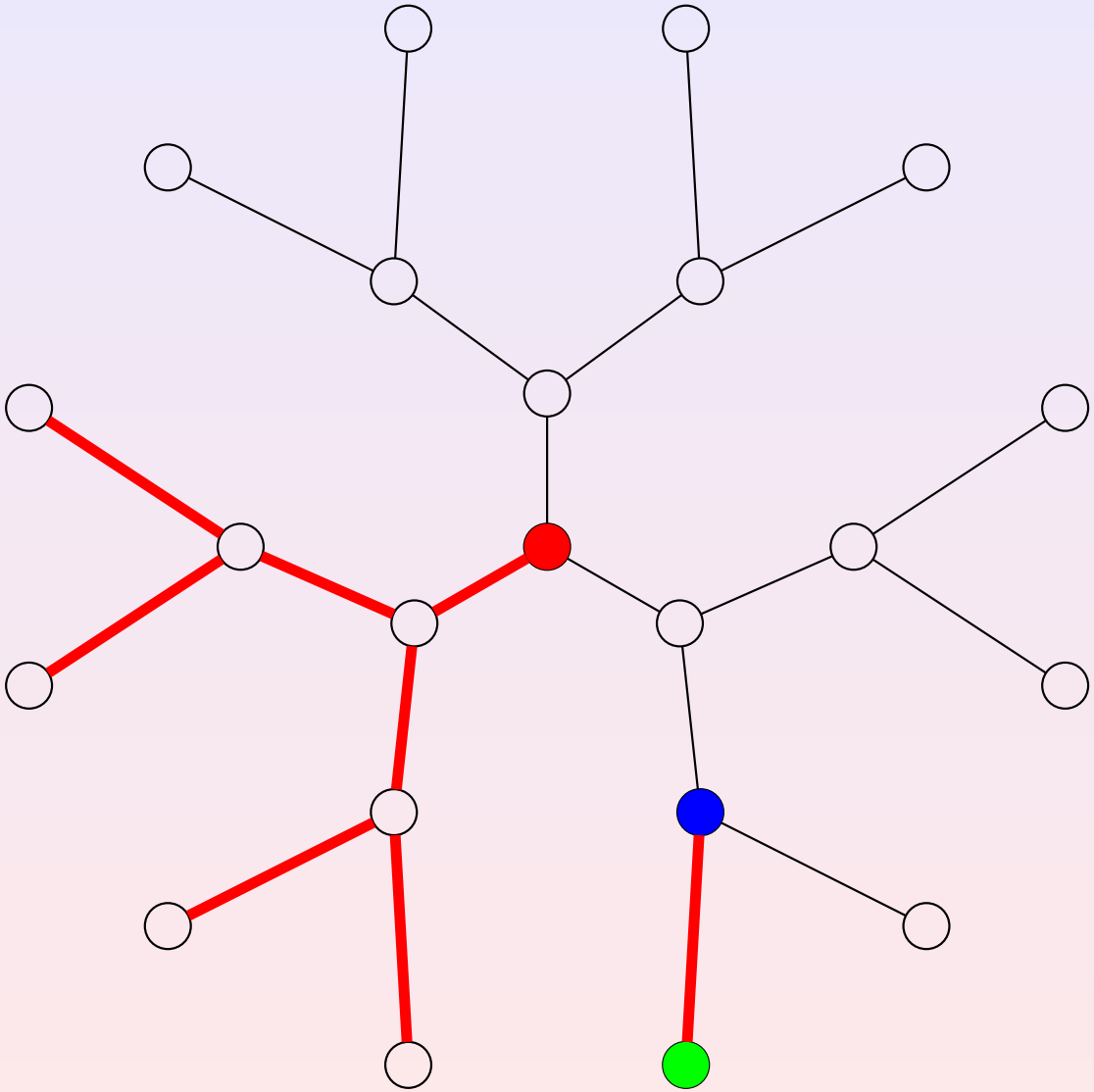
Encerclement dans un arbre



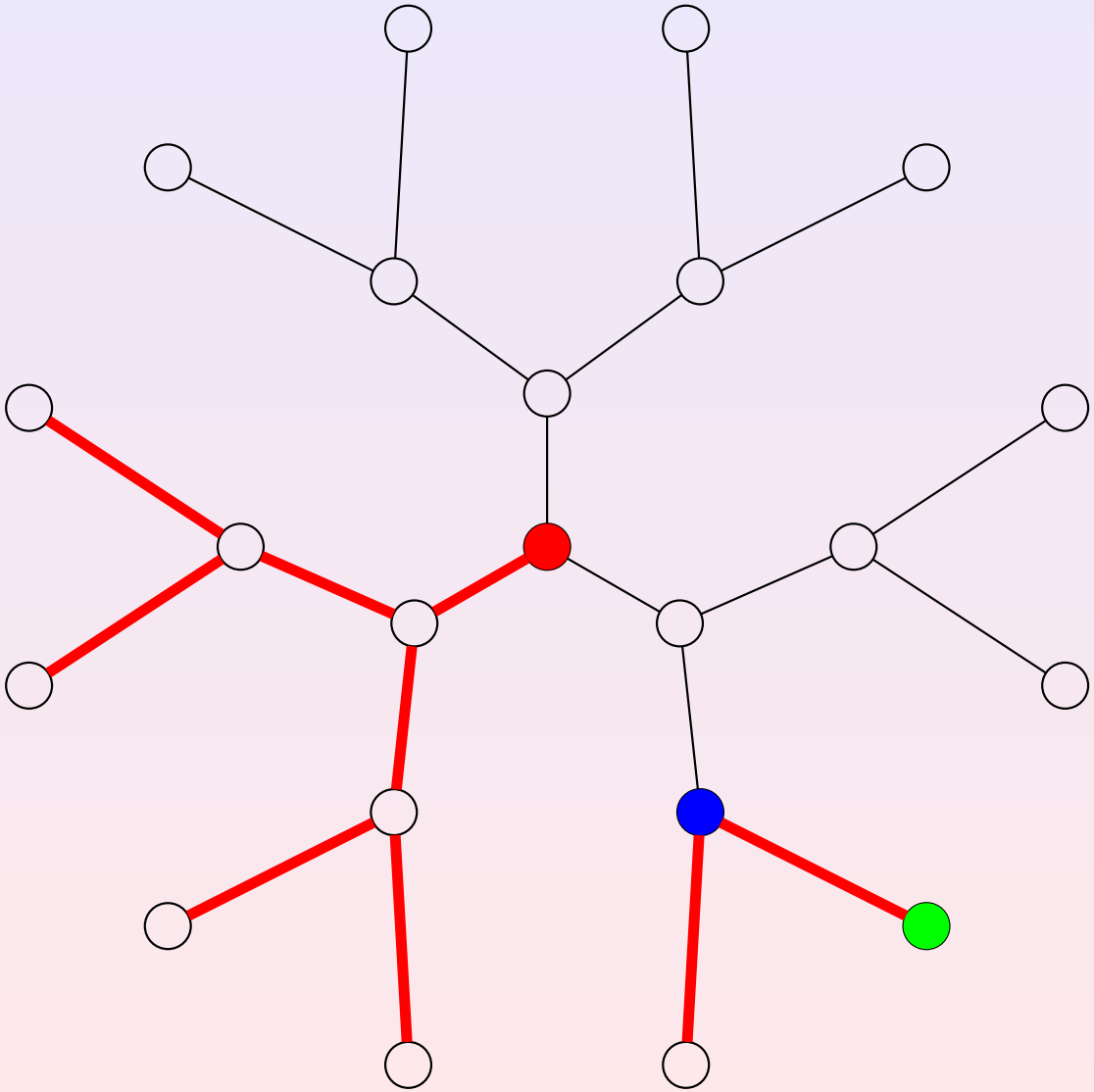
Encerclement dans un arbre



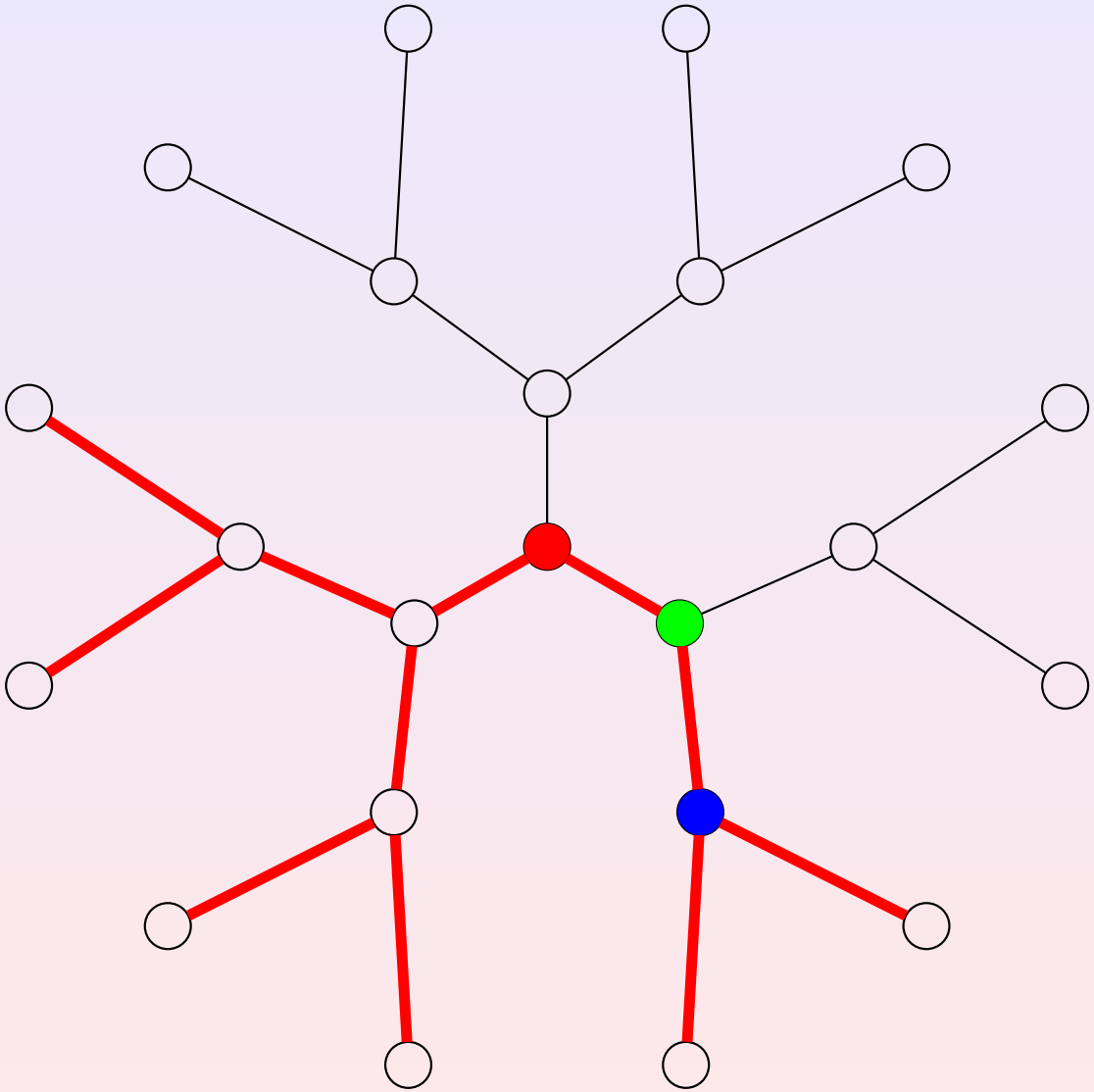
Encerclement dans un arbre



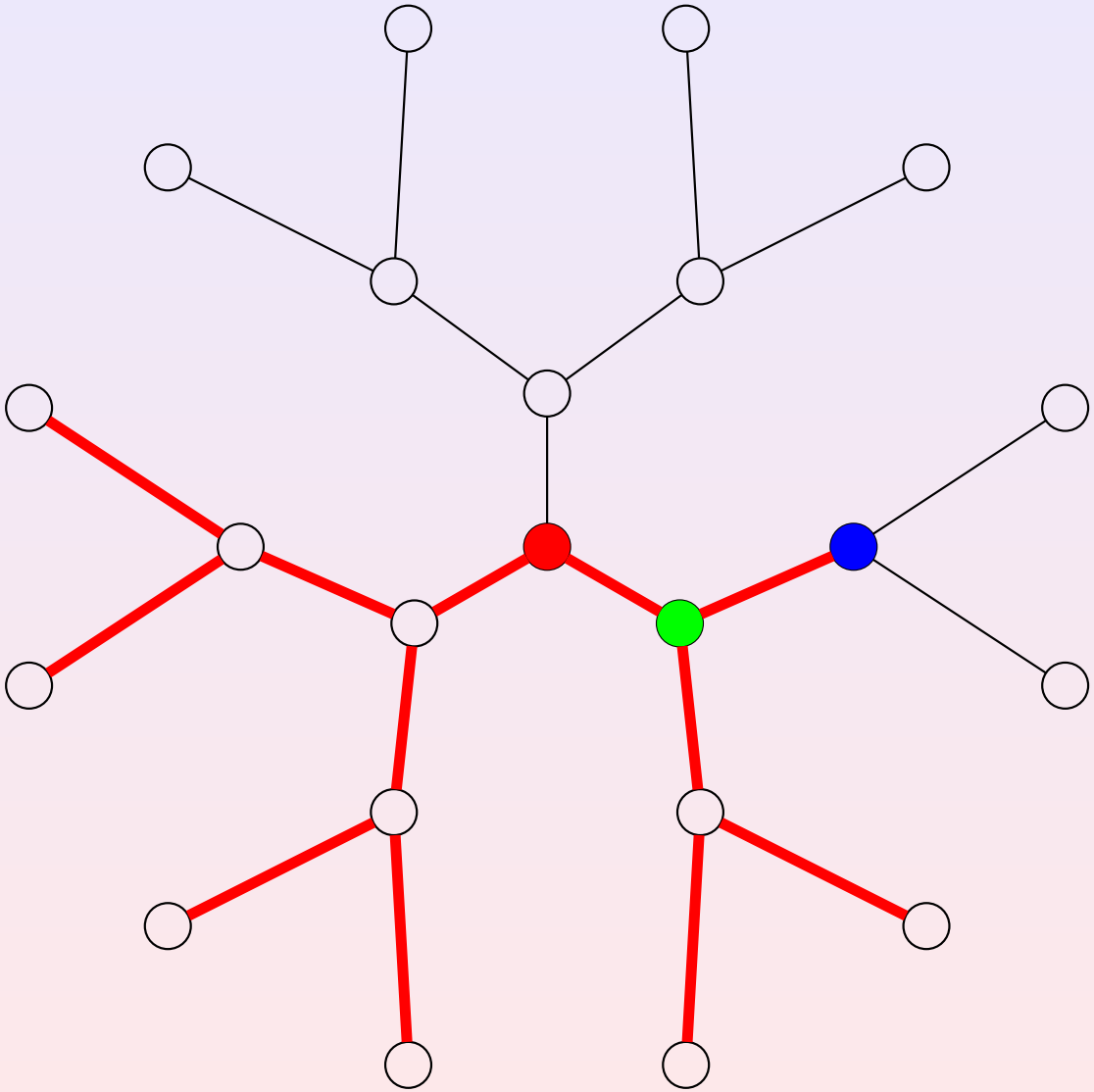
Encerclement dans un arbre



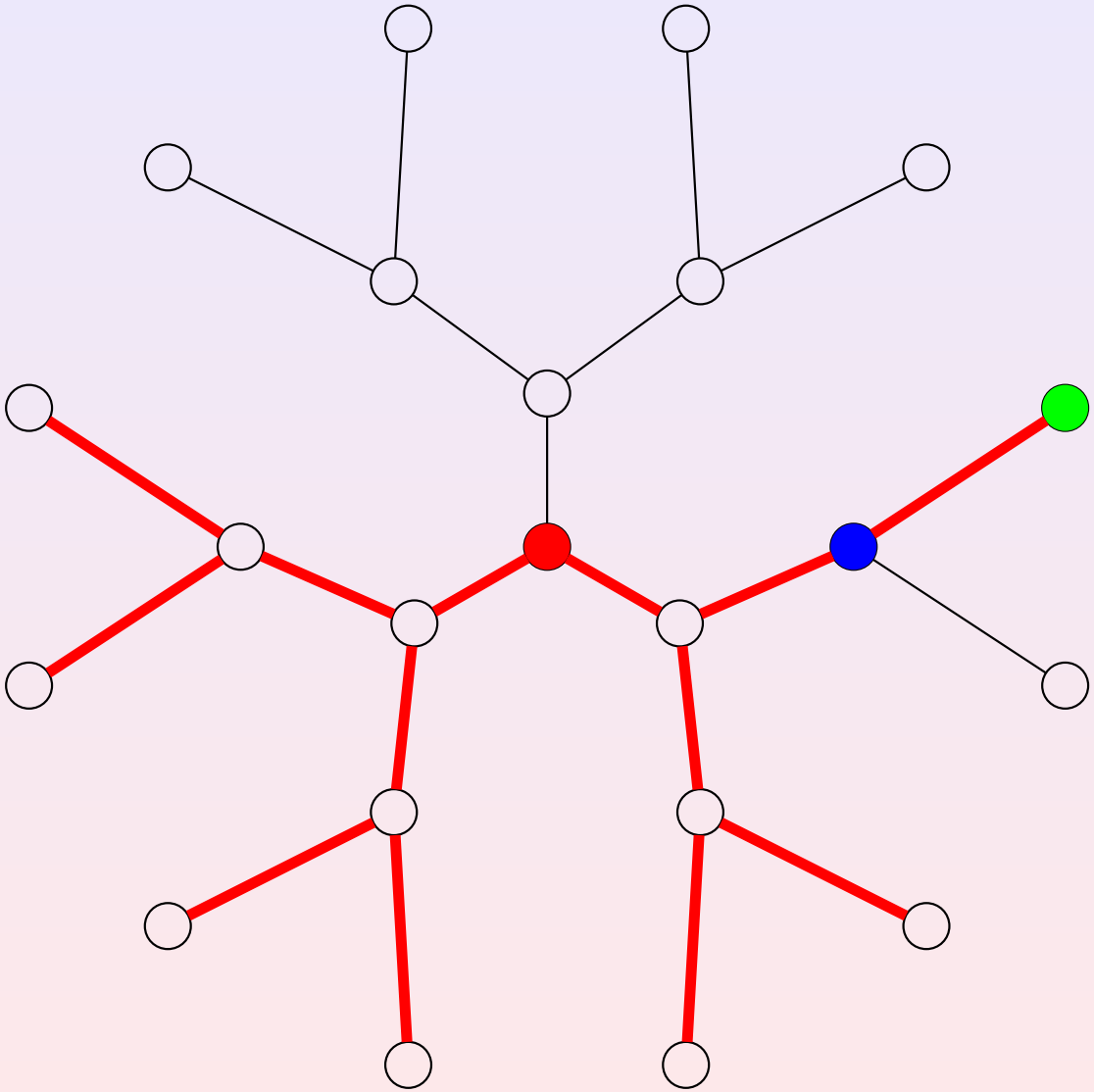
Encerclement dans un arbre



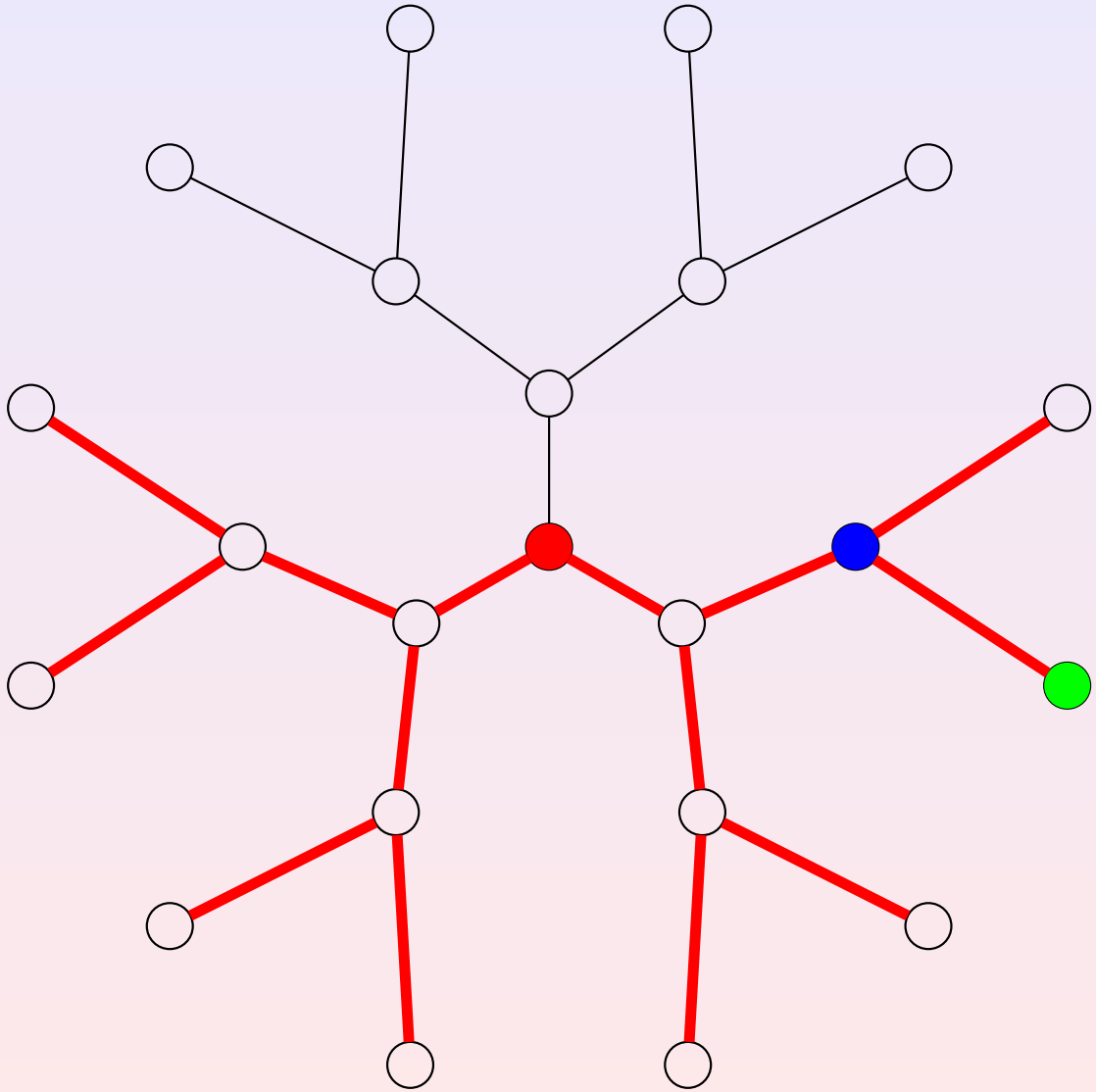
Encerclement dans un arbre



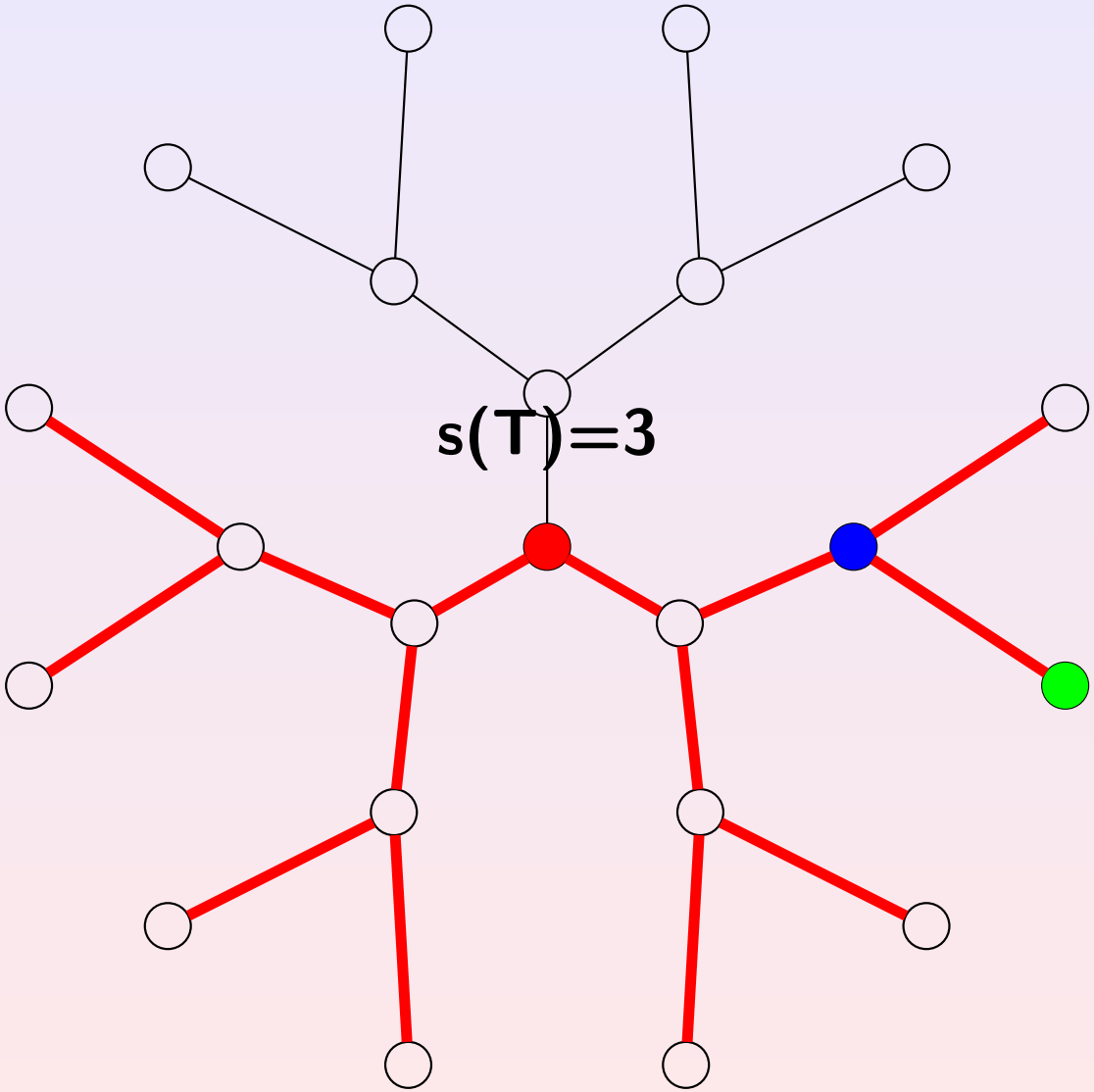
Encerclement dans un arbre



Encerclement dans un arbre



Encerclement dans un arbre



Encerclement visible

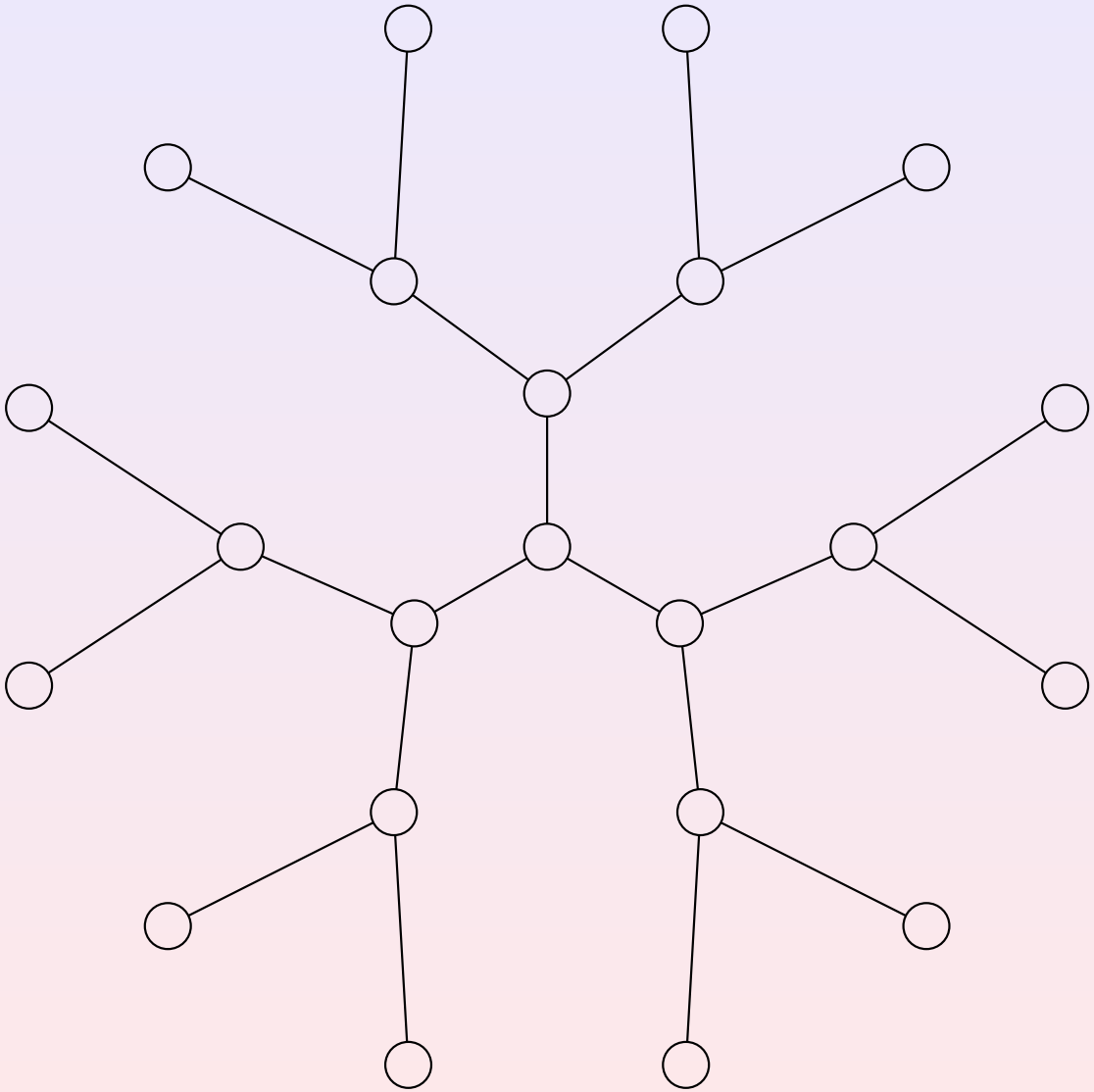
Visibilité du fugitif

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.

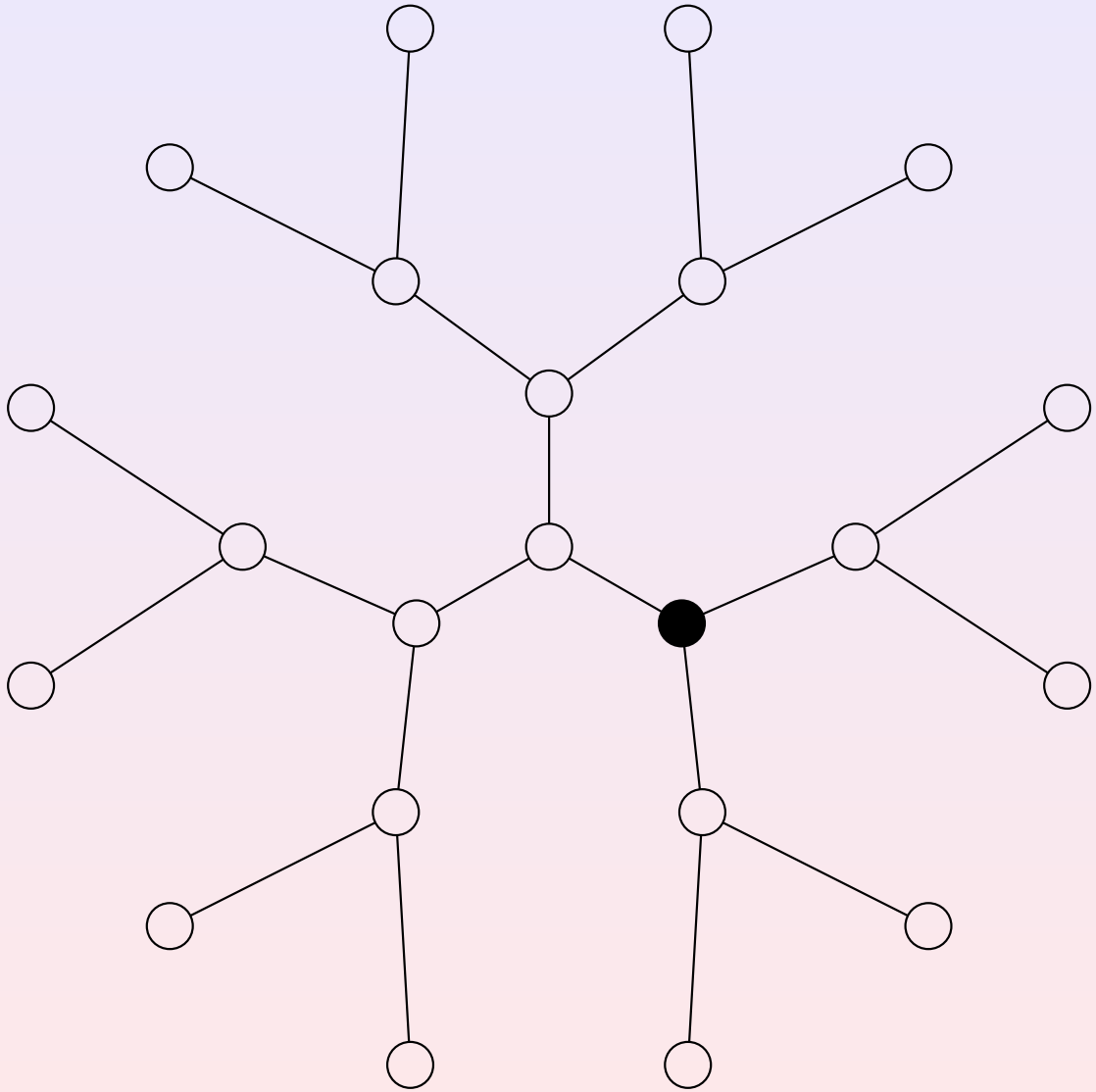
Paramètre associé

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

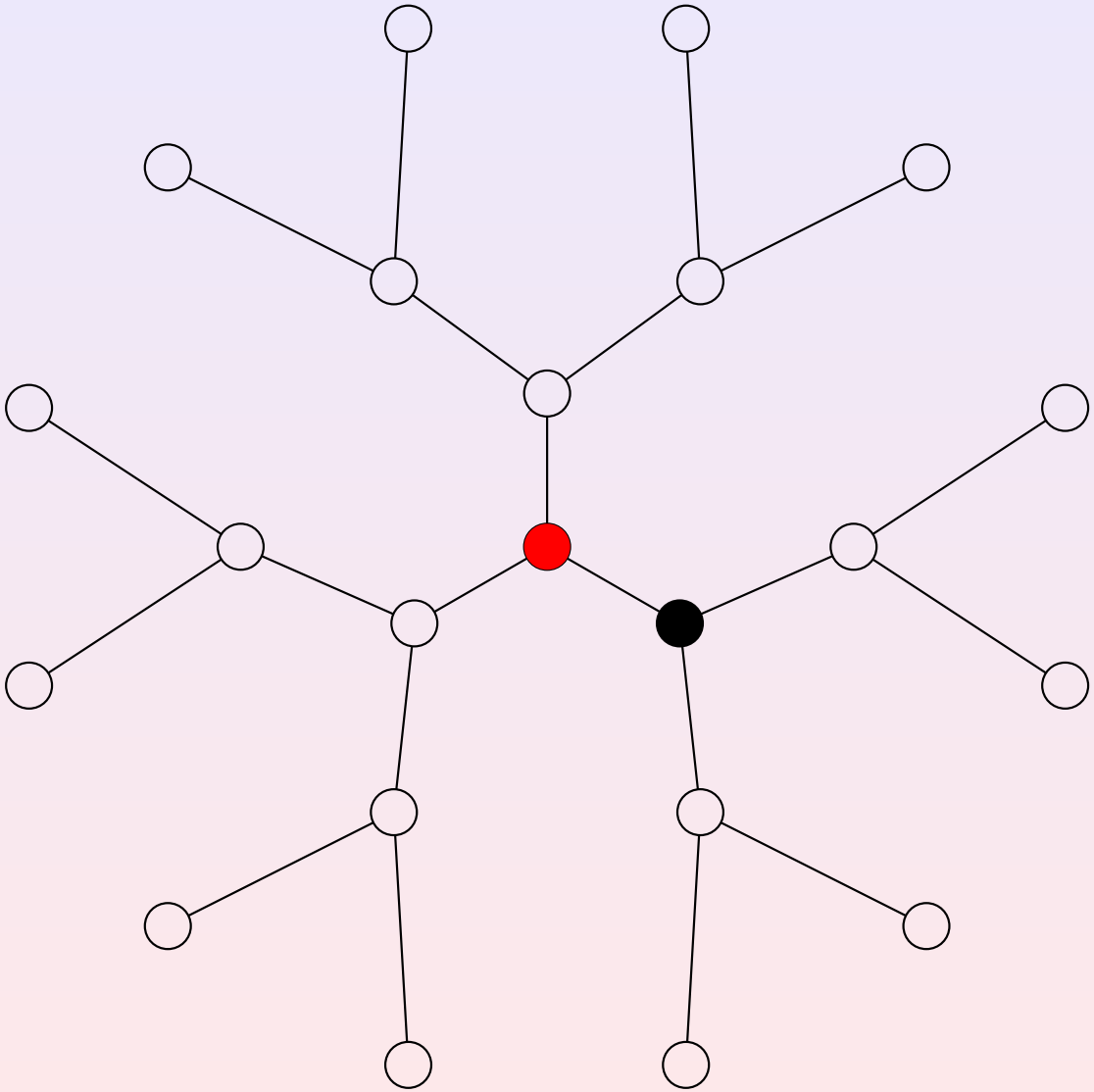
Encerclement d'un fugitif visible dans un arbre



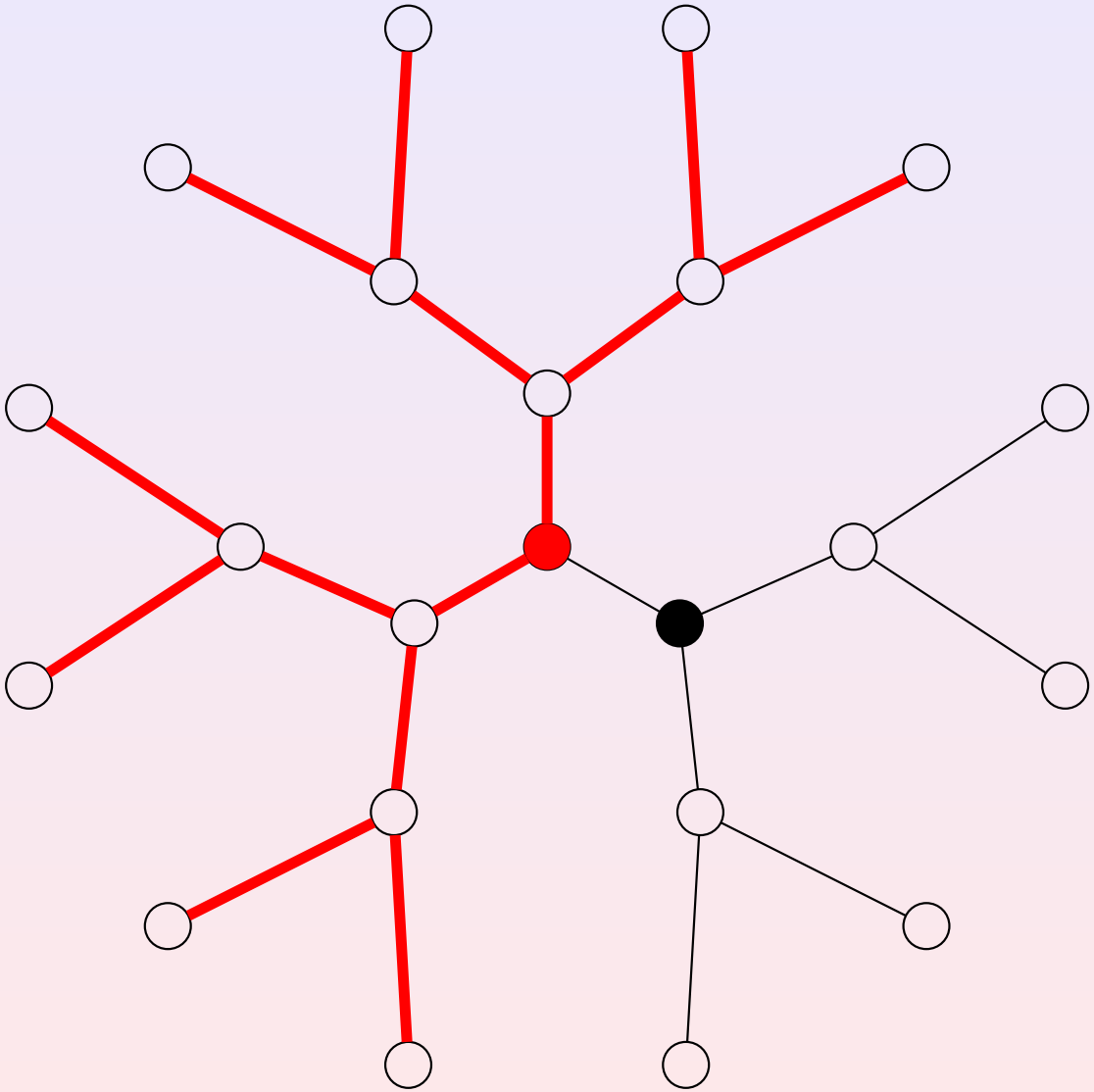
Encerclement d'un fugitif visible dans un arbre



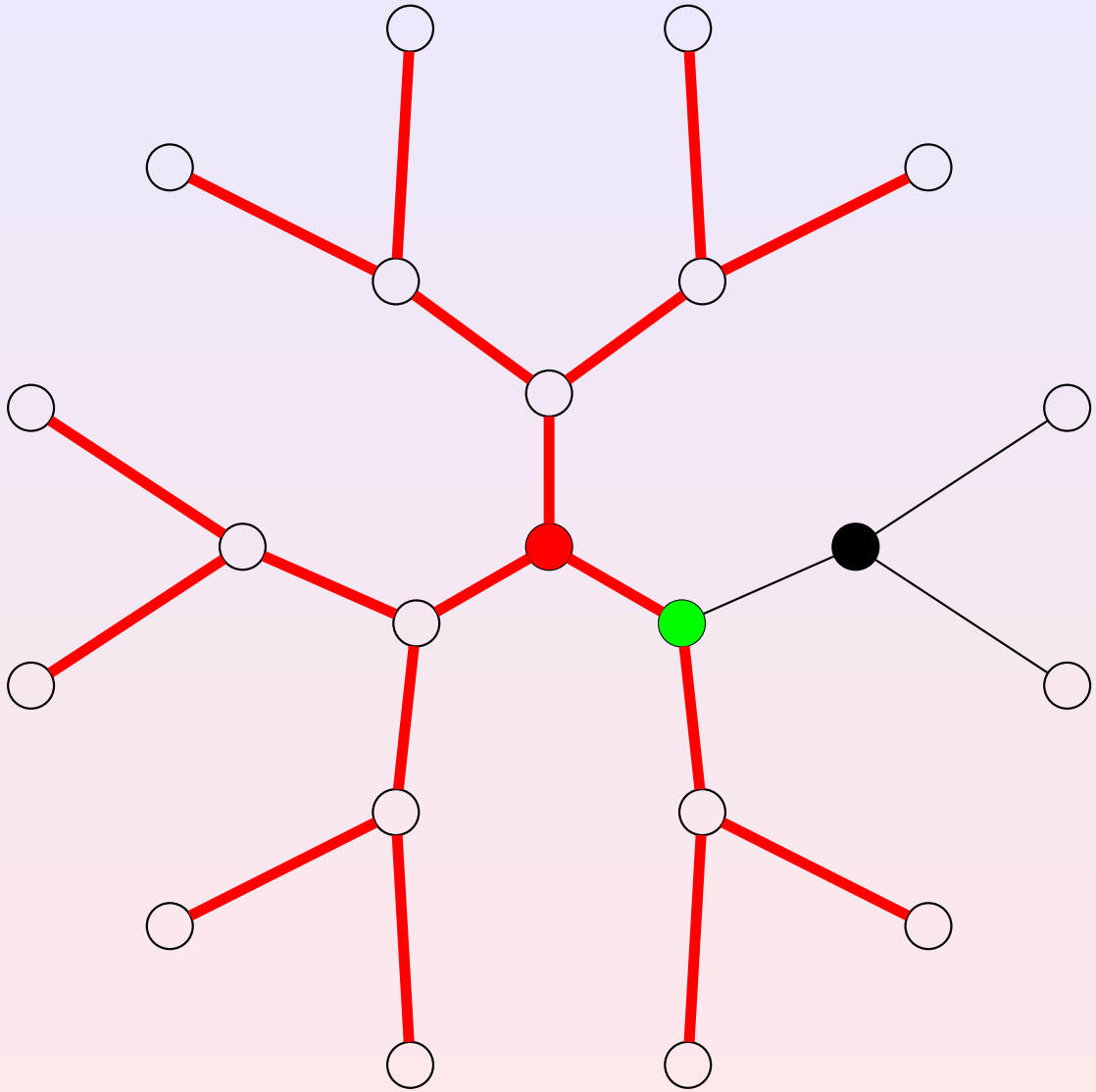
Encerclement d'un fugitif visible dans un arbre



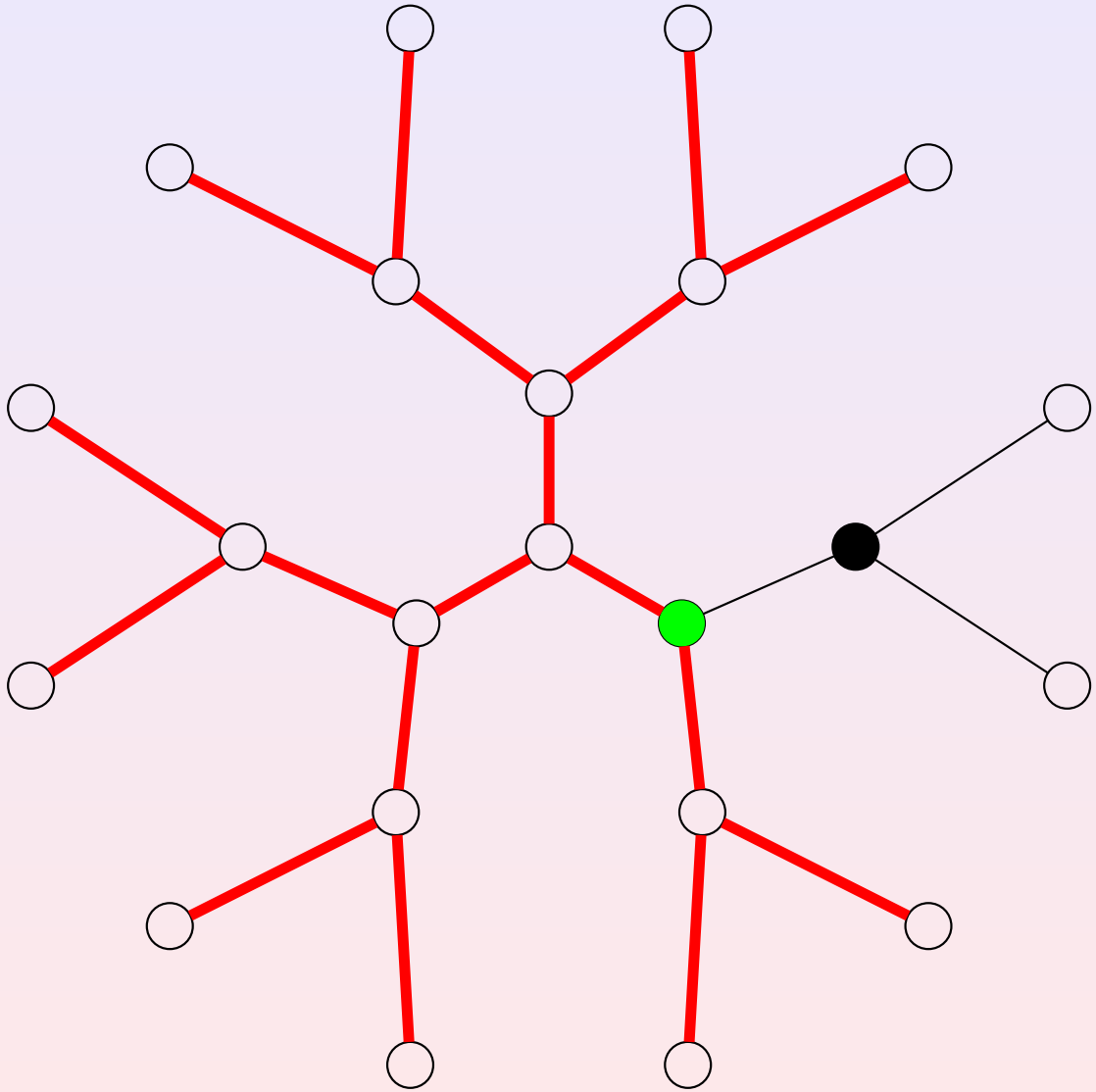
Encerclement d'un fugitif visible dans un arbre



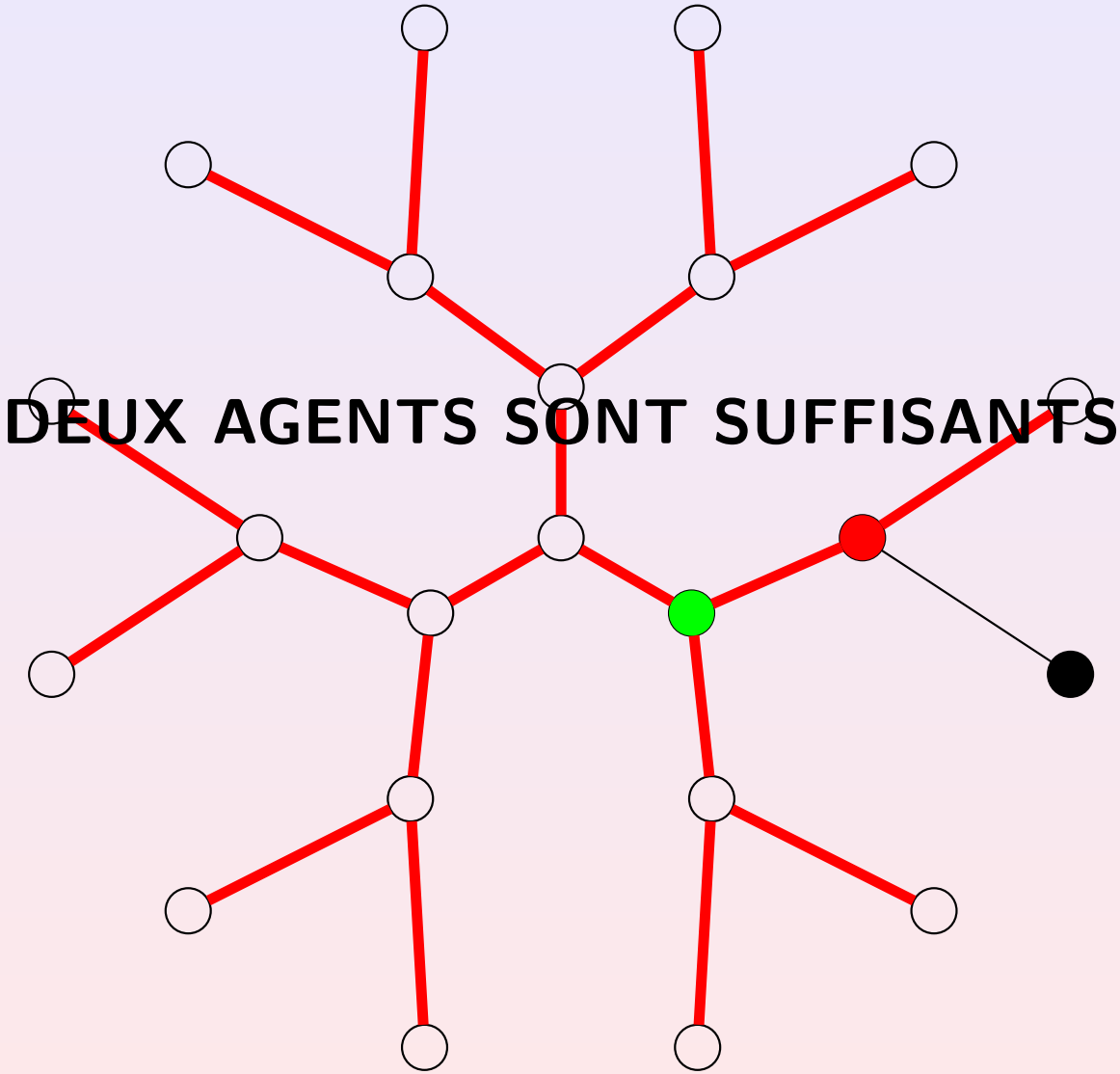
Encerclement d'un fugitif visible dans un arbre



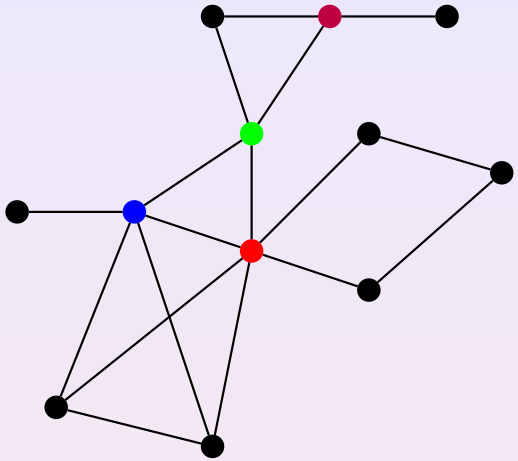
Encerclement d'un fugitif visible dans un arbre



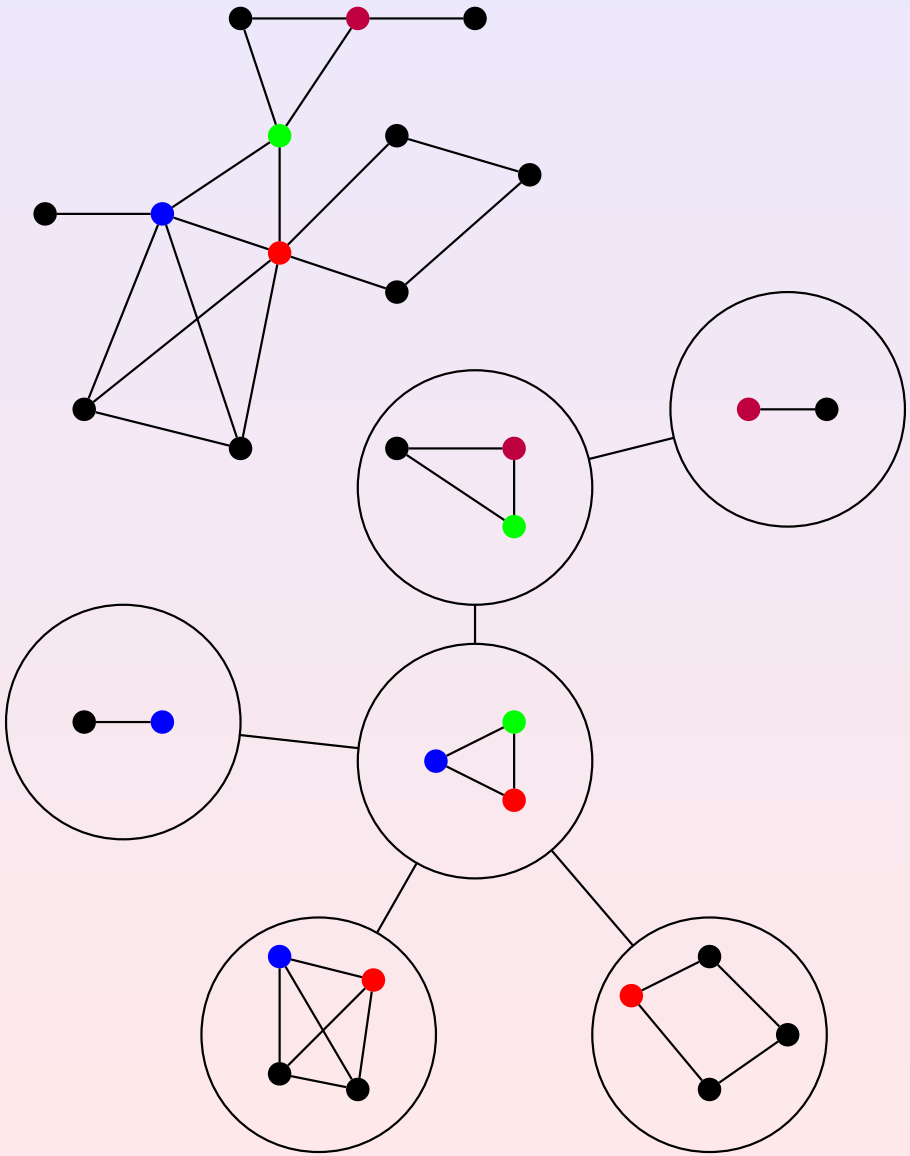
Encerclement d'un fugitif visible dans un arbre



Décompositions arborescente et linéaire

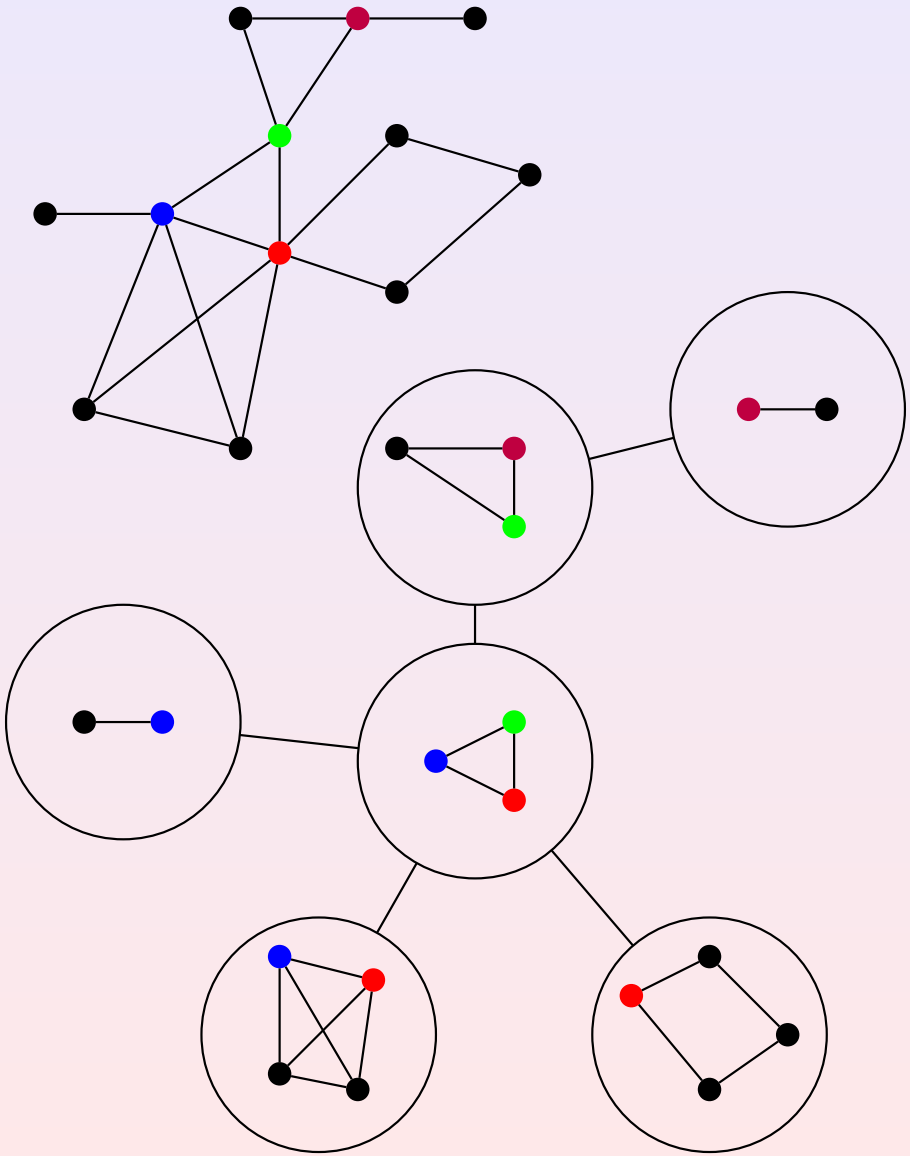


Décompositions arborescente et linéaire



arbre T et "sacs" $(X_t)_{t \in V(T)}$

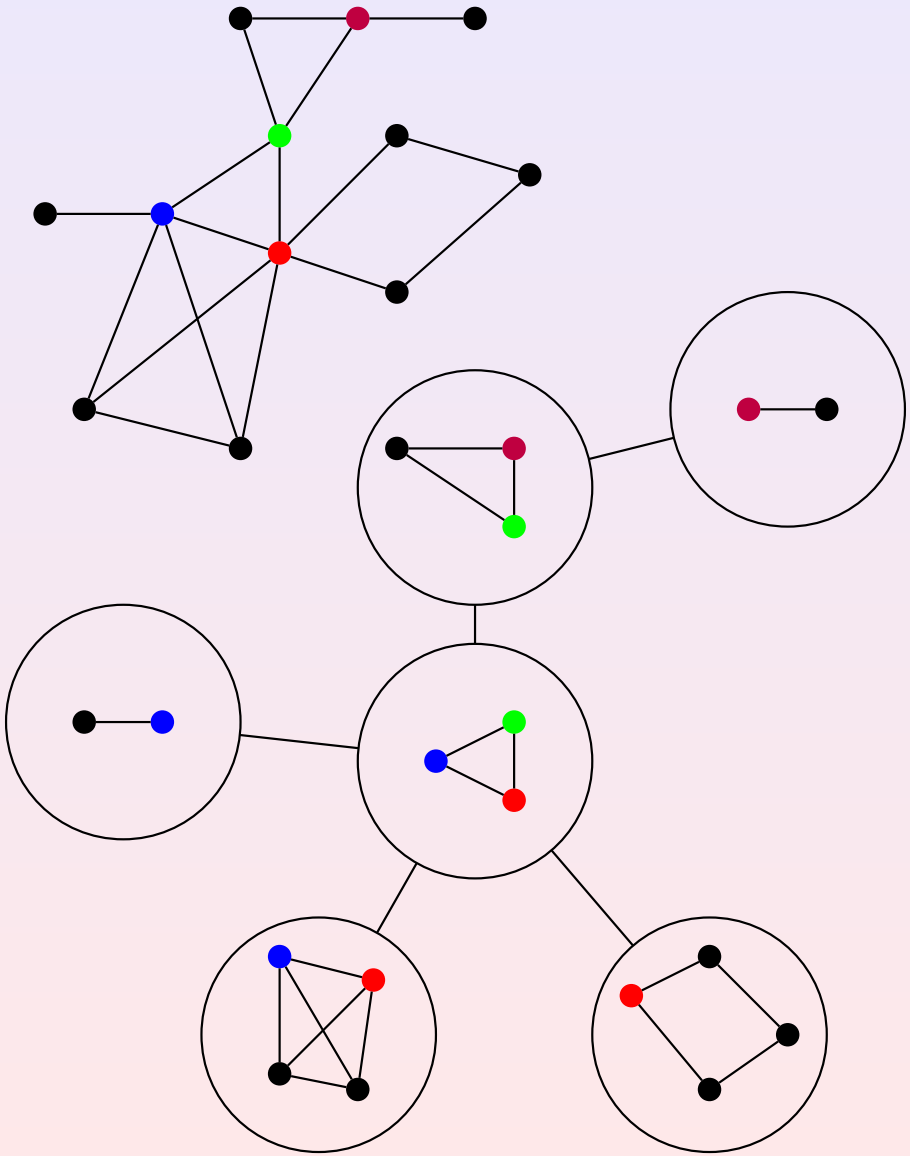
Décompositions arborescente et linéaire



arbre T et "sacs" $(X_t)_{t \in V(T)}$

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

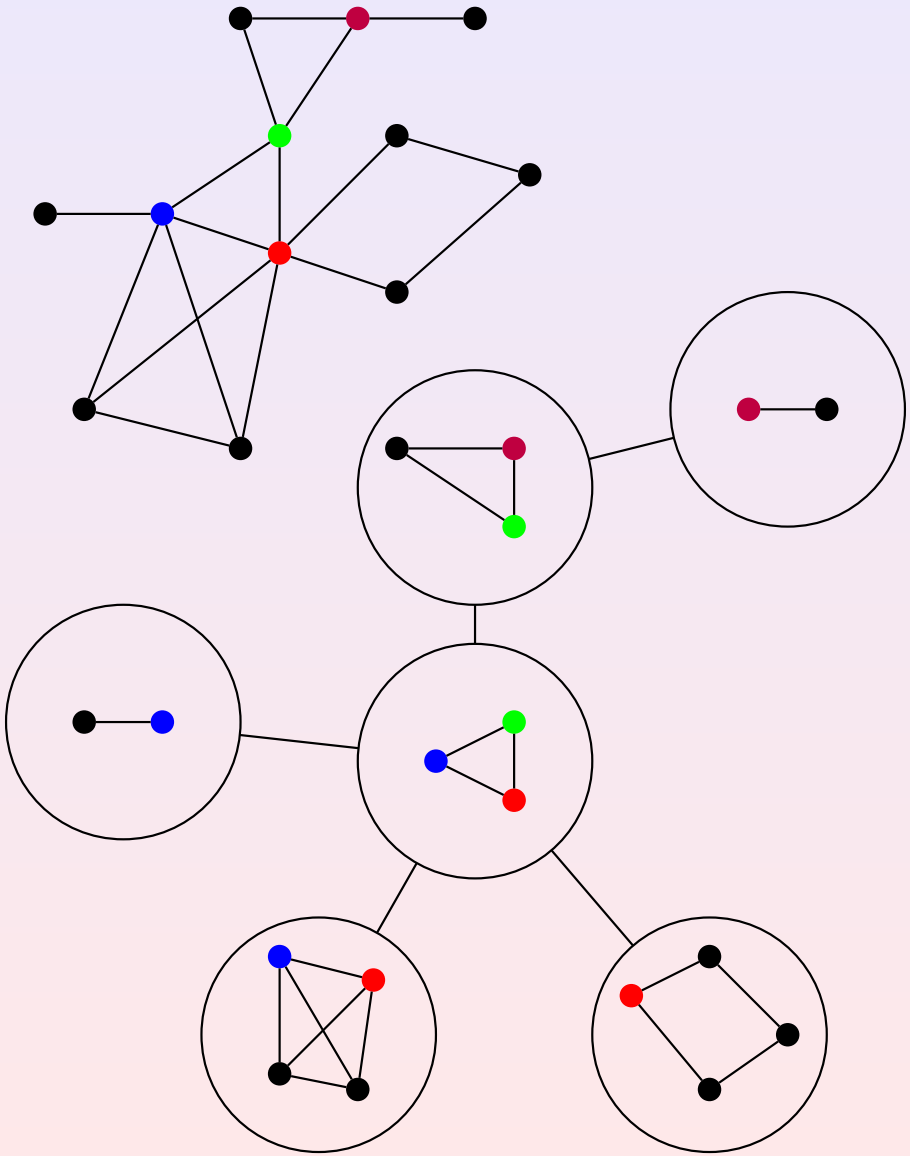
Décompositions arborescente et linéaire



arbre T et "sacs" $(X_t)_{t \in V(T)}$

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

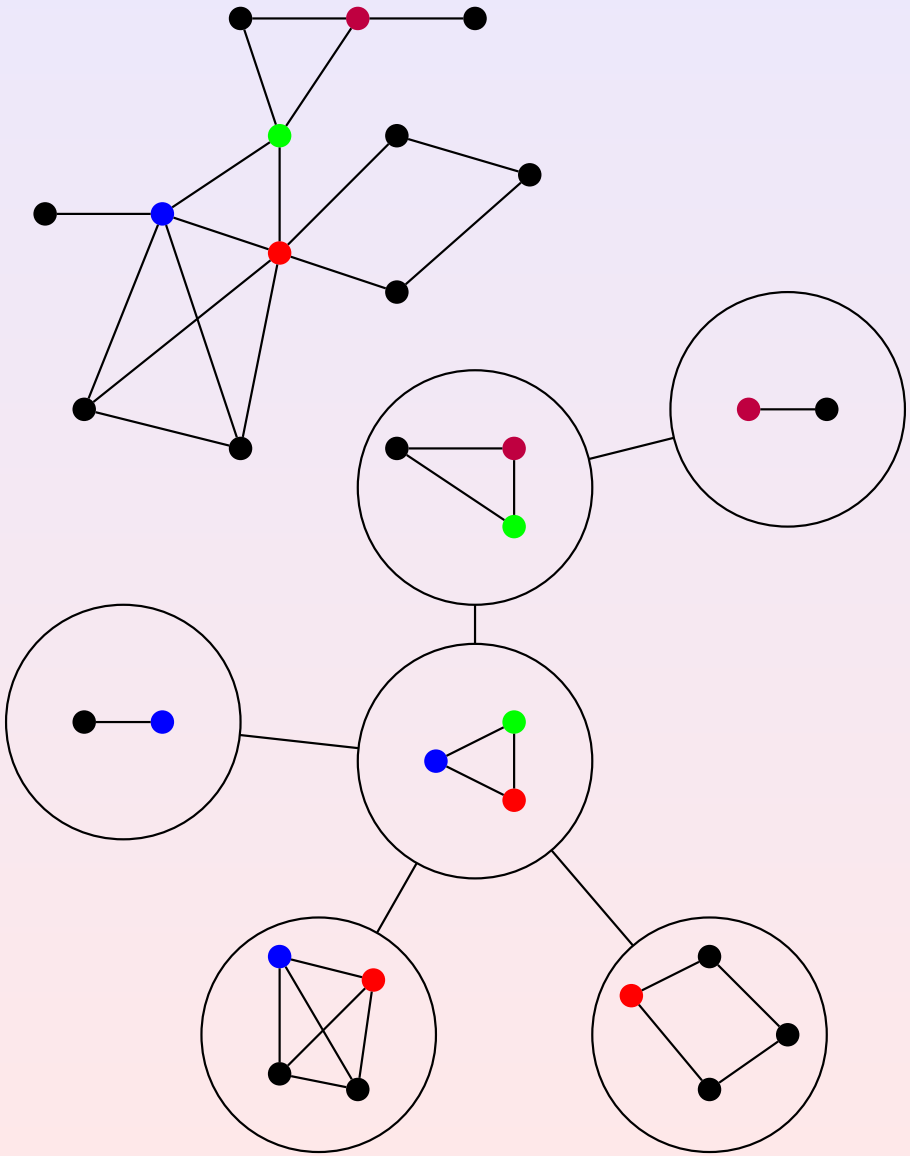
Décompositions arborescente et linéaire



arbre T et "sacs" $(X_t)_{t \in V(T)}$

- chaque **sommet** de G est contenu dans au moins un sac ;
- les 2 extrémités d'une **arête** sont contenues dans au moins un sac ;
- Pour tout sommet de G, l'ensemble des sacs qui contiennent ce sommet forme un sous-arbre.

Décompositions arborescente et linéaire

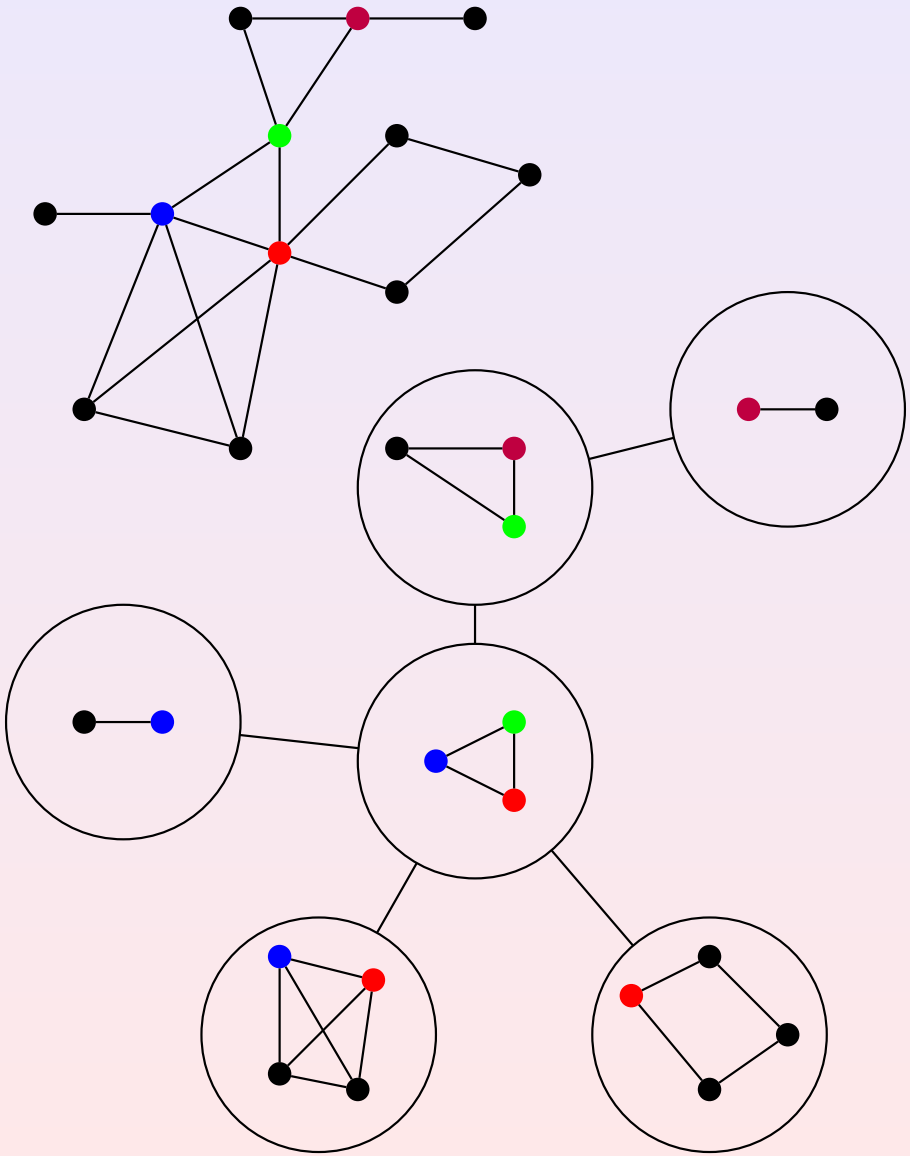


arbre T et "sacs" $(X_t)_{t \in V(T)}$

- chaque **sommet** de G est contenu dans au moins un sac ;
- les 2 extrémités d'une **arête** sont contenues dans au moins un sac ;
- Pour tout sommet de G, l'ensemble des sacs qui contiennent ce sommet forme un **sous-arbre**.

Largeur = #plus grand sac -1

Décompositions arborescente et linéaire



arbre T et "sacs" $(X_t)_{t \in V(T)}$

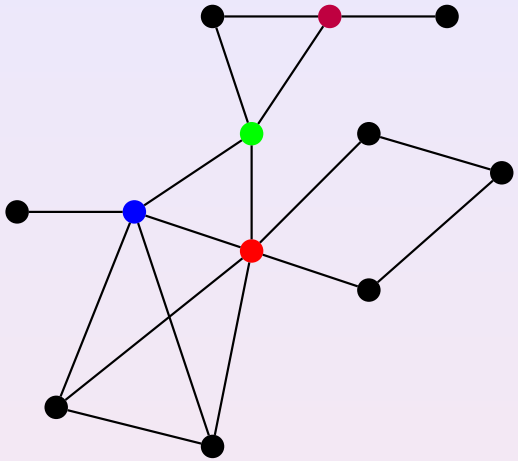
- chaque **sommet** de G est contenu dans au moins un sac ;
- les 2 extrémités d'une **arête** sont contenues dans au moins un sac ;
- Pour tout sommet de G, l'ensemble des sacs qui contiennent ce sommet forme un **sous-arbre**.

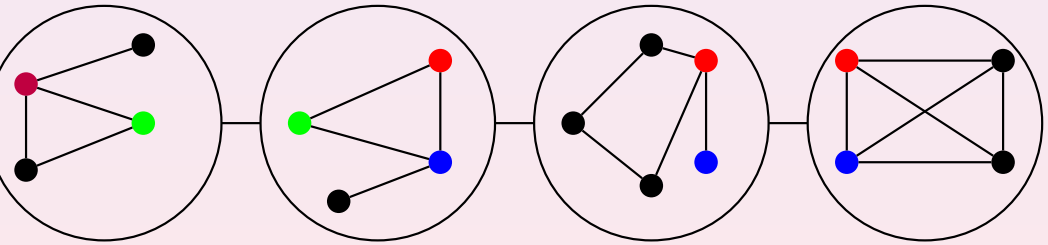
Largeur = #plus grand sac -1

Largeur arborescente de G

tw(G), largeur minimum parmi les décompositions arborescente

Décompositions arborescente et linéaire





chemin P et "sacs" $(X_t)_{t \in V(P)}$

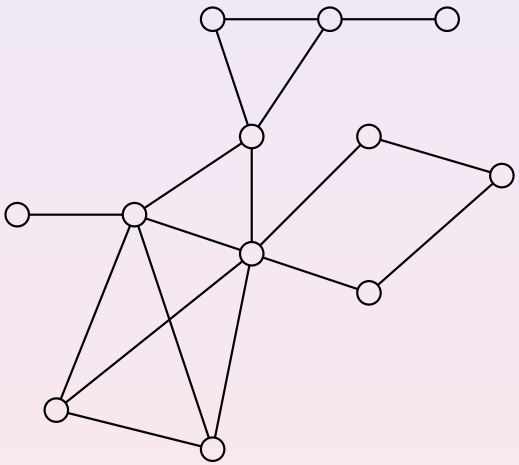
- chaque **sommet** de G est contenu dans au moins un sac ;
- les 2 extrémités d'une **arête** sont contenues dans au moins un sac ;
- Pour tout sommet de G, l'ensemble des sacs qui contiennent ce sommet forme un **sous-chemin**.

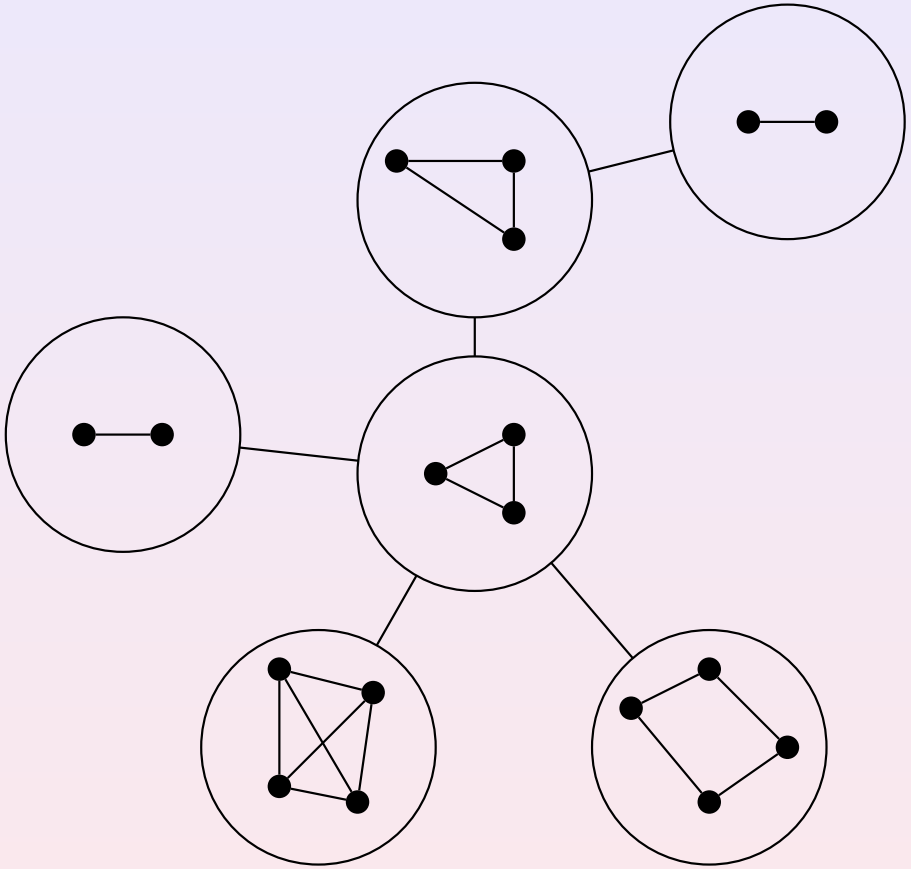
Largeur = #plus grand sac -1

Largeur linéaire de G

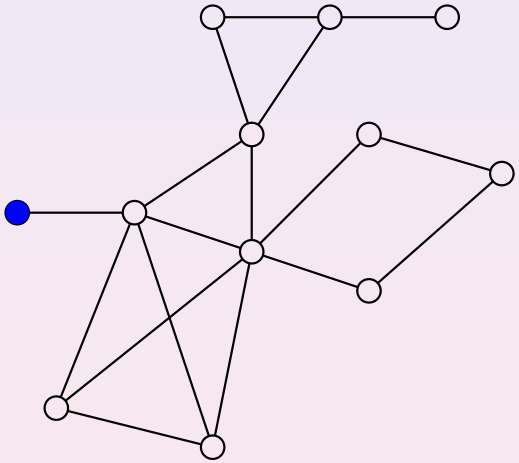
pw(G), largeur minimum parmi les décompositions linéaires

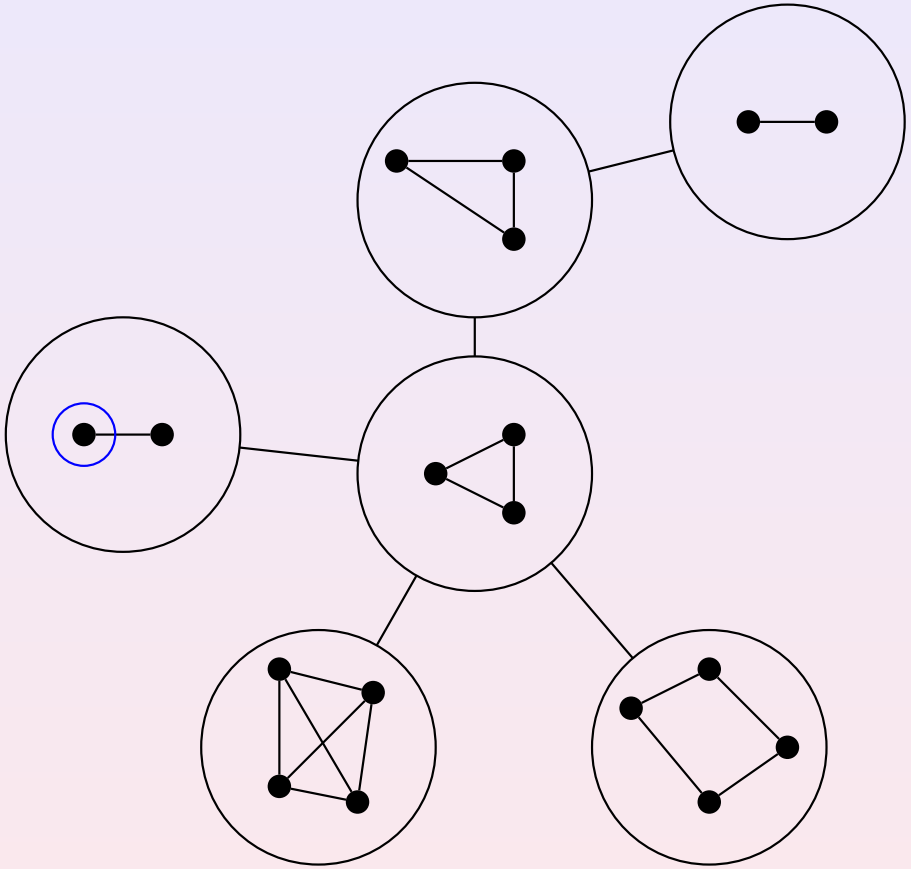
Décomposition arborescente et Encerclement



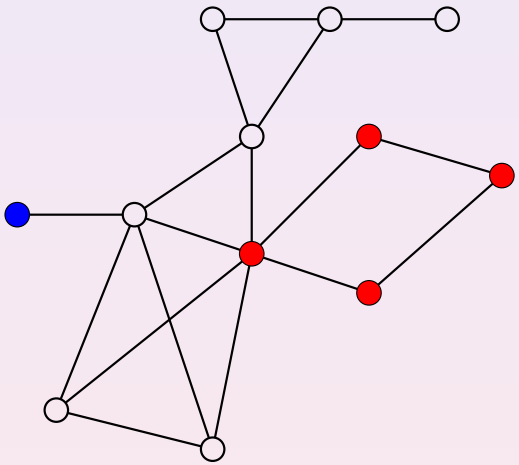


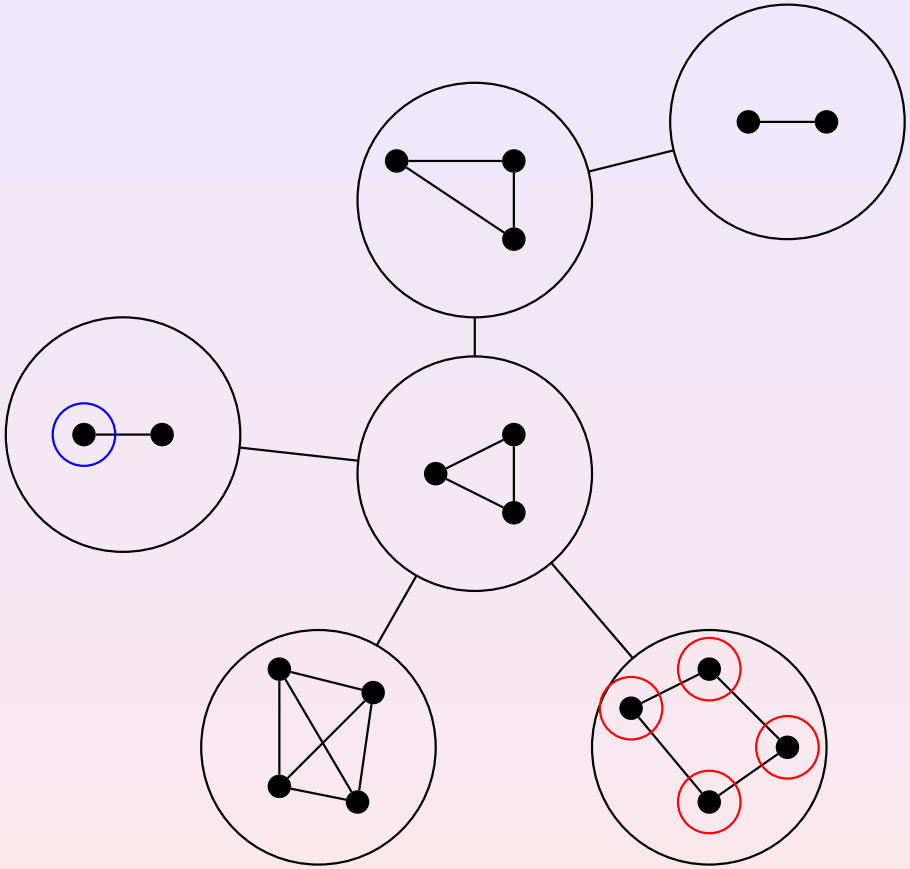
Décomposition arborescente et Encerclement



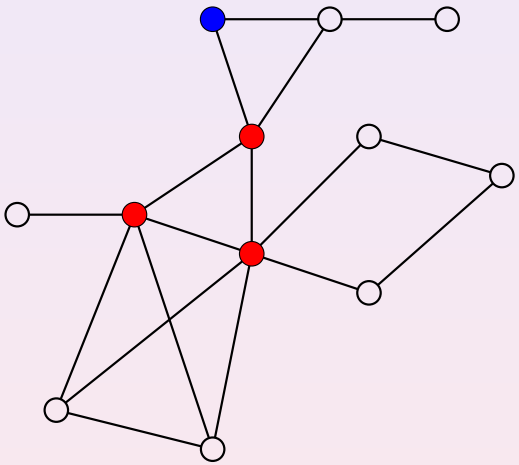


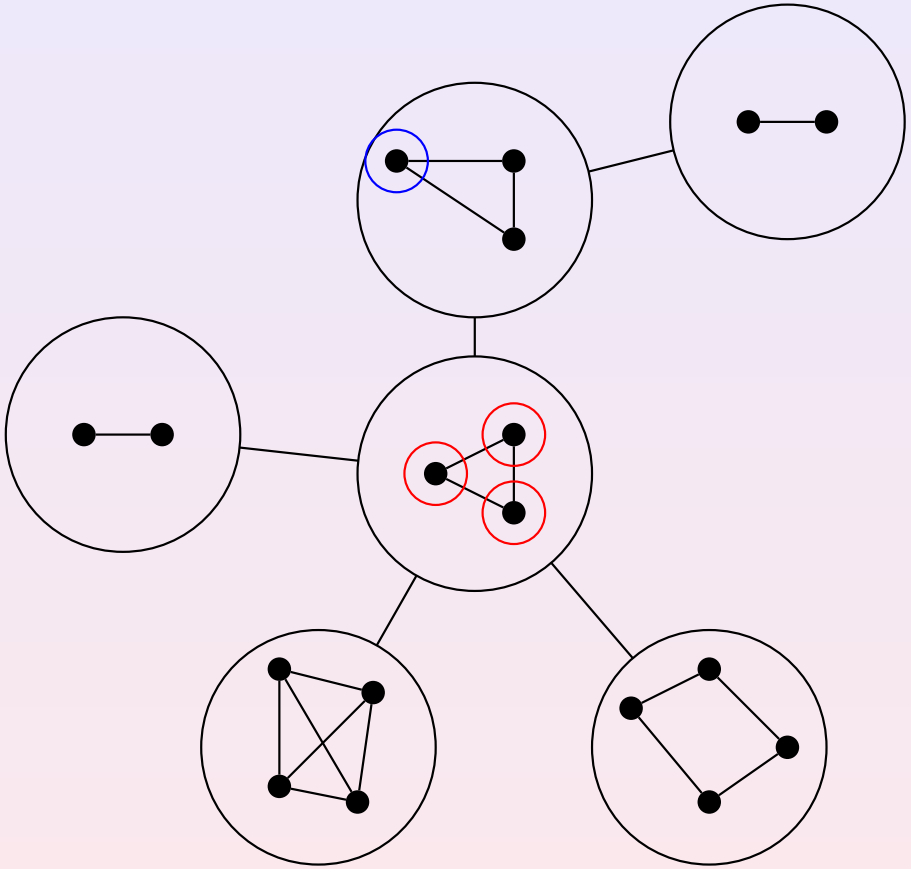
Décomposition arborescente et Encerclement



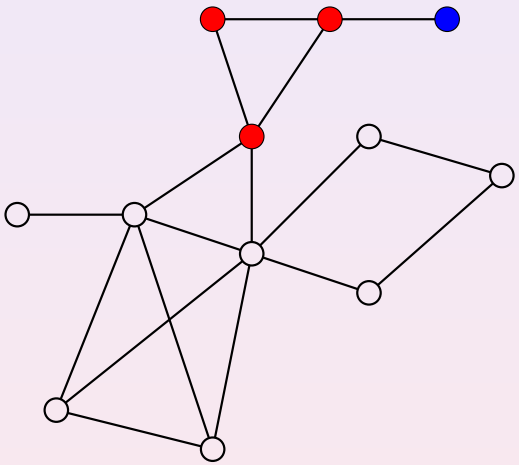


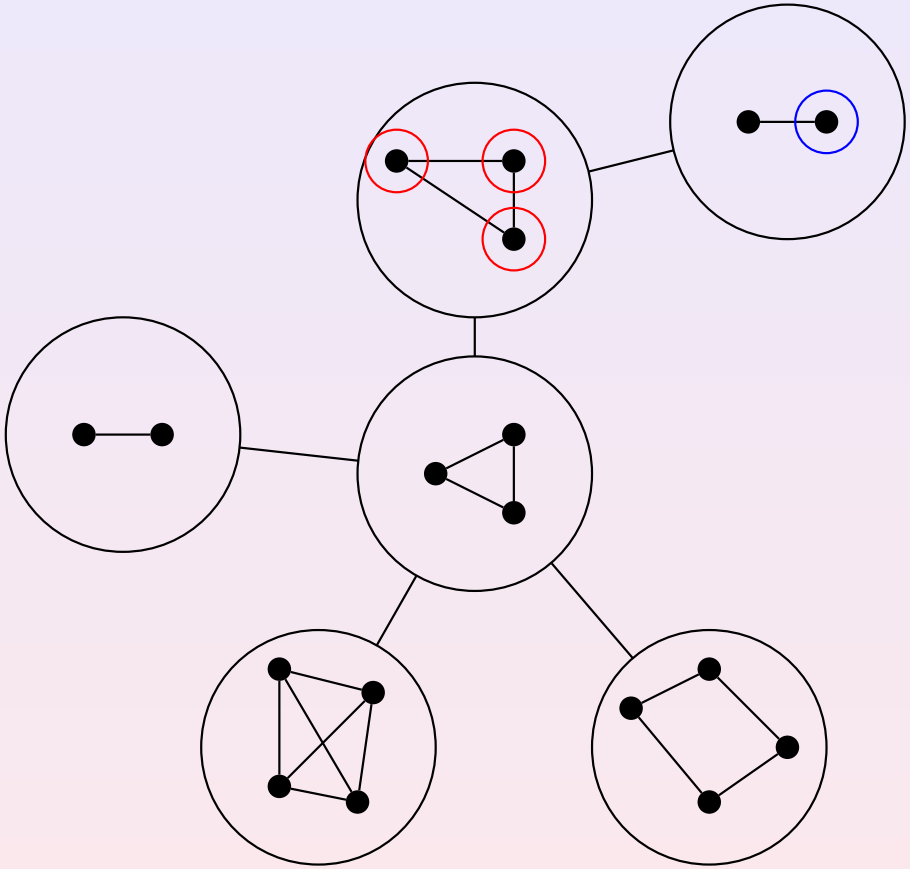
Décomposition arborescente et Encerclement



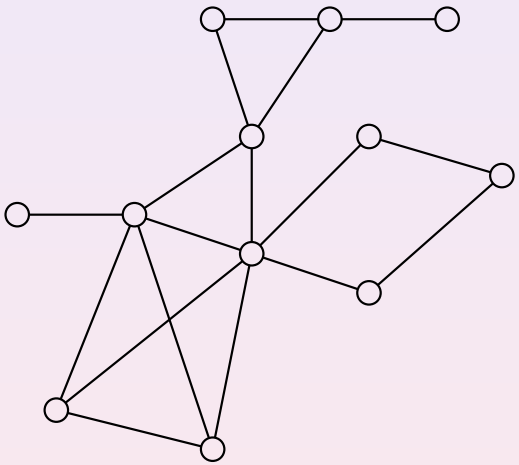


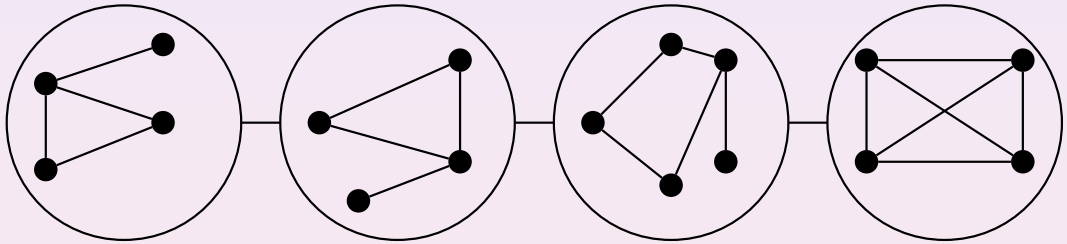
Décomposition arborescente et Encerclement



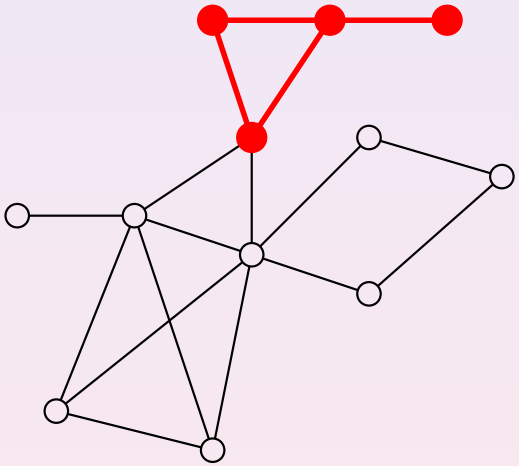


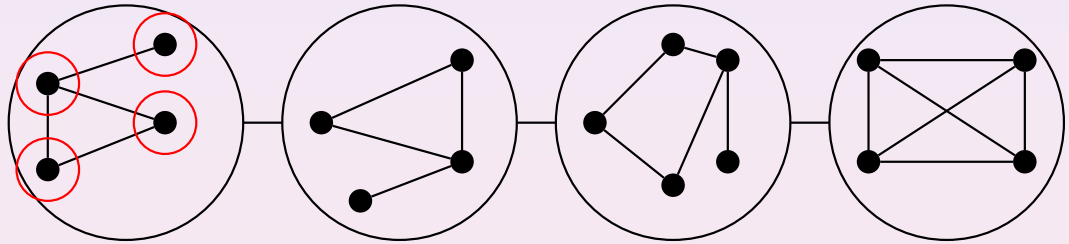
Décomposition linéaire et Encerclement



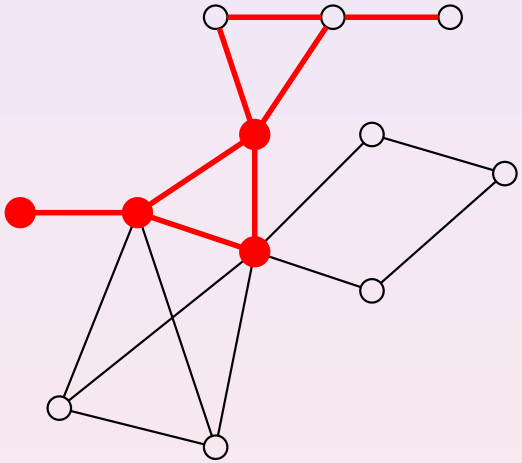


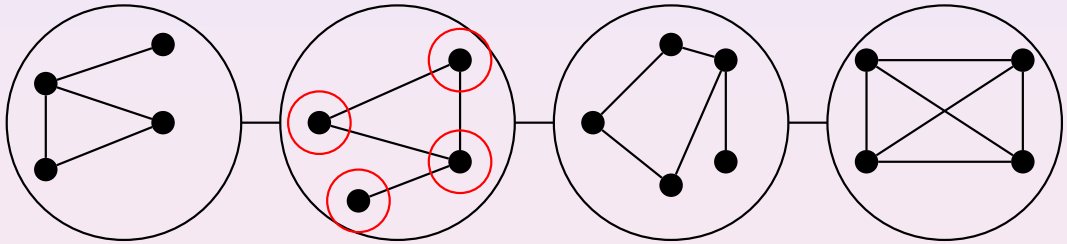
Décomposition linéaire et Encerclement



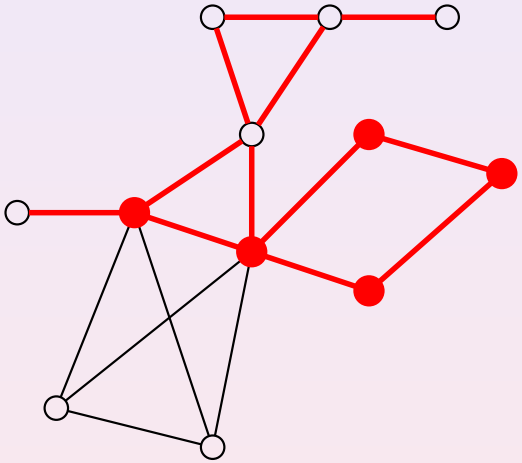


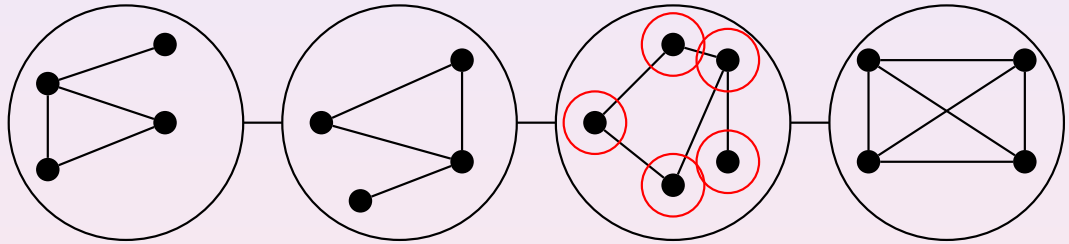
Décomposition linéaire et Encerclement





Décomposition linéaire et Encerclement





État de l'art

$s(G) = pw(G) + 1$

- **Kirousis et Papadimitriou**, Theor. Comp. Sc., 1986
  Searching and Pebbing.
- **Ellis, Sudborough et Turner**. Inf. and Comp., 1994
  The vertex separation and search number of a graph

$s_{visible}(G) = tw(G) + 1$

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

Encerclement non déterministe

Opération supplémentaire : Question

- oracle ;
- compromis entre le nombre d'agents et celui de questions ;
- encerclement q-limité non déterministe, $s_q(G)$.

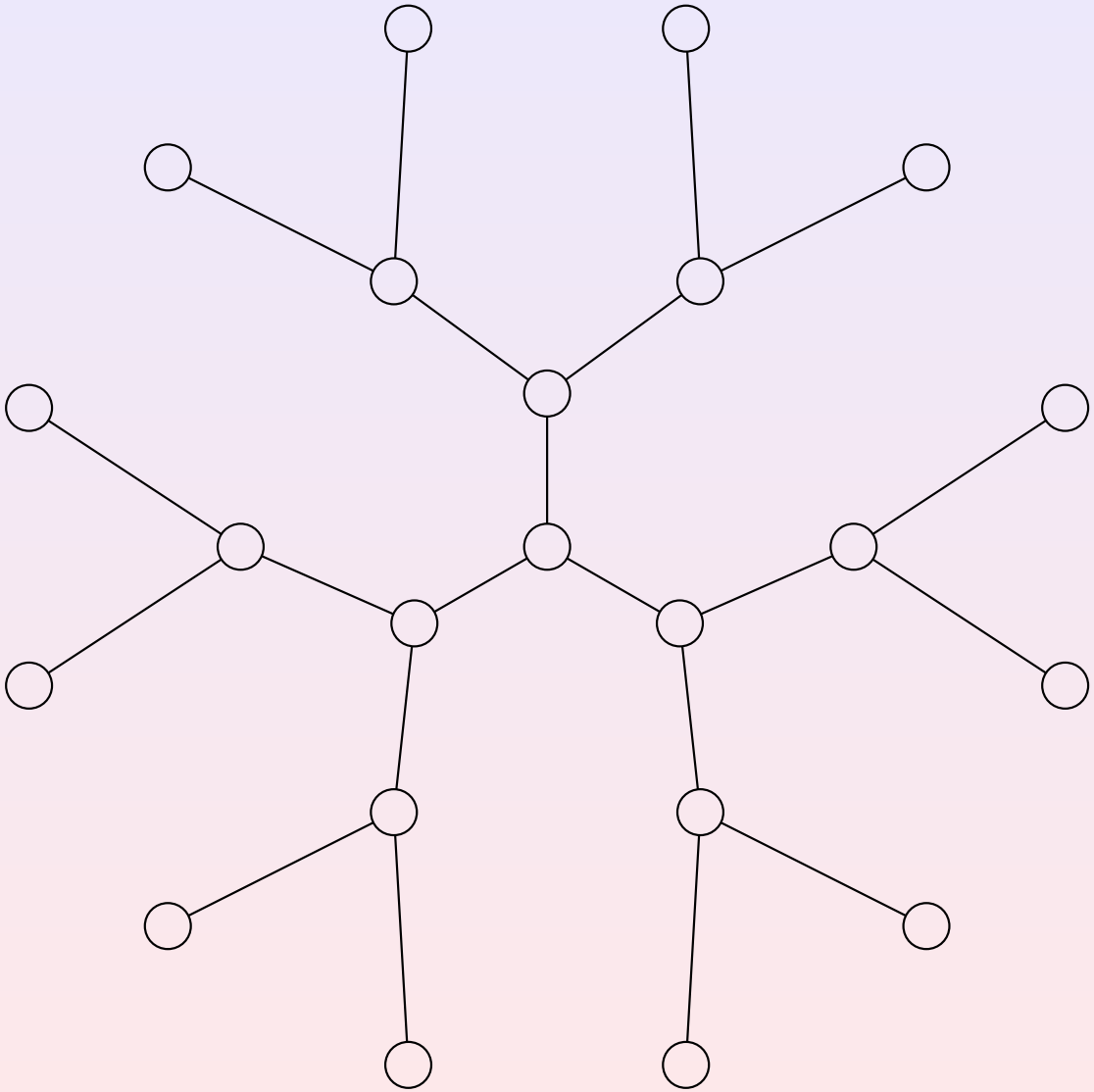
Relation avec l'Encerclement

- $s(G) = s_0(G)$;
- $s_{visible}(G) = s_\infty(G)$.

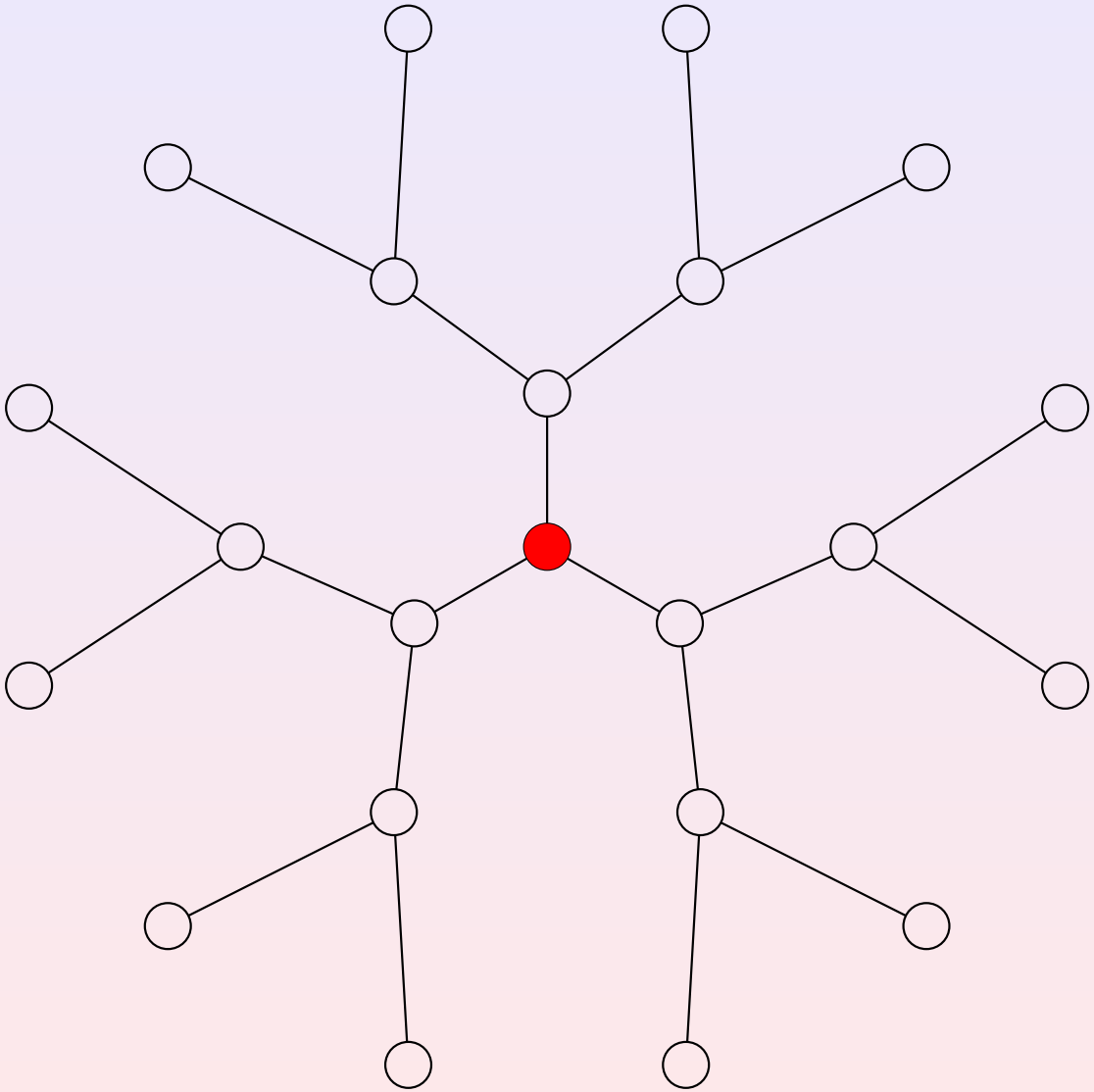
Remarque

Dans la suite, on ne considère que des stratégies monotones.

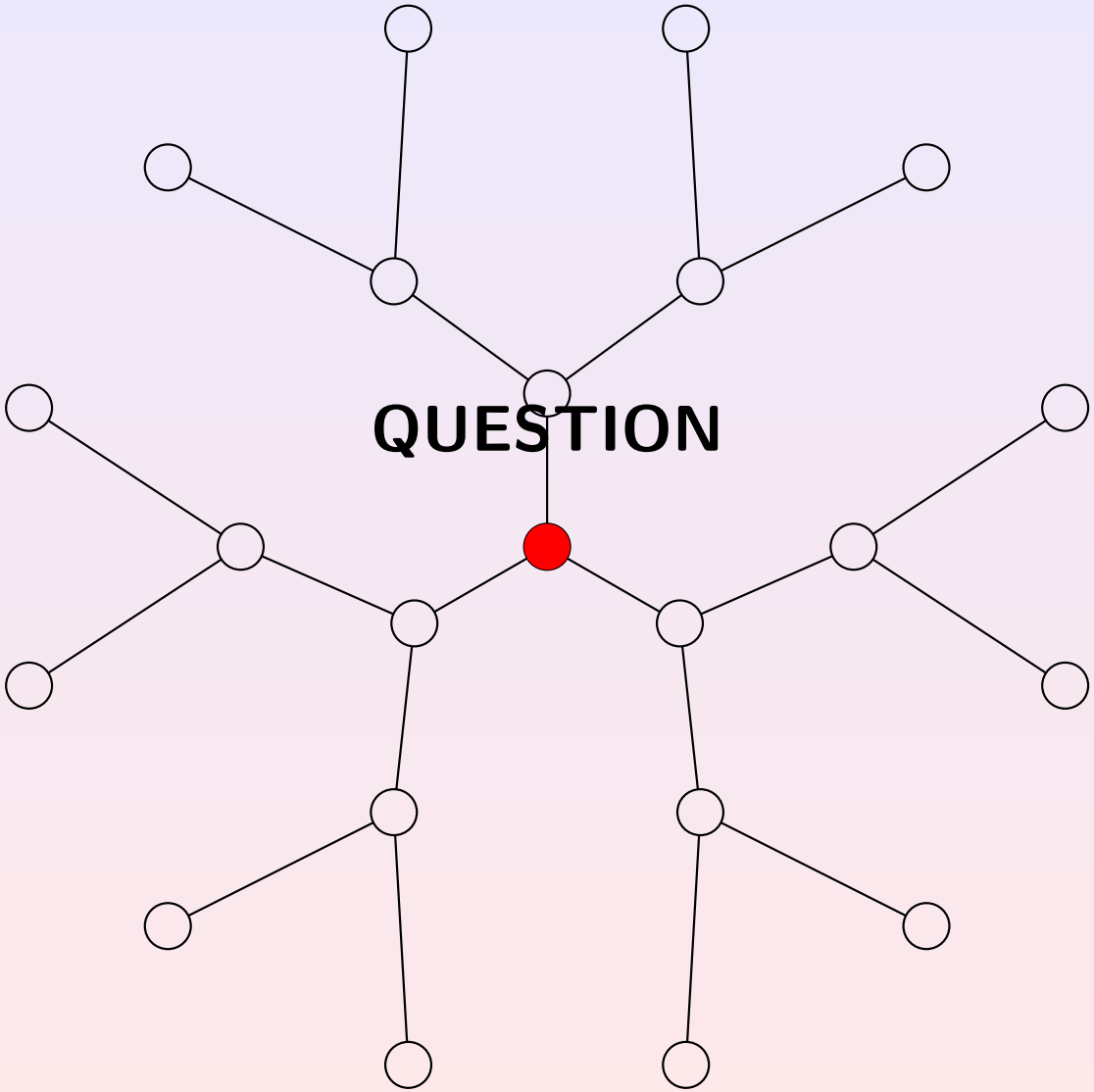
Encerclement non déterministe avec 2 questions



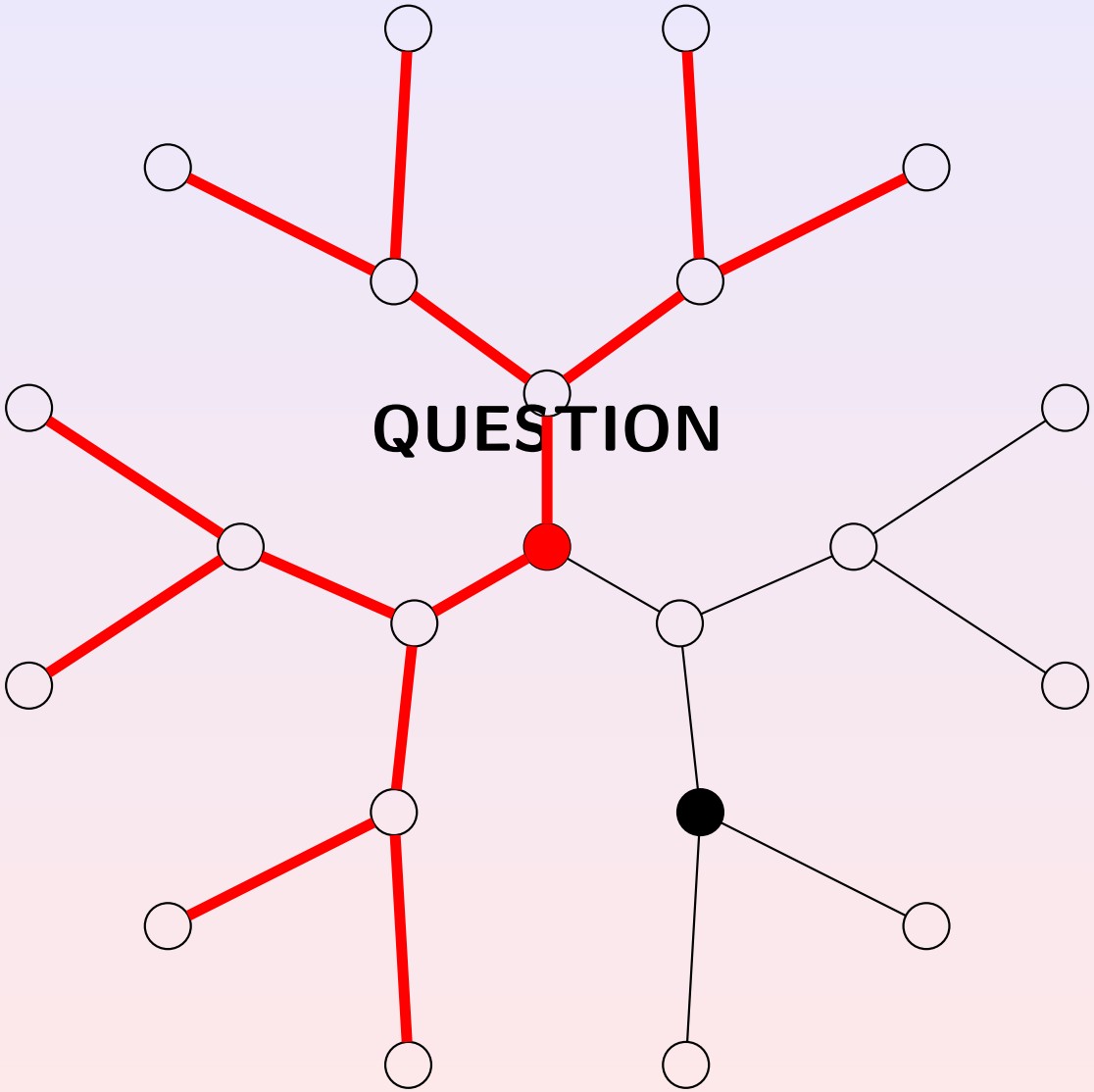
Encerclement non déterministe avec 2 questions



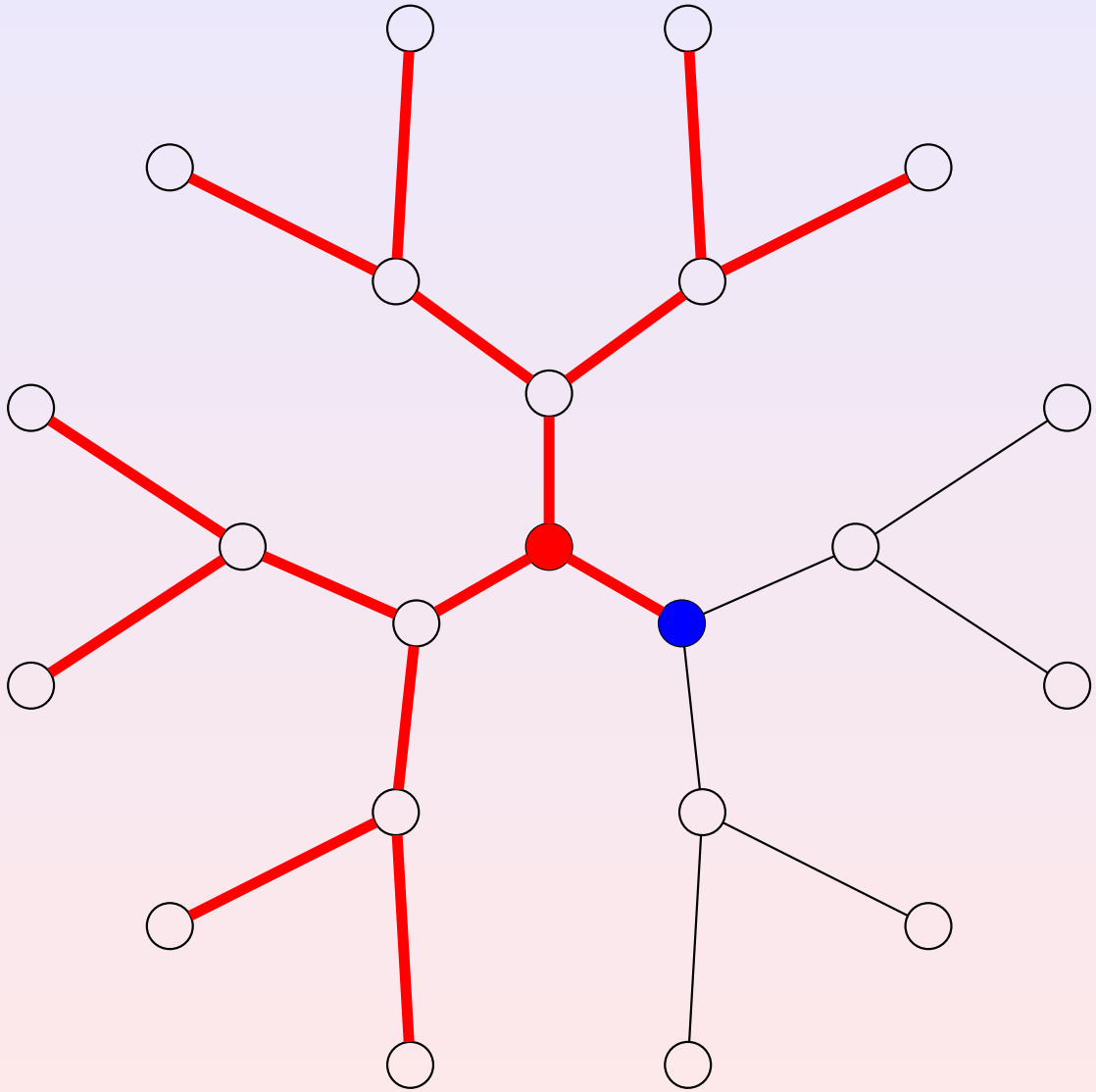
Encerclement non déterministe avec 2 questions



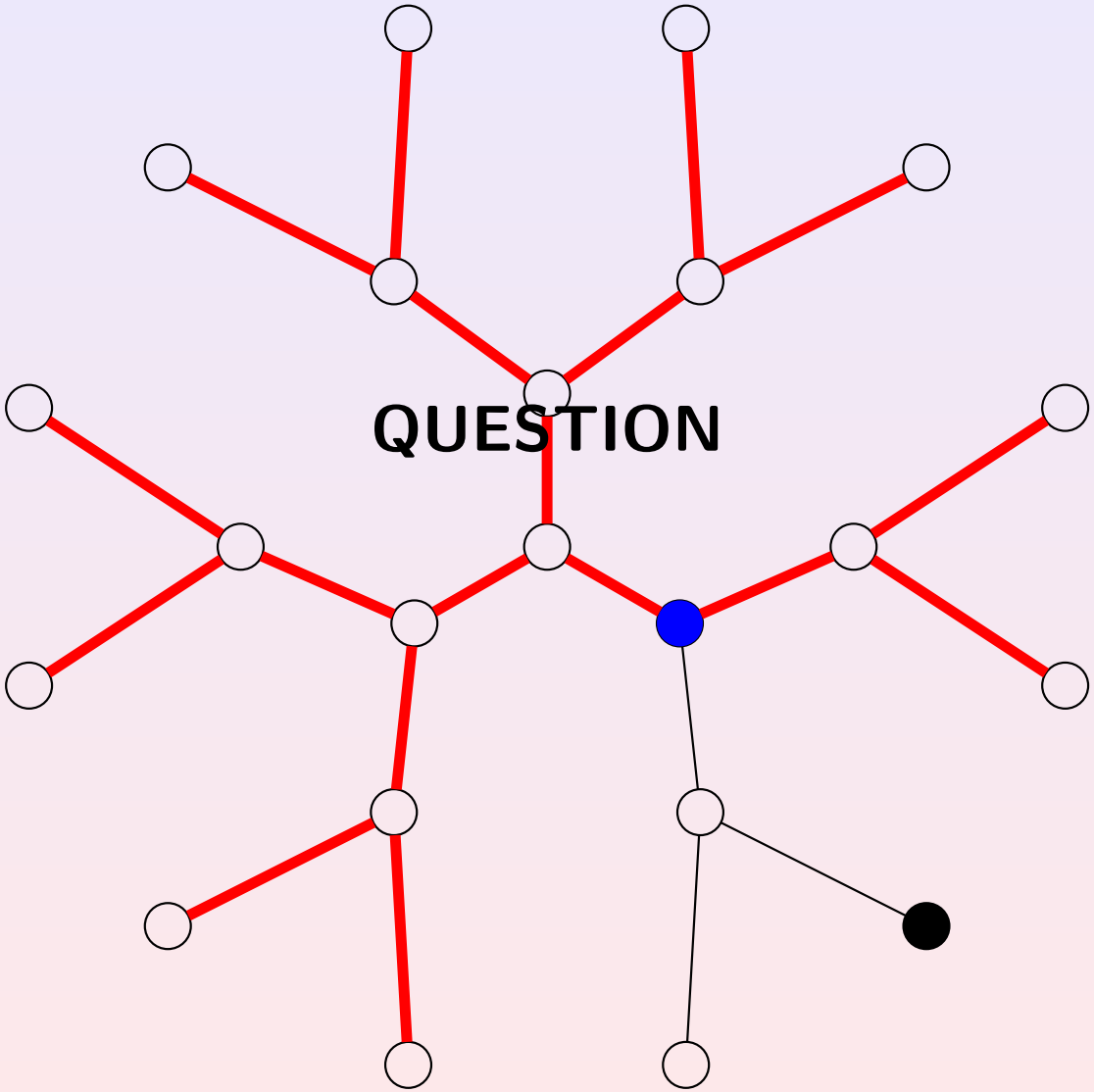
Encerclement non déterministe avec 2 questions



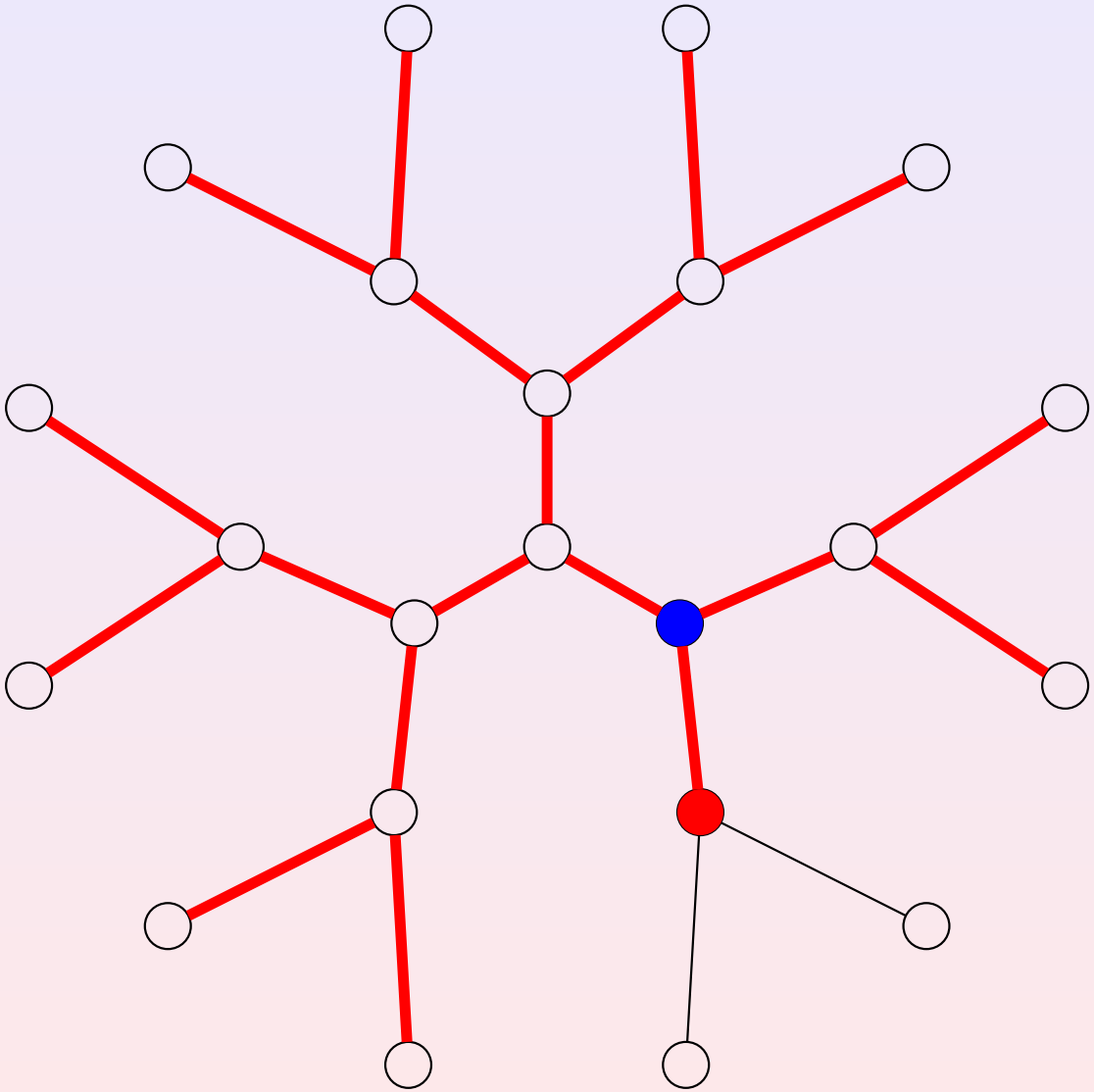
Encerclement non déterministe avec 2 questions



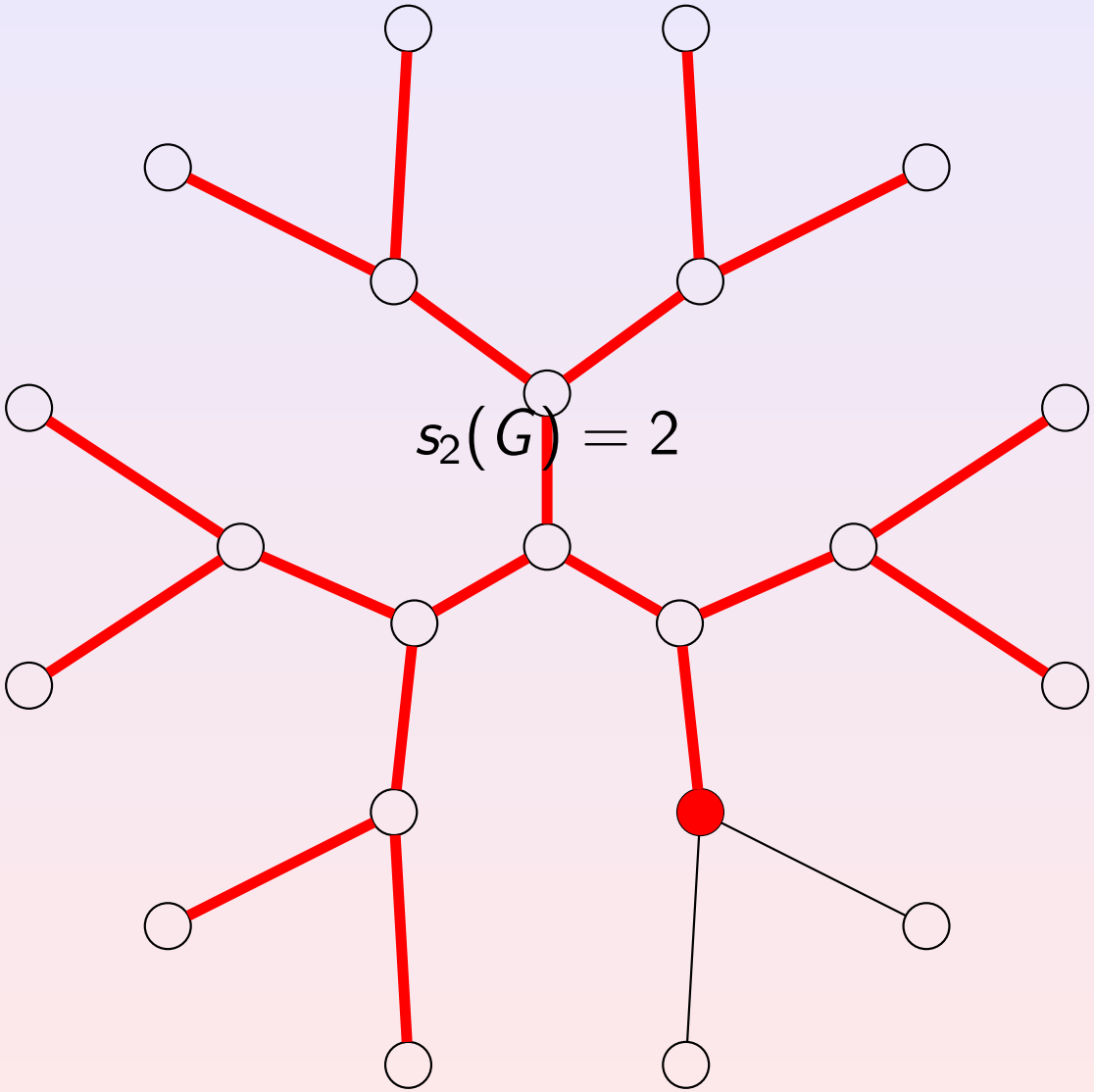
Encerclement non déterministe avec 2 questions



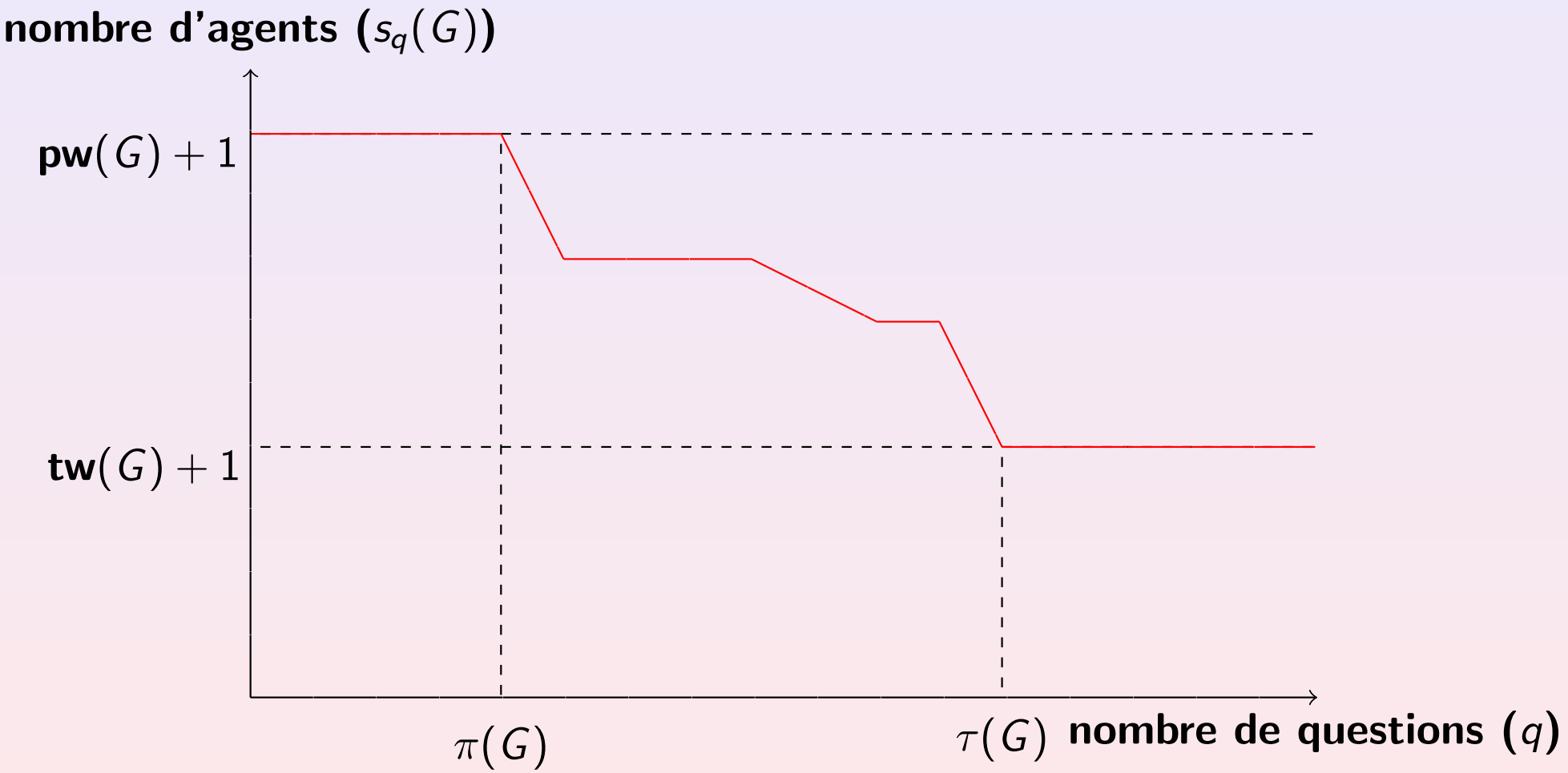
Encerclement non déterministe avec 2 questions



Encerclement non déterministe avec 2 questions



Contrôle du Non déterminisme



Décomposition arborescente branchée

décomposition arborescente q-branchée

- décomposition arborescente enracinée ;
- sommet d'embranchement (avec au moins 2 fils) ;
- chaque chemin de la racine à une feuille contient au plus q sommets d'embranchement ;
- largeur arborescente q-branchée, $tw_q(G)$.

- décomposition linéaire = décomposition arborescente 0-branchée : $\mathbf{pw}(G) = \mathbf{tw}_0(G)$;
- décomposition arborescente = décomposition arborescente ∞-branchée : $\mathbf{tw}(G) = \mathbf{tw}_\infty(G)$;

Résultats (1)

Décomposition arborescente branchée vs. Encerclement non déterministe

Theorème 1 : Pour tout $q \geq 0$, pour tout graphe G, $\mathbf{s}_q(G) = \mathbf{tw}_q(G) + 1$.

NP-Complétude

Décider si $\mathbf{tw}_q(G) \leq k$ est NP-complet pour tout q.

Résultats (2)

Algorithme Exponentiel Exact

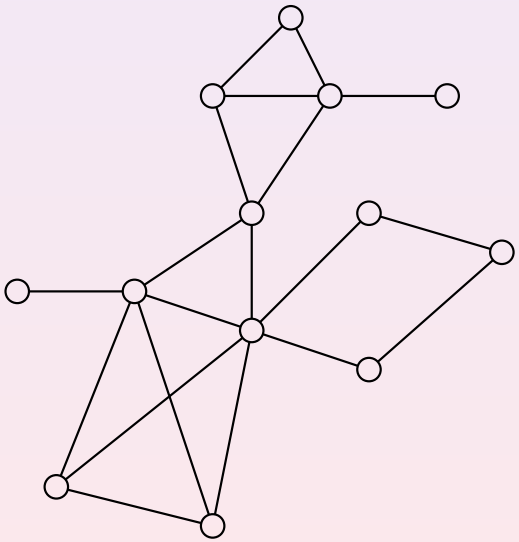
Theorème 2 : Il existe un algorithme qui, pour tout graphe G et pour tout $q \geq 0$, calcule $\mathbf{tw}_q(G)$ et une décomposition arborescente q-branchée optimale de G.

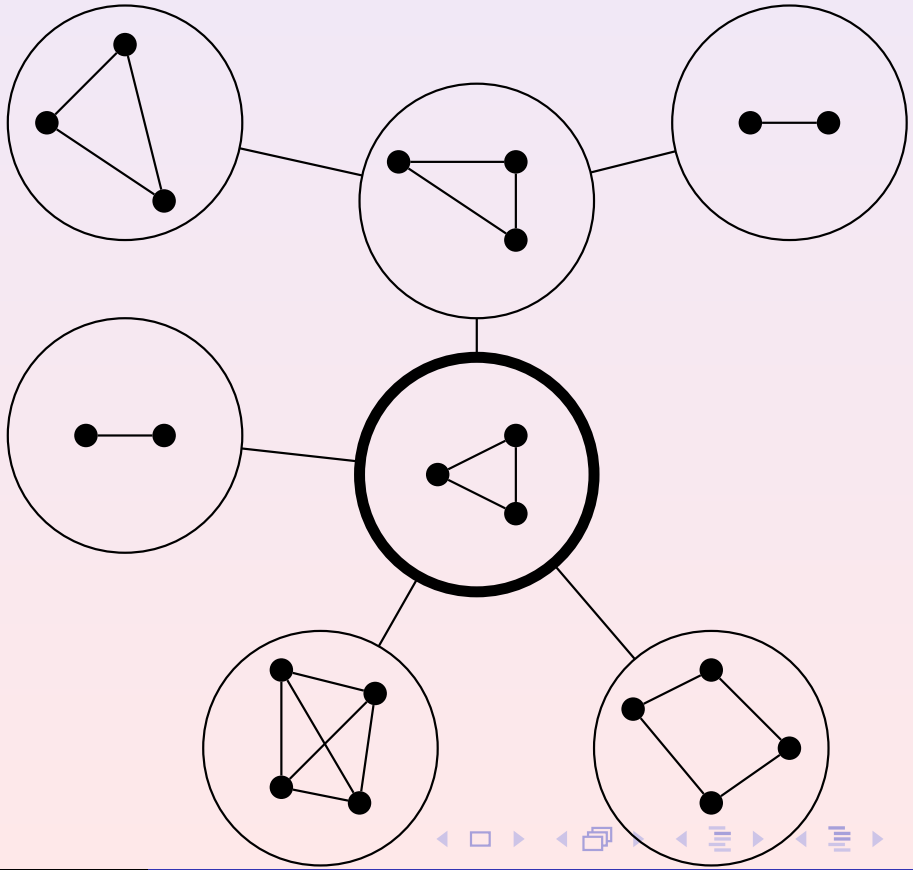
Borne sur le Non déterminisme

Theorème 3 : Pour tout $q \geq 1$, pour tout graphe G, $\mathbf{tw}_{q-1}(G) \leq 2\ \mathbf{tw}_q(G)$.

Théorème 1 : $\mathbf{s}_q(G) = \mathbf{tw}_q(G) + 1$

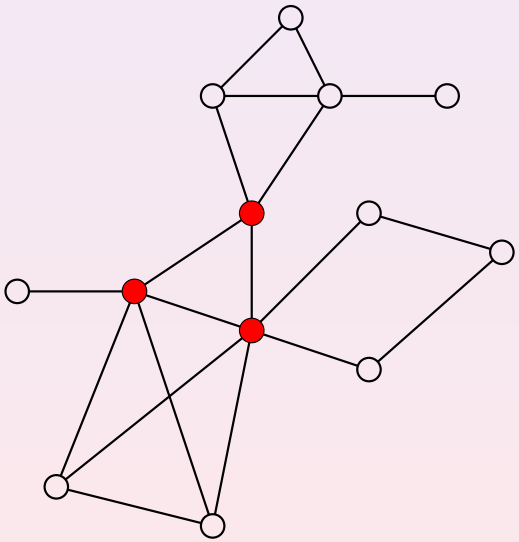
- placer des agents sur les sommets de la racine ;
- poser une question à chaque fois que l'on rencontre un embranchement.

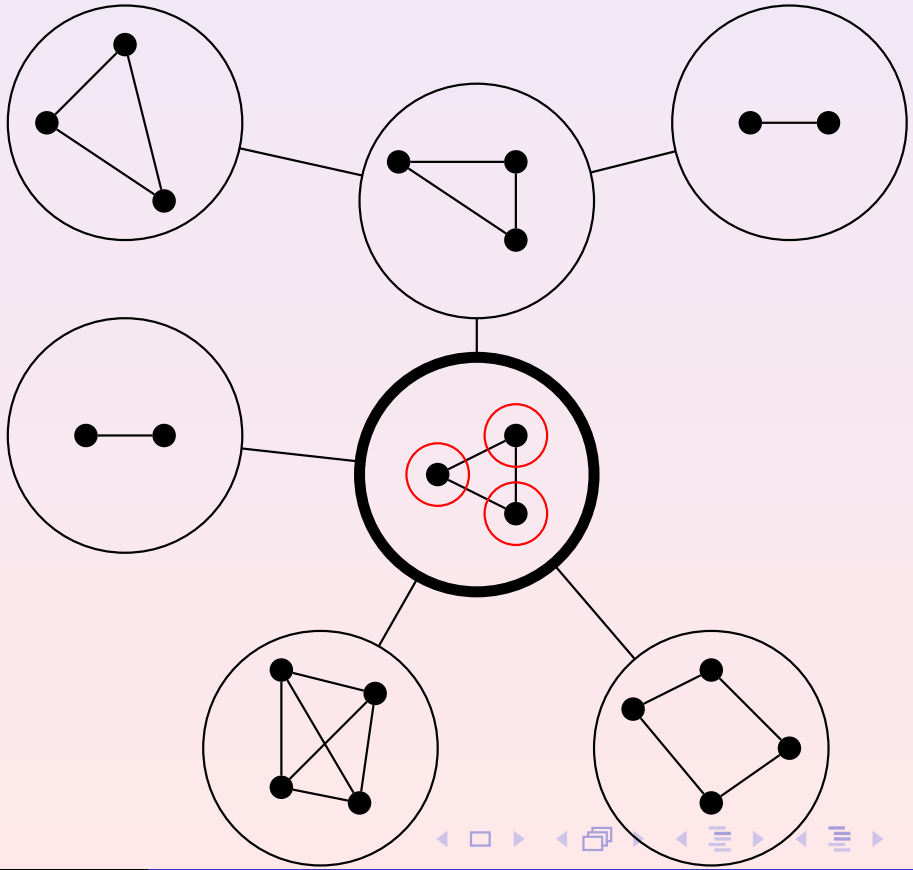




Théorème 1 : $\mathbf{s}_q(G) = \mathbf{tw}_q(G) + 1$

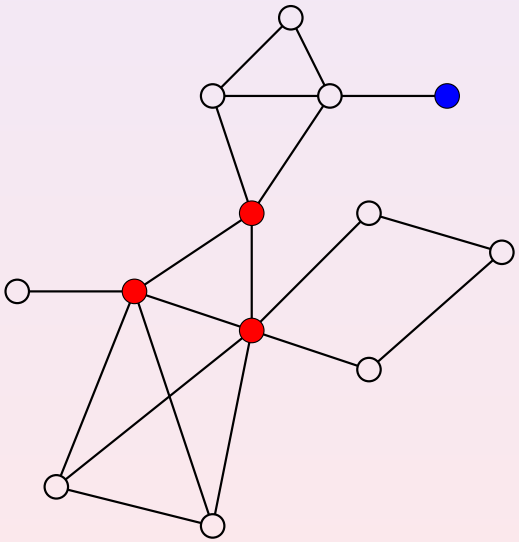
- placer des agents sur les sommets de la racine ;
- poser une question à chaque fois que l'on rencontre un embranchement.

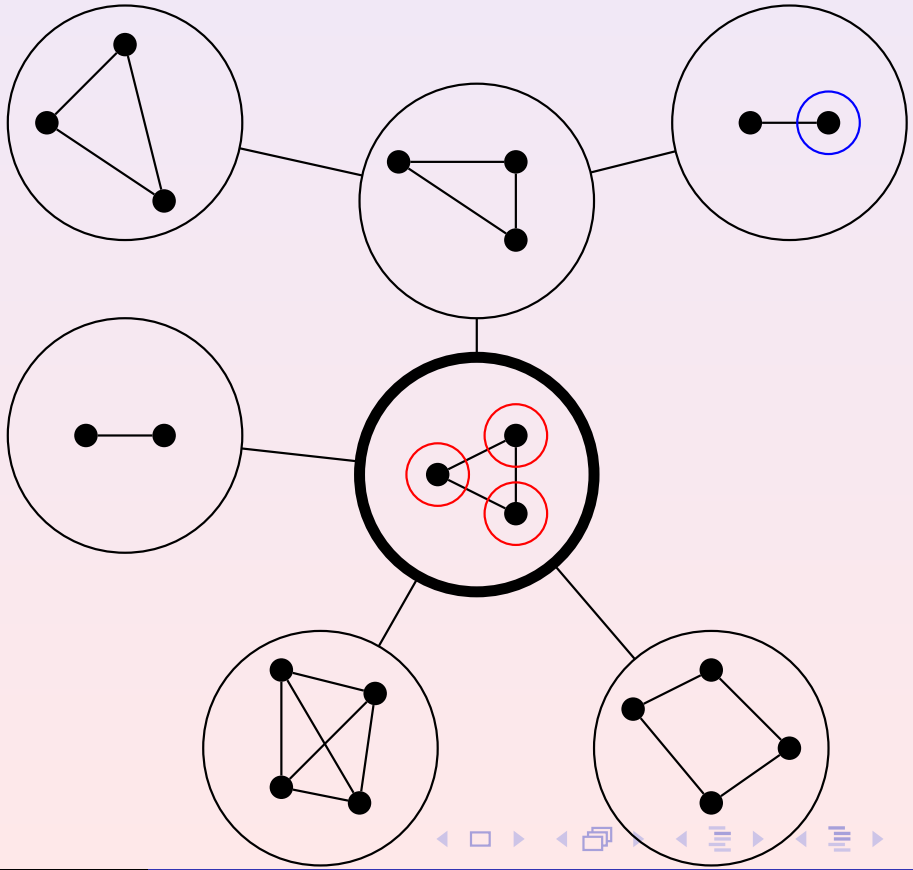




Théorème 1 : $\mathbf{s}_q(G) = \mathbf{tw}_q(G) + 1$

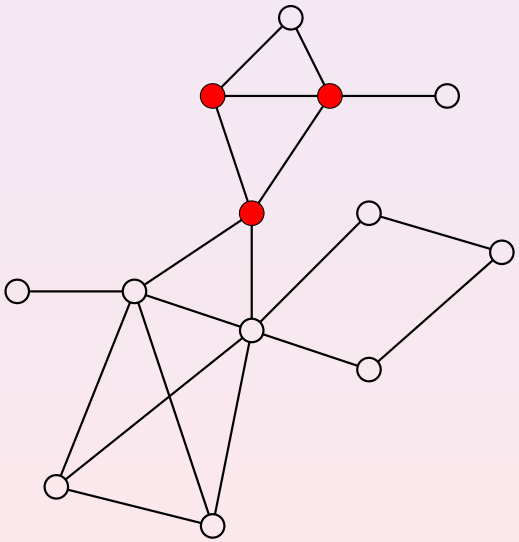
- placer des agents sur les sommets de la racine ;
- poser une question à chaque fois que l'on rencontre un embranchement.

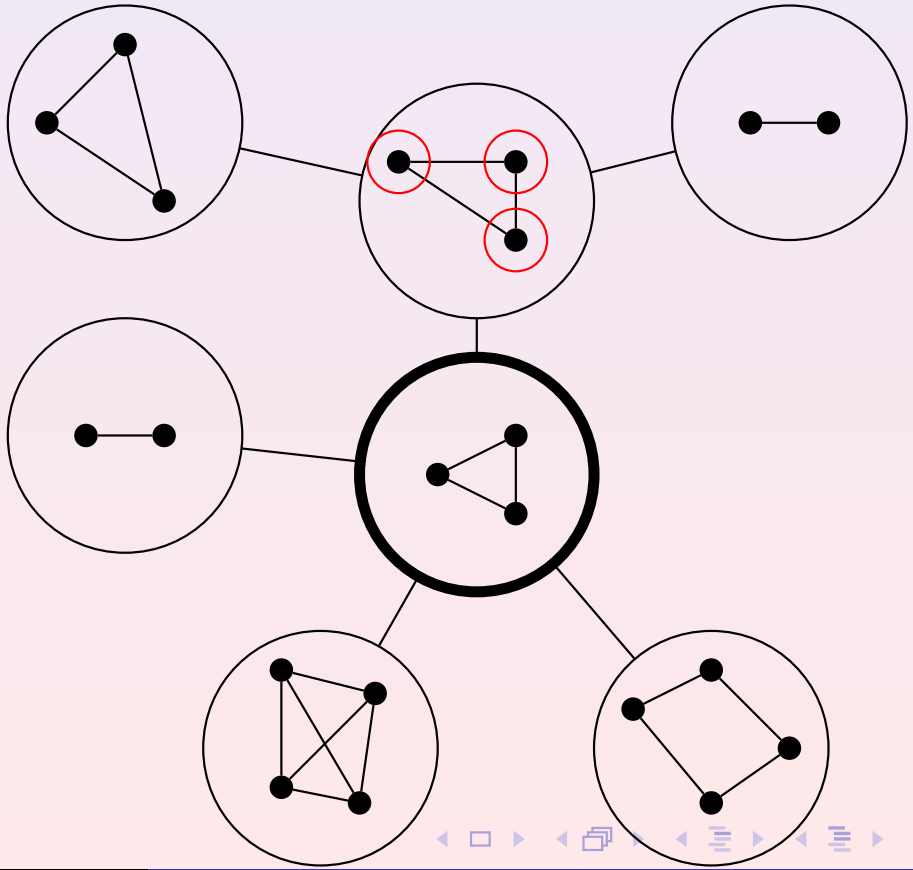




Théorème 1 : $\mathbf{s}_q(G) = \mathbf{tw}_q(G) + 1$

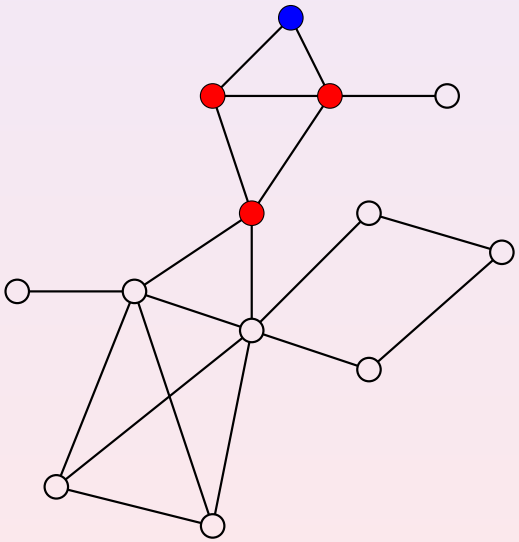
- placer des agents sur les sommets de la racine ;
- poser une question à chaque fois que l'on rencontre un embranchement.

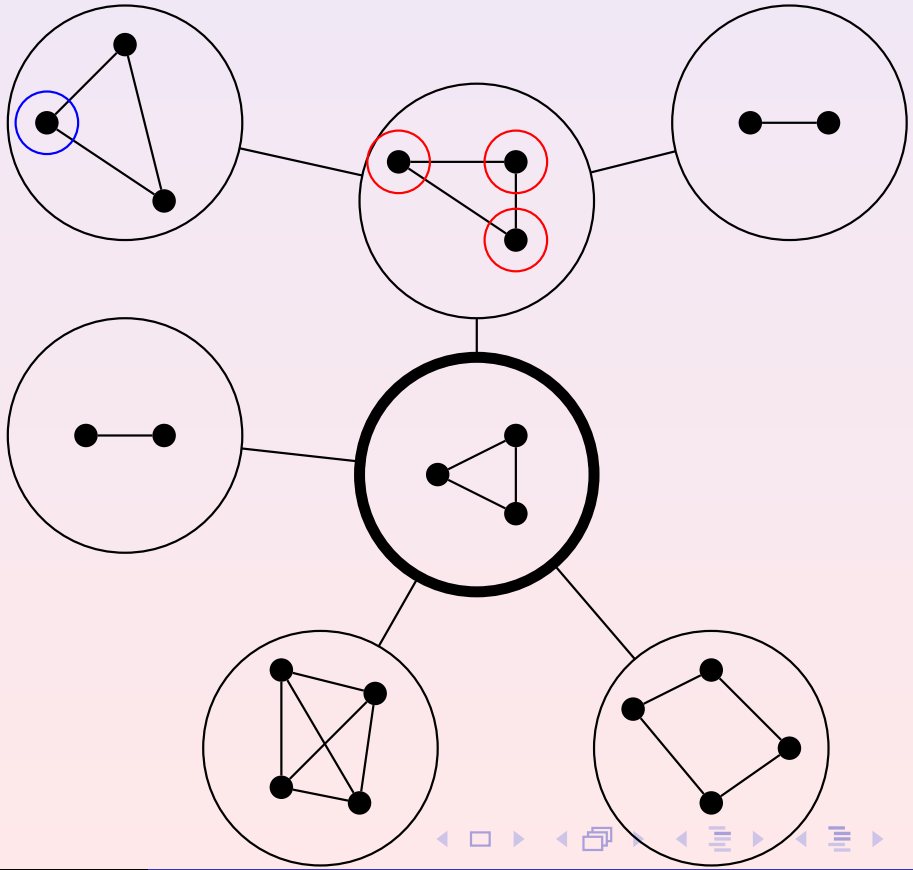




Théorème 1 : $\mathbf{s}_q(G) = \mathbf{tw}_q(G) + 1$

- placer des agents sur les sommets de la racine ;
- poser une question à chaque fois que l'on rencontre un embranchement.





Algorithme exponentiel exact

Théorème 2

Il existe un algorithme qui, pour tout graphe G et pour tout $q \geq 0$, calcule $\mathbf{tw}_q(G)$ et une décomposition arborescente q-branchée optimale de G.

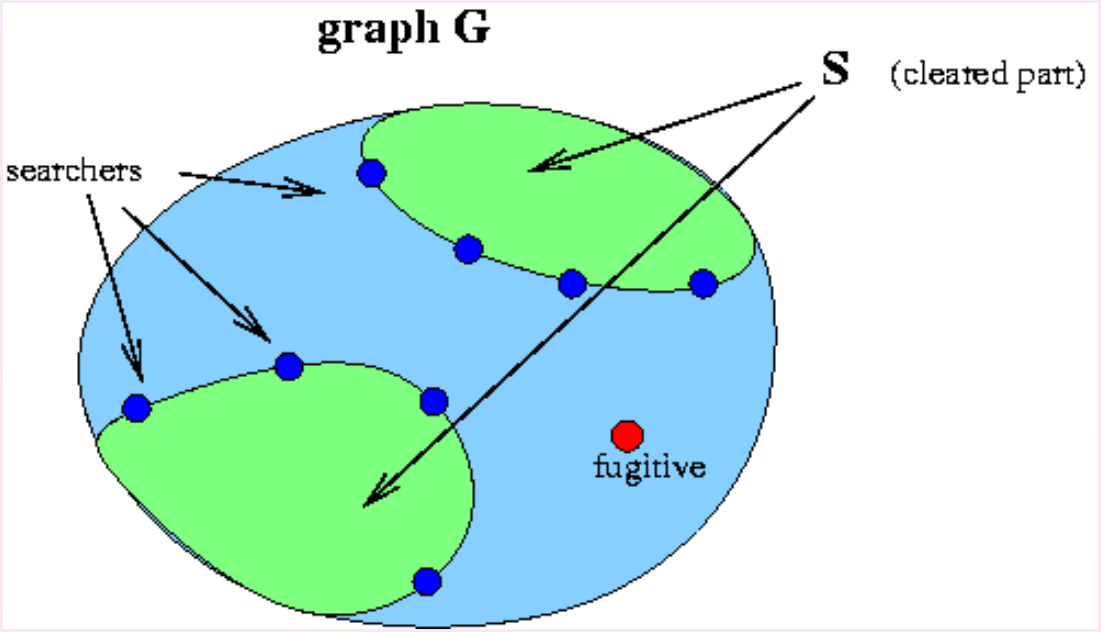
Grace au Théorème 1, il suffit de prouver :

Pour tout k et pour tout graphe G, notre algorithme calcule le minimum q tel qu'il existe une strategie d'encerclement non déterministe de G, utilisant k agents et q requêtes. Notre algorithme calcule de plus cette stratégie.

Algorithme exponentiel exact

Graphe des Configurations H

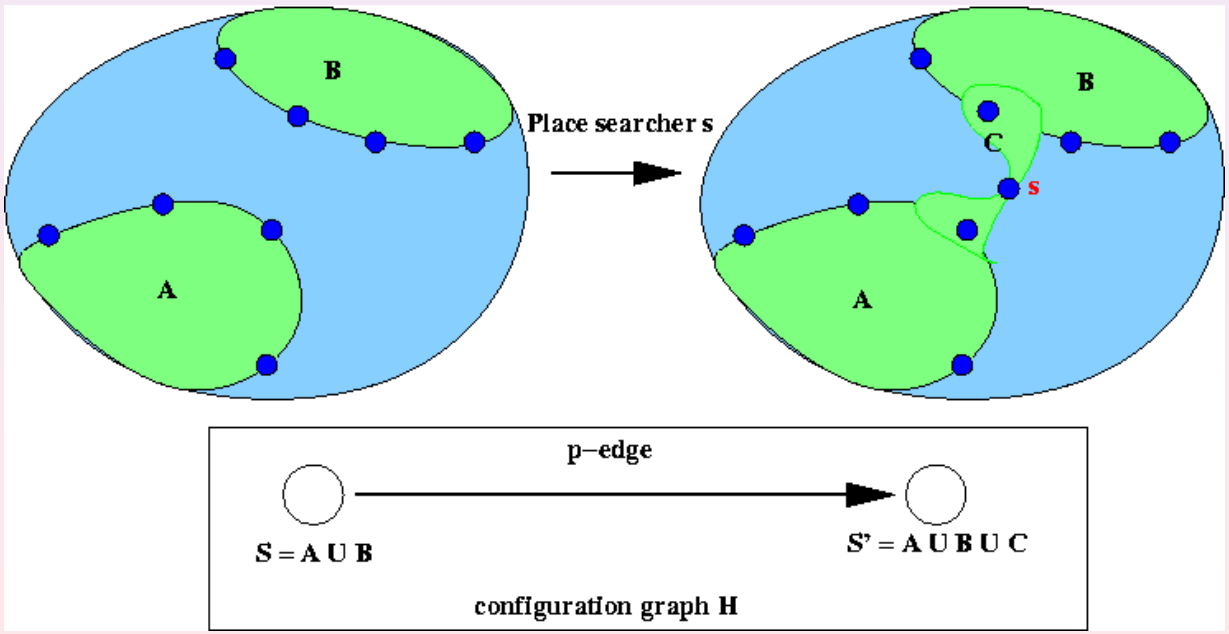
- un **sommet** S de H est un *ensemble de sommets protégés* de G *accessible* par une stratégie utilisant k agents.
  $V(H) = \{S \subseteq V(G)\ t.q.\ |\delta(S)| \leq k\}$;



Algorithme exponentiel exact

Graphe des Configurations H

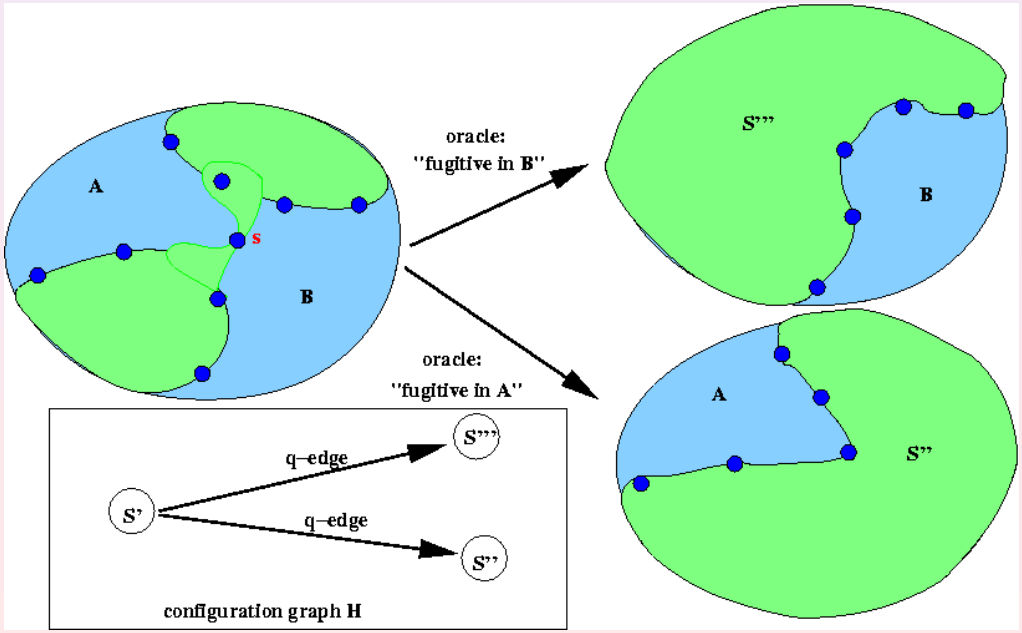
- une **arête dirigée** $\{S, S'\}$ de H est une *opération élémentaire* qui permet d'atteindre S' à partir de S. arête de **placement** et arête de *requête* ;



Algorithme exponentiel exact

Configuration digraph H

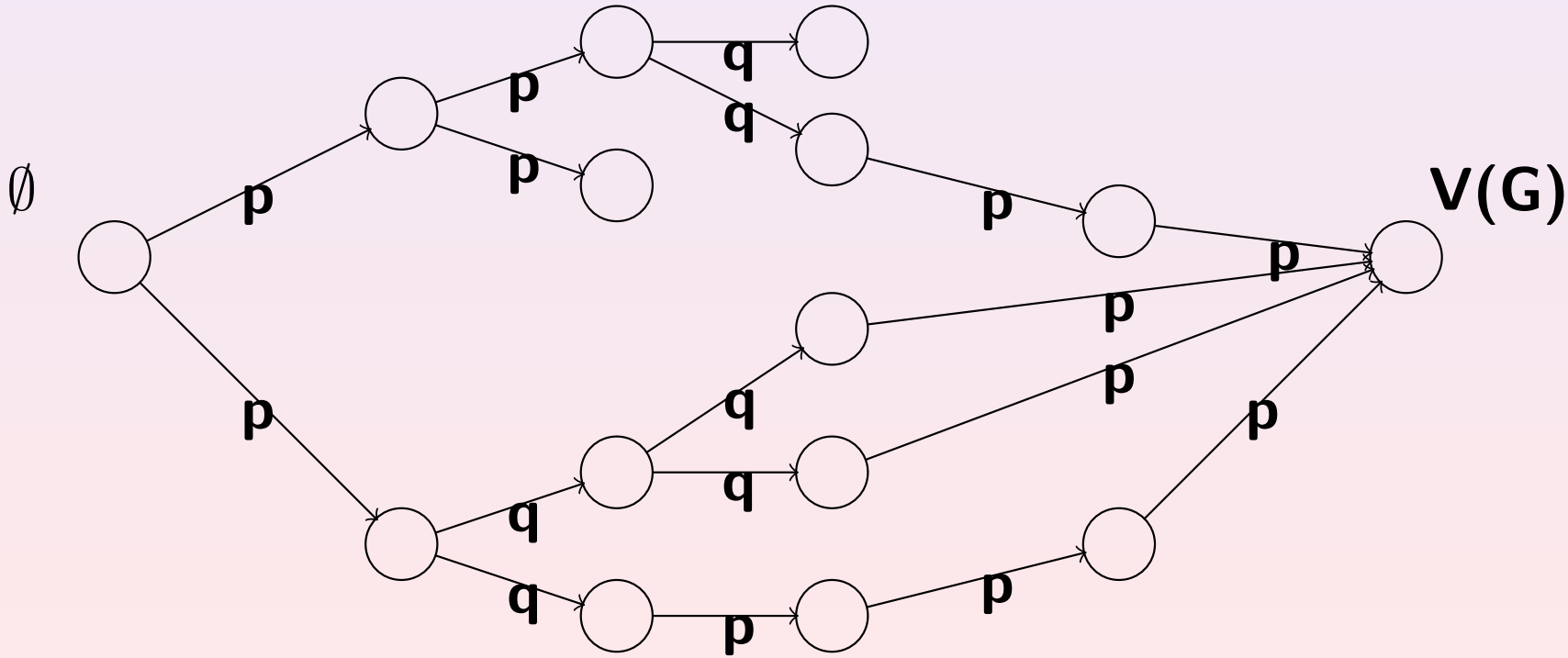
- une **arête dirigée** $\{S, S'\}$ de H est une *opération élémentaire*
  qui permet d'atteindre S' à partir de S.
  arête de *placement* et arête de **requête** ;



Algorithme exponentiel exact

Principe de l'Algorithme

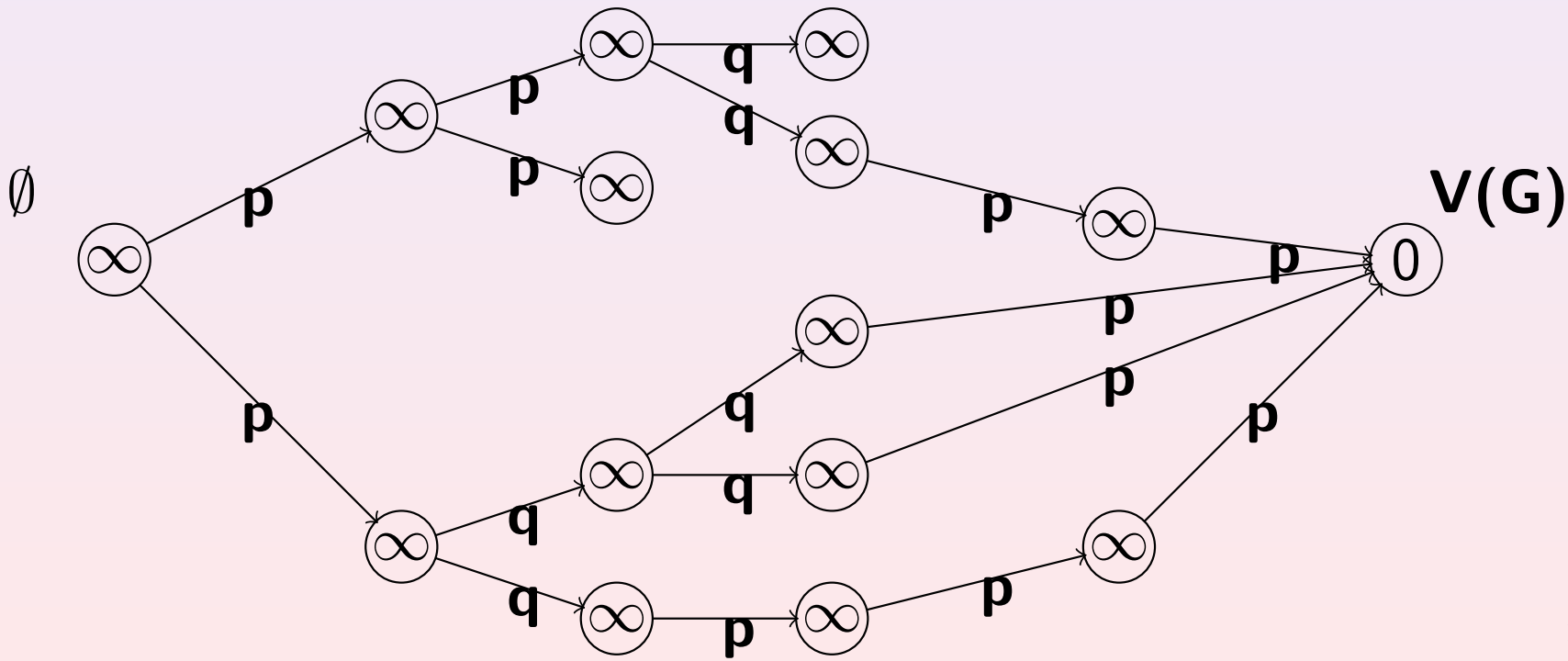
- Trouver un chemin dans H de $S = \emptyset$ à $S = V(G)$.
  Correspondance avec une stratégie d'encerclement.
- Etiquetage des sommets du graphe des configurations.



Algorithme exponentiel exact

Principe de l'Algorithme

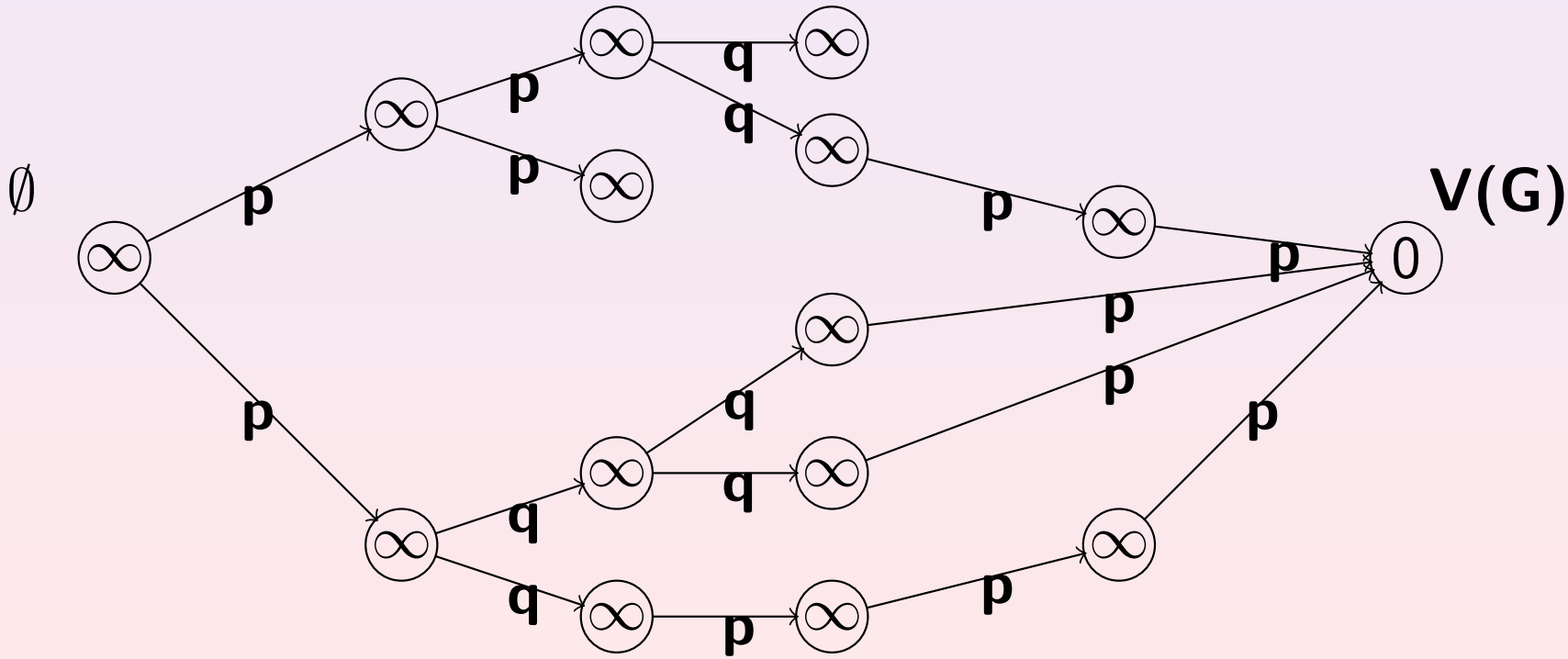
- étiquette(S)= #questions requises pour atteindre $V(G)$;
- Propagation de l'étiquetage : étiquette($V(G)$) = 0 ; étiquette($\emptyset$) = minimum q tel que $\mathbf{s}_q(G) = k$;



Algorithme exponentiel exact

Règles d'étiquetage

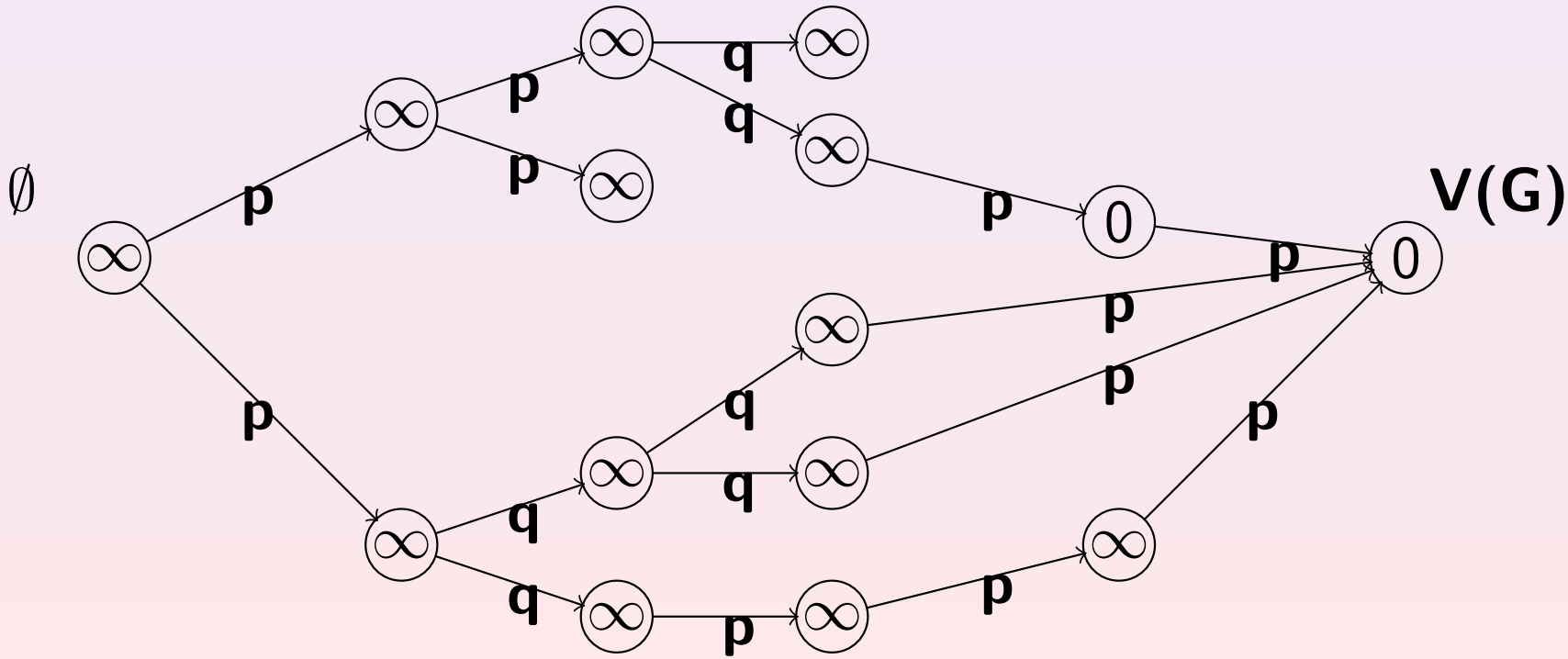
- (S, S') p-edge : $label(S) = \min(label(S), label(S')$;
- (S, S_i) q-edges :
  $label(S) = \min(label(S), 1 + \max label(S_i))$.



Algorithme exponentiel exact

Règles d'étiquetage

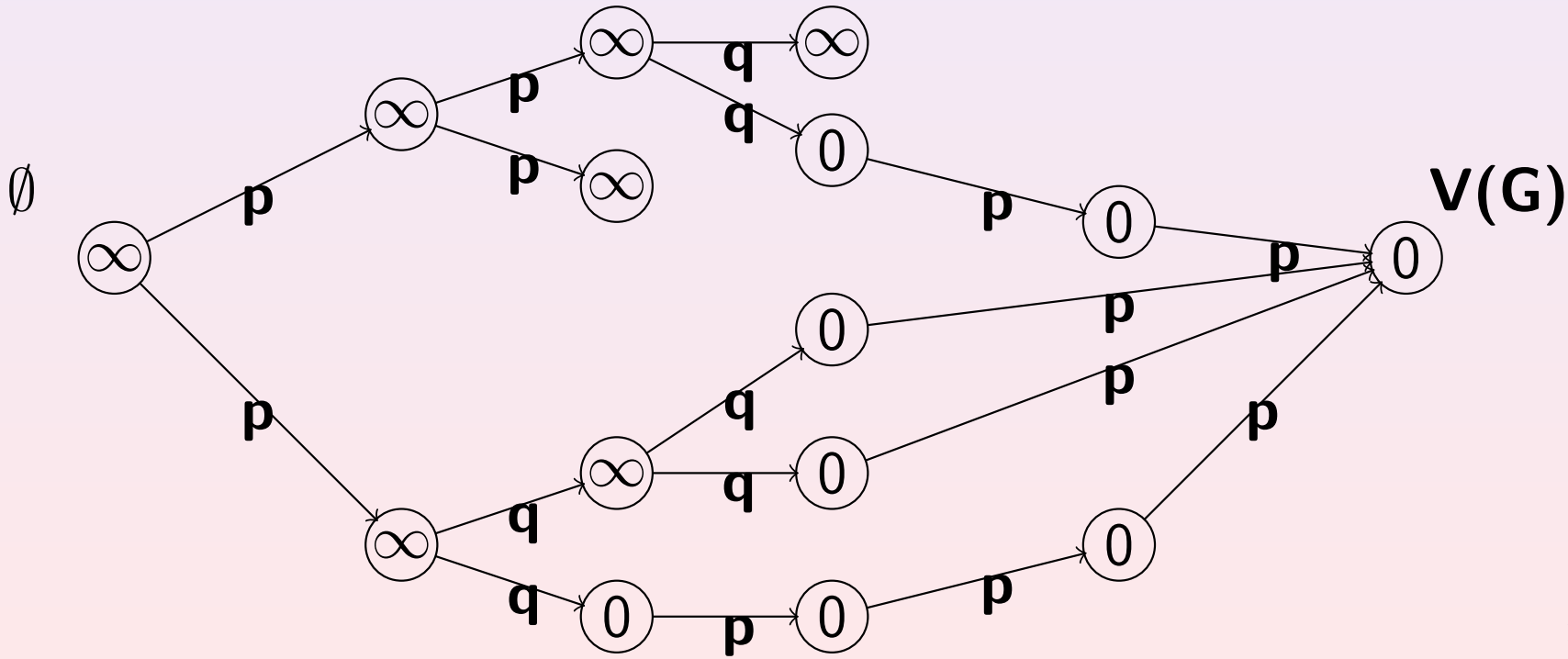
- (S, S') p-edge : $label(S) = \min(label(S), label(S')$;
- (S, S_i) q-edges :
  $label(S) = \min(label(S), 1 + \max label(S_i))$.



Algorithme exponentiel exact

Règles d'étiquetage

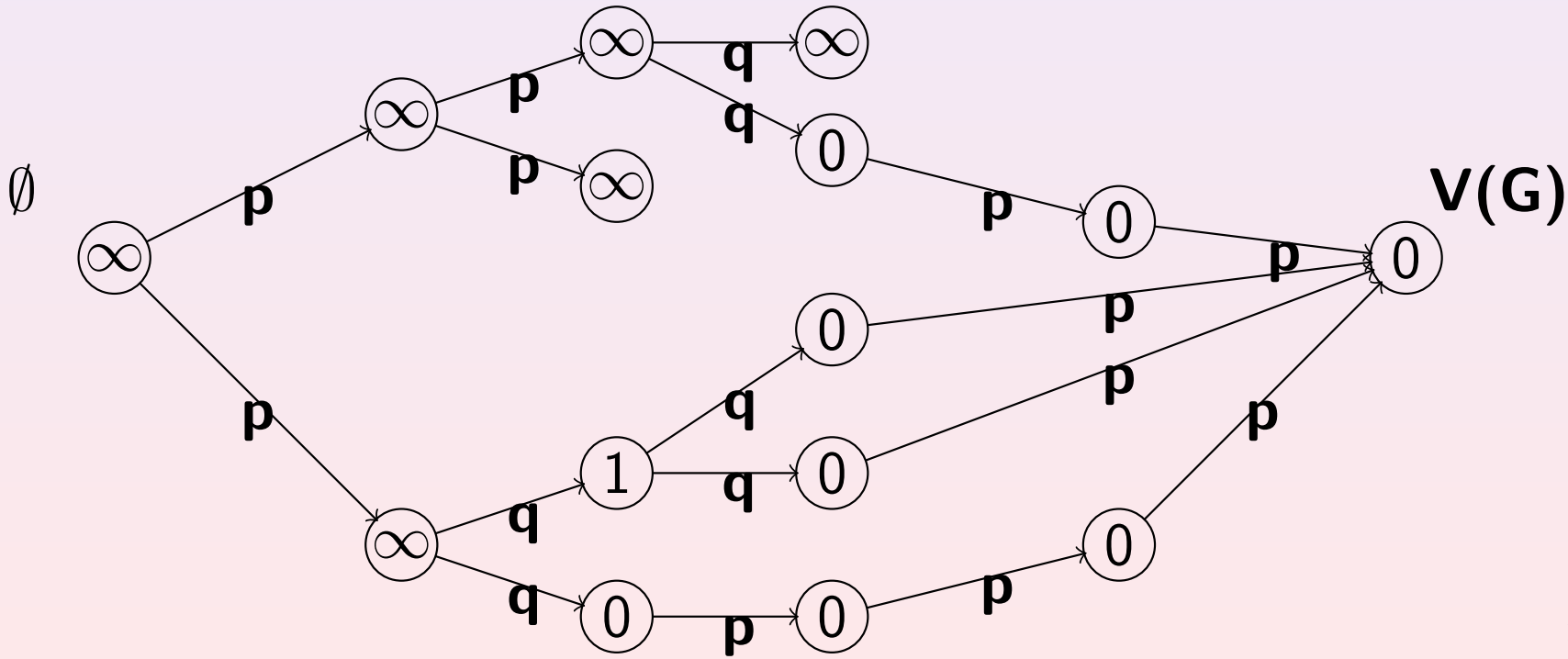
- (S, S') p-edge : $label(S) = \min(label(S), label(S'))$;
- (S, S_i) q-edges :
  $label(S) = \min(label(S), 1 + \max label(S_i))$.



Algorithme exponentiel exact

Règles d'étiquetage

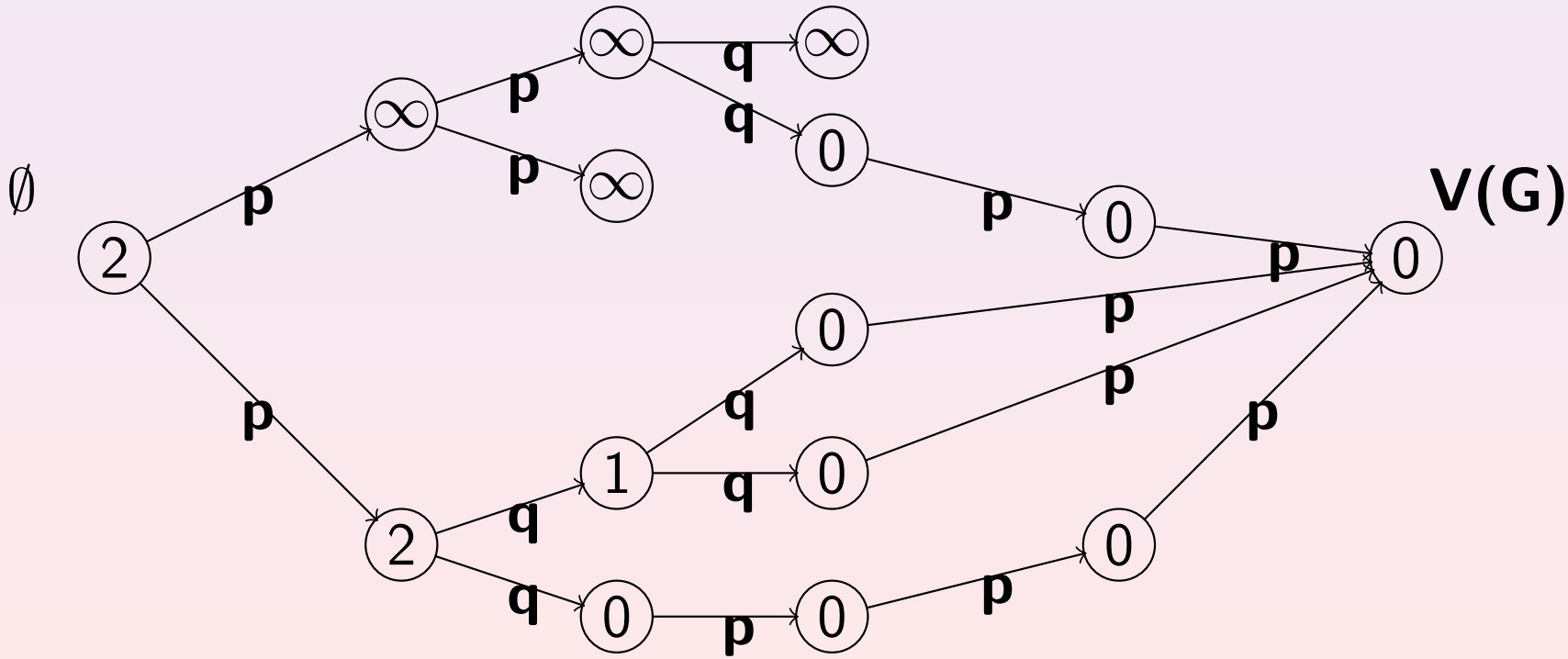
- (S, S') p-edge : $label(S) = \min(label(S), label(S'))$;
- (S, S_i) q-edges :
  $label(S) = \min(label(S), 1 + \max label(S_i))$.



Algorithme exponentiel exact

Règles d'étiquetage

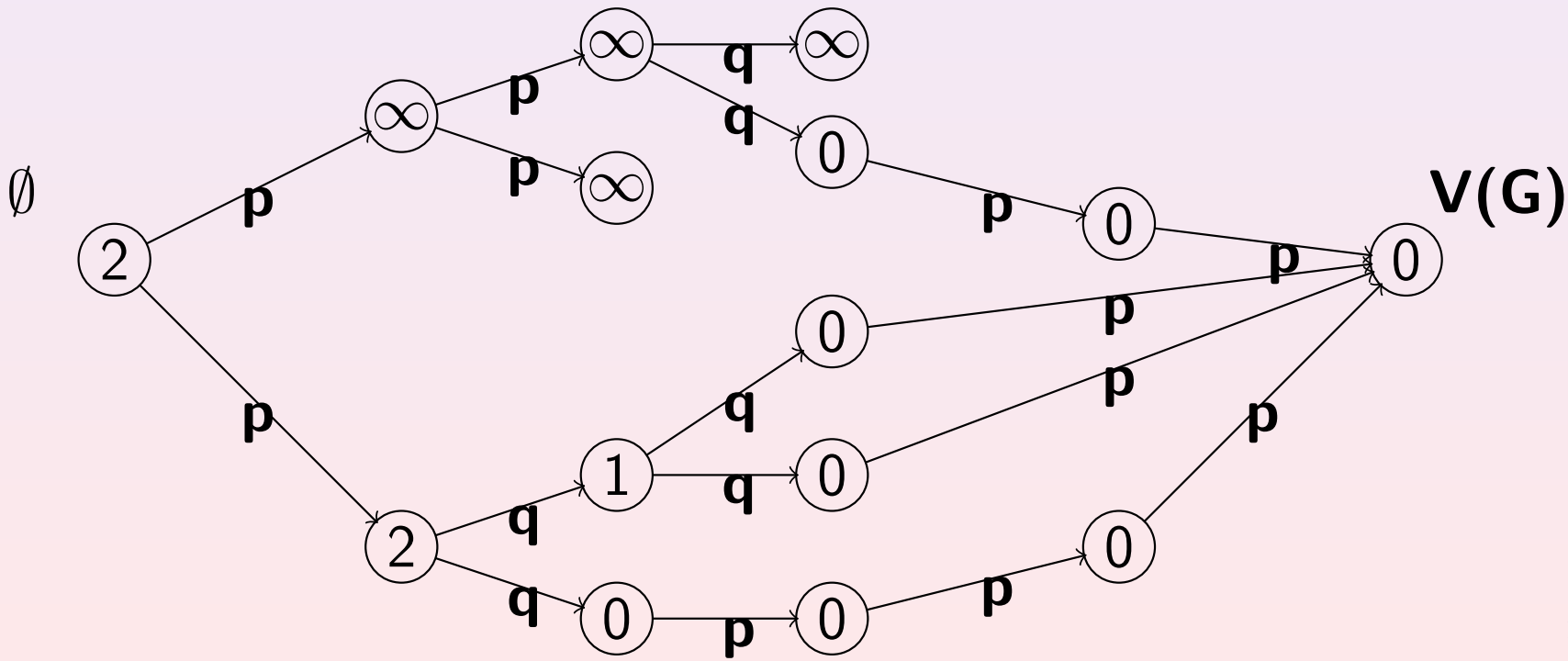
- (S, S') p-edge : $label(S) = \min(label(S), label(S')$;
- (S, S_i) q-edges :
  $label(S) = \min(label(S), 1 + \max label(S_i))$.



Algorithme exponentiel exact

Complexité de l'algorithme

- $|V(H)| = 2^k$, $|E(H)| \leq 2n \times 2^k$; Algorithme en $O(2^n)$;
- Même complexité que le meilleur algorithme connu pour calculer la largeur linéaire (**pw**).



Conclusion et Perspectives

Nouvelle variante des stratégies d'encerclement

- encerclement non déterministe ;
- unifie décompositions linéaire et arborescente.

Perspectives

- rôle de la recontamination ;
- algorithme exact en temps $O(c^n)$ $(\mathbf{c}<\mathbf{2})$;
- algorithme linéaire pour les arbres.

Dernière minute

L'encerclement nondéterministe est monotone.[Mazoit,N]

Conclusion et Perspectives

Nouvelle variante des stratégies d'encerclement

- encerclement non déterministe ;
- unifie décompositions linéaire et arborescente.

Perspectives

- rôle de la recontamination ;
- algorithme exact en temps $O(c^n)$ $(\mathbf{c}<\mathbf{2})$;
- algorithme linéaire pour les arbres.

Dernière minute

L'encerclement nondéterministe est monotone.[Mazoit,N]