Résultats de complexité pour le jeu d'acquisition

Valentin Gledel

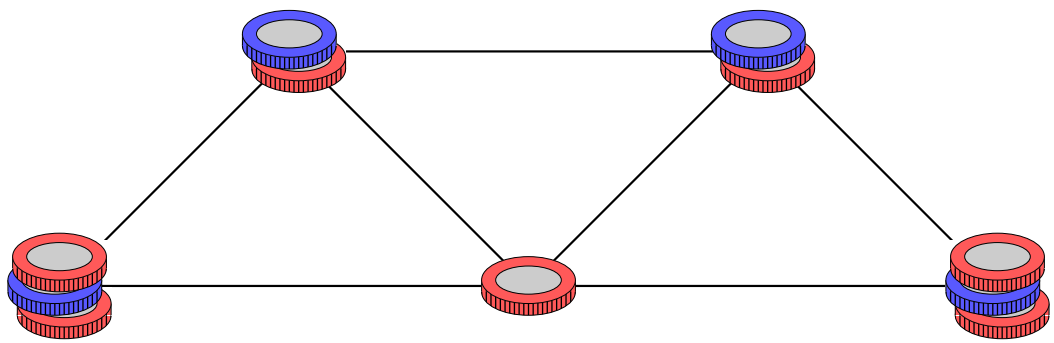
JGA 2019, Bruxelles

En collaboration avec Guillaume Bagan, Marc Heinrich et Fionn Mc Inerney

Le jeu d'acquisition

Jeu d'acquisition (Lampert et Slater, 1995)

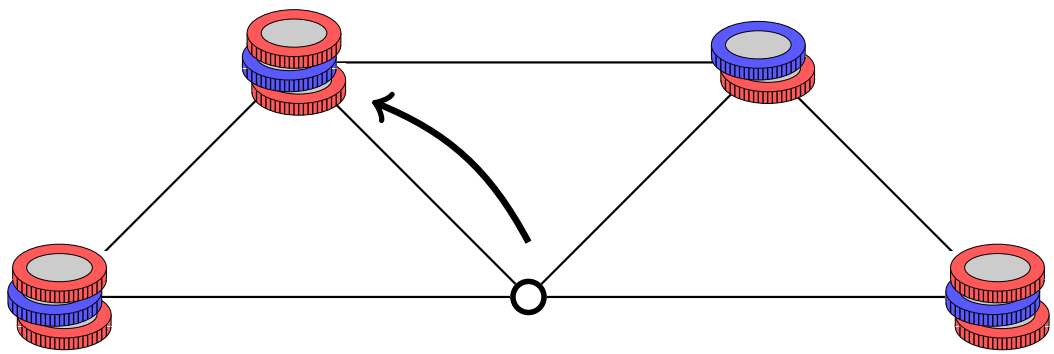
- Jeu à un joueur sur un graphe $G = (V, E)$.
- Il y a une pile de jeton sur chaque sommet.
- A chaque tour le joueur déplace une pile vers une pile adjacente et plus grande.
- Le but du jeu est de minimiser le nombre de piles final.



Le jeu d'acquisition

Jeu d'acquisition (Lampert et Slater, 1995)

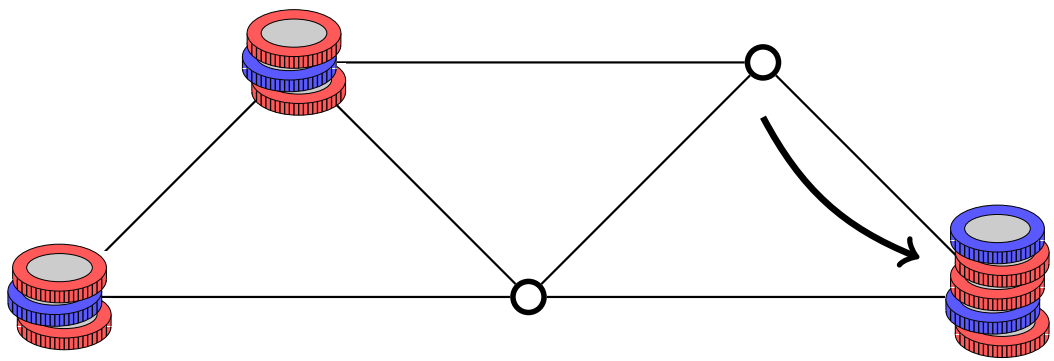
- Jeu à un joueur sur un graphe $G = (V, E)$.
- Il y a une pile de jeton sur chaque sommet.
- A chaque tour le joueur déplace une pile vers une pile adjacente et plus grande.
- Le but du jeu est de minimiser le nombre de piles final.



Le jeu d'acquisition

Jeu d'acquisition (Lampert et Slater, 1995)

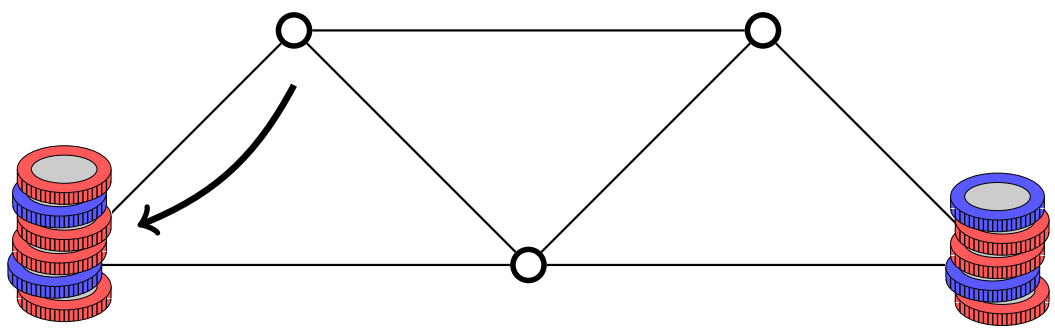
- Jeu à un joueur sur un graphe $G = (V, E)$.
- Il y a une pile de jeton sur chaque sommet.
- A chaque tour le joueur déplace une pile vers une pile adjacente et plus grande.
- Le but du jeu est de minimiser le nombre de piles final.



Le jeu d'acquisition

Jeu d'acquisition (Lampert et Slater, 1995)

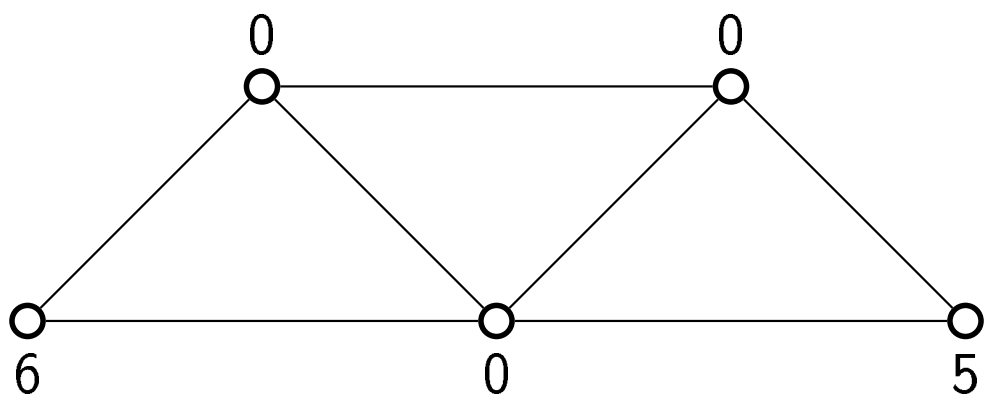
- Jeu à un joueur sur un graphe $G = (V, E)$.
- Il y a une pile de jeton sur chaque sommet.
- A chaque tour le joueur déplace une pile vers une pile adjacente et plus grande.
- Le but du jeu est de minimiser le nombre de piles final.



Le jeu d'acquisition

Jeu d'acquisition (Lampert et Slater, 1995)

- Jeu à un joueur sur un graphe $G = (V, E)$.
- Il y a une pile de jeton sur chaque sommet.
- A chaque tour le joueur déplace une pile vers une pile adjacente et plus grande.
- Le but du jeu est de minimiser le nombre de piles final.



Le jeu d'acquisition

- Introduit par Lampert et Slater en 1995
- Principalement étudié dans le cas de positions initiales avec un jeton par sommet
- Étudié sur différentes classes de graphes (grilles, arbres, graphes aléatoires)
- NP-complet de savoir s'il reste un seul tas à la fin (Slater et Wang, 2008)

Le jeu d'acquisition

- Introduit par Lampert et Slater en 1995
- Principalement étudié dans le cas de positions initiales avec un jeton par sommet
- Étudié sur différentes classes de graphes (grilles, arbres, graphes aléatoires)
- **NP-complet de savoir s'il reste un seul tas à la fin (Slater et Wang, 2008)**

Nos résultats

Nous étudions la complexité dans le cas de position avec un nombre arbitraire de jetons sur chaque sommet.

- NP-complet sur les étoiles subdivisées une fois
- NP-complet de savoir si on peut finir avec un seul tas sur $K_{3,n}$
- Fortement NP-complet sur les graphes scindés

Nos résultats

Nous étudions la complexité dans le cas de position avec un nombre arbitraire de jetons sur chaque sommet.

- NP-complet sur les étoiles subdivisées une fois
- NP-complet de savoir si on peut finir avec un seul tas sur $K_{3,n}$
- Fortement NP-complet sur les graphes scindés

- Pseudo-polynomial sur les graphes de largeur arborescente bornée et de degré maximal borné
- Algorithme en $W^{O(\log(W))}$ sur les arbres.

Réduction depuis PARTITION

PARTITION

Entrée : Un ensemble A de n éléments w_i
Question : Existe-t-il une partition de A en deux ensembles A_1 et A_2 telle que $\sum_{w_i \in A_1} w_i = \sum_{w_i \in A_2} w_i$?

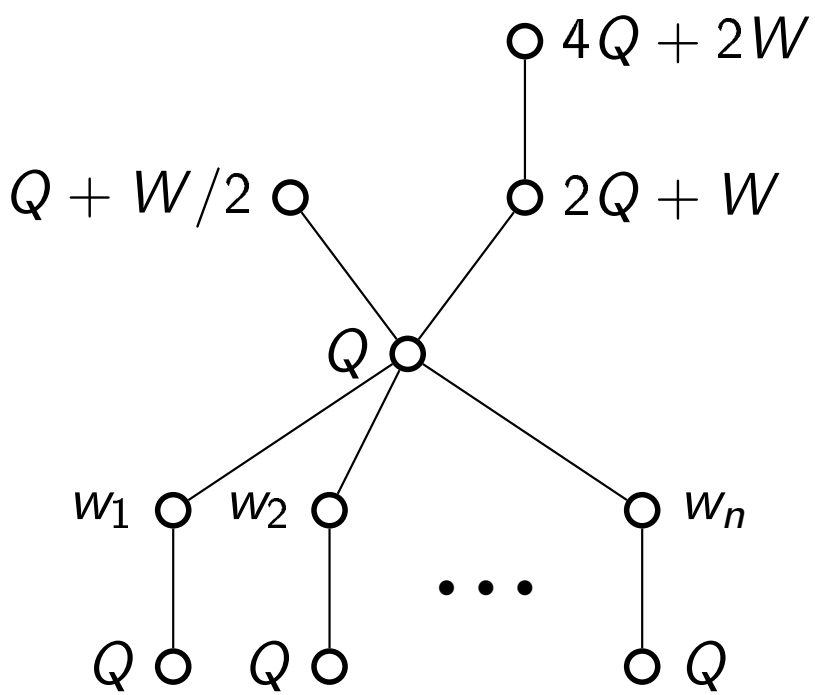
Théorème (Karp, 1972)

Le problème PARTITION est NP-complet.

Étoiles subdivisées

Théorème

Le jeu d'acquisition est NP-complet sur les étoiles subdivisées une fois



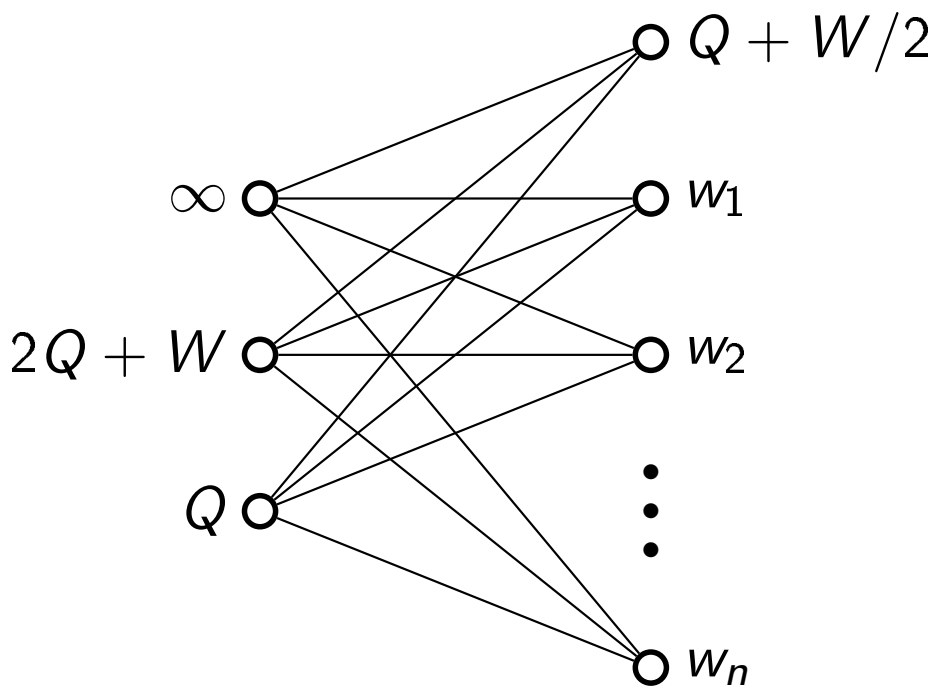
$$W = \sum w_i$$

$$Q > w_i, \forall i \in \{1, ..., n\}$$

Graphes bipartis complets

Théorème

Il est NP-complet de savoir s'il reste un seul tas pour le jeu d'acquisition sur les graphes bipartis complets $K_{3,k}$



Autres résultats de NP-complétudes

Théorème

Le jeu d'acquisition est NP-complet sur les chenille de degré au plus 3.

Théorème

Il est NP-complet de savoir s'il reste un seul tas pour le jeu d'acquisition sur les graphes scindés, les graphes de largeur arborescente 2, les graphes d'intervalles unitaires.

Problèmes fortement NP-complets

NP-complétude forte

Un problème est fortement NP-complet s'il reste NP-complet lorsque ses valeur numériques sont codées en unaire et non en binaire.

Problèmes fortement NP-complets

NP-complétude forte

Un problème est fortement NP-complet s'il reste NP-complet lorsque ses valeur numériques sont codées en unaire et non en binaire.

PARTITION n'est **pas** fortement NP-complet car il peut résolu à l'aide de programmation dynamique.

3-PARTITION

3-PARTITION

Entrée : Un ensemble A de $3n$ élément w_i avec $\sum w_i = nB$, avec $\frac{B}{4} < w_i < \frac{B}{2}$

Question : Existe-t-il une partition de A en n ensembles A_j telle que $\sum_{w_i \in A_j} w_i = B$?

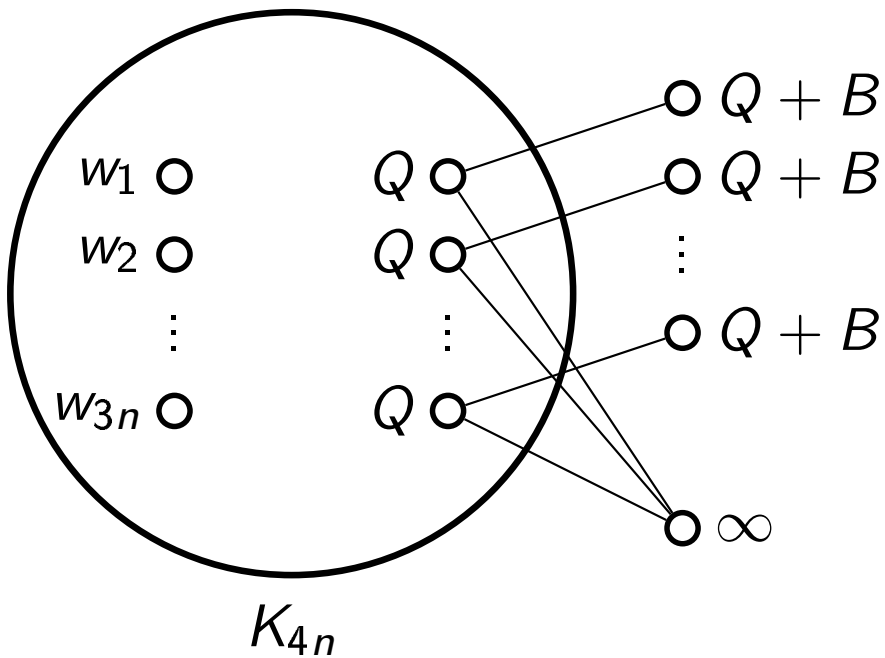
Théorème (Garey et Johnson, 1975)

Le problème 3-PARTITION est fortement NP-complet.

NP-complétude forte pour le jeu d'acquisition

Théorème

Le jeu d'acquisition est fortement NP-complet sur les graphes scindés.

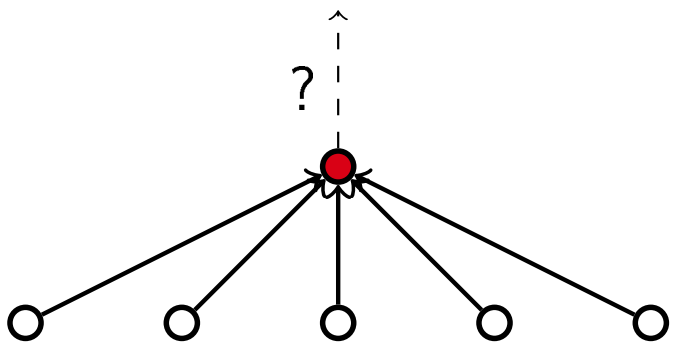


Algorithme pseudo-polynomial

Théorème

Le jeu d'acquisition est pseudo polynomial sur les arbres de degré borné

La preuve se fait par programmation dynamique :

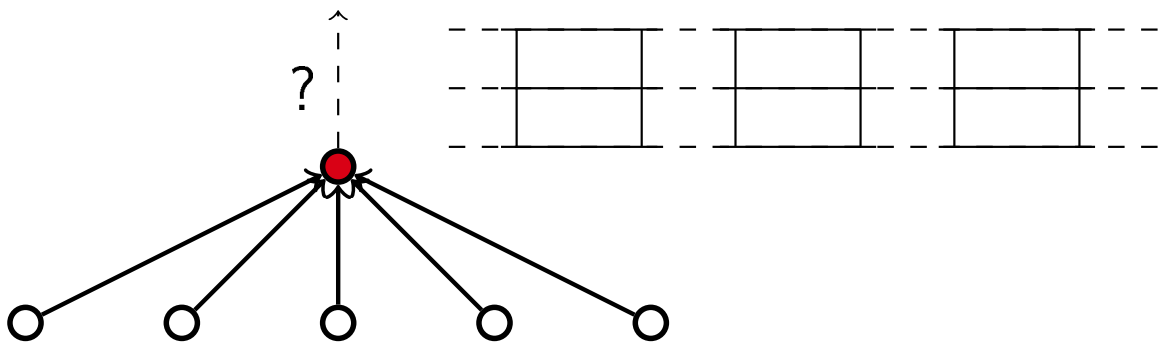


Algorithme pseudo-polynomial

Théorème

Le jeu d'acquisition est pseudo polynomial sur les arbres de degré borné

La preuve se fait par programmation dynamique :

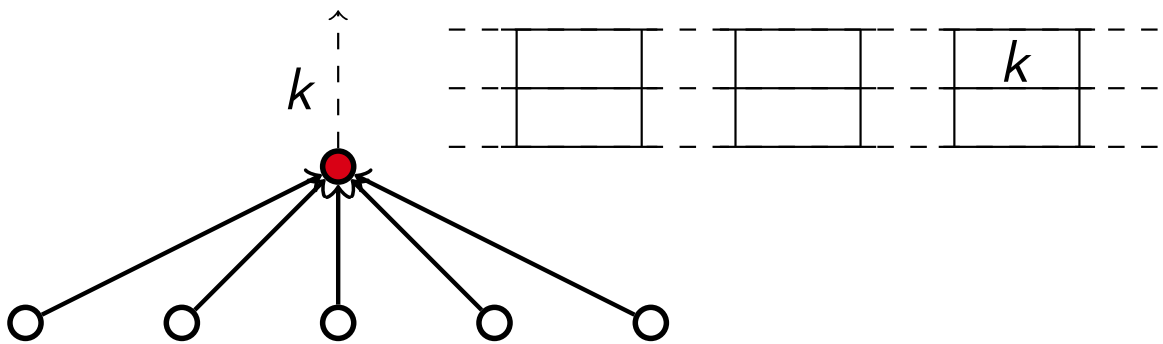


Algorithme pseudo-polynomial

Théorème

Le jeu d'acquisition est pseudo polynomial sur les arbres de degré borné

La preuve se fait par programmation dynamique :

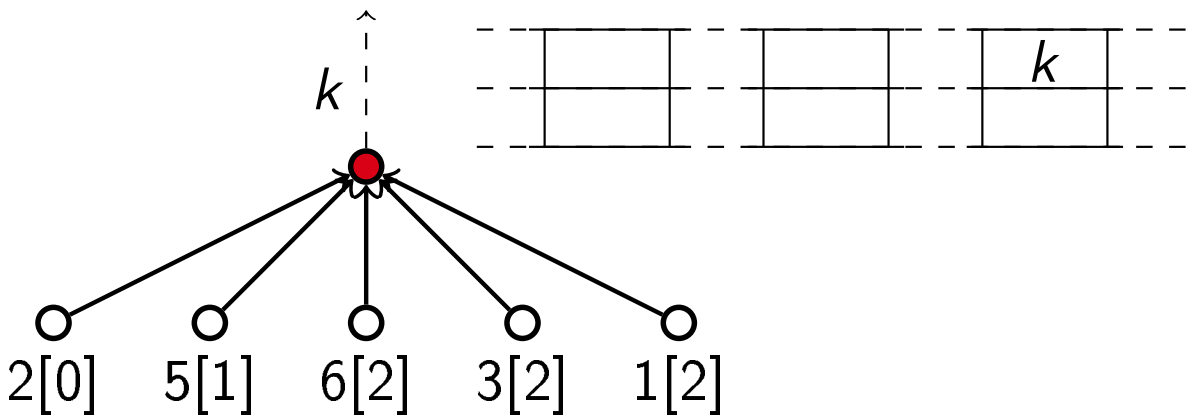


Algorithme pseudo-polynomial

Théorème

Le jeu d'acquisition est pseudo polynomial sur les arbres de degré borné

La preuve se fait par programmation dynamique :

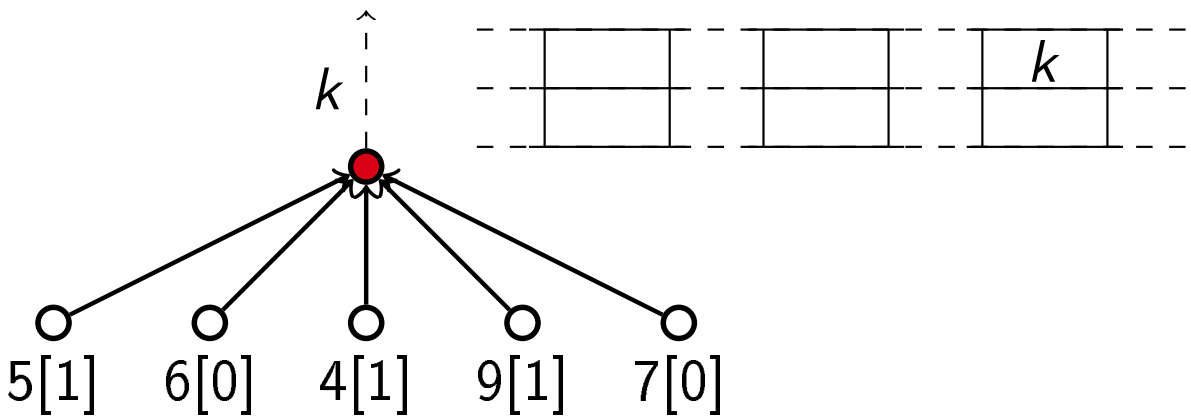


Algorithme pseudo-polynomial

Théorème

Le jeu d'acquisition est pseudo polynomial sur les arbres de degré borné

La preuve se fait par programmation dynamique :

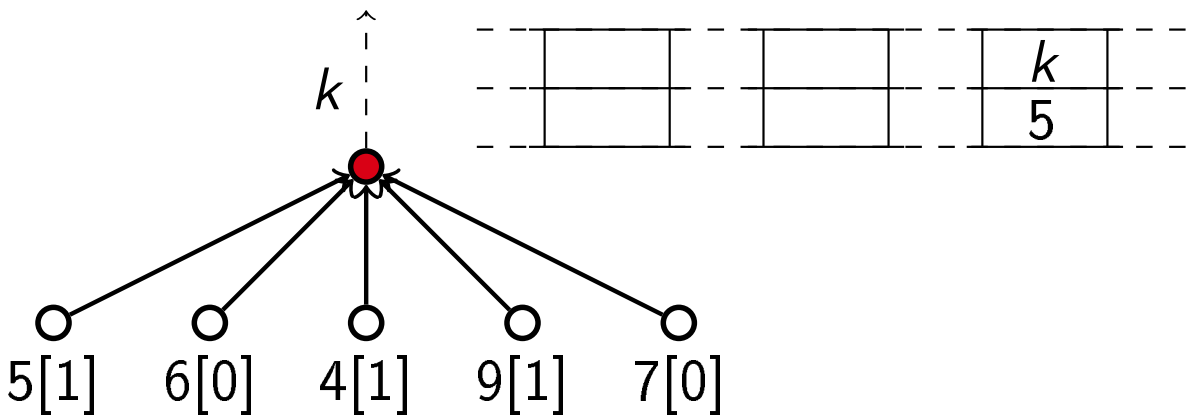


Algorithme pseudo-polynomial

Théorème

Le jeu d'acquisition est pseudo polynomial sur les arbres de degré borné

La preuve se fait par programmation dynamique :

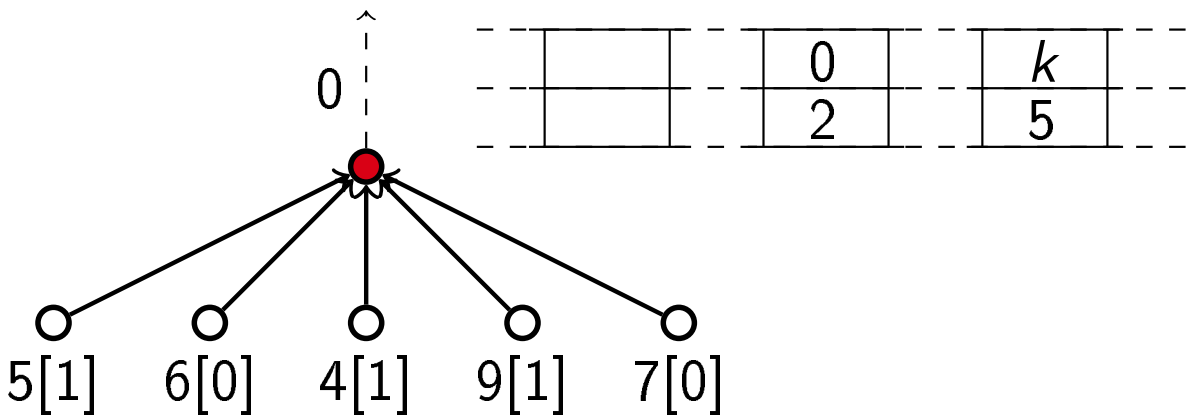


Algorithme pseudo-polynomial

Théorème

Le jeu d'acquisition est pseudo polynomial sur les arbres de degré borné

La preuve se fait par programmation dynamique :

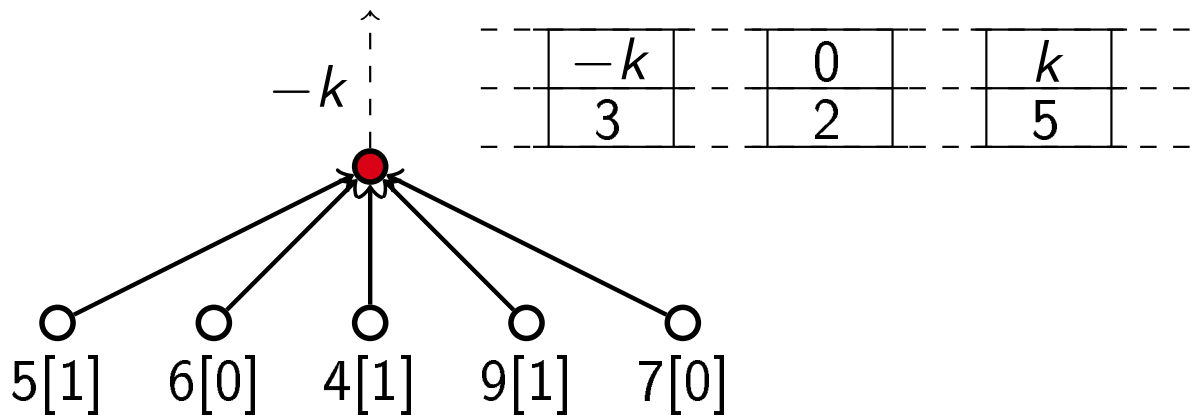


Algorithme pseudo-polynomial

Théorème

Le jeu d'acquisition est pseudo polynomial sur les arbres de degré borné

La preuve se fait par programmation dynamique :

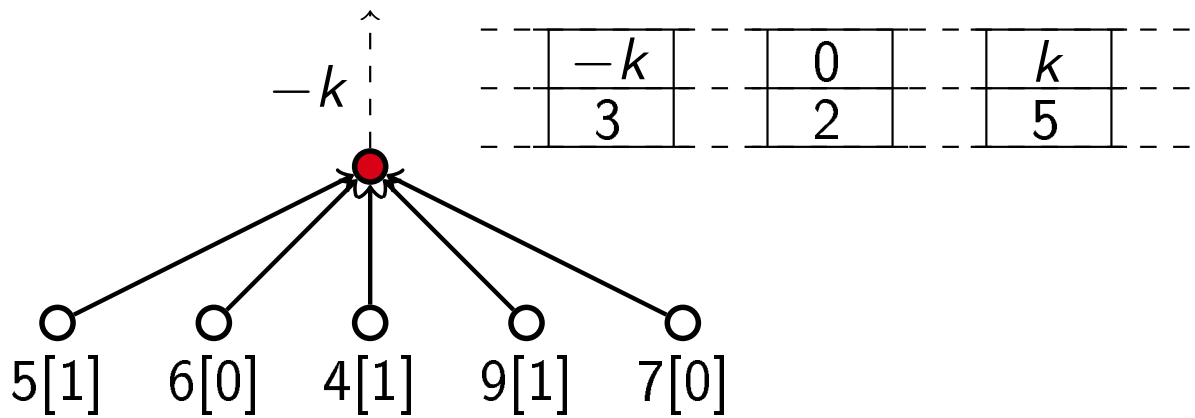


Algorithme pseudo-polynomial

Théorème

Le jeu d'acquisition est pseudo polynomial sur les arbres de degré borné

La preuve se fait par programmation dynamique :



La programmation dynamique permet de résoudre le jeu d'acquisition en $O(W^{\Delta})$.

Algorithme pseudo-polynomial

Théorème

Le jeu d'acquisition est pseudo polynomial sur les graphes de largeur arborescente bornée et de degré borné

Algorithme pseudo-polynomial

Théorème

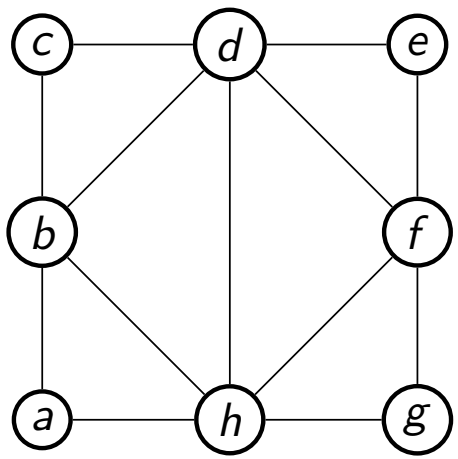
Le jeu d'acquisition est pseudo polynomial sur les graphes de largeur arborescente bornée et de degré borné

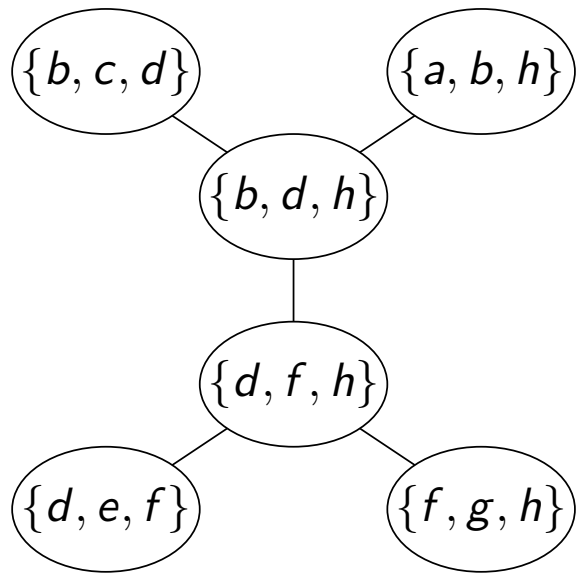
Qu'est-ce que la largeur arborescente ?

Décomposition arborescente

Une décomposition arborescente de G est un arbre T tel que :

- Les nœuds de T sont des sous ensembles de $V(G)$

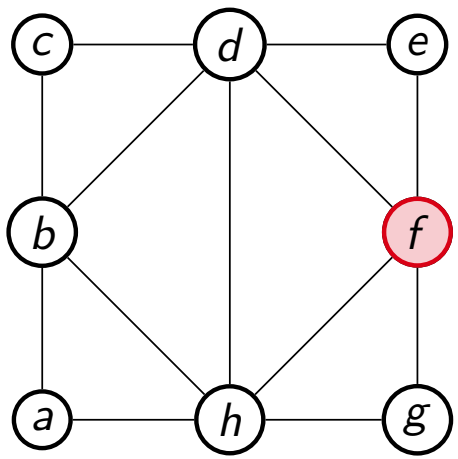


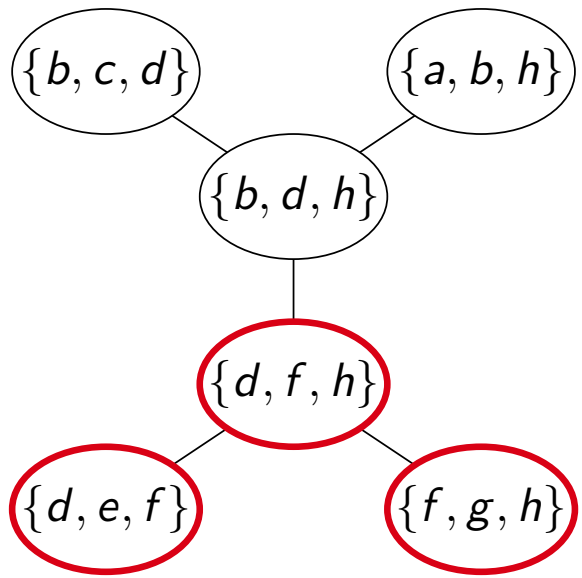


Décomposition arborescente

Une décomposition arborescente de G est un arbre T tel que :

- Les nœuds de T sont des sous ensembles de $V(G)$
- Chaque sommet de G est contenu dans un sous-arbre de T

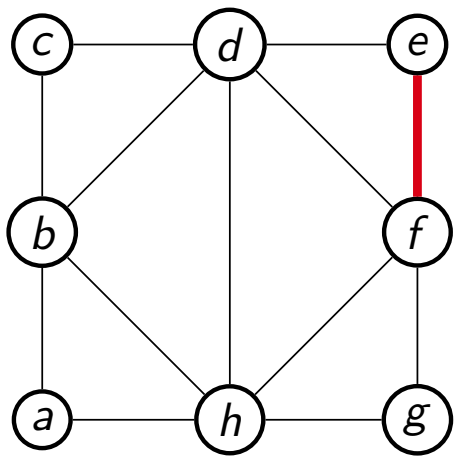


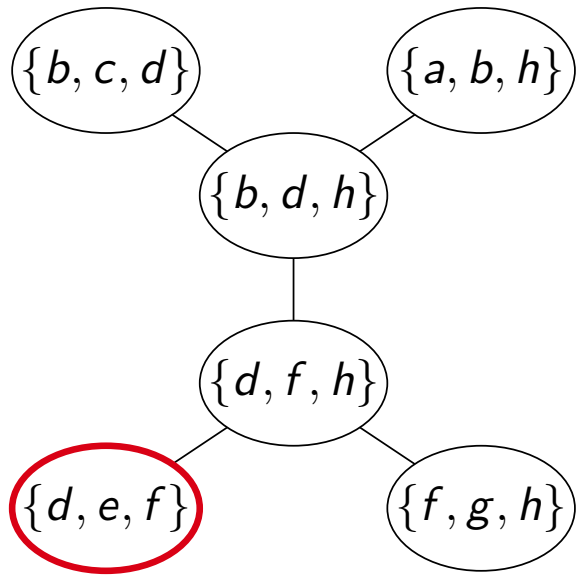


Décomposition arborescente

Une décomposition arborescente de G est un arbre T tel que :

- Les nœuds de T sont des sous ensembles de $V(G)$
- Chaque sommet de G est contenu dans un sous-arbre de T
- Chaque arête de G est contenu dans l'un des nœuds de T

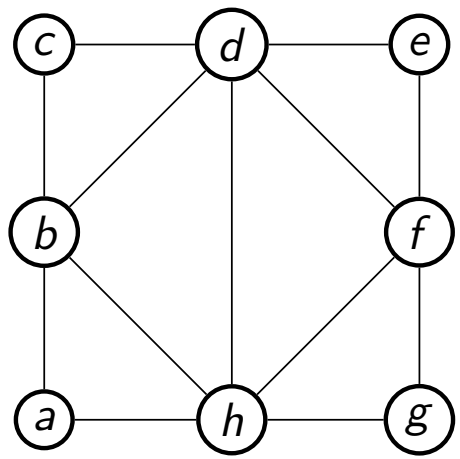


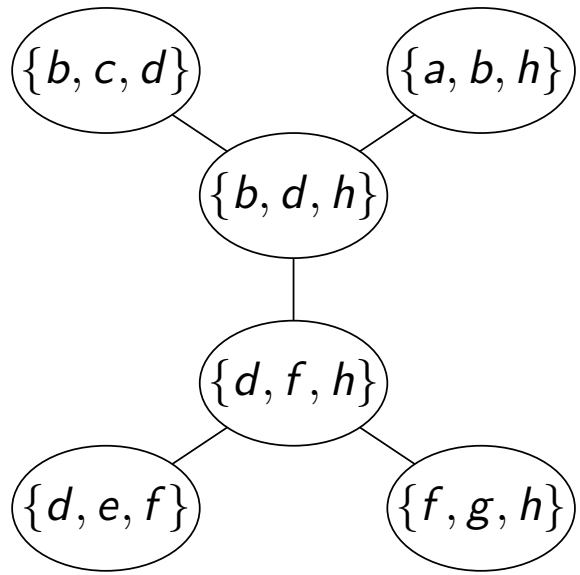


Décomposition arborescente

Une décomposition arborescente de G est un arbre T tel que :

- Les nœuds de T sont des sous ensembles de $V(G)$
- Chaque sommet de G est contenu dans un sous-arbre de T
- Chaque arête de G est contenu dans l'un des nœuds de T



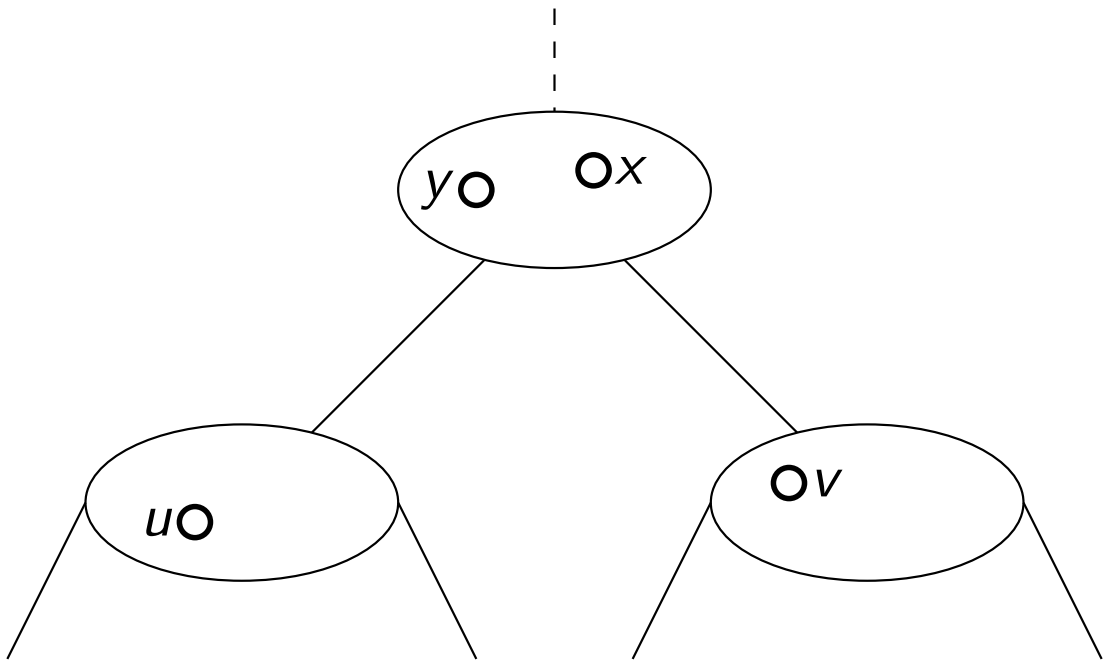


La largeur arborescente d'une décomposition est le cardinal maximum de ses nœuds -1.

La largeur arborescente de G, $tw(G)$, est le minimum des largeurs arborescente de ses décomposition.

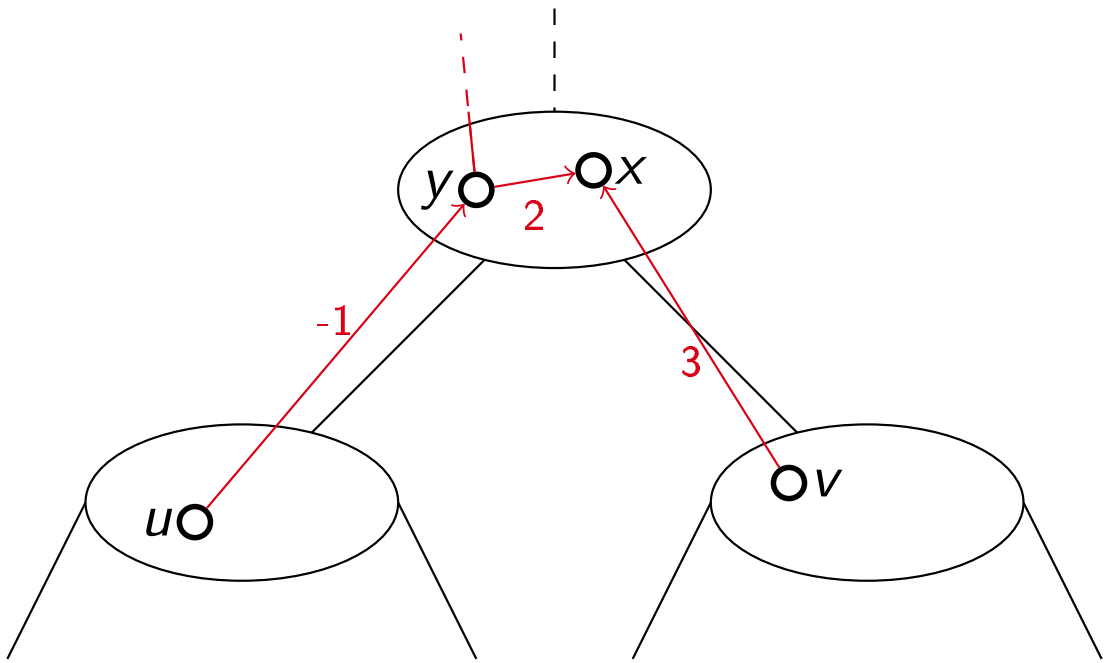
Graphes de largeur arborescente bornée

Pour un nœud donné de la décomposition, une configuration est une attribution de poids à chaque arête incidente à un sommet du nœud.



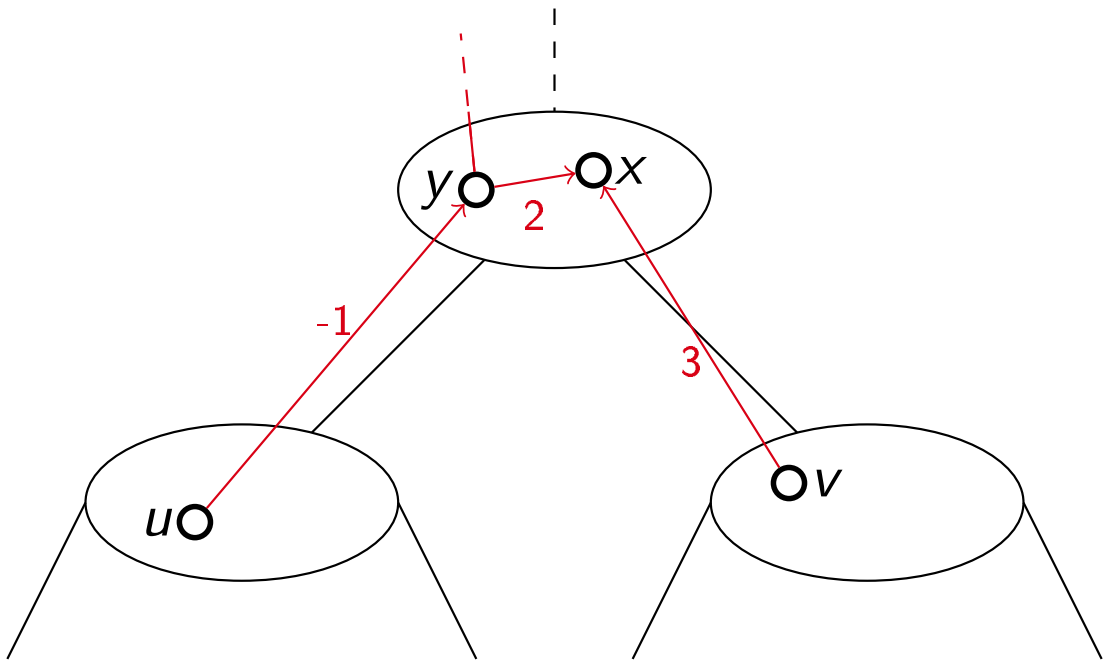
Graphes de largeur arborescente bornée

Pour un nœud donné de la décomposition, une configuration est une attribution de poids à chaque arête incidente à un sommet du nœud.



Graphes de largeur arborescente bornée

Pour un nœud donné de la décomposition, une configuration est une attribution de poids à chaque arête incidente à un sommet du nœud.

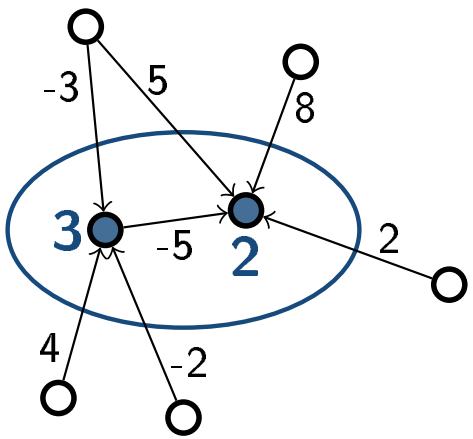


Il y a au plus $W^{tw(G)\Delta}$ configurations pour chaque nœud.

Graphes de largeur arborescente bornée

Une configuration est valide si pour chaque sommet :

- Il ne donne son poids a au plus un sommet.
- S'il donne son poids a un sommet, la somme des valeurs entrantes est égale à la valeur sortante
- Les valeurs entrantes sont compatibles avec les règles d'acquisition

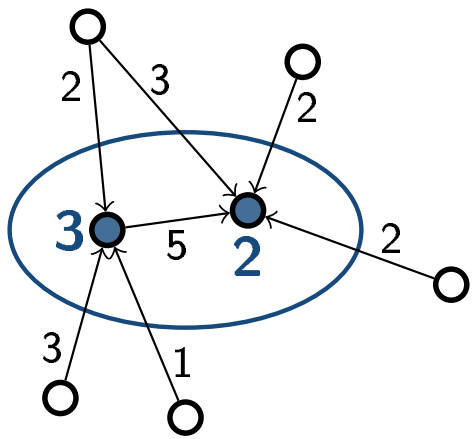


Non valide

Graphes de largeur arborescente bornée

Une configuration est valide si pour chaque sommet :

- Il ne donne son poids a au plus un sommet.
- S'il donne son poids a un sommet, la somme des valeurs entrantes est égale à la valeur sortante
- Les valeurs entrantes sont compatibles avec les règles d'acquisition

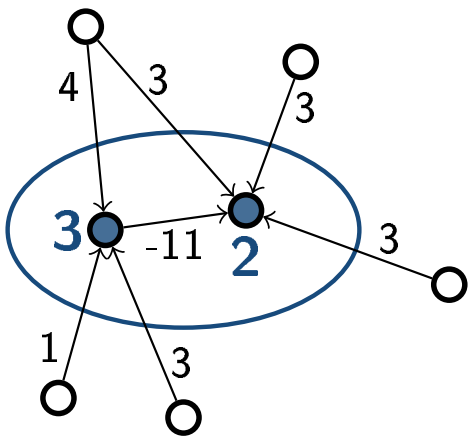


Non valide

Graphes de largeur arborescente bornée

Une configuration est valide si pour chaque sommet :

- Il ne donne son poids a au plus un sommet.
- S'il donne son poids a un sommet, la somme des valeurs entrantes est égale à la valeur sortante
- Les valeurs entrantes sont compatibles avec les règles d'acquisition

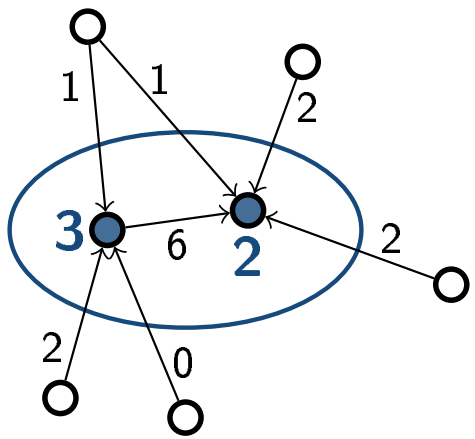


Non valide

Graphes de largeur arborescente bornée

Une configuration est valide si pour chaque sommet :

- Il ne donne son poids a au plus un sommet.
- S'il donne son poids a un sommet, la somme des valeurs entrantes est égale à la valeur sortante
- Les valeurs entrantes sont compatibles avec les règles d'acquisition

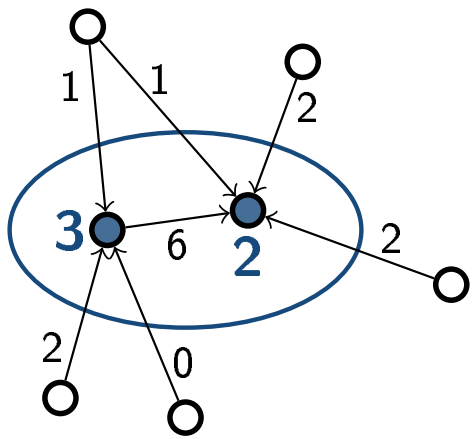


Valide

Graphes de largeur arborescente bornée

Une configuration est valide si pour chaque sommet :

- Il ne donne son poids a au plus un sommet.
- S'il donne son poids a un sommet, la somme des valeurs entrantes est égale à la valeur sortante
- Les valeurs entrantes sont compatibles avec les règles d'acquisition



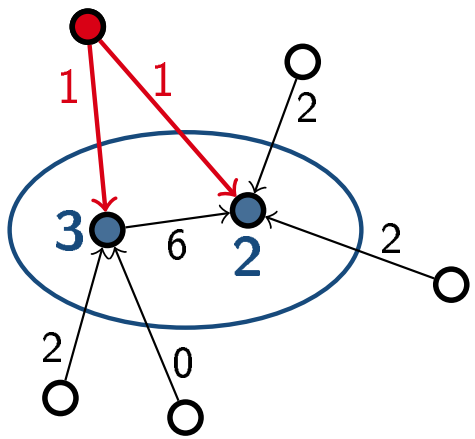
Valide

On attribue la valeur ∞ à toute configuration non valide.

Graphes de largeur arborescente bornée

Une configuration est valide si pour chaque sommet :

- Il ne donne son poids a au plus un sommet.
- S'il donne son poids a un sommet, la somme des valeurs entrantes est égale à la valeur sortante
- Les valeurs entrantes sont compatibles avec les règles d'acquisition



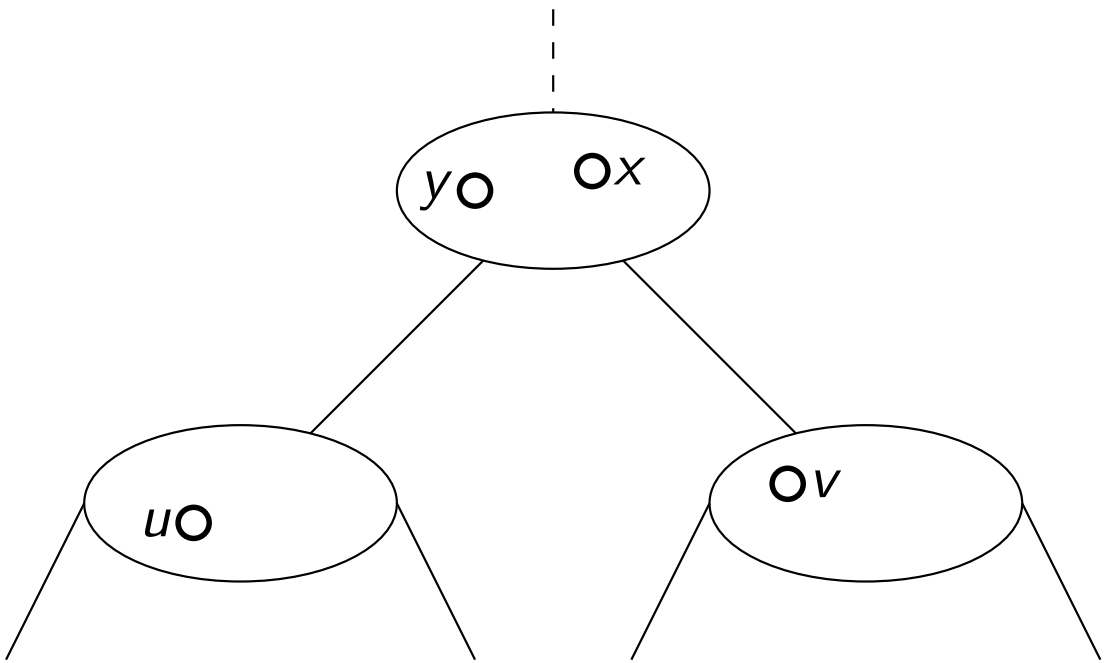
Valide

On attribue la valeur ∞ à toute configuration non valide.

Graphes de largeur arborescente bornée

Deux configurations sont compatibles si les sommets qu'elles ont en commun donnent et reçoivent les mêmes poids pour chaque arête.

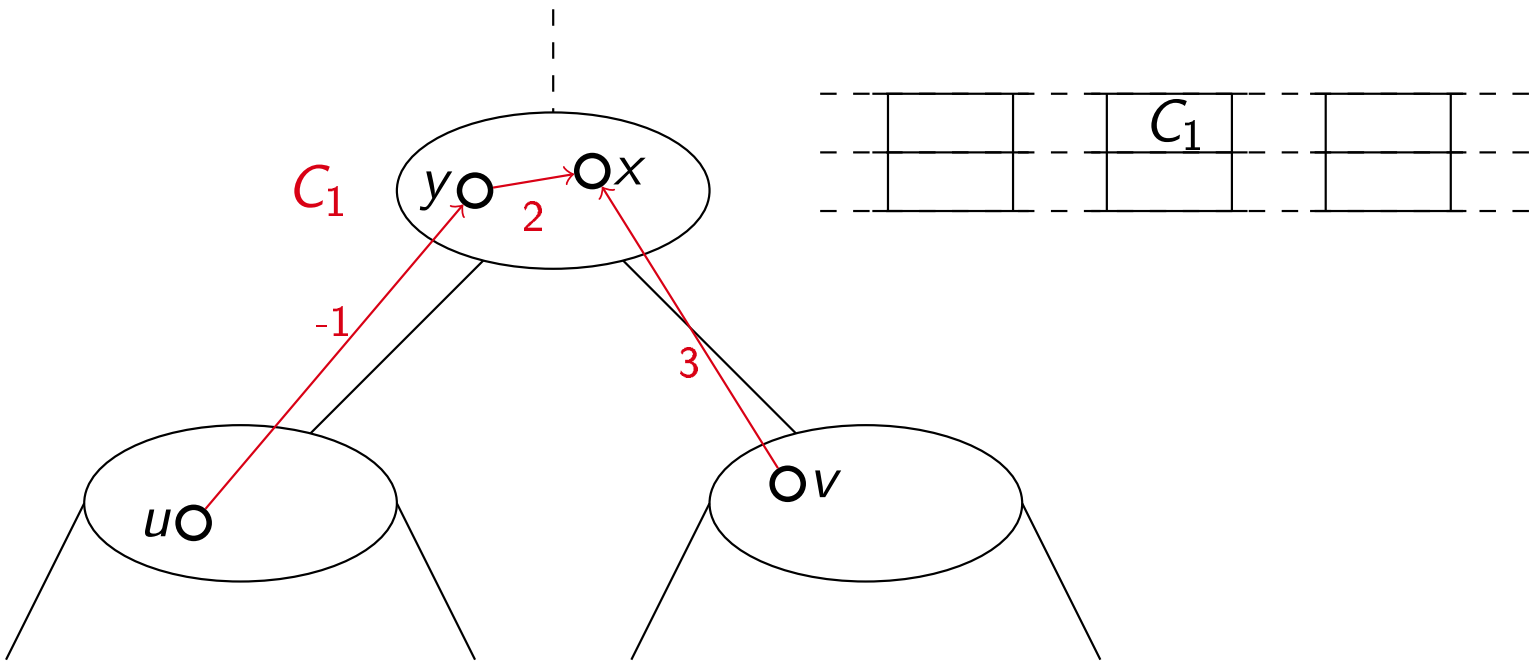
On effectue de la programmation dynamique sur les configurations compatibles entre nœuds adjacents de la décomposition.



Graphes de largeur arborescente bornée

Deux configurations sont compatibles si les sommets qu'elles ont en commun donnent et reçoivent les mêmes poids pour chaque arête.

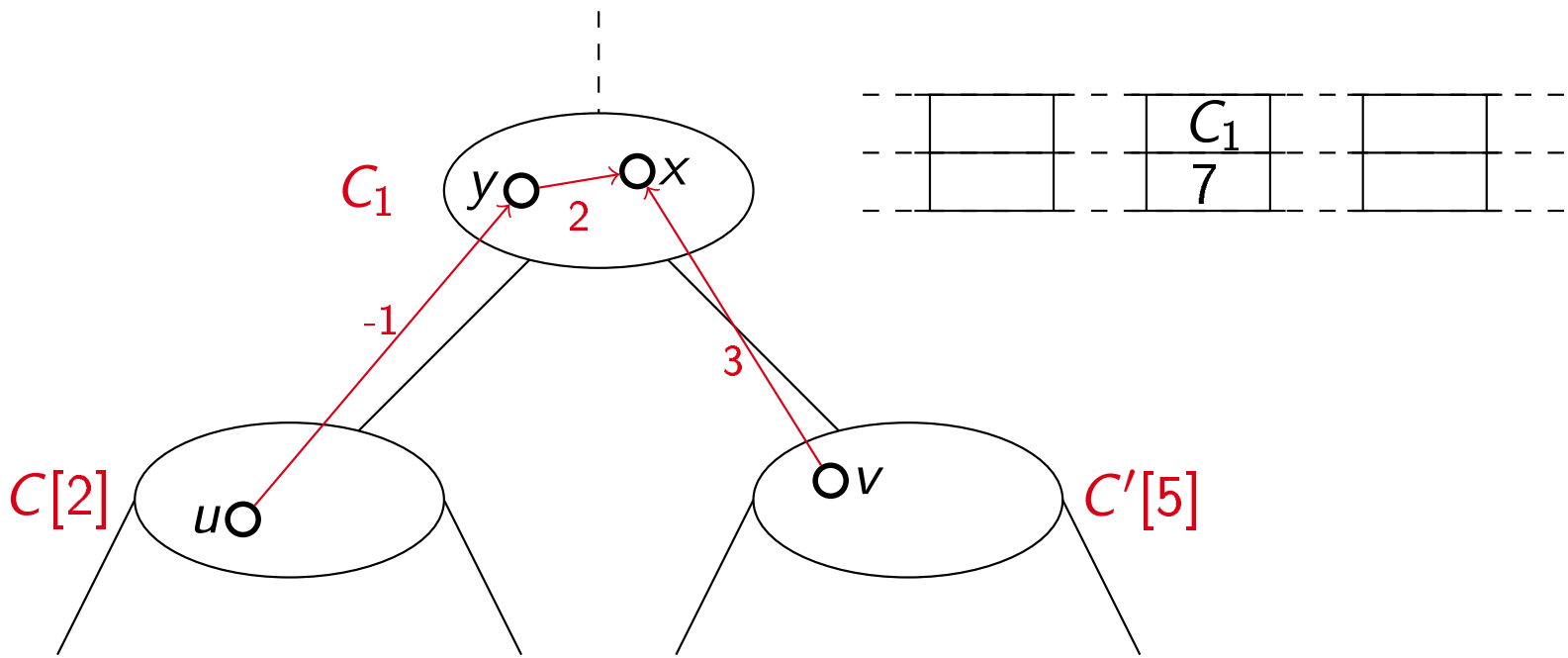
On effectue de la programmation dynamique sur les configurations compatibles entre nœuds adjacents de la décomposition.



Graphes de largeur arborescente bornée

Deux configurations sont compatibles si les sommets qu'elles ont en commun donnent et reçoivent les mêmes poids pour chaque arête.

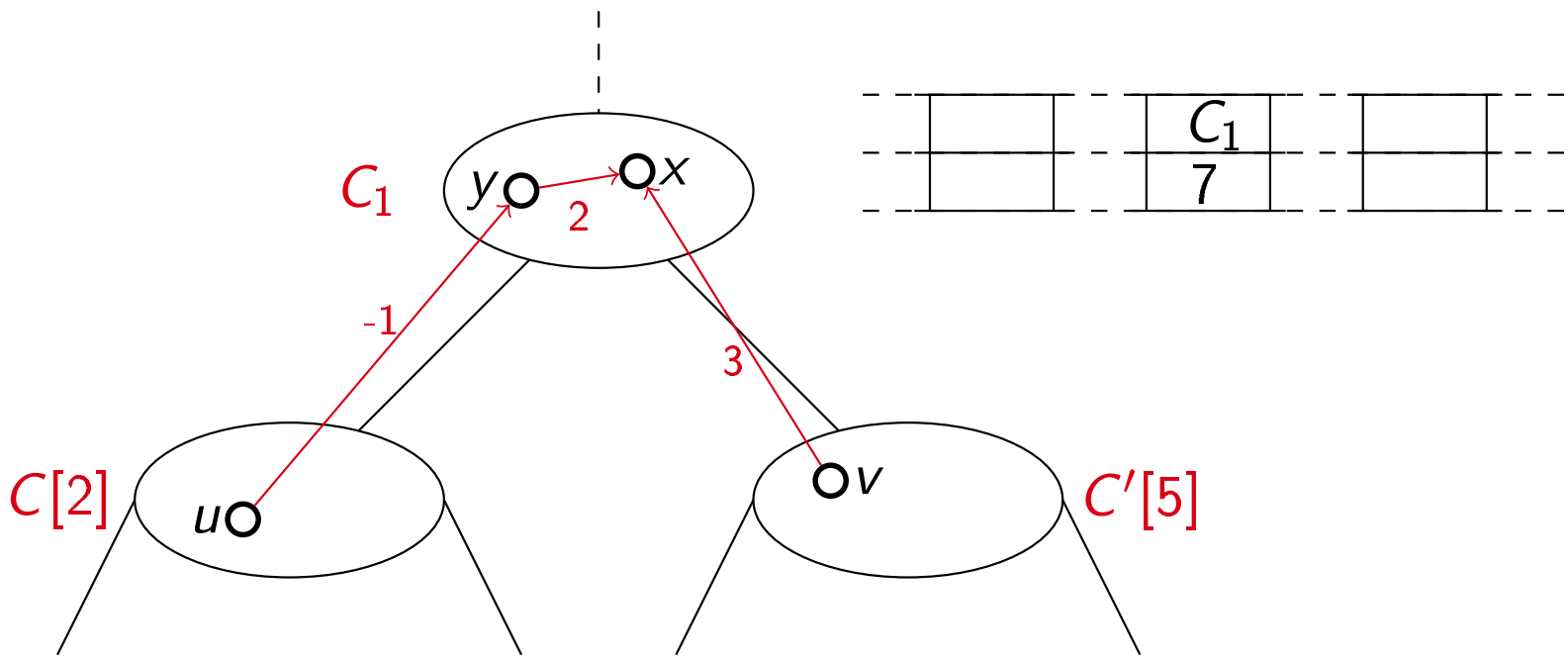
On effectue de la programmation dynamique sur les configurations compatibles entre nœuds adjacents de la décomposition.



Graphes de largeur arborescente bornée

Deux configurations sont compatibles si les sommets qu'elles ont en commun donnent et reçoivent les mêmes poids pour chaque arête.

On effectue de la programmation dynamique sur les configurations compatibles entre nœuds adjacents de la décomposition.



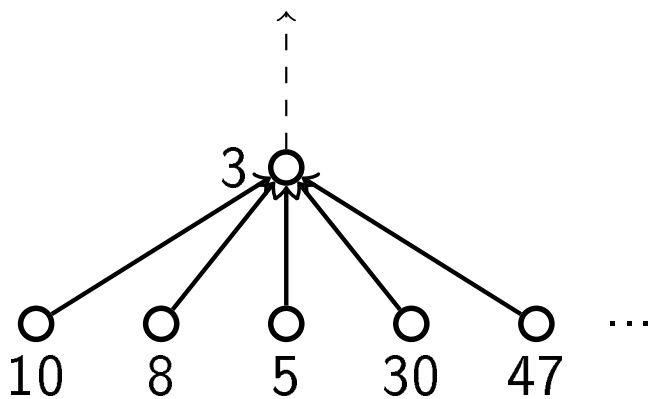
On peu considérer chaque nœud fils indépendamment de tous les autres $\Longrightarrow$ Algorithme en $W^{O(tw(G)\Delta)}$

Revenons aux arbres

Théorème

Le jeu d'acquisition peut être résolu en temps $W^{O(\log(W))}$ sur les arbres.

On remarque que notre algorithme précédent peut être amélioré :

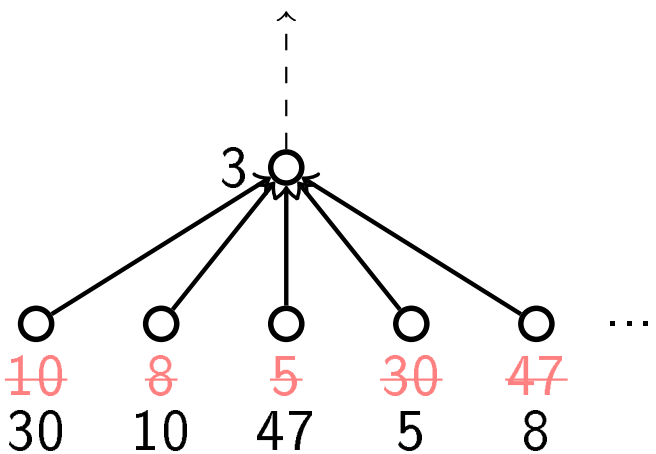


Revenons aux arbres

Théorème

Le jeu d'acquisition peut être résolu en temps $W^{O(\log(W))}$ sur les arbres.

On remarque que notre algorithme précédent peut être amélioré :

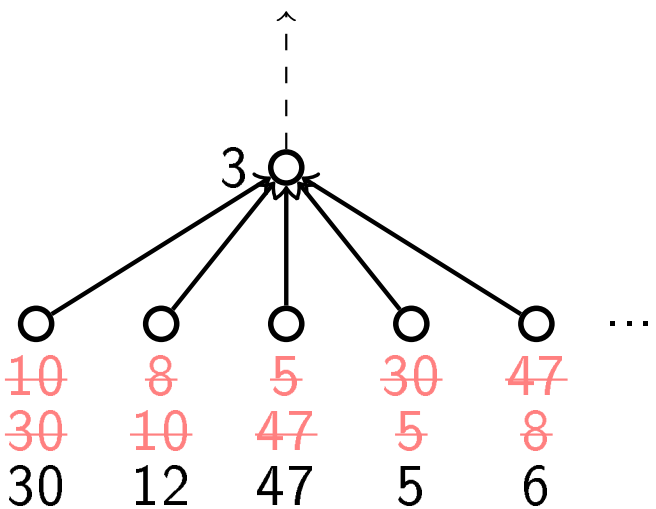


Revenons aux arbres

Théorème

Le jeu d'acquisition peut être résolu en temps $W^{O(\log(W))}$ sur les arbres.

On remarque que notre algorithme précédent peut être amélioré :

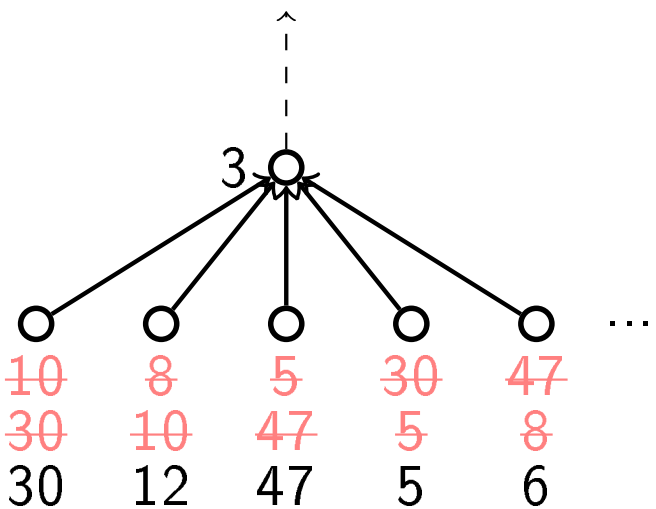


Revenons aux arbres

Théorème

Le jeu d'acquisition peut être résolu en temps $W^{O(\log(W))}$ sur les arbres.

On remarque que notre algorithme précédent peut être amélioré :



On crée un *"enregistrement"* de l'état actuel du parcours :

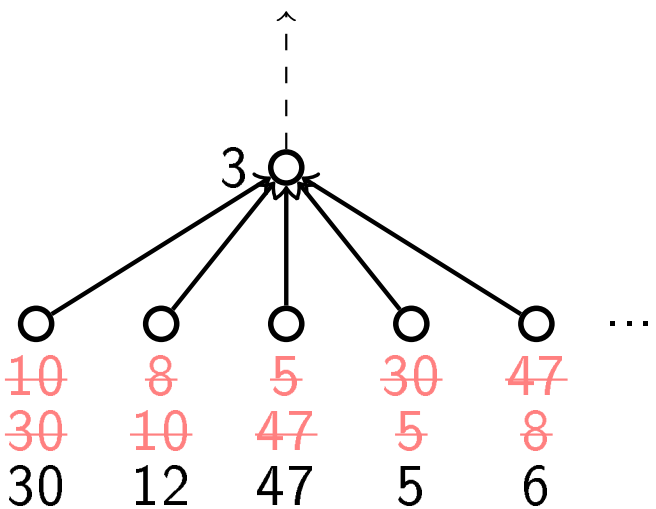
$\{[0, 3]; [5, 28]; [30, 107]\}$

Revenons aux arbres

Théorème

Le jeu d'acquisition peut être résolu en temps $W^{O(\log(W))}$ sur les arbres.

On remarque que notre algorithme précédent peut être amélioré :



On crée un *"enregistrement"* de l'état actuel du parcours :

$\{[0,3];[5,28];[30,107]\}$

↑ ↑ ↑

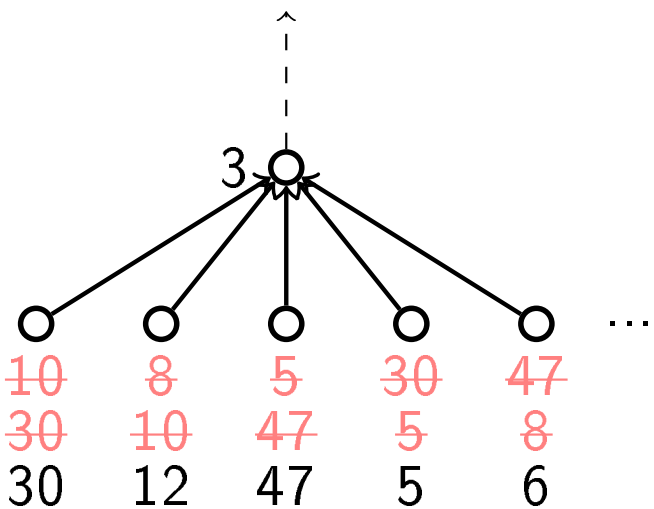
Valeurs "bloquantes"

Revenons aux arbres

Théorème

Le jeu d'acquisition peut être résolu en temps $W^{O(\log(W))}$ sur les arbres.

On remarque que notre algorithme précédent peut être amélioré :



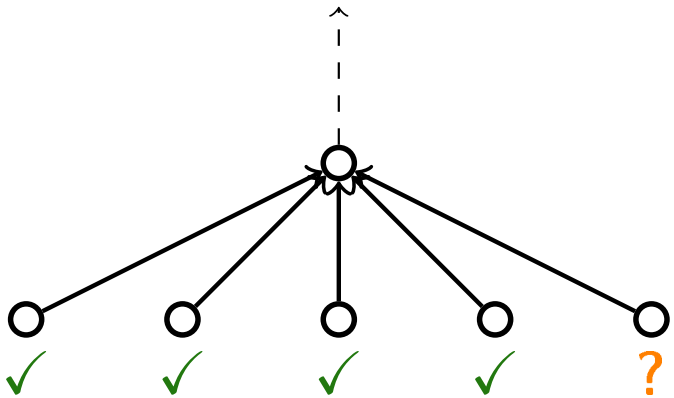
On crée un *"enregistrement"* de l'état actuel du parcours :

$$\{[0,3];[5,28];[30,107]\}$$

Valeurs atteintes si "débloqué"

Revenons aux arbres

Mise à jour des enregistrements :

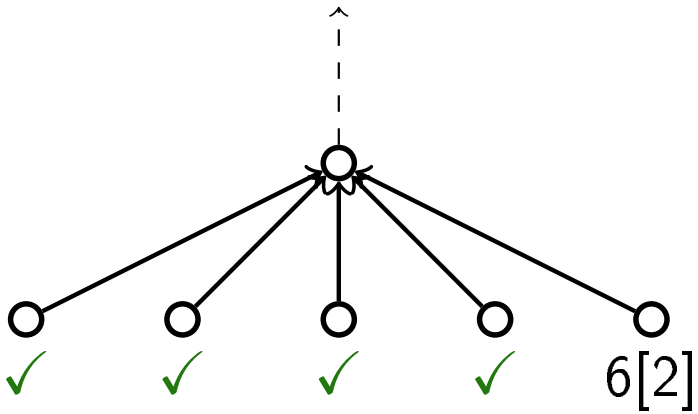


Ensemble des enregistrements :

$$\{...\}[k]$$
$$\{[0,3];[5,28];[30,107]\}[5]$$
$$\{...\}[k']$$

Revenons aux arbres

Mise à jour des enregistrements :

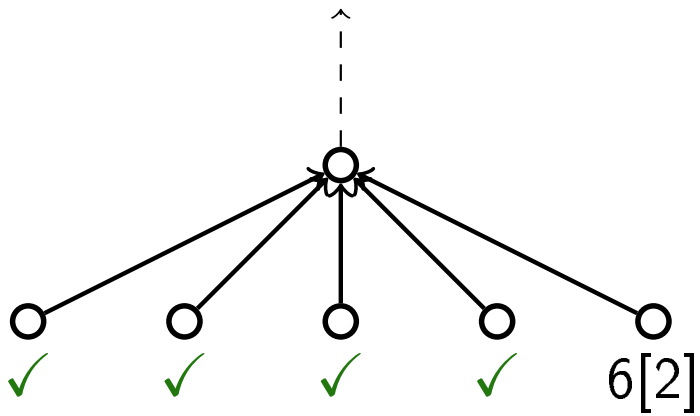


Ensemble des enregistrements :

$$\{...\}[k]$$
$$\{[0,3];[5,28];[30,107]\}[5]$$
$$\{...\}[k']$$

Revenons aux arbres

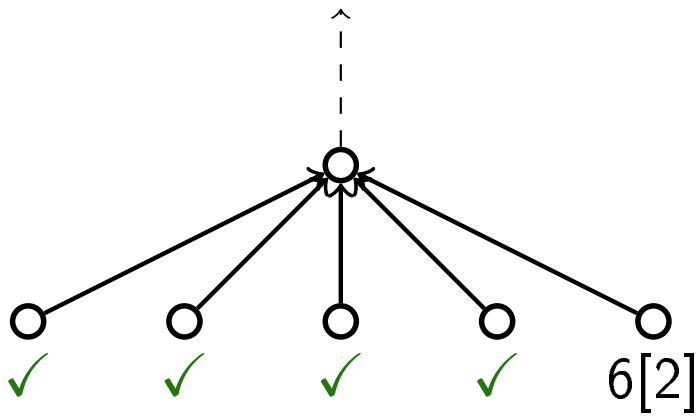
Mise à jour des enregistrements :



$$\{[0,3];\underbrace{[5,28]}_{\mathbf{6}};[30,107]\}[5]$$

Revenons aux arbres

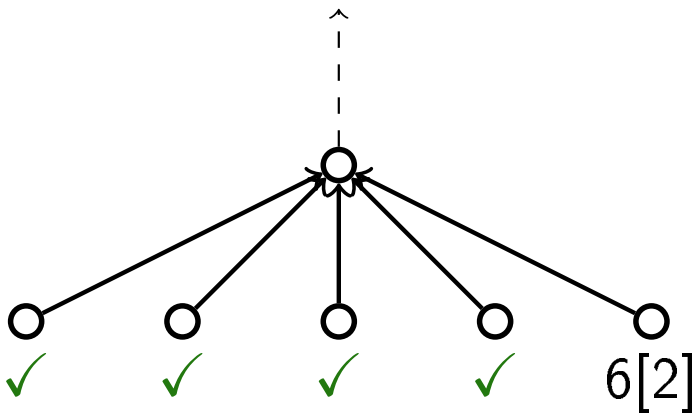
Mise à jour des enregistrements :



$\{[0,3];[5,\mathbf{34}];[30,107]\}[7]$

Revenons aux arbres

Mise à jour des enregistrements :

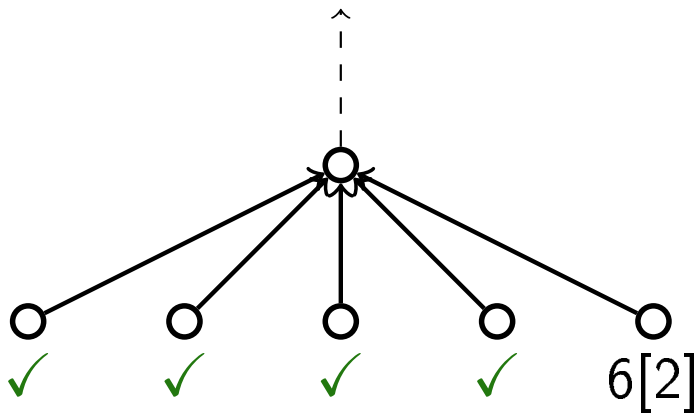


$\{[0,3];[5,\mathbf{34}];[30,107]\}[7]$

$\geq$

Revenons aux arbres

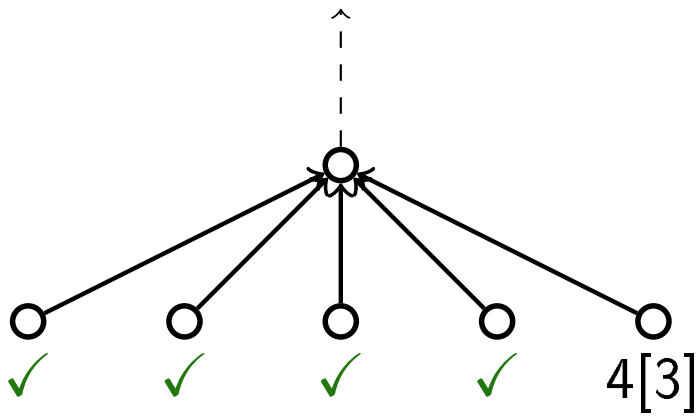
Mise à jour des enregistrements :



$\{[0,3];[5,\mathbf{111}]\}[7]$

Revenons aux arbres

Mise à jour des enregistrements :



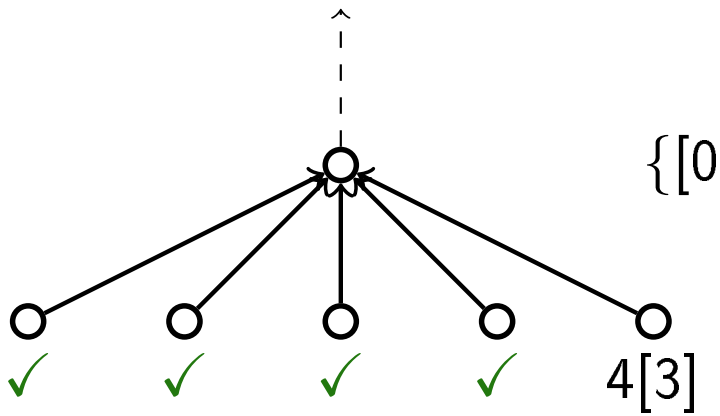
$\{[0,3];[5,28];[30,107]\}[5]$

↑

4

Revenons aux arbres

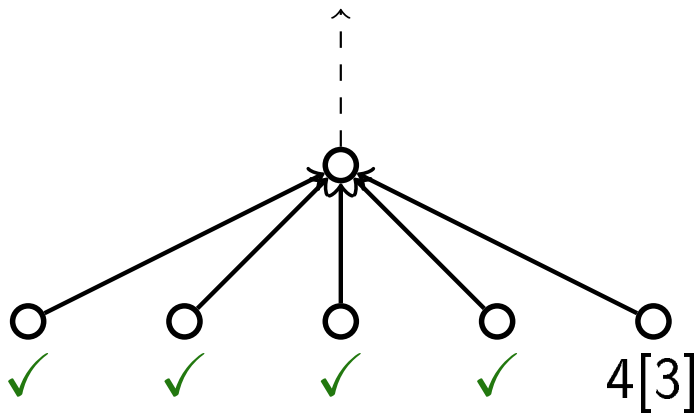
Mise à jour des enregistrements :



$\{[0,3];\mathbf{[4,8]};[5,28];[30,107]\}[8]$

Revenons aux arbres

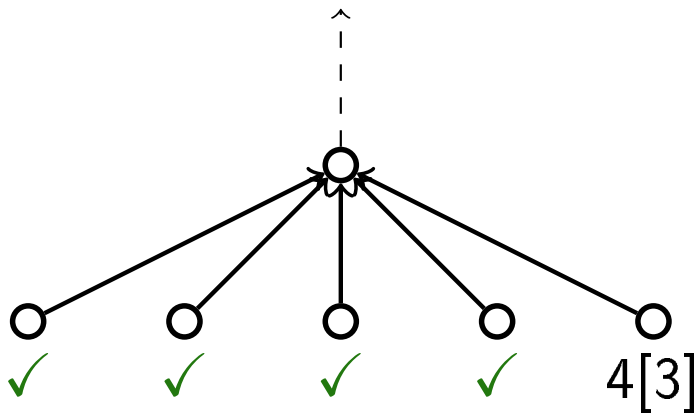
Mise à jour des enregistrements :



$\{[0,3];[4,\mathbf{33}];[30,107]\}[8]$

Revenons aux arbres

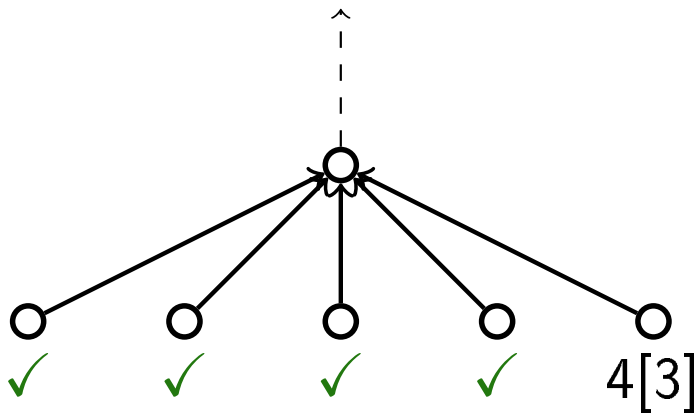
Mise à jour des enregistrements :



$\{[0,3];[4,\mathbf{110}]\}[8]$

Revenons aux arbres

Mise à jour des enregistrements :



À la fin, les enregistrements de la forme $\{[0, k]\}$ donne le poids optimal pour donner un poids k au père.

Analyse de l'algorithme

Regardons plus en détails les enregistrements

$$\{[0, 3]; [5, 28]; [30, 107]\}$$

Analyse de l'algorithme

Regardons plus en détails les enregistrements

$$\{[0, 3]; [5, 28]; [30, 107]\}$$

- La valeur finale ne peut pas être plus grande que W

Analyse de l'algorithme

Regardons plus en détails les enregistrements

$$\{[0,3];[5,28];[30,107]\}$$
$$\uparrow \quad \uparrow$$

- La valeur finale ne peut pas être plus grande que W
- La fin de chaque intervalle est au moins le double de son début.

Analyse de l'algorithme

Regardons plus en détails les enregistrements

$$\{[0, 3]; [5, 28]; [30, 107]\}$$

- La valeur finale ne peut pas être plus grande que W
- La fin de chaque intervalle est au moins le double de son début.

$\Longrightarrow$ Il y a au plus $\log(W)$ intervalles dans un enregistrement.

Analyse de l'algorithme

Regardons plus en détails les enregistrements

$$\{[0, 3]; [5, 28]; [30, 107]\}$$

- La valeur finale ne peut pas être plus grande que W
- La fin de chaque intervalle est au moins le double de son début.

$\Longrightarrow$ Il y a au plus $\log(W)$ intervalles dans un enregistrement.
$\Longrightarrow$ Il y a au plus $W^{\log(W)}$ enregistrements.

Analyse de l'algorithme

Regardons plus en détails les enregistrements

$$\{[0,3];[5,28];[30,107]\}$$

- La valeur finale ne peut pas être plus grande que W
- La fin de chaque intervalle est au moins le double de son début.

$\Longrightarrow$ Il y a au plus $\log(W)$ intervalles dans un enregistrement.
$\Longrightarrow$ Il y a au plus $W^{\log(W)}$ enregistrements.

Théorème

Le jeu d'acquisition peut être résolu en temps $W^{O(\log(W))}$ sur les arbres.

Perspectives

- Application des "enregistrements" aux graphes de largeur arborescente bornée.
- Algorithme pseudo-polynomial pour les arbres sans borner le degré.
- Algorithme pseudo-polynomial pour des classes de graphes dont la largeur arborescente n'est pas bornée.

Perspectives

- Application des "enregistrements" aux graphes de largeur arborescente bornée.
- Algorithme pseudo-polynomial pour les arbres sans borner le degré.
- Algorithme pseudo-polynomial pour des classes de graphes dont la largeur arborescente n'est pas bornée.