Leçon N° n: décomposition arborescente, séparateur, routage et distance

Cyril Gavoille

LaBRI, Université de Bordeaux, France

Mars 2007

Problème

Constuire des structures de données pour les réseaux, *distribuées* sur les nœuds, supportant des requêtes comme la distance ou le routage.

Problème

Constuire des structures de données pour les réseaux, *distribuées* sur les nœuds, supportant des requêtes comme la distance ou le routage.

Definition

Soit P une propriété de graphe. Un **P-schéma d'étiquetage** pour une famille de graphes $\mathcal{F}$ est une paire (L, f) de fonctions vérifiant, pour tout graphe $G = (V, E) \in \mathcal{F}$,

- pour tout $x \in V$, $L(x, G)$ est une étiquette binaire
- pour tous $x, y \in V$, $f(L(x, G), L(y, G)) = P(x, y, G)$

Problème

Constuire des structures de données pour les réseaux, *distribuées* sur les nœuds, supportant des requêtes comme la distance ou le routage.

Definition

Soit P une propriété de graphe. Un **P-schéma d'étiquetage** pour une famille de graphes $\mathcal{F}$ est une paire (L, f) de fonctions vérifiant, pour tout graphe $G = (V, E) \in \mathcal{F}$,

- pour tout $x \in V$, $L(x, G)$ est une étiquette binaire
- pour tous $x, y \in V$, $f(L(x, G), L(y, G)) = P(x, y, G)$

Objectif : construire, pour une propriété P donnée, des P-schémas d'étiquetage utilisant les étiquettes les plus courtes possibles.

Exemples

Soient $\lambda_1 = L(x, G)$ et $\lambda_2 = L(y, G)$ deux étiquettes

- Distance entre deux sommets
  $f(\lambda_1, \lambda_2) = P(x, y, G) =$ distance entre x et y dans G.

Exemples

Soient $\lambda_1 = L(x, G)$ et $\lambda_2 = L(y, G)$ deux étiquettes

- Distance entre deux sommets
  $f(\lambda_1, \lambda_2) = P(x, y, G) =$ distance entre x et y dans G.
- Routage à base d'adresse
  $f(\lambda_1, \lambda_2) = P(x, y, G) =$ arête (ou numéro de port) incidente à x sur un plus courts chemin entre x et y dans G.

Observations

Si (L, f) étiquetage de distance avec $O(k \log n)$ bits/étiquette et la complexité en temps de f est τ, alors il existe une structure de données (classique) d'**espace** $O(kn)$ supportant les requêtes de distance en temps $O(\tau)$.

Observations

Si (L, f) étiquetage de distance avec $O(k \log n)$ bits/étiquette et la complexité en temps de f est τ, alors il existe une structure de données (classique) d'**espace** $O(kn)$ supportant les requêtes de distance en temps $O(\tau)$.

Si (L, f) routage avec $O(k \log n)$ bits/étiquette et la complexité en temps de f est τ, alors il existe une structure de données (classique) d'**espace** $O(kn)$ permettant l'extraction de plus courts chemins de longueur ℓ en temps $O(\ell \cdot \tau)$.

Observations

Si (L, f) étiquetage de distance avec $O(k \log n)$ bits/étiquette et la complexité en temps de f est τ, alors il existe une structure de données (classique) d'**espace** $O(kn)$ supportant les requêtes de distance en temps $O(\tau)$.

Si (L, f) routage avec $O(k \log n)$ bits/étiquette et la complexité en temps de f est τ, alors il existe une structure de données (classique) d'**espace** $O(kn)$ permettant l'extraction de plus courts chemins de longueur ℓ en temps $O(\ell \cdot \tau)$.

A priori, moins d'informations dans les étiquettes de routage que dans les étiquettes de distance ... la route étant calculée avec plus de requêtes (une seule pour la distance).

Cas d'école : distances dans les arbres

Definition

$S \subseteq V(G)$ est un **demi-séparateur** pour G si les composantes connexes de $G \setminus S$ sont de taille au plus $|V(G)|/2$.

Cas d'école : distances dans les arbres

Definition

$S \subseteq V(G)$ est un **demi-séparateur** pour G si les composantes connexes de $G \setminus S$ sont de taille au plus $|V(G)|/2$.

Tout arbre possède un demi-séparateur de taille 1.

[Faire la preuve ...]

Décomposition hiéarchique

Soit A un arbre à n sommets.

Constuire une décomposition ($\Rightarrow$ arbre H) induite par l'appel récursif du demi-séparateur. Hauteur de H, au plus $\log_2 n$.

Décomposition hiéarchique

Soit A un arbre à n sommets.

Constuire une décomposition ($\Rightarrow$ arbre H) induite par l'appel récursif du demi-séparateur. Hauteur de H, au plus $\log_2 n$.

À la fin tout sommet de l'arbre A est un demi-séparateur, il se retrouve dans H.

Étiquetage (de distance)

$x \mapsto s(x) = \langle x_1, \ldots, x_h = x \rangle$ chemin dans H de la racine à x.

Étiquetage (de distance)

$x \mapsto s(x) = \langle x_1, \ldots, x_h = x \rangle$ chemin dans H de la racine à x.

$x \mapsto d(x) = \langle d_A(x, x_1), \ldots, d_A(x, x_h) = 0 \rangle$ vecteur de distance de x aux séparateurs ancêtres de x.

Étiquetage (de distance)

$x \mapsto s(x) = \langle x_1, \ldots, x_h = x \rangle$ chemin dans H de la racine à x.

$x \mapsto d(x) = \langle d_A(x, x_1), \ldots, d_A(x, x_h) = 0 \rangle$ vecteur de distance de x aux séparateurs ancêtres de x.

Étiquetage : $L(x, A) = (s(x), d(x))$.

Étiquetage (de distance)

$x \mapsto s(x) = \langle x_1, \ldots, x_h = x \rangle$ chemin dans H de la racine à x.

$x \mapsto d(x) = \langle d_A(x, x_1), \ldots, d_A(x, x_h) = 0 \rangle$ vecteur de distance de x aux séparateurs ancêtres de x.

Étiquetage : $L(x, A) = (s(x), d(x))$.

Distance entre $(s(x), d(x))$ et $(s(x'), d(x'))$:

- calculer i la longueur du plus grand préfixe commun entre $s(x)$ et $s(x')$
- retourner $d(x)[i] + d(x')[i]$.

... et le routage

$x \mapsto s(x) =$ ... (idem)

$x \mapsto p(x) = \langle p(x, x_1), \ldots, p(x, x_h) \rangle$ où $p(x, x_i)$ est l'arête pour aller de x à x_i.

Étiquetage : $L(x, A) = (s(x), p(x))$.

Routage entre $(s(x), d(x))$ et $(s(x'), d(x'))$:

- calculer i ... (idem)
- router le message sur le port $p(x, x_i)$

Résultats

Dans les deux cas : $O(\log^2 n)$ bits/étiquette

Résultats

Dans les deux cas : $O(\log^2 n)$ bits/étiquette

Notes :

- Pour la distance, il n'est pas possible de faire moins que $\frac{1}{8}\log^2 n$ bits.

Résultats

Dans les deux cas : $O(\log^2 n)$ bits/étiquette

Notes :

- Pour la distance, il n'est pas possible de faire moins que $\frac{1}{8}\log^2 n$ bits.
- Pour le routage, il est possible de faire $O(\log^2 n/\log\log n)$, et pas possible de faire moins que $\frac{1}{7}\log^2 n/\log\log n$ bits.

Résultats (bis)

Il est possible de coder le champs $s(x)$ par une autre structure de données taille $O(\log n)$ bits (au lieu du $O(\log^2 n)$ naïf) supportant "longueur du plus grand préfixe commun" en temps $O(1)$ dans le model RAM.

Résultats (bis)

Il est possible de coder le champs $s(x)$ par une autre structure de données taille $O(\log n)$ bits (au lieu du $O(\log^2 n)$ naïf) supportant "longueur du plus grand préfixe commun" en temps $O(1)$ dans le model RAM.

Donc le temps de calcul est constant pour la distance et le routage.

Résultats (bis)

Il est possible de coder le champs $s(x)$ par une autre structure de données taille $O(\log n)$ bits (au lieu du $O(\log^2 n)$ naïf) supportant "longueur du plus grand préfixe commun" en temps $O(1)$ dans le model RAM.

Donc le temps de calcul est constant pour la distance et le routage.

Autre application : approximation de la distance, $\varepsilon > 0$

$$d_A(x,y) \leqslant f(L(x,A), L(y,A)) \leqslant (1+\varepsilon) \cdot d_A(x,y)$$

Codage du champs $d(x)$ avec $O(\log(\varepsilon^{-1} \log n))$ bits.

Résultats (bis)

Il est possible de coder le champs $s(x)$ par une autre structure de données taille $O(\log n)$ bits (au lieu du $O(\log^2 n)$ naïf) supportant "longueur du plus grand préfixe commun" en temps $O(1)$ dans le model RAM.

Donc le temps de calcul est constant pour la distance et le routage.

Autre application : approximation de la distance, $\varepsilon > 0$

$$d_A(x,y) \leqslant f(L(x,A), L(y,A)) \leqslant (1+\varepsilon) \cdot d_A(x,y)$$

Codage du champs $d(x)$ avec $O(\log(\varepsilon^{-1} \log n))$ bits.

$\Rightarrow$ on peut approximer la distance avec un facteur $1 + o(1)$ avec des étiquettes de taille $O(\log n \cdot \log\log n)$, et c'est la meilleure borne possible si $\varepsilon \leqslant 6/\log n$.

... et la **treewidth** !

Largeur arborescente : rappel

Definition

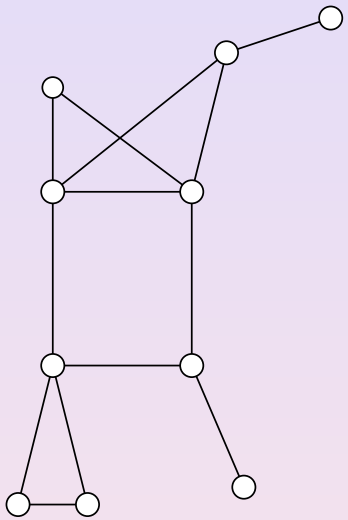
Une **décomposition arborescente** d'un graphe G est un arbre T dont les sommets, appelés **sacs**, sont des sous-ensembles de sommets de G, tel que :

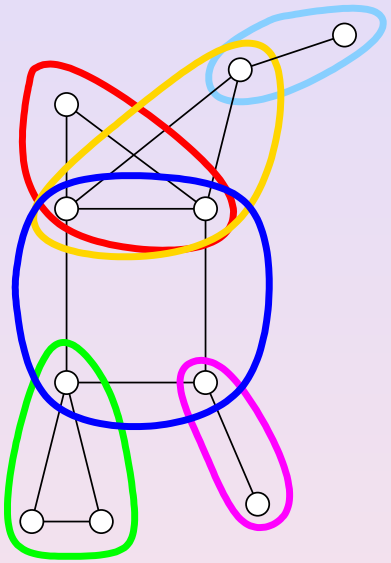
1. tout sommet est contenu dans au moins un sac ;
2. toute arête a ses deux extrémité dans un même sac ;
3. l'ensemble des sacs contenant un sommet donné de G induit un arbre dans T.

La **largeur de** T est le nombre de sommet -1 du plus gros sac de T et la **largeur arborescente** de G (*treewidth*) est la plus petite largeur de ses décompositions arborescentes.

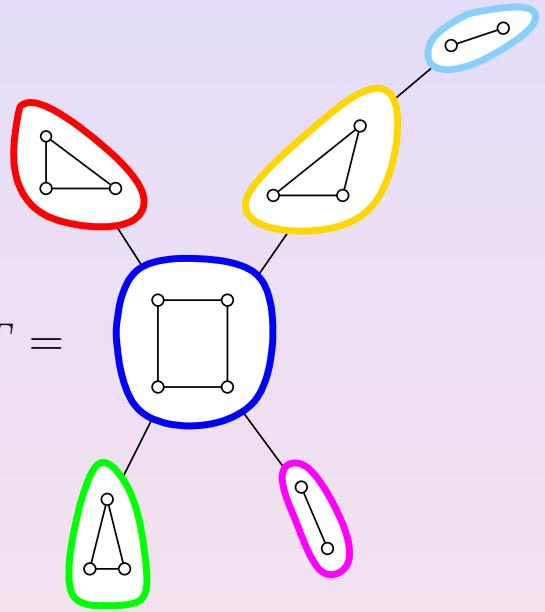
Les arbres sont de treewidth 1, si il y a au moins une arête.

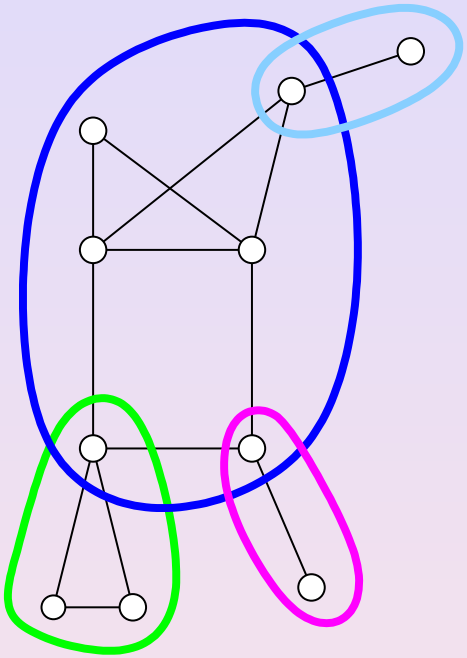
$G =$

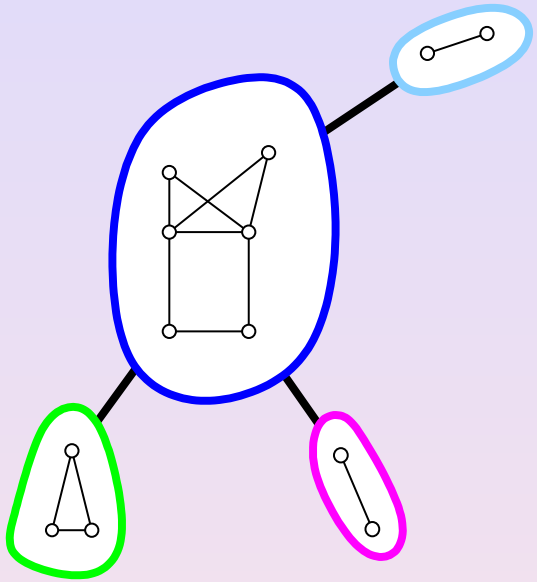


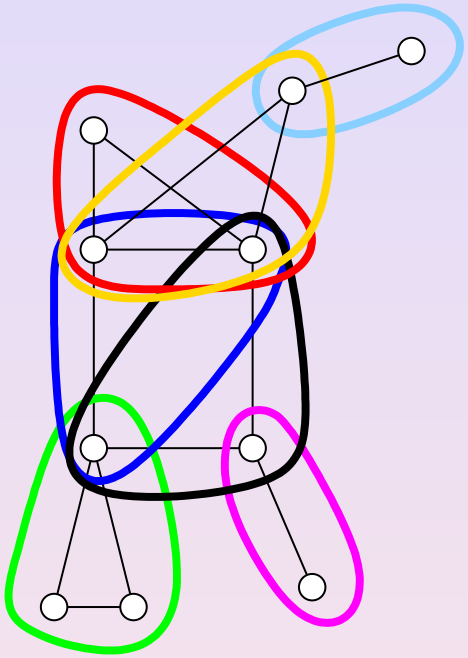


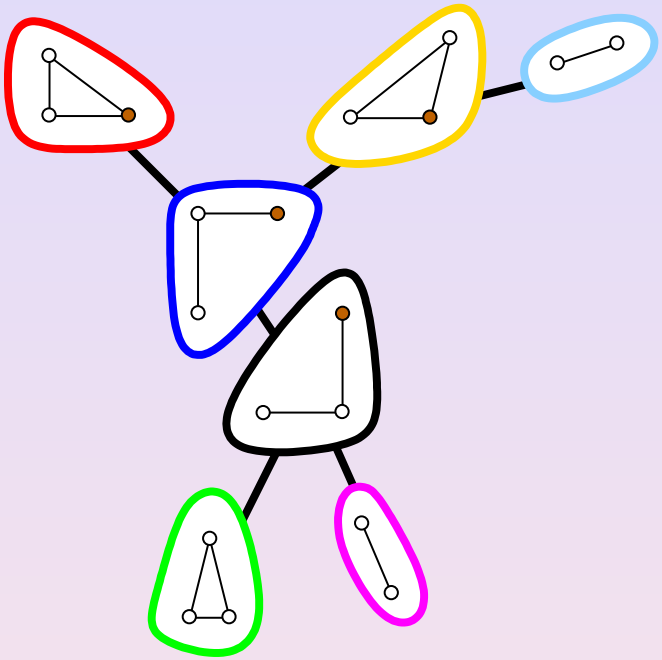
$T =$











Largeur arborescente v.s. mineur

Les graphes de largeur arborescente k excluent K_{k+2} comme mineur.

Largeur arborescente v.s. mineur

Les graphes de largeur arborescente k excluent K_{k+2} comme mineur.

Les graphes qui exclus un graphe planaire H comme mineur sont de largeur arborescente au plus $f(H)$. La meilleure borne connue est $f(H) \leqslant 2^{2^{2^{2^{2|V(H)|+4|E(H)|}}}}$.

Largeur arborescente v.s. mineur

Les graphes de largeur arborescente k excluent K_{k+2} comme mineur.

Les graphes qui exclus un graphe planaire H comme mineur sont de largeur arborescente au plus $f(H)$. La meilleure borne connue est $f(H) \leqslant 20^{2(2|V(H)|+4|E(H)|)^5}$.

Pour $f(K_4) \leqslant 20^{2(2\cdot 4+4\cdot 6)^5} \approx 20^{2^{26}} = 2^{67\,108\,864}$, mais en fait la valeur optimale est $f(K_4) = 2$. Il est conjecturé que la vraie borne est $f(H) = O(|V(H)|^2 \log |V(H)|)$.

Distance

Idée : faire comme pour les arbres en notant que le schéma n'utilise pas vraiment la définition de "décomposition arborescente" ...

Distance

Idée : faire comme pour les arbres en notant que le schéma n'utilise pas vraiment la définition de "décomposition arborescente" ...

La propriété qui est utile :

Definition

Un graphe est k-**séparable** si tout sous-graphe possède un demi-séparateur de taille au plus k.

Note : donc les arbres sont 1-séparables.

Étiquetage pour les graphes k-séparables

Étiquetage pour les graphes k-séparables

1. Construire une décomposition hiéarchique de G ($\Rightarrow$ arbre H de hauteur $\log n$)

Étiquetage pour les graphes k-séparables

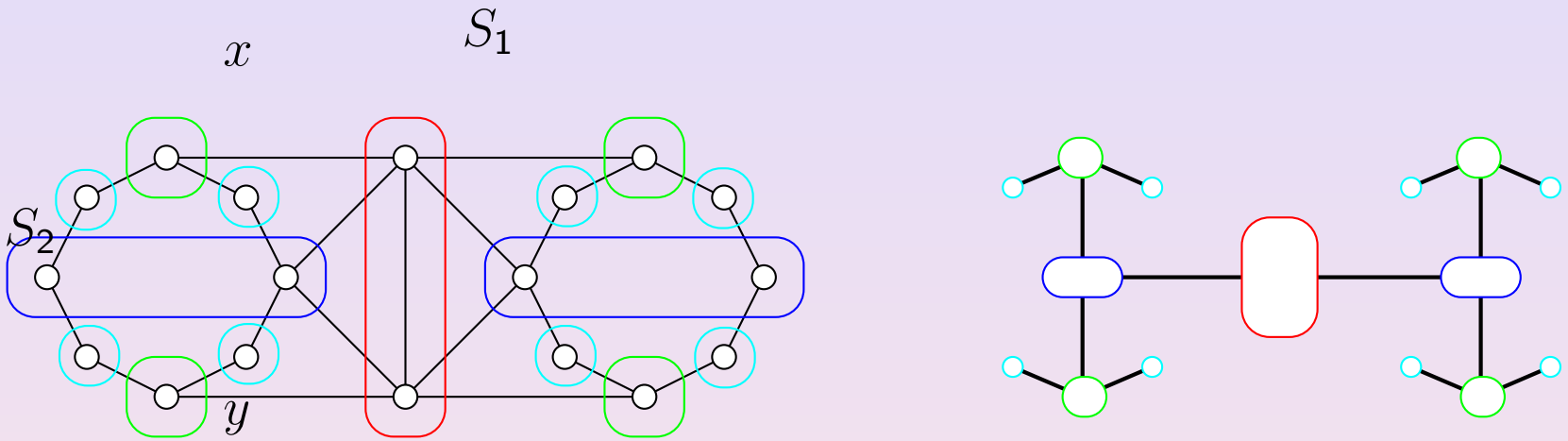
1. Construire une décomposition hiéarchique de G ($\Rightarrow$ arbre H de hauteur $\log n$)
2. $x \mapsto S(x) = \langle S_1, \ldots, S_h \rangle$ où S_i est le i^{e} séparateur de la composante contenant x dans H

Étiquetage pour les graphes k-séparables

1. Construire une décomposition hiéarchique de G ($\Rightarrow$ arbre H de hauteur $\log n$)
2. $x \mapsto S(x) = \langle S_1, \ldots, S_h \rangle$ où S_i est le i^{e} séparateur de la composante contenant x dans H
3. $x \mapsto d(x) = \langle \vec{d}(x, S_1), \ldots, \vec{d}(x, S_h) \rangle$ où $\vec{d}(x, S_i)$ est la liste des distances entre x et tout $s \in S_i$ dans G

Étiquetage pour les graphes k-séparables

1. Construire une décomposition hiéarchique de G ($\Rightarrow$ arbre H de hauteur $\log n$)
2. $x \mapsto S(x) = \langle S_1, \ldots, S_h \rangle$ où S_i est le i^{e} séparateur de la composante contenant x dans H
3. $x \mapsto d(x) = \langle \vec{d}(x, S_1), \ldots, \vec{d}(x, S_h) \rangle$ où $\vec{d}(x, S_i)$ est la liste des distances entre x et tout $s \in S_i$ dans G
4. Calcul de la distance : ...


x
S_1
S_2
y

Calcul de la distance

Attention ! Dans H, entre deux séparateurs S et S' il n'y a pas d'arête (et donc de chemin) si S et S' ne sont pas ancêtre l'un de l'autre. Mais, à partir de $k \geqslant 2$, des arêtes peuvent exister entre un S et n'importe quels de ces ancêtres dans H.

Calcul de la distance

Attention ! Dans H, entre deux séparateurs S et S' il n'y a pas d'arête (et donc de chemin) si S et S' ne sont pas ancêtre l'un de l'autre. Mais, à partir de $k \geqslant 2$, des arêtes peuvent exister entre un S et n'importe quels de ces ancêtres dans H.

Distance :

$$d_G(x,y) \;=\; \min_{S_i \in S(x) \cap S(y)} \left\{ \min_{s \in S_i} \{ d(x,s) + d(y,s) \} \right\}$$

Résultats

Tout graphe k-séparable possède un étiquetage de distance (et de routage !) avec $O(k \log^2 n)$ bits/étiquette.

Résultats

Tout graphe k-séparable possède un étiquetage de distance (et de routage !) avec $O(k \log^2 n)$ bits/étiquette.

Note : pour le routage il faut stocker la distance **et** le numéro de l'arête.

Résultats

Tout graphe k-séparable possède un étiquetage de distance (et de routage !) avec $O(k \log^2 n)$ bits/étiquette.

Note : pour le routage il faut stocker la distance **et** le numéro de l'arête.

Problème ouvert : peut-on faire $o(k \log^2 n)$ bit/étiquettes pour le routage ? (pour $k = 1$, oui ; les planaires-extérieures aussi. Pour $k > 1$ on ne sait pas faire sans évaluer la distance)

Treewidth v.s. *Séparabilité*

Tout graphe de largeur arborescente k est $(k+1)$-séparable.

($\Leftrightarrow$ si G n'a pas de demi-séparateur de taille s, alors sa *treewidth* est au moins s.)

Treewidth v.s. *Séparabilité*

Tout graphe de largeur arborescente k est $(k+1)$-séparable.

($\Leftrightarrow$ si G n'a pas de demi-séparateur de taille s, alors sa *treewidth* est au moins s.)

Tout graphe s-séparable est de largeur arborescente au plus $3s-1$.

Treewidth v.s. *Séparabilité*

Tout graphe de largeur arborescente k est $(k+1)$-séparable.

($\Leftrightarrow$ si G n'a pas de demi-séparateur de taille s, alors sa *treewidth* est au moins s.)

Tout graphe s-séparable est de largeur arborescente au plus $3s-1$.

Dit autrement, si G est de *treewidth* k et de *séparabilité* s, alors

$$\boxed{\tfrac{1}{3}(k+1) \leqslant s \leqslant k+1}$$

Les graphes de treewidth k sont $(k+1)$-séparables

[Faire la preuve ...]

Est-ce $k+1$ la meilleure borne possible ?