Étiquetage des graphes
pour le calcul de distance

Cyril Gavoille

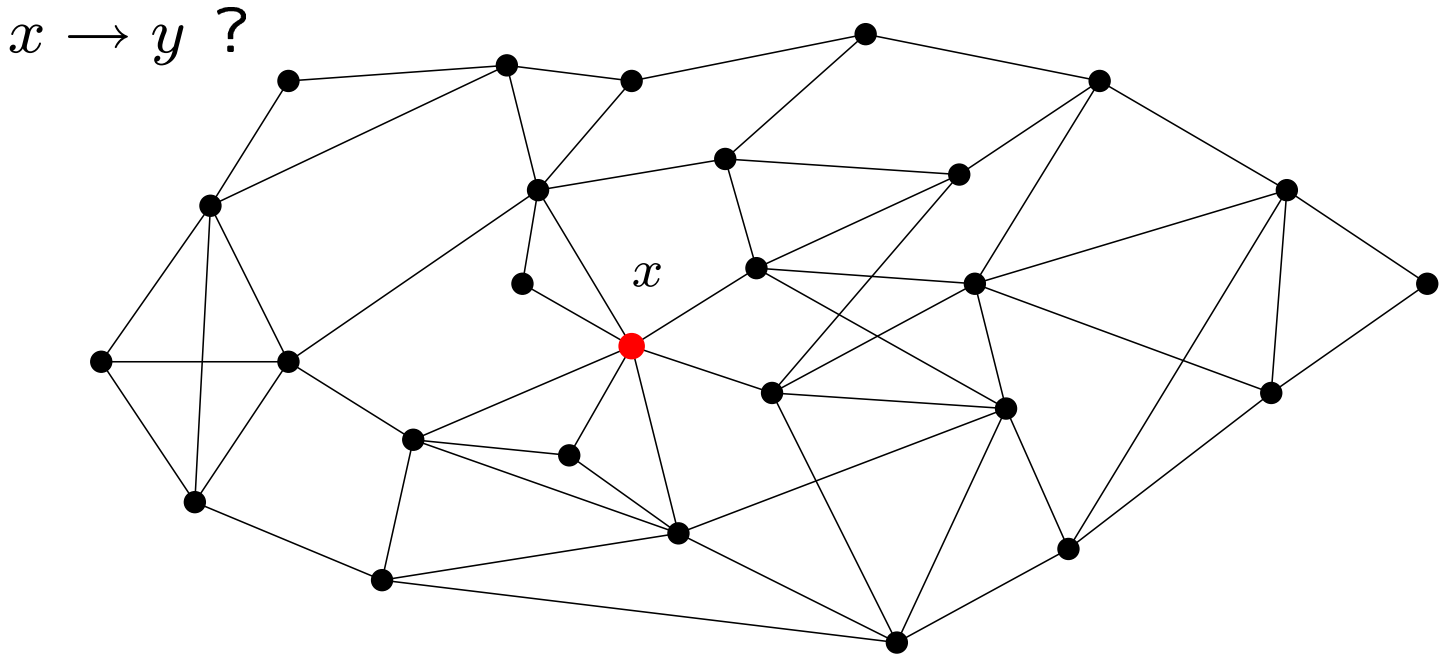
8 avril 2005 – Montpellier
École Jeunes Chercheurs “Algorithmique et Calcul Formel”

Plan

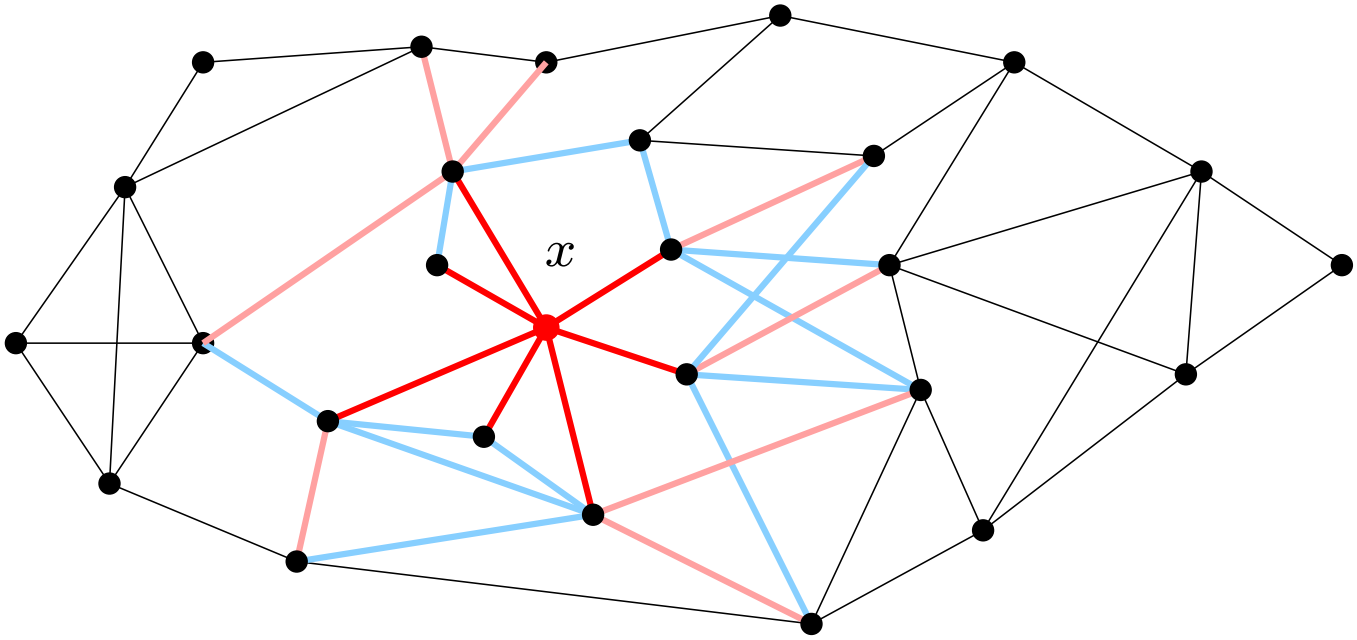
1. Motivation
2. Les arbres
3. Cas général
4. Conclusion

1. Motivation

Établissement d'une connexion dans un système distribué



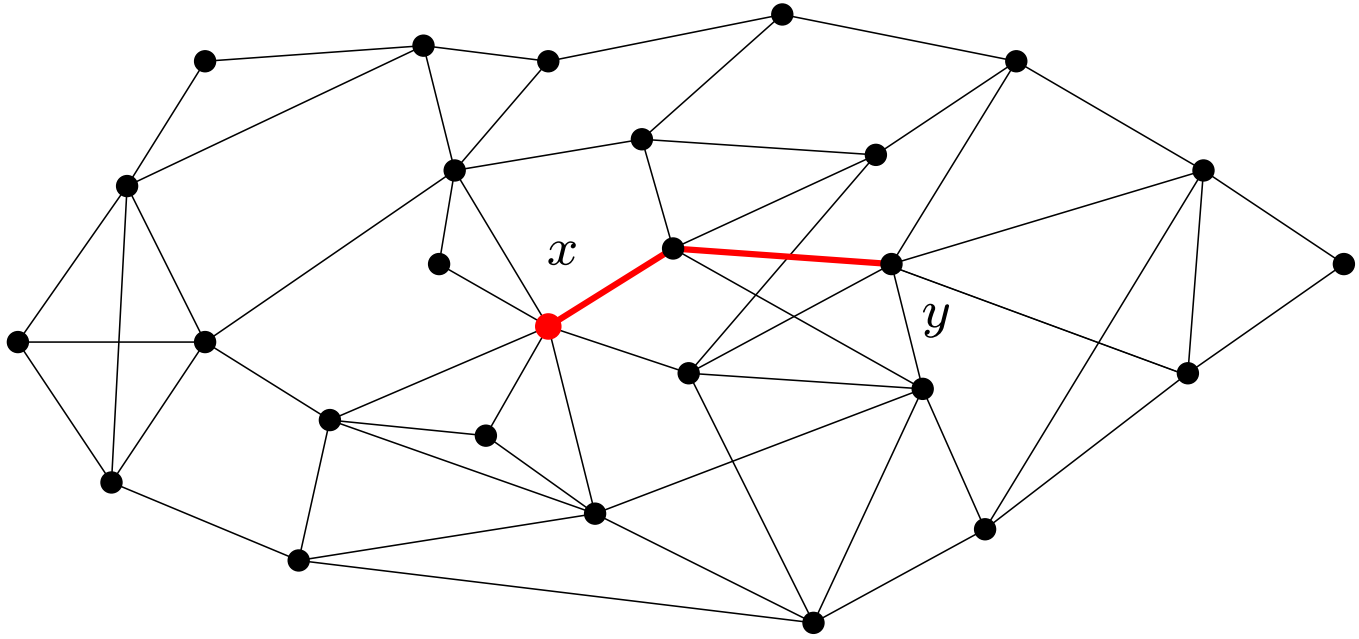
$$\text{coût}(x \longrightarrow y) = \underbrace{\text{coût(connexion)}}_{\text{trouver un chemin}} + \underbrace{\text{coût(transmission)}}_{\text{trans. des paquets}}$$



Solution 1 : choisir le chemin en diffusant un message de connexion depuis x.

$\boxed{+}$: plus courts chemins, donc rapide pour l'utilisateur

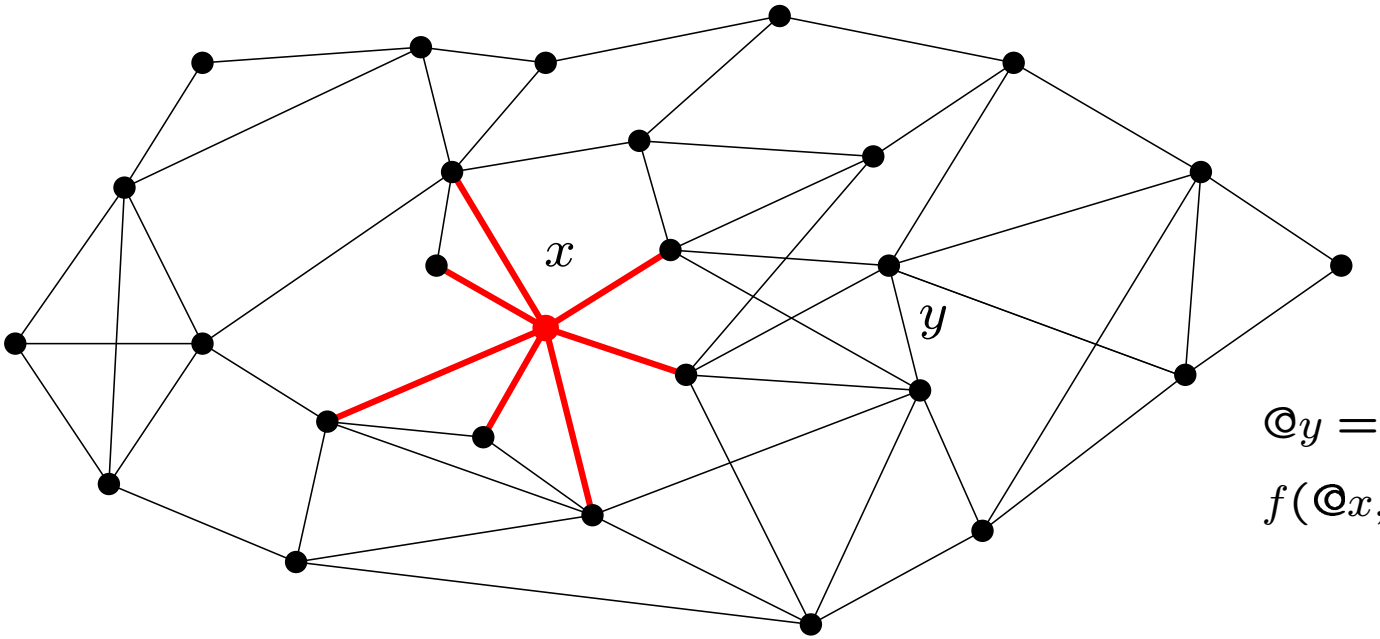
$\boxed{-}$: sature la bande passante du réseau, $O(m)$ messages même si $\mathrm{d}(x, y) = 2$ ($m = \#$ d'arêtes)



Solution 2 : on précalcule les routes de x à $\forall y$.

$\boxed{+}$: plus courts chemins, d(x, y) messages pour la connexion

$\boxed{-}$: besoin de précalculer les tables (mais faites 1 seule fois), stocker la table de routage $x \to \forall y$: coût mémoire de $n \log n$ bits, $n = \#$ sommets.



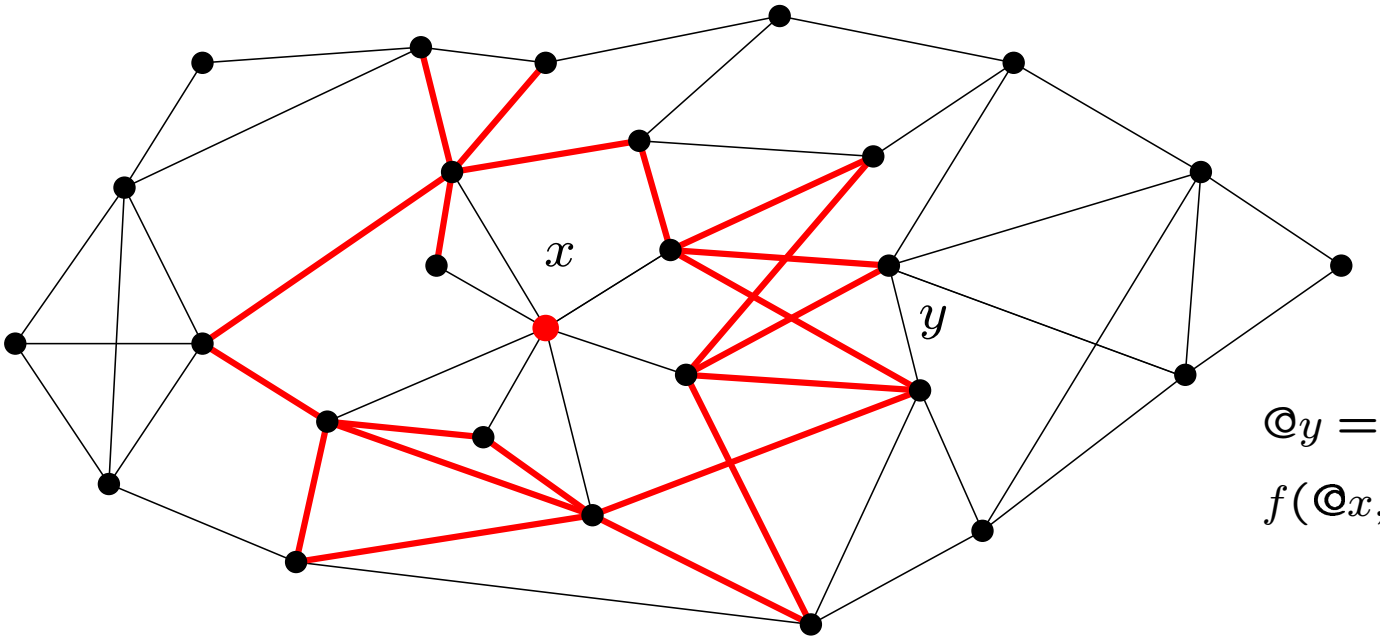
@y = adresse modifiée

$f($@$x,$ @$y) = \mathsf{d}(x, y)$

Solution 3 (hybride) : on précalcule une *étiquette* que l'on joint à chaque sommet permettant de calculer la distance. On diffuse jusqu'à la distance $d = \mathsf{d}(x, y)$ (ajoût d'un compteur dans les messages).

$\boxed{+}$: # messages réduit, plus courts chemins, pas de table

$\boxed{-}$: taille les étiquettes ?



@y = adresse modifiée

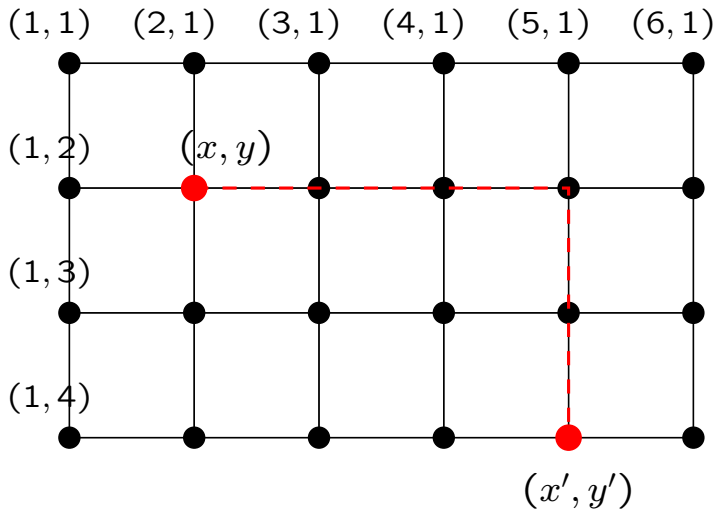
$f($@$x,$ @$y) = \mathrm{d}(x, y)$

Solution 3 (hybride) : on précalcule une *étiquette* que l'on joint à chaque sommet permettant de calculer la distance. On diffuse jusqu'à la distance $d = \mathrm{d}(x, y)$ (ajoût d'un compteur dans les messages).

$\boxed{+}$: # messages réduit, plus courts chemins, pas de table

$\boxed{-}$: taille les étiquettes ?

Étiquetage de distance (exemple)



$@X = (x, y)$

$@Y = (x', y')$

$$f(@X, @Y) = |x' - x| + |y' - y|$$

Étiquetage de distance (définition)

Soit $\mathcal{F}$ une famille de graphes.
(Ex. : $\mathcal{F}$ = arbres, graphes planaires, graphes de degré borné, ...)

Le couple (L, f) est un *étiquetage de distance* pour $\mathcal{F}$ si $\forall G \in \mathcal{F}$ et $\forall x \neq y$ sommets de G,

- $L(x, G) \in \{0, 1\}^*$ (fonction d'*étiquetage*)
- $f(L(x, G), L(y, G)) = \mathrm{d}_G(x, y)$ (fonction de *distance*)

Étiquetage de distance (définition)

Soit $\mathcal{F}$ une famille de graphes.
(Ex. : $\mathcal{F}$ = arbres, graphes planaires, graphes de degré borné, ...)

Le couple (L, f) est un *étiquetage de distance* pour $\mathcal{F}$ si $\forall G \in \mathcal{F}$ et $\forall x \neq y$ sommets de G,

- $L(x, G) \in \{0, 1\}^*$ (fonction d'*étiquetage*)
- $f(L(x, G), L(y, G)) = \mathrm{d}_G(x, y)$ (fonction de *distance*)

Objectifs :

1) minimiser la taille de $L(x, G)$ (en bits) ;
2) minimiser la complexité en temps de f ;
3) calcul des étiquettes

$\mathcal{F}_n$ = grilles rectangulaires à n sommets

Soit G une grille $a \times b$ à n sommets :

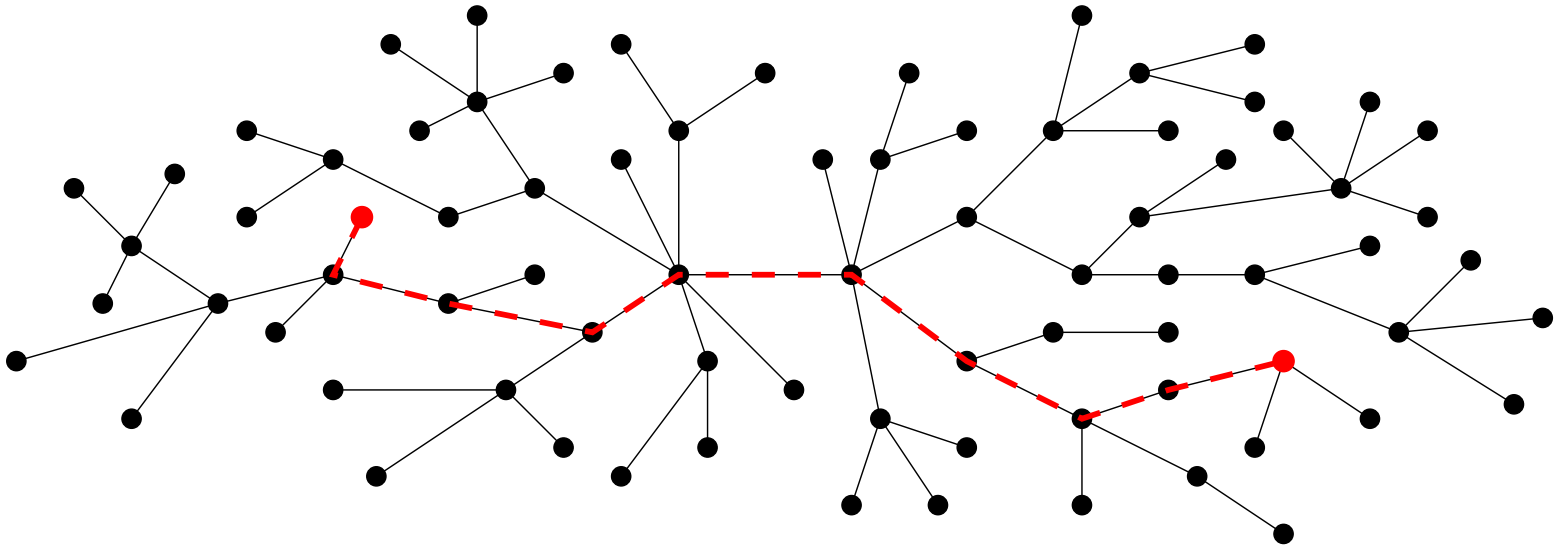
$$@X = (x, y) \in [1, a] \times [1, b] \quad \rightarrow \quad \{0, \ldots, n-1\}$$

$$f(@X, @Y) = |@X/a - @Y/a| + |@X\%a - @Y\%a|$$

Complexité en temps de f : $O(1)$ (constant)
Taille des étiquettes : $\lceil \log_2 n \rceil$ bits

2. Les arbres

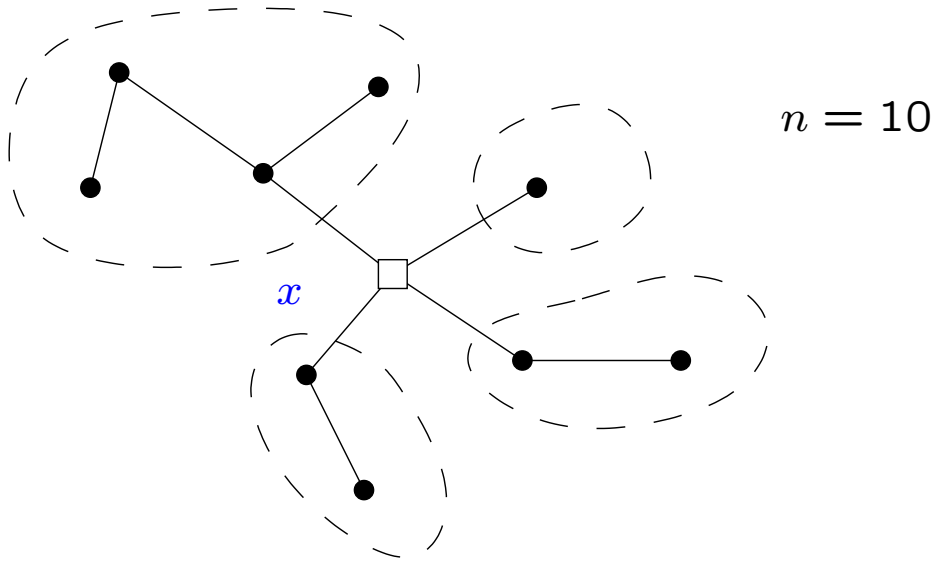
$\mathcal{F}_n$ = arbres à n sommets
(n est grand)



Séparateur

Le sommet x est un séparateur d'un arbre T à n sommets si toutes les composantes connexes de $T \setminus \{x\}$ ont au plus $n/2$ sommets.

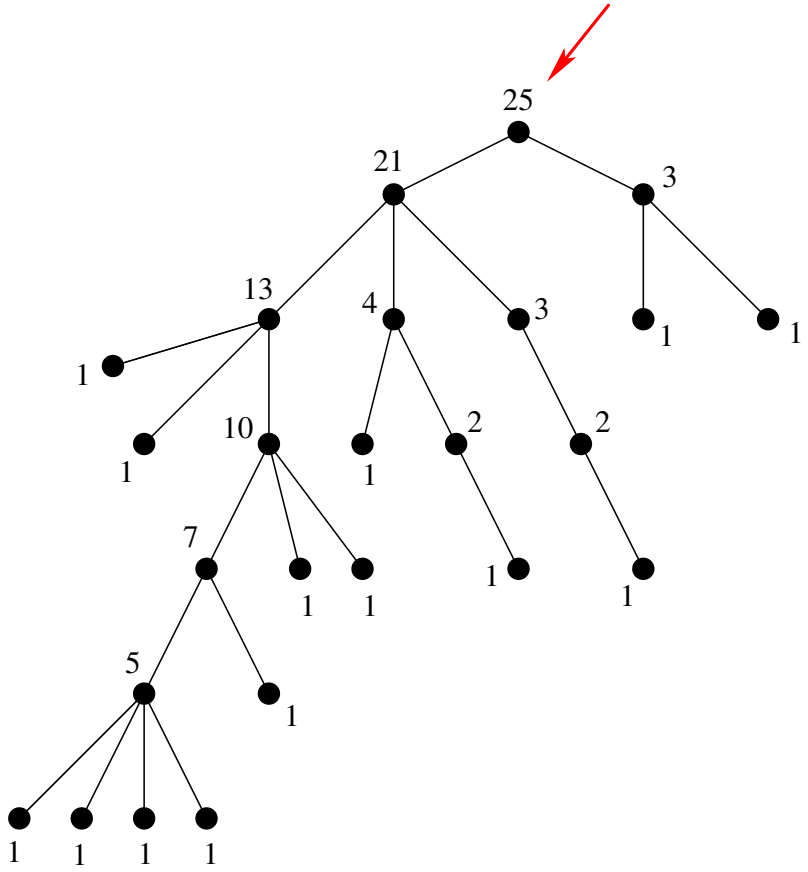
T



Tout arbre possède au moins 1 séparateur.

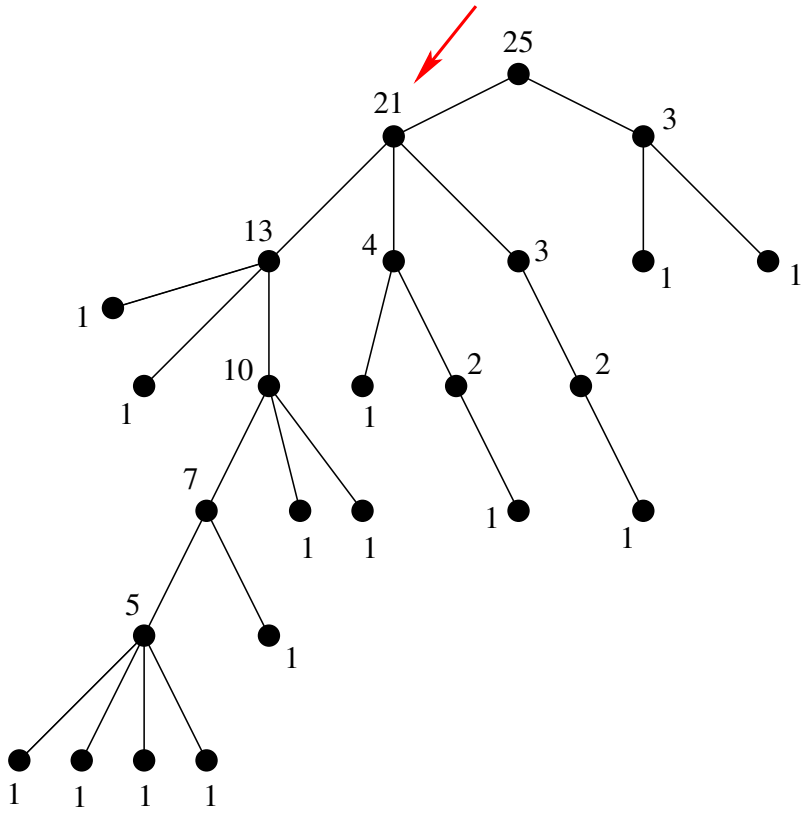
Comment trouver un séparateur ?

$n = 25$
$n/2 = 12.5$



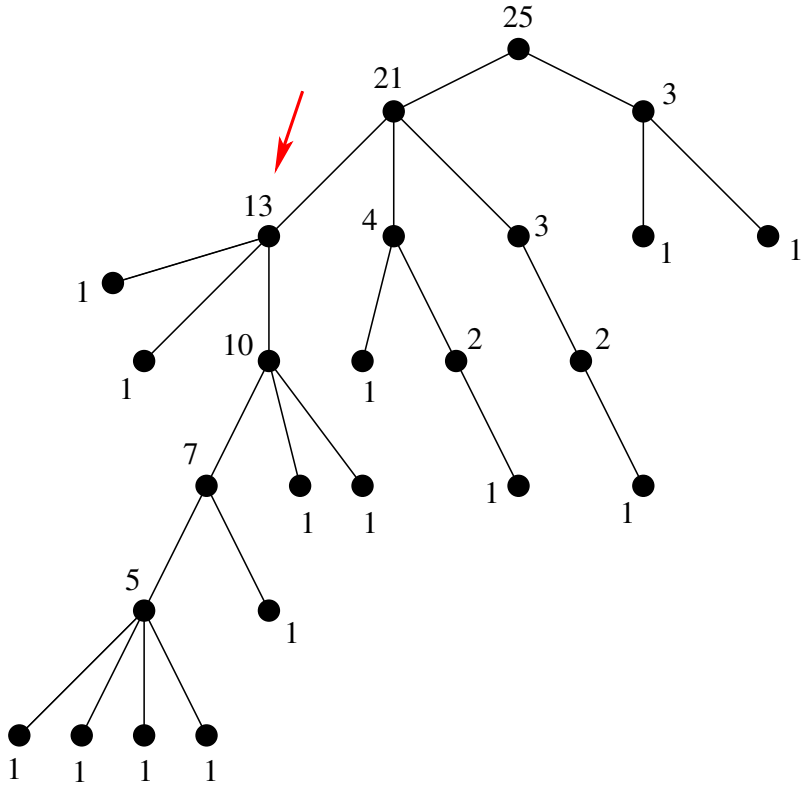
Comment trouver un séparateur ?

$n = 25$
$n/2 = 12.5$

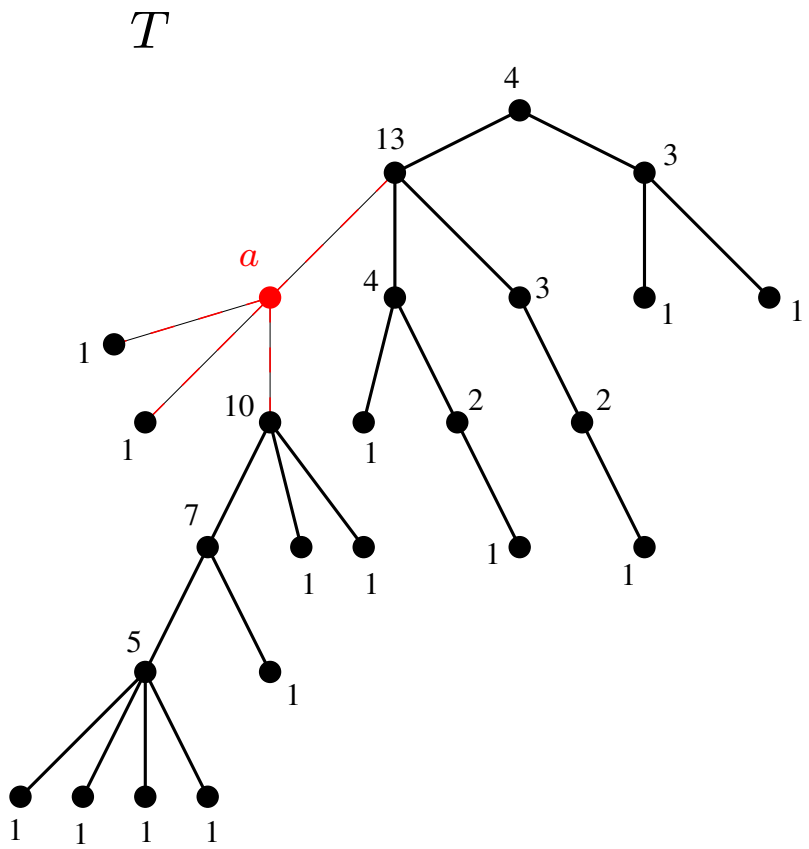


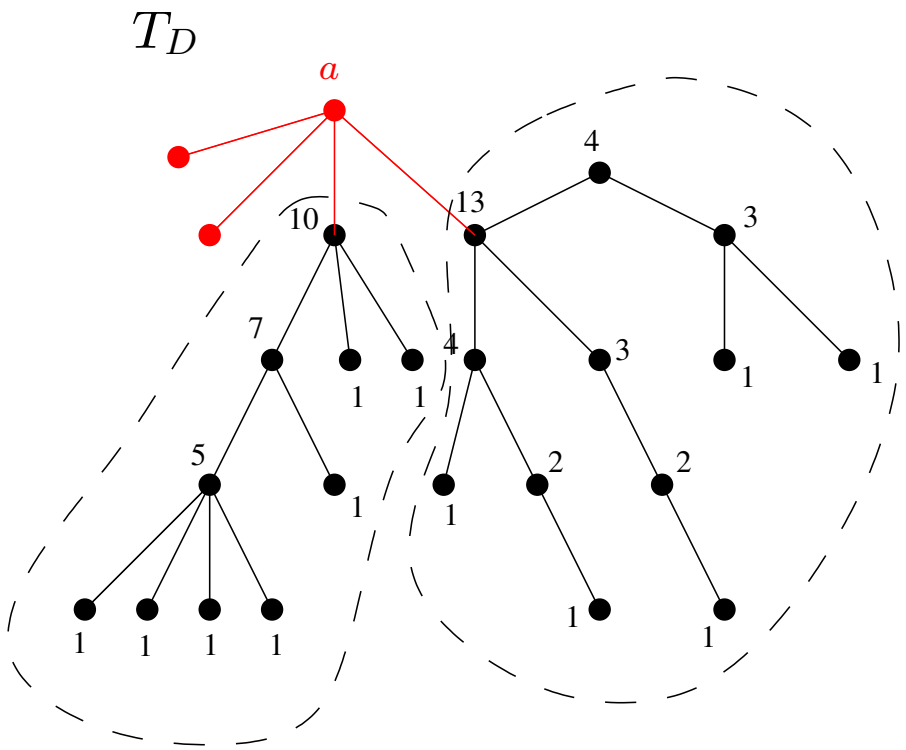
Comment trouver un séparateur ?

$n = 25$
$n/2 = 12.5$

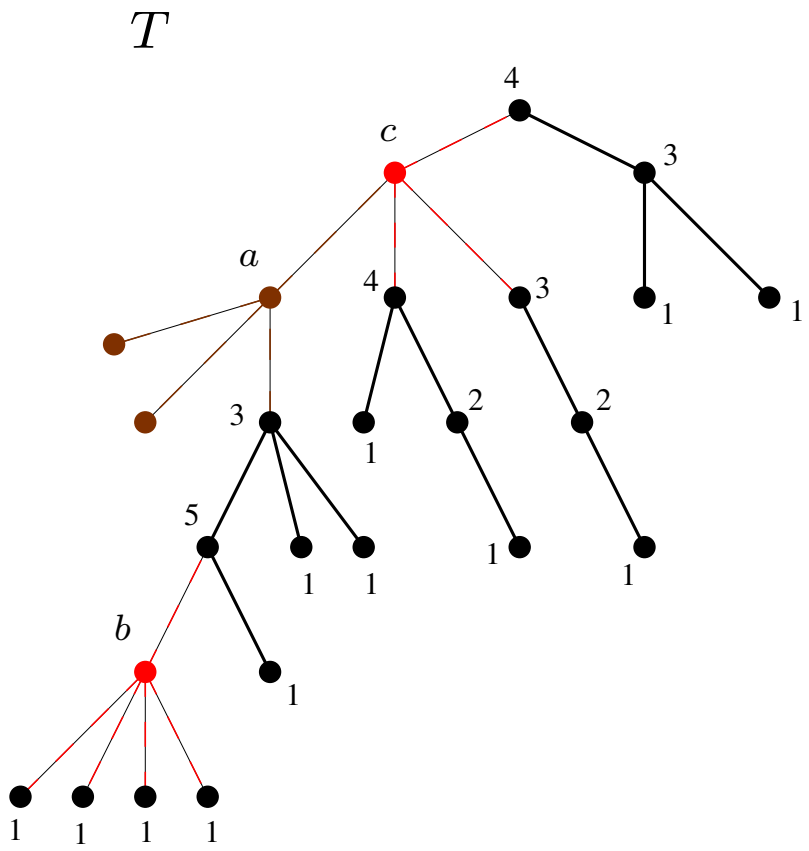


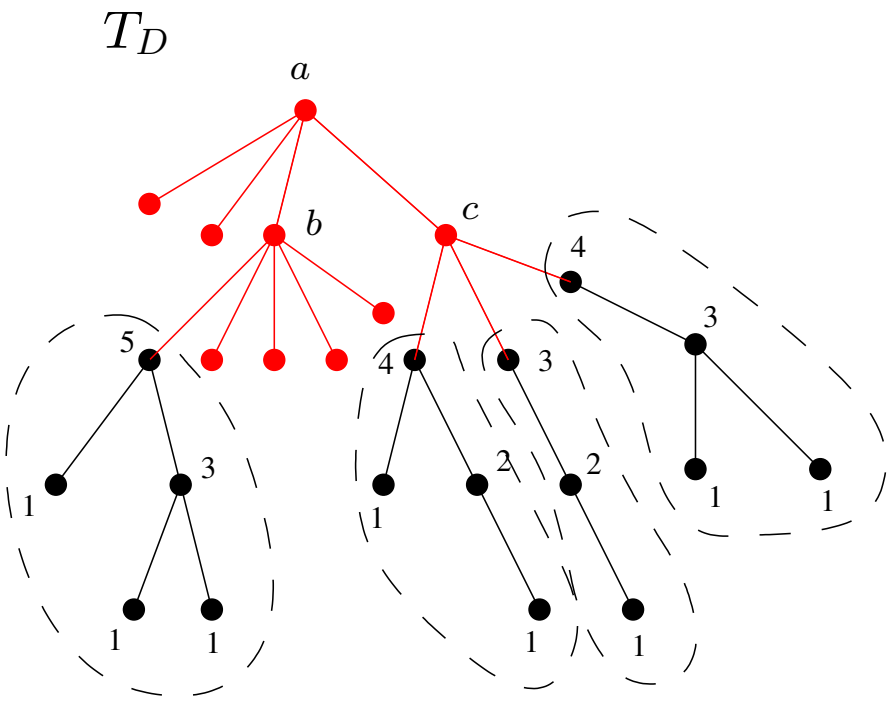
Arbre de décomposition : T_D (1/6)



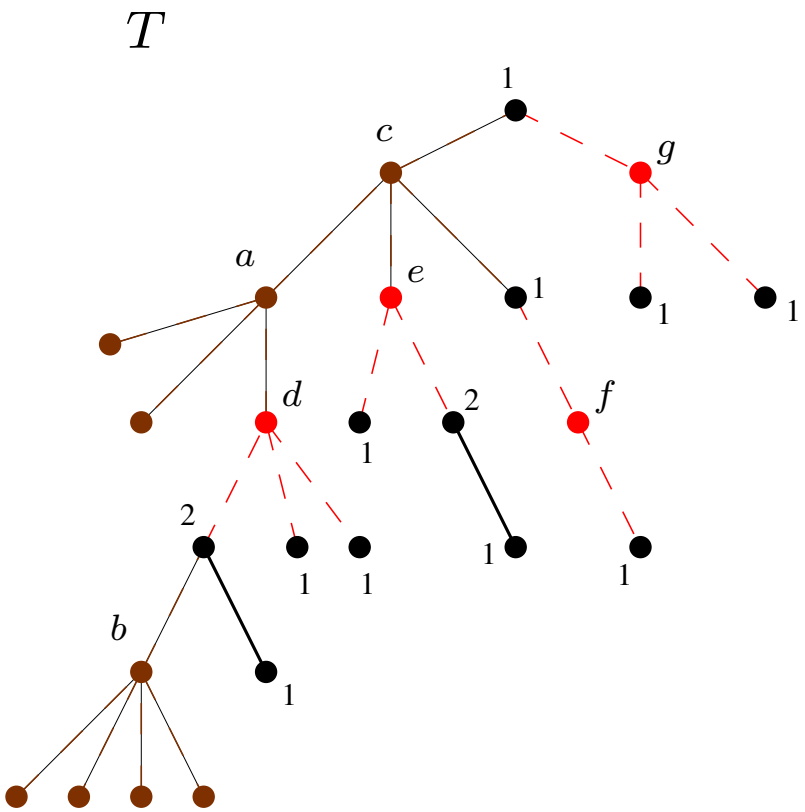


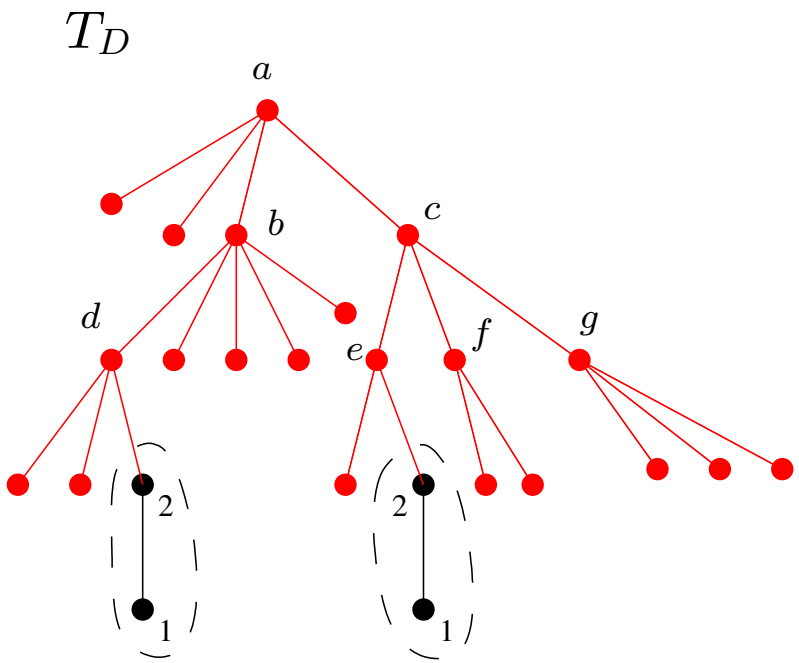
Arbre de décomposition : T_D (2/6)



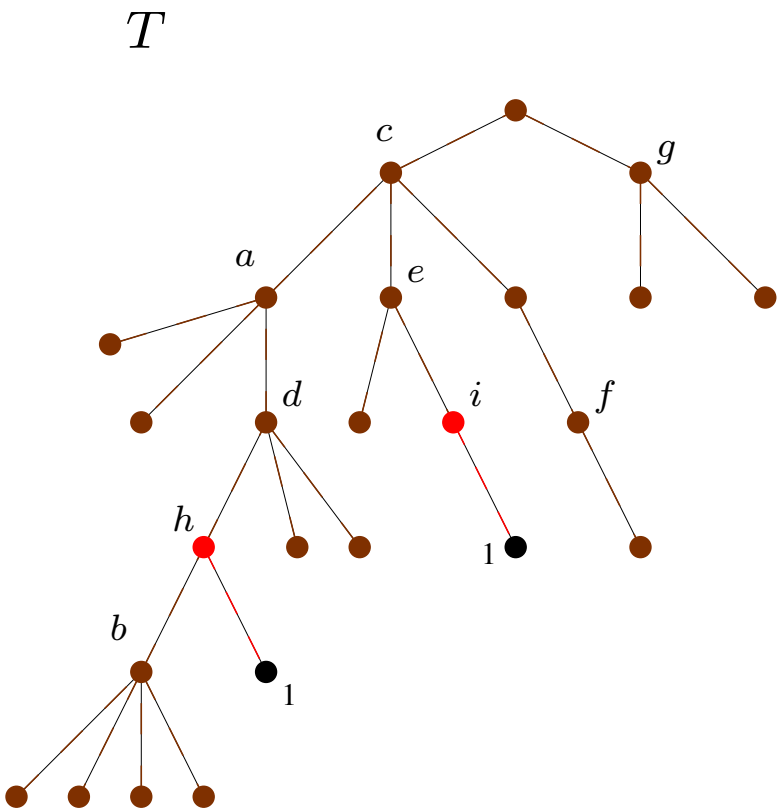


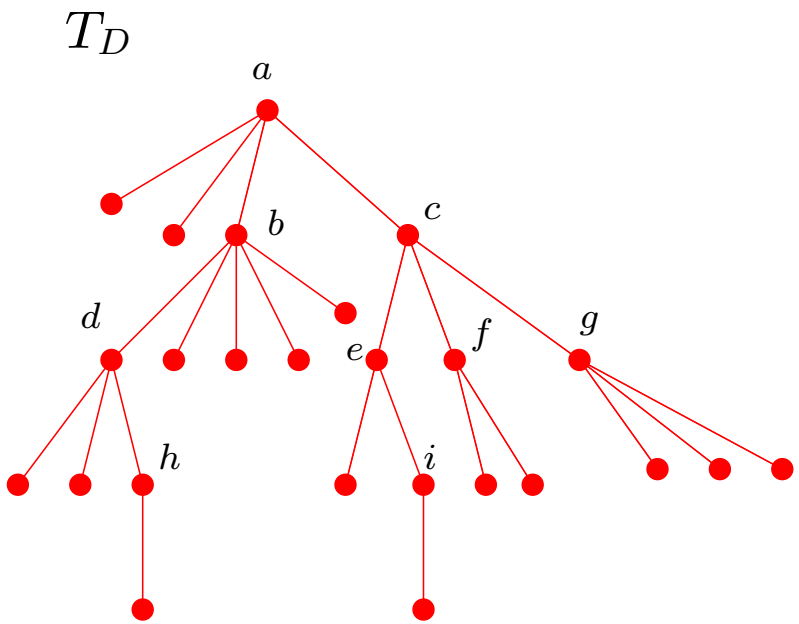
Arbre de décomposition : T_D (3/6)



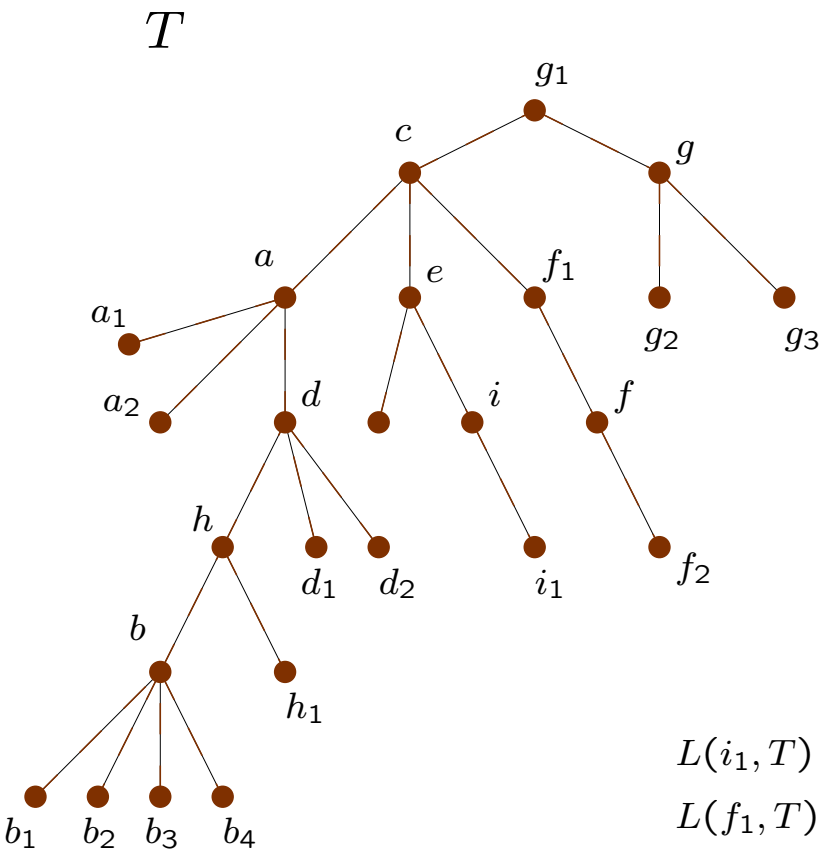


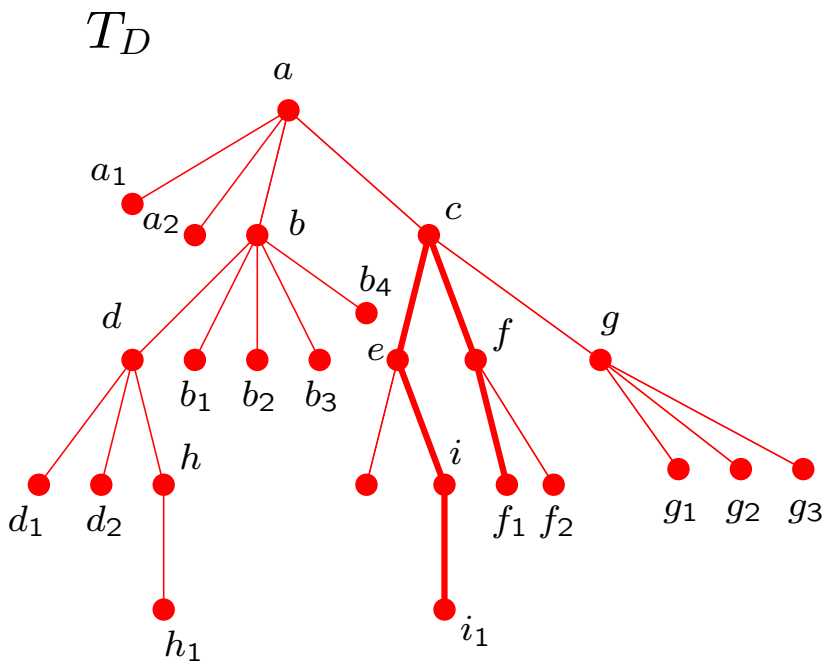
Arbre de décomposition : T_D (4/6)





Arbre de décomposition : T_D (5/6)



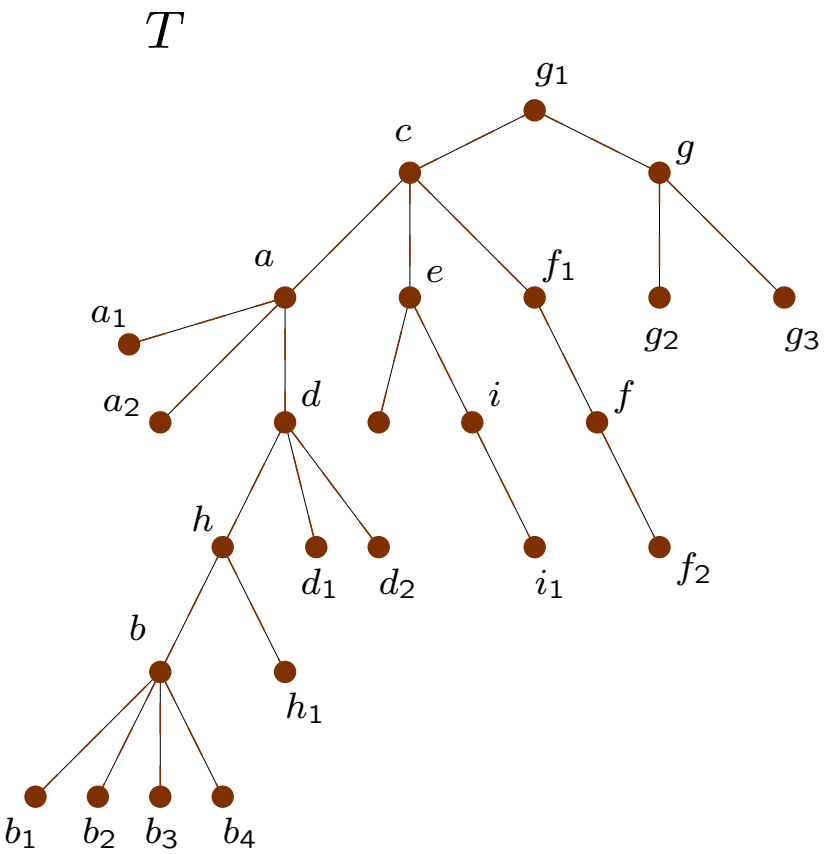


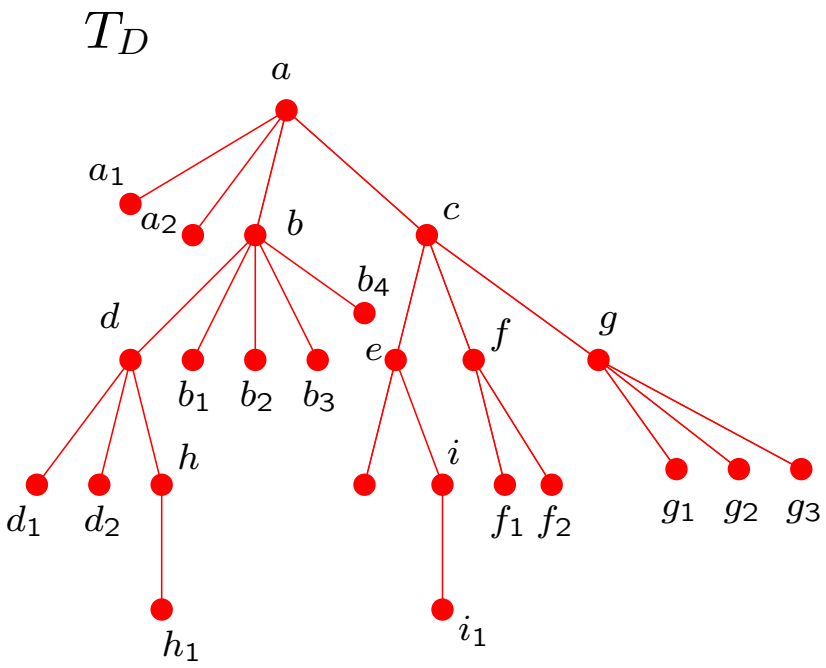
$L(i_1, T) = ((a, c, e, i), (4, 3, 2, 1))$

$L(f_1, T) = ((a, c, f), (2, 1, 1))$

$\mathsf{d}_T(i_1, f_1) = \mathsf{d}_T(i_1, c) + \mathsf{d}_T(f_1, c)$

Arbre de décomposition : T_D (6/6)





$h(n) \leqslant 1 + h(\lfloor n/2 \rfloor)$ et $h(0) = h(1) = 0$

$\Rightarrow h \leqslant \log n$

Étiquettes de x dans T

$L(x, T) = (S, D)$ avec

1) $h =$ hauteur de x dans T_D ;
2) $S = ()$, $D = ()$ pour la racine. $s_{h+1} = x$;
3) $S = (s_1, \ldots, s_h)$ ancêtres de x dans T_D, $s_1 =$ racine de T_D ;
4) $D = (d_1, \ldots, d_h)$ où $d_i = \mathrm{d}_T(x, s_i)$;

$f((S, D), (S', D')) =$

1) trouver le plus grand i_0 tel que $s_{i_0} = s'_{i_0}$;
2) retourner $d_{i_0} + d'_{i_0}$

Complexités

Taille des étiquettes : $2h \log n \leqslant 2(\log n)^2$ bits

Temps de calcul de f : $O(h) = O(\log n)$ opérations

***

Complexités

Taille des étiquettes : $2h \log n \leqslant 2(\log n)^2$ bits

Temps de calcul de f : $O(h) = O(\log n)$ opérations

***

Question : Peut-on faire mieux ?

Remarque : la complexité en temps de f provient du calcul de i_0.

Calcul de i_0 en temps $O(1)$

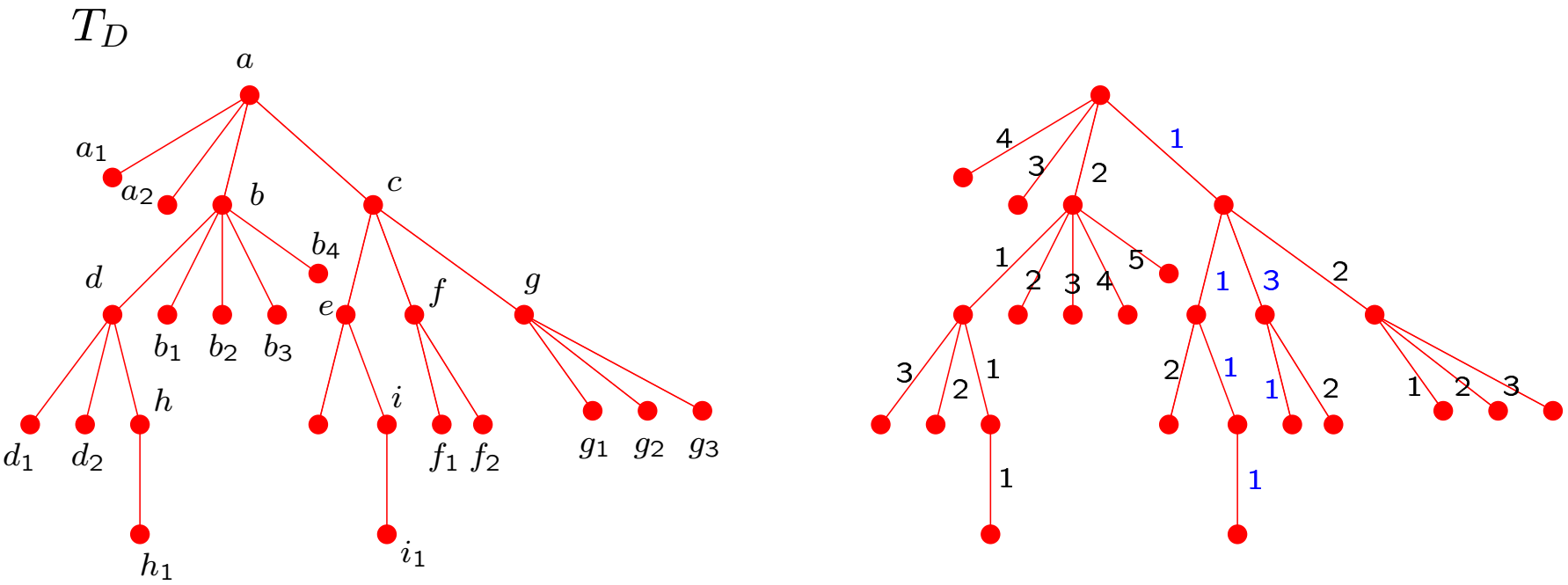
Modèle de calcul : Une opération arithmétique "standard" (addition entière, multiplication entière, décalage, opérations booléennes, ...) sur des registres de $\lceil \log n \rceil$ bits prend un temps constant.

Calcul de i_0 en temps $O(1)$

Modèle de calcul : Une opération arithmétique "standard" (addition entière, multiplication entière, décalage, opérations booléennes, ...) sur des registres de $\lceil \log n \rceil$ bits prend un temps constant.

Idée : Compresser la suite des séparateurs S en la représentant sur un nombre constant de registres, c'est-à-dire sur $O(\log n)$ bits, au lieu d'utiliser $h \log n = O(\log^2 n)$ bits. Puis effectuer un nombre constant d'opérations pour extraire i_0.

Renumérotation des arêtes de T_D



$L(i_1, T) = ((1, 1, 1, 1), (4, 3, 2, 1))$

$L(f_1, T) = ((1, 3, 1), (2, 1, 1))$

$$\boldsymbol{x \to C = (c_1, \ldots, c_h)}$$

$C =$ suite des numéros d'arête de la racine au père de x dans T_D.
$c_i =$ numéro de la composante triée par taille décroissante.
$c_i \geqslant 1$.

$$x \to C = (c_1, \ldots, c_h)$$

$C =$ suite des numéros d'arête de la racine au père de x dans T_D.
$c_i =$ numéro de la composante triée par taille décroissante.
$c_i \geqslant 1$.

Dans T_D, la composante numérotée c_1 à au plus n/c_1 sommets, car sinon

$$\sum_{c=1}^{c_1} \frac{n}{c} > \sum_{c=1}^{c_1} \frac{n}{c_1} = n$$

D'où,

la sous-composante numérotée c_2 (dans c_1) a au plus $(n/c_1)/c_2$ sommets.

...

la sous-composante numérotée c_h (dans c_{h-1}), qui contient exactement 1 sommet (c'est-à-dire x), a au plus

$$\frac{n}{\prod_{i=1}^{h} c_i} \geqslant 1 \quad \text{sommets} \quad \Rightarrow \quad \prod_{i=1}^{h} c_i \leqslant n$$

Codage de C sur $O(\log n)$ bits

$(c_1, \ldots, c_h) \longrightarrow (A, B)$ deux chaînes binaires de k bits.

A est la concaténation des écritures binaires de chaque c_i (en fait de $c_i - 1 \in \{0, 1, 2, \ldots\}$), et B indique la position du début de chaque valeur c_i dans A.

Ex. : $C = (4, 10, 3, 1, 6, 3, 5, 7) \longrightarrow (A, B)$

$c_i - 1 =$	3	9	2	0	5	2	4	6
$A =$	11	1001	10	0	101	10	100	110
$B =$	10	1000	10	1	100	10	100	100

$$k \leqslant \sum_{i=1}^{h} \lceil \log c_i \rceil \leqslant \log \left(\prod_{i=1}^{h} c_i \right) + h \leqslant 2 \log n$$

Calcul de i_0

Ex. : $k = 10$, deux sommets $x \to C = (A, B)$ et $x' \to C' = (A', B')$

```
                    ... 4  3  2   1  0                          ... 4  3   2  1  0
A  = | 0  1  0 | 0 | 1  1  1  0 | 0  1 |      A' = | 0  1  0 | 0 | 1  1  1 | 0 | 0 | 1 |
B  =   1  0  0   1   1  0  0  0   1  0          B' =   1  0  0   1   1  0  0   1   1   1
```

```
p=A xor A';                          (p=0000000000)
si p=0, a=0 sinon a=LOG2(p);         (a=0)
p=B xor B';                          (p=0000000101)
si p=0, b=0 sinon b=LOG2(p);         (b=2)
p=max{a,b};                          (p=2)
```

On trouve i_0 en comptant le nombre de 1 dans B (ou B') situés à gauche du bit numéro `p`. Ici, $i_0 = 3$: C et C' diffèrent à partir de la 3^{e} composante.

Calcul du nombre de 1 à gauche d'une position p

Soit $f_B(p)$ la fonction qui pour toute position p retourne le nombre de 1 dans B situés à gauche de p, p non compris. B est fixe pour le sommet x. Par contre p dépend des autres sommets.

Il est possible de précalculer 3 registres de k bits tels que $f_B(p)$ puisse être évalué en temps constant.

$k = 18,\ k_1 = 6,\ k_2 = 2$

B	11	01	11	01	10	01	11	10	11
R_2	0	2	3	0	1	2	0	2	3
R_1			0			5			8

R_3	0	1
00	0	0
01	0	0
10	1	0
11	1	0

Choisir : $k_1 \approx \log k$ et $k_2 \approx \log k_1$

Taille de R_1 : $(k/k_1) \cdot \log k = k$ bits
Taille de R_2 : $(k/k_2) \cdot \log k_1 = k$ bits
Taille de R_3 : $2^{k_2} \cdot k_2 \cdot \log k_2 < k$ bits.

$k = 18,\ k_1 = 6,\ k_2 = 2$

B	11	01	11	01	10	01	11	10	11
R_2	0	2	3	0	1	2	0	2	3
R_1			0			5			8

R_3	0	1
00	0	0
01	0	0
10	1	0
11	1	0

Choisir : $k_1 \approx \log k$ et $k_2 \approx \log k_1$

Taille de R_1 : $(k/k_1) \cdot \log k = k$ bits
Taille de R_2 : $(k/k_2) \cdot \log k_1 = k$ bits
Taille de R_3 : $2^{k_2} \cdot k_2 \cdot \log k_2 < k$ bits.

Remarque : on peut faire avec 1 seul registre de $o(k)$ bits.

... donc

avec des étiquettes de taille au plus $(\log n)^2 + O(\log n)$ bits, on peut calculer la distance en un temps constant dans un arbre à n sommets.

***

... donc

avec des étiquettes de taille au plus $(\log n)^2 + O(\log n)$ bits, on peut calculer la distance en un temps constant dans un arbre à n sommets.

***

Question : Peut-on faire mieux ?

Théorème. *Tout étiquetage de distance sur la famille des arbres à n sommets nécessite une étiquette de taille au moins*

$$\frac{1}{8}(\log n)^2 - O(\log n) \quad \textit{bits.}$$

Idée de la preuve ...

Étant donné un étiquetage de distance $E = (L, f)$ sur $\mathcal{F}_n = \{$ arbres $\}$, on définit

$$\mathcal{L}(E) = \{L(x, T) \mid x \in T, \quad T \in \mathcal{F}_n\}\,.$$

Remarque : $\mathcal{L}(E)$ est un ensemble de chaînes binaires.

Soit E^* un étiquetage de distance sur $\mathcal{F}_n$ tel que $\mathcal{L}(E^*)$ soit de cardinalité minimale. On pose $c = |\mathcal{L}(E^*)|$. Alors,

Lemme. *$\exists\ T_0 \in \mathcal{F}_n$ et $x_0 \in T_0$ tels que $L(x_0, T_0)$ est de taille au moins* $\log(c+1) - 1$ *bits.*

Lemme. *$\exists\ T_0 \in \mathcal{F}_n$ et $x_0 \in T_0$ tels que $L(x_0, T_0)$ est de taille au moins $\log(c+1) - 1$ bits.*

Preuve. Soit $t = \log(c+1) - 1$. Supposons que $\forall \ell \in \mathcal{L}(E^*)$, ℓ est de taille $\leqslant t-1$. Alors

$$|\mathcal{L}(E^*)| \ \leqslant \ \sum_{i=0}^{t-1} 2^i = 2^t - 1 = 2^{\log(c+1)-1} - 1 = c - 1,$$

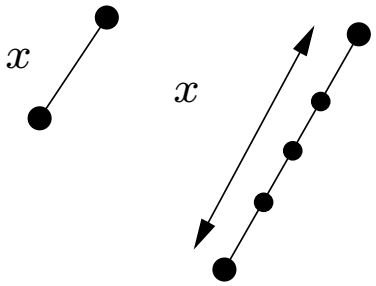
contradiction. Donc $\exists \ell_0 \in \mathcal{L}(E^*)$ de taille $\geqslant t$. $\mathcal{L}(E^*)$ minimal donc ℓ_0 est associé à au moins un sommet dans au moins un arbre de $\mathcal{F}_n$. □

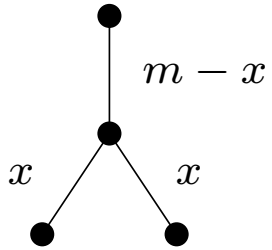
Donc pour prouver le Théorème il suffit de calculer $|\mathcal{L}(E^*)|$ ...

Idée de la preuve (suite)

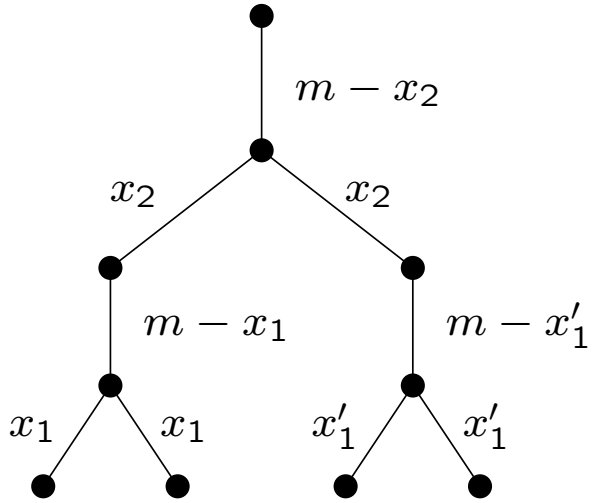
sous-famille $\mathcal{T}_{h,m}$

$m \sim \sqrt{n}$
$h \sim \log \sqrt{n}$





un arbre de $\mathcal{T}_{1,m}$



total d'étiquettes pour tous les arbres $\mathcal{T}_{h,m}$ est au moins $m^{h/2}$

	1	2	3	...	n
T_1	ℓ_1	ℓ_2	ℓ_3	...	
T_2	ℓ_i	ℓ_1	ℓ_j	...	
T_3	ℓ_4	ℓ_i	ℓ_k	...	
$\vdots$		...	ℓ_t	...	

$\Rightarrow \exists\ \ell_0$ de taille $\frac{h}{2} \log m \ \geqslant\ \frac{1}{8} \log^2 n$ bits.

Compression vs. qualité

Peut-on battre $\frac{1}{8}(\log n)^2$ bits, quitte à perdre sur la qualité de l'estimateur de distance ? (approximation de la distance)

Définitions

1) Un étiquetage de distance (L, f) est s-approximé pour une famille $\mathcal{F}$ si $\forall G \in \mathcal{F}$, $\forall x \neq y \in G$,

$$\mathrm{d}_G(x, y) \leqslant f(L(x, G), L(y, G)) \leqslant s \cdot \mathrm{d}_G(x, y)$$

2) Un couple de fonctions (λ, δ) est un (s, n)-estimateur de taille k si $\forall x \in \{1, \ldots, n\}$, $\lambda(x) \in \{0, 1\}^k$ et $\delta : \{0, 1\}^k \to \mathbb{N}$ tels que

$$x \leqslant \delta(\lambda(x)) \leqslant s \cdot x$$

(λ: compression, δ: décodeur)

Généralisation de l'étiquetage "exact"

$L(x, T) = (C, D)$, $D = (d_1, \ldots, d_h)$, $d_i \in \{1, \ldots, n\}$

$$(C, D) \longrightarrow (C, \tilde{D})$$

Soit (λ, δ) un (s, n)-estimateur de taille k.
$\tilde{D} = (\lambda(d_1), \ldots, \lambda(d_h))$ sur $kh = k \log n$ bits

Taille des nouvelles étiquettes : $k \log n + O(\log n)$ bits

Qualité de l'approximation

Soit $x, x' \in T$ et $(C, \tilde{D})$ et $(C', \tilde{D}')$ leurs étiquettes.
On a $\mathsf{d}_T(x, x') = \mathsf{d}_T(x, s_{i_0}) + \mathsf{d}_T(s_{i_0}, x')$.

Donc

$$\delta(\lambda(d_{i_0})) + \delta(\lambda(d_{i_0})) \;\leqslant\; s \cdot \mathsf{d}_T(x, s_{i_0}) + s \cdot \mathsf{d}_T(s_{i_0}, x') \;\leqslant\; s \cdot \mathsf{d}_T(x, x')$$

$\Rightarrow$ étiquetage s-approximé

(s, n)-estimateur de taille k

$$x = 10101011$$

Estimateur I : $\lambda(x) = \text{bin}(x)$ (écriture binaire de l'entier x)

$\Rightarrow k = \lceil \log n \rceil$ bits et $s = 1$

Estimateur II : $\lambda(x) = \lfloor \log x \rfloor$ = position du bit le plus à gauche dans bin(x). Si $x \in \{1, \ldots, n\}$ alors $\lambda(x)$ représentable sur $\lceil \log \log n \rceil$ bits. $\delta(\lambda(x)) = 2^{\lambda(x)+1} - 1 = 11111111$
$x \leqslant \delta(\lambda(x)) \leqslant x + x - 1 \leqslant 2x$.

$\Rightarrow k = \lceil \log \log n \rceil$ bits et $s = 2$

$$x = 10101011$$

Estimateur III : $\lambda(x) = (b, p)$ avec
$b = \text{bin}(x, \lceil \log \log n \rceil)$ et $p = \max\{\lfloor \log x \rfloor - \lceil \log \log n \rceil, 0\}$

Ex. : $\lambda(x) = (\text{"}101\text{"}, 7-3)$ pour $n = 2^8$. $\delta(\lambda(x)) = b \cdot 2^{p+1} + 2^{p+2} - 1$
Ici $\delta(\lambda(x)) = 10100000 + 11111 = 10111111$

$x \leqslant \delta(\lambda(x)) \leqslant x + 2^{\lfloor \log x \rfloor - \lceil \log \log n \rceil} - 1 \leqslant x + x/\log n.$

$\Rightarrow k = 2\lceil \log \log n \rceil$ bits et $s = 1 + 1/\log n$.

... donc

Avec des étiquettes de taille au plus $2 \log n \log \log n + O(\log n)$ bits, on peut approximer la distance à un facteur $1 + 1/\log n$ près en un temps constant dans un arbre à n sommets.

... donc

Avec des étiquettes de taille au plus $2 \log n \log \log n + O(\log n)$ bits, on peut approximer la distance à un facteur $1 + 1/\log n$ près en un temps constant dans un arbre à n sommets.

... et on ne peux pas faire mieux.

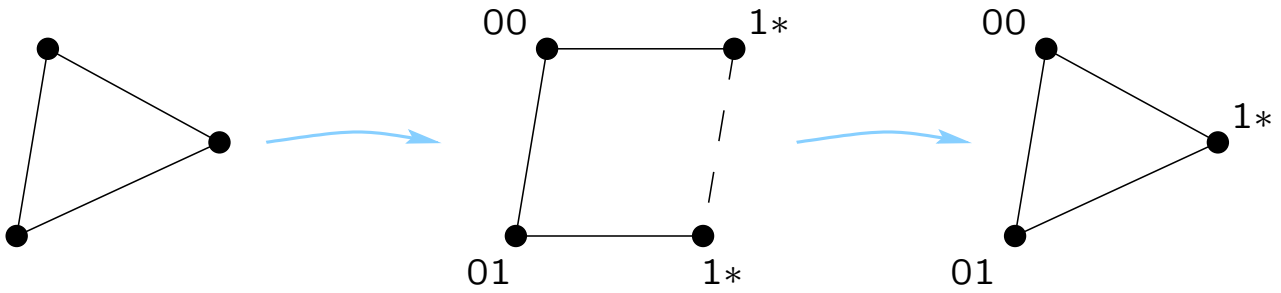
Théorème. *Tout étiquetage de distance* $(1 + 1/\log n)$*-approximé sur la famille des arbres à* n *sommets nécessite une étiquette de taille au moins*

$$c \log n \log \log n \quad \textit{bits}$$

pour une certaine constante $c > 0$.

3. Cas général

Plongement de G dans un Hypercube



$q = 2$

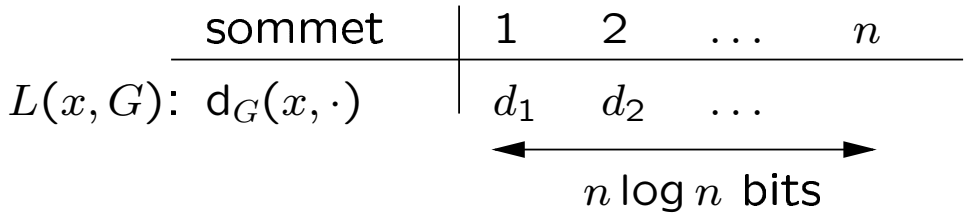
Étiquette : $L(x, G) \in \{0, 1, *\}^q$

Distance: distance de Hamming avec $d(0, *) = d(1, *) = d(*, *) = 0$

Graham-Pollak 1978 : La *cube-dimension* d'un graphe connexe G est au moins r, où r est le maximum du nombre de valeurs propres positives et de valeurs propres négatives de la matrice de distances de G. (Conjecture: $q_{opt} \leqslant n - 1$.)

Bornes générales

Étiquettes en $n \log n$ bits avec un temps constant pour le calcul de la distance :



Winkler en 1984, a prouvé la conjecture de Graham-Pollack ($q_{opt} \leqslant n - 1$) $\Rightarrow$ des étiquettes de $n \log_2(3) \approx 1.58n$ bits, mais un temps en $O(n)$ pour le calcul de la distance.

On peut montrer $11n$ bits et un temps en $O(\log \log n)$

4. Conclusion

De nombreux problèmes sont encore ouverts pour familles des graphes particulières : graphes planaires par exemple (taille optimale des étiquettes comprises entre $\sqrt[3]{n}$ et $\sqrt{n}$ bits), graphes définis sur les groupes (graphes de Cayley), ...

Pour en savoir plus ...

dept-info.labri.fr/~gavoille