Oracles pour les arbres et les graphes

Cyril Gavoille

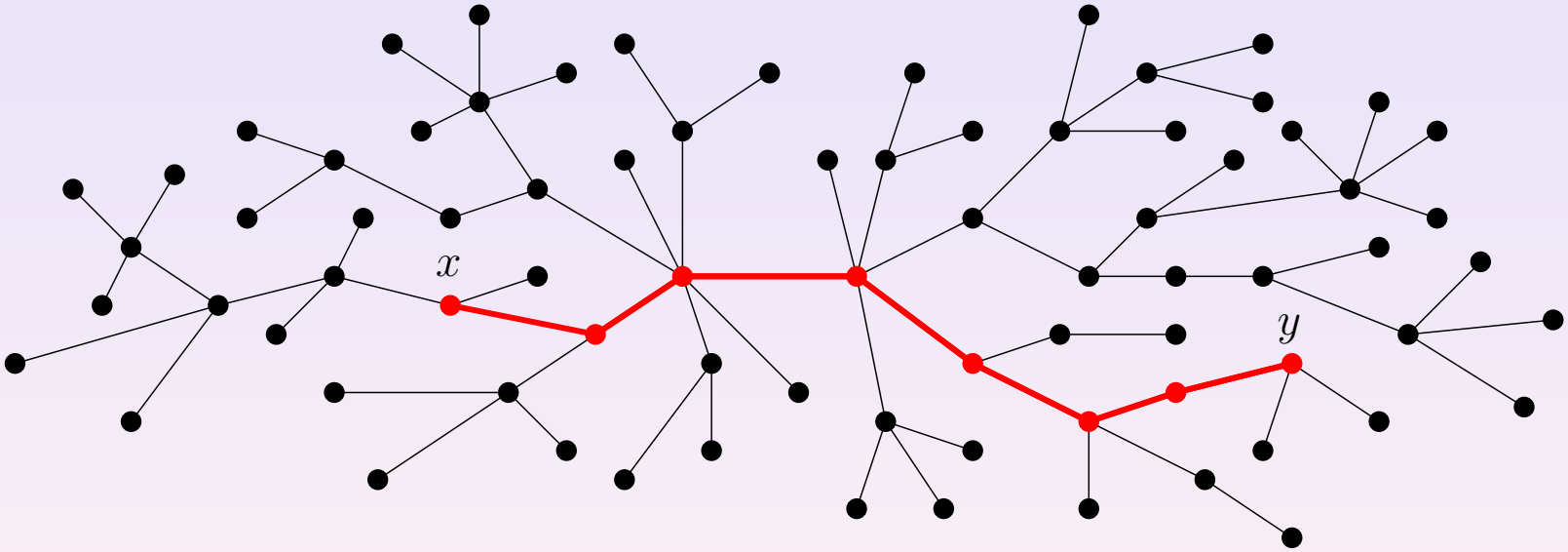
LaBRI - Université de Bordeaux - IUF

Séminaire MaMux - IRCAM - 20 mai 2011

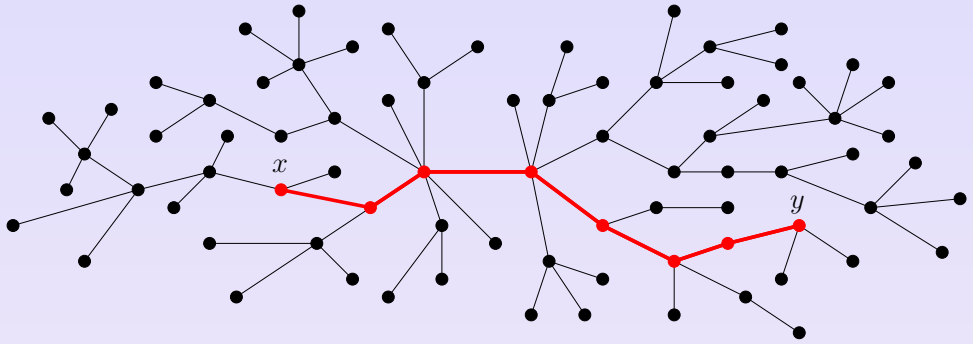
(Mathématiques, musique et relations avec d'autres disciplines)

Quels types de problèmes ?

Étant donné un arbre T à n nœuds répondre rapidement à des requêtes de **distances** : « Quelle est la longueur du chemin reliant le nœud x au nœud y dans T ? »



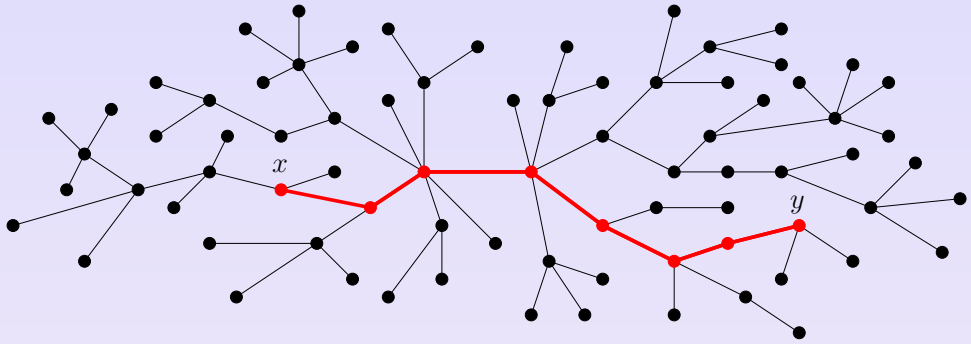
Problème de structures de données



- **Sans pré-calcul :** réponse en temps $O(n)$
  [Ex : parcourir T à partir de x]

  $\Rightarrow$ pas assez rapide !

Problème de structures de données



- **Sans pré-calcul :** réponse en temps $O(n)$
  [Ex : parcourir T à partir de x]

  $\Rightarrow$ pas assez rapide !
- **Avec pré-calcul :** réponse en temps constant
  [Ex : pré-calculer la matrice des distances]

  $\Rightarrow$ pré-calcul trop coûteux ! espace en $O(n^2)$

Oracle de distances

Un **oracle de distances** est une structure de données permettant de répondre rapidement à des requêtes de distances.

On souhaite :

- temps de réponse constant (idéal), voir $\mathrm{polylog}(n)$
- pré-calcul linéaire en n (idéal), voir $n \cdot \mathrm{polylog}(n)$

Oracle de distances

Un **oracle de distances** est une structure de données permettant de répondre rapidement à des requêtes de distances.

On souhaite :

- temps de réponse constant (idéal), voir $\mathrm{polylog}(n)$
- pré-calcul linéaire en n (idéal), voir $n \cdot \mathrm{polylog}(n)$

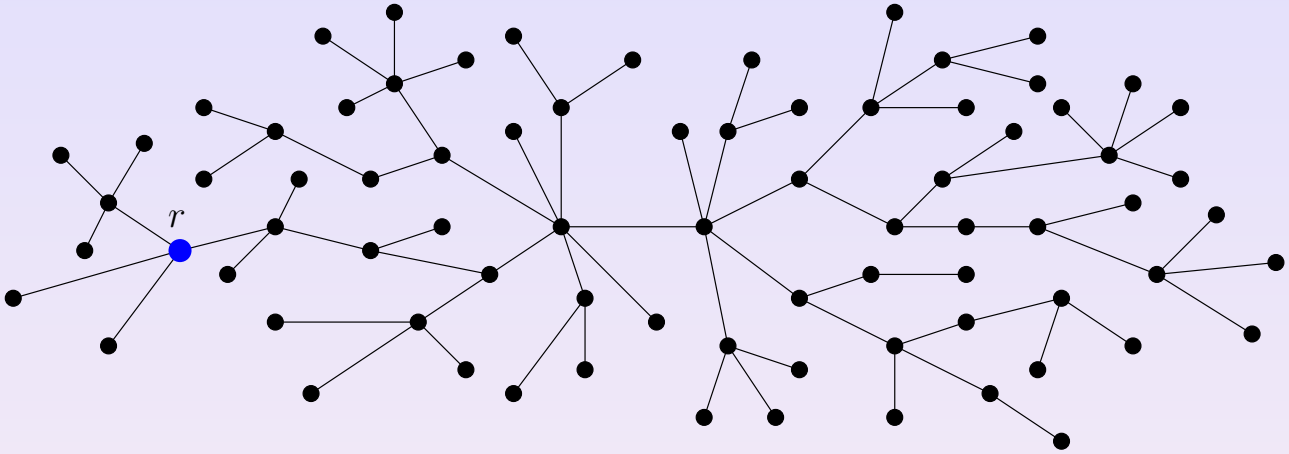
$\Rightarrow$ taille de la structure de données linéaire ou $n \cdot \mathrm{polylog}(n)$

Un oracle simple pour les arbres

Idée générale : compresser et « localiser » les informations

Un oracle simple pour les arbres

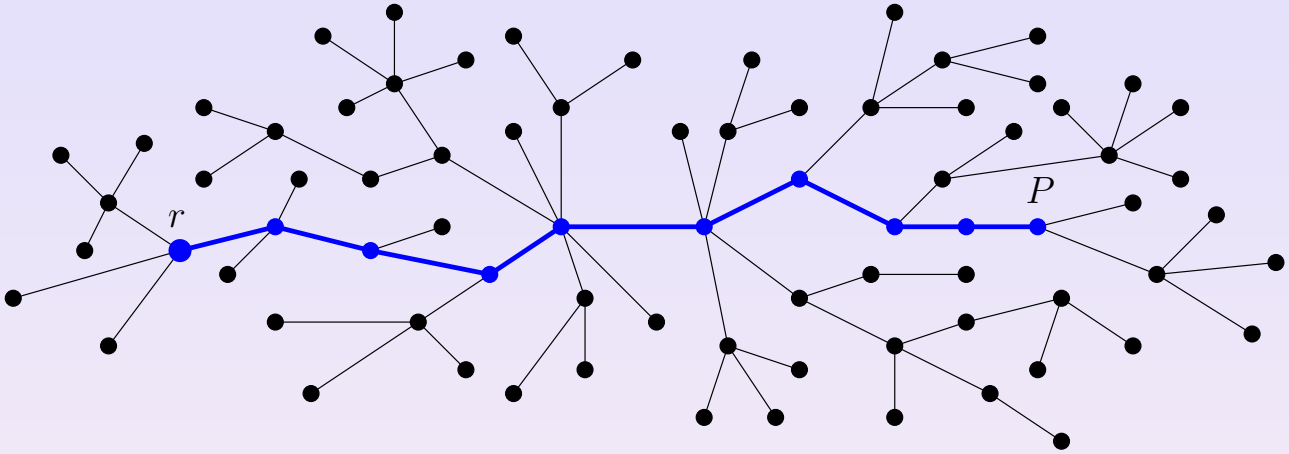
Idée générale : compresser et « localiser » les informations



1. Choisir un nœud r arbitraire comme racine de T

Un oracle simple pour les arbres

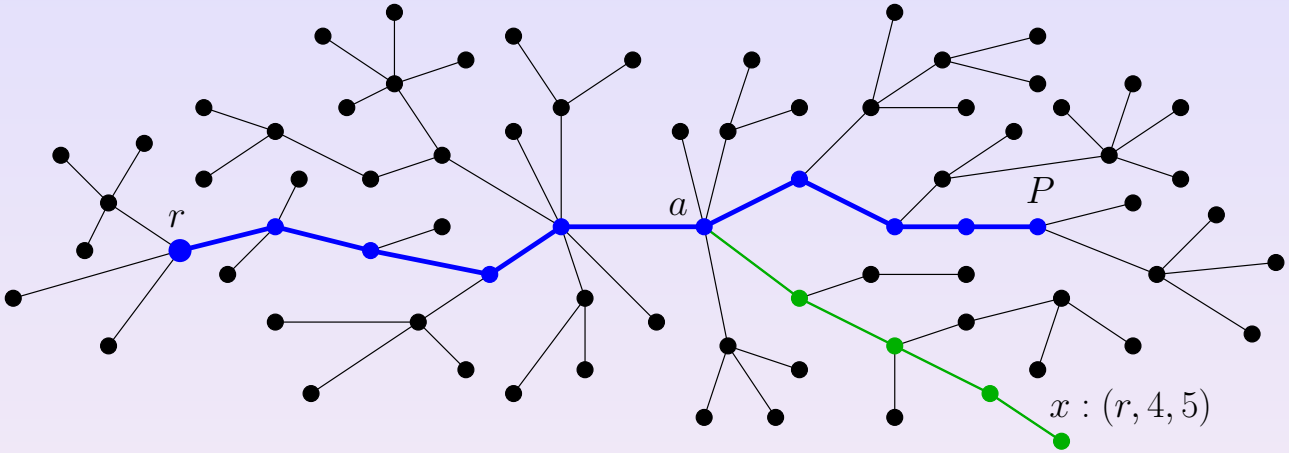
Idée générale : compresser et « localiser » les informations



1. Choisir un nœud r arbitraire comme racine de T
2. Prendre un chemin P qui « coupe en deux » T

Un oracle simple pour les arbres

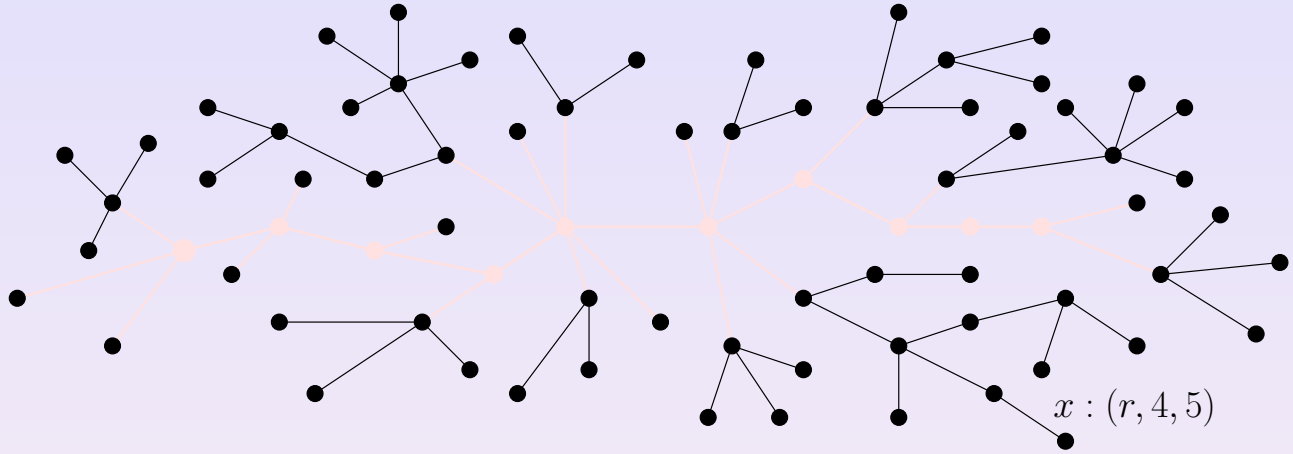
Idée générale : compresser et « localiser » les informations



1. Choisir un nœud r arbitraire comme racine de T
2. Prendre un chemin P qui « coupe en deux » T
3. Le nœud x stocke (r, d, h) où $d = d_T(x, a)$ et $h = d_T(r, a)$

Un oracle simple pour les arbres

Idée générale : compresser et « localiser » les informations



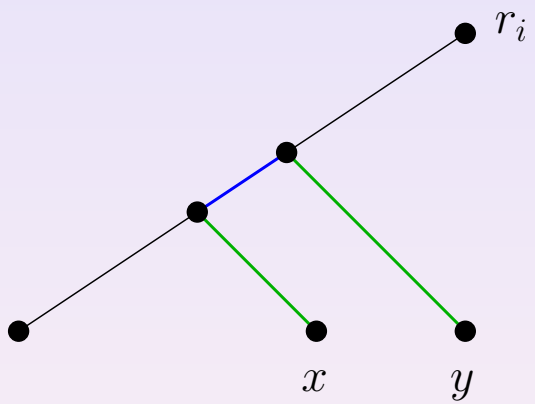
1. Choisir un nœud r arbitraire comme racine de T
2. Prendre un chemin P qui « coupe en deux » T
3. Le nœud x stocke (r, d, h) où $d = d_T(x, a)$ et $h = d_T(r, a)$
4. Recommencer avec les nœuds restant de $T \setminus P$

Décodage de la distance et analyse

Le nœud x stocke : $(r_1, d_1, h_1), \ldots, (r_k, d_k, h_k)$ où $d_k = 0$

Distance entre x et y :

1. Calculer le plus grand i tq $r_i(x) = r_i(y)$
2. renvoyer $d_i(x) + d_i(y) + |h_i(x) - h_i(y)|$

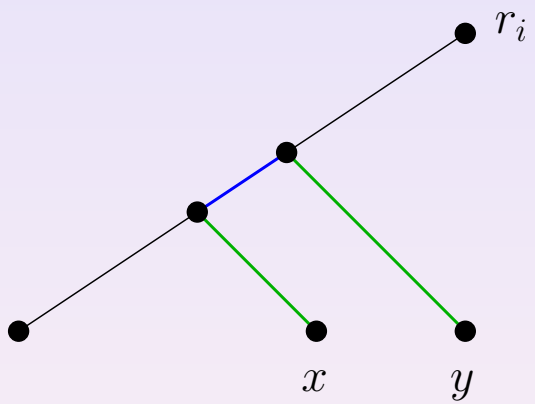


Décodage de la distance et analyse

Le nœud x stocke : $(r_1, d_1, h_1), \ldots, (r_k, d_k, h_k)$ où $d_k = 0$

Distance entre x et y :

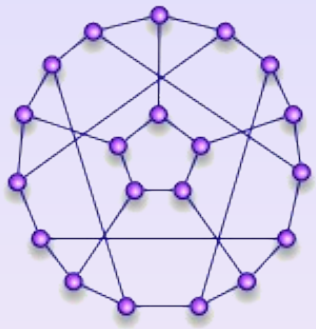
1. Calculer le plus grand i tq $r_i(x) = r_i(y)$
2. renvoyer $d_i(x) + d_i(y) + |h_i(x) - h_i(y)|$

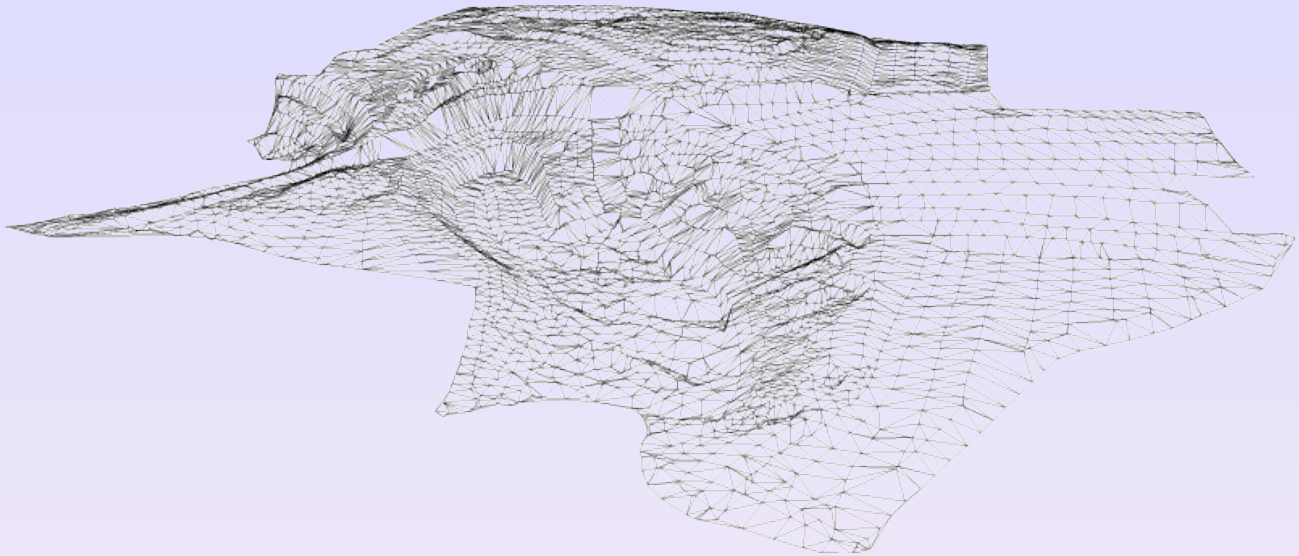


Complexité : $O(k) = O(\log n)$ [$O(1)$ possible]

Pré-calcul : $O(nk) = O(n \log n)$ [$O(n)$ possible]

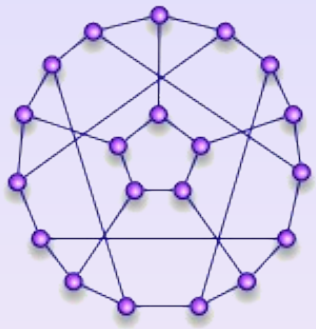
Oracles pour les graphes

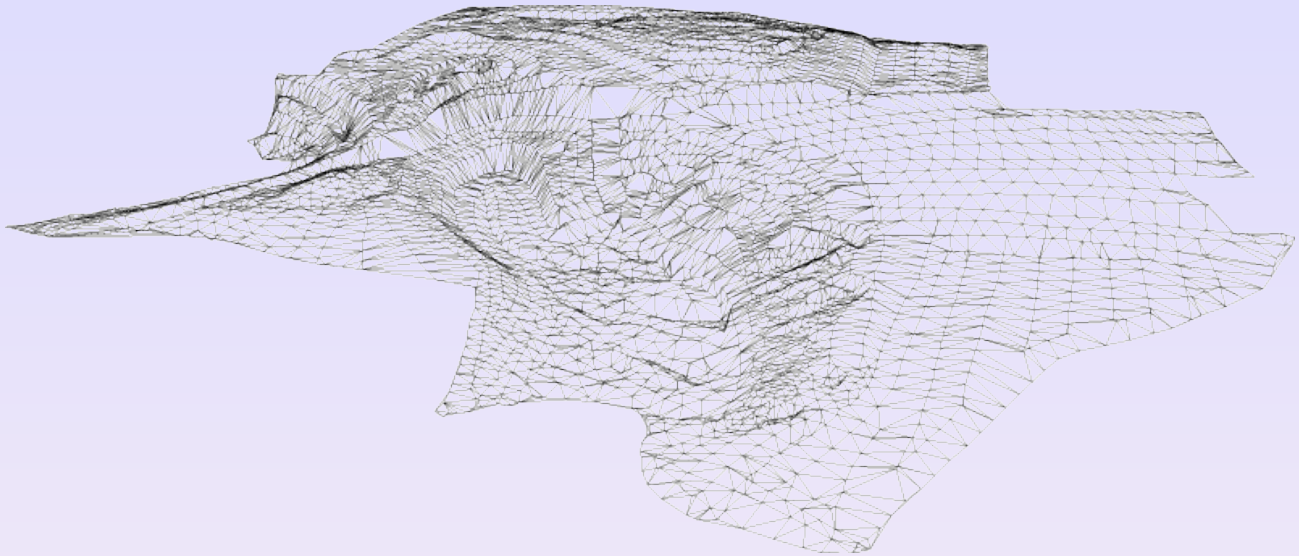




G un graphe arbitraire à n sommets
$\Rightarrow$ problème **APSP** (= *All Pairs Shortest Path*)

Oracles pour les graphes





G un graphe arbitraire à n sommets
$\Rightarrow$ problème **APSP** (= *All Pairs Shortest Path*)

On cherche ici un oracle qui « approxime » la distance :

$$d_G(x,y) \leqslant \hat{d}(x,y) \leqslant f(d_G(x,y))$$

où $f(\cdot)$ est la fonction d'*étirement* [Ex : $f(d) = 3d$]

Un résultat phare du domaine

2010 IEEE 51st Annual Symposium on Foundations of Computer Science

Distance Oracles Beyond the Thorup–Zwick Bound

Mihai Pătraşcu Liam Roditty

Theorem 1. *For any unweighted graph, there exists a distance oracle of size $O(n^{5/3})$ that, given any nodes u and v at distance d, returns a distance of at most $2d+1$ in constant time. A path of this length can be listed in $O(1)$ time per hop.*

Caractéristiques d'un « bon » oracle

- taille $\ll n^2$
- temps de réponse constant
- pré-calcul en temps polynomial

Caractéristiques d'un « bon » oracle

- taille $\ll n^2$
- temps de réponse constant
- pré-calcul en temps polynomial

+

- l'oracle peut être découpé en n « étiquettes » équilibrées

Compression
(pour avoir une taille en $\ll n^2$)

Astuce : simplifier le graphe !

Calculer un **sous-graphe** couvrant qui approche les distances

(par exemple pour un graphe complet géométrique, on calcule une triangulation de Delaunay. Prendre un arbre couvrant ne donne pas un bon étirement)

Compression
(pour avoir une taille en $\ll n^2$)

Astuce : simplifier le graphe !

Calculer un **sous-graphe** couvrant qui approche les distances

(par exemple pour un graphe complet géométrique, on calcule une triangulation de Delaunay. Prendre un arbre couvrant ne donne pas un bon étirement)

Bonne nouvelle : Tout graphe G possède un sous-graphe couvrant avec $O(n^{1.5})$ arêtes et une fonction d'étirement $3d$.

Compression : étirement $d+2$ et $O(n^{1.5})$ arêtes

Preuve :

1. $H := \varnothing$
2. Tantque $\exists\, u \in V$, $\deg(u) \geqslant \sqrt{n}$:
   - $H := H \cup \mathrm{BFS}(u, G)$
   - $G := G \setminus N(u)$
3. Renvoyer $H \cup G$

Compression : étirement $d+2$ et $O(n^{1.5})$ arêtes

Preuve :

1. $H := \varnothing$
2. Tantque $\exists\, u \in V$, $\deg(u) \geqslant \sqrt{n}$:
   - $H := H \cup \mathrm{BFS}(u, G)$
   - $G := G \setminus N(u)$
3. Renvoyer $H \cup G$

Taille : $O(n\sqrt{n}\,)$

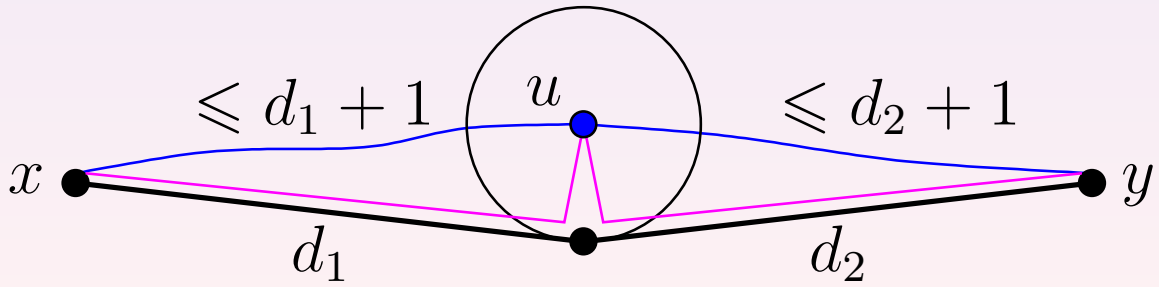
Compression : étirement $d+2$ et $O(n^{1.5})$ arêtes

Preuve :

1. $H := \varnothing$
2. Tantque $\exists\, u \in V$, $\deg(u) \geqslant \sqrt{n}$:
   - $H := H \cup \mathrm{BFS}(u, G)$
   - $G := G \setminus N(u)$
3. Renvoyer $H \cup G$

Taille : $O(n\sqrt{n})$

Étirement : si un plus court chemin entre x et y n'intersecte pas un voisin d'un nœud sélectionné u, alors l'étirement est d. Sinon, $d_H(x, y) \leqslant d_1 + d_2 + 2 = d + 2$.



Compression : ✔

Temps de réponse ?

Comment calculer $d_H(x, y)$?

Certes on peut répondre en temps $O(n + m_H) = O(n^{1.5})$ au lieu de $O(n + m_G) = O(n^2)$, mais c'est loin de $O(1)$.

Compression : ✔

Temps de réponse ?

Comment calculer $d_H(x, y)$?

Certes on peut répondre en temps $O(n + m_H) = O(n^{1.5})$ au lieu de $O(n + m_G) = O(n^2)$, mais c'est loin de $O(1)$.

Que faire si G est très peu dense ? si G est cubic, planaire, ... ?

Compression : ✔
Temps de réponse ?

Comment calculer $d_H(x, y)$?

Certes on peut répondre en temps $O(n + m_H) = O(n^{1.5})$ au lieu de $O(n + m_G) = O(n^2)$, mais c'est loin de $O(1)$.

Que faire si G est très peu dense ? si G est cubic, planaire, ... ?

Borne inférieure [Sommer *et al.* (FOCS'09)] Tout oracle de distances d'étirement $O(d)$ et de temps de réponse t doit avoir une taille $n^{1+\Omega(1/t)}$ même pour des graphes avec $n \cdot \mathrm{polylog}(n)$ arêtes.

Le temps de réponse contraint la taille de l'oracle. Même si les graphes ont $O(n)$ arêtes, la taille de l'oracle doit être en $n^{1+\epsilon}$.

Un nouvel oracle de distances

[Abraham, G. (2011)] Soit k un entier $\geqslant 2$.
Tout graphe non valué possède un oracle de distances d'étirement $(2k-2)d+1$, de taille $\tilde{O}(n^{1+2/(2k-1)})$, et avec un temps de réponse $O(k)$.

[L'oracle, calculable en temps polynomial, peut être découpé en n étiquettes équilibrées]

Pour k=2 :
$\Rightarrow$ étirement $2d+1$
$\Rightarrow$ taille $n^{1+2/3} = n^{5/3} = n^{1.66}$
$\Rightarrow$ temps de réponse constant

Preuve pour $k = 2$.

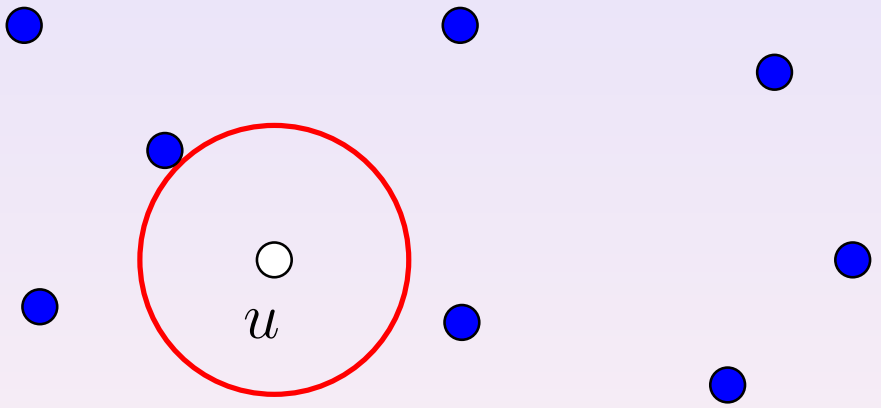
(on veut n étiquettes de taille $n^{2/3}$ et un étirement $2d + 1$)

Preuve pour $k = 2$.

(on veut n étiquettes de taille $n^{2/3}$ et un étirement $2d + 1$)

Définitions : Étant donné un ensemble de *landmarks* $L \subset V$:

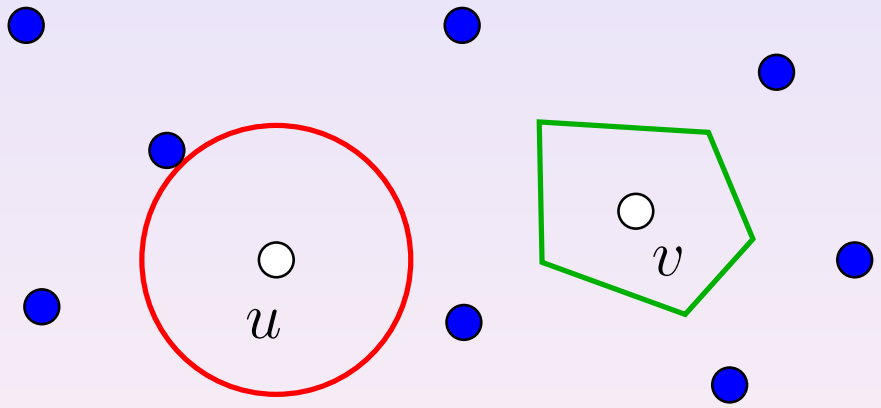
$B_L(u) = \{v \in V : d(u, v) < d(u, L)\}$



Preuve pour $k = 2$.
(on veut n étiquettes de taille $n^{2/3}$ et un étirement $2d + 1$)

Définitions : Étant donné un ensemble de *landmarks* $L \subset V$:

$B_L(u) = \{v \in V : d(u, v) < d(u, L)\}$
$C_L(v) = \{u \in V : v \in B_L(u)\}$



☞ $v \in B_L(u)$ ssi $u \in C_L(v)$

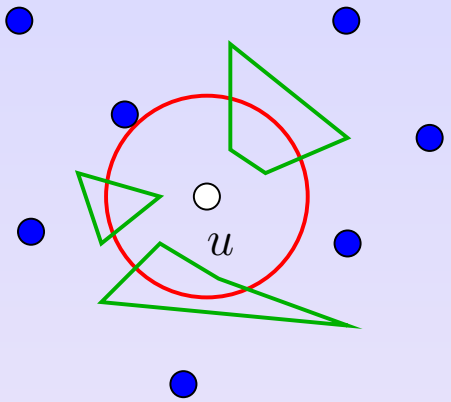
Échantillonage

On peut choisir un ensemble de *landmarks* :

- $|L| \sim n^{2/3}$
- $\forall u$, $|B_L(u)|$ & $|C_L(u)| \sim n^{1/3}$

Idée. Échantilloner les nœuds de V avec probabilité $n^{-1/3}$. $|B_L(u)|$ est ok avec grande probabilité. Calculer l'ensemble W des w ayant un $C_L(w)$ trop **grand**. Si $|W| \leqslant n^{1/3}$, ajouter W à L. Sinon, échantilloner W avec probabilité $n^{1/3}/|W|$. Alors, la moitié des w trop **grands** vont devenir relativement **petits** (Markov + double comptage).

Stockage

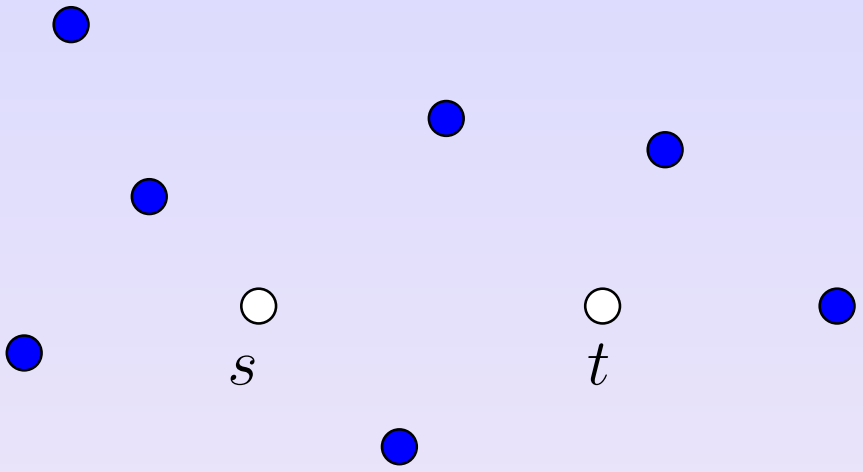


$$I(u) \;:=\; L \cup B_L(u) \cup \left(\bigcup_{v \in B_L(u)} C_L(v) \right)$$

Stockage pour u **:** $I(u)$ et la distance entre u et tous les $v \in I(u)$, plus son plus proche *landmarks* l_v.

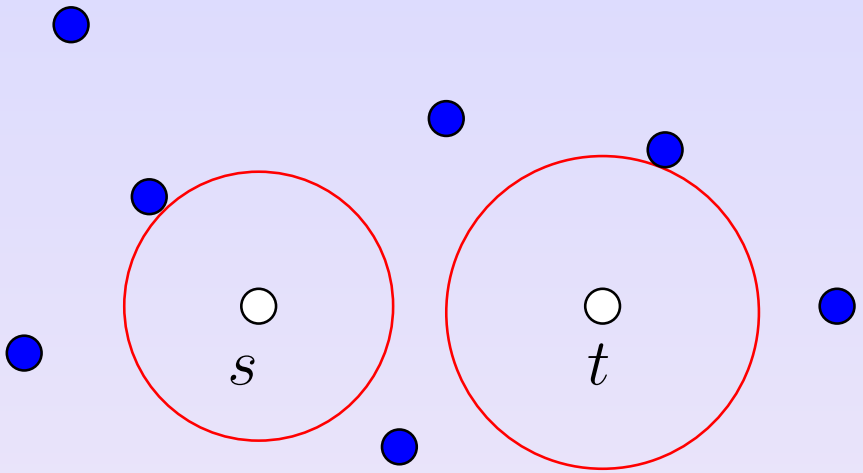
[$|I(u)| \sim n^{2/3}$, stockage ✔]

Distance $\hat{d}$ entre s and t



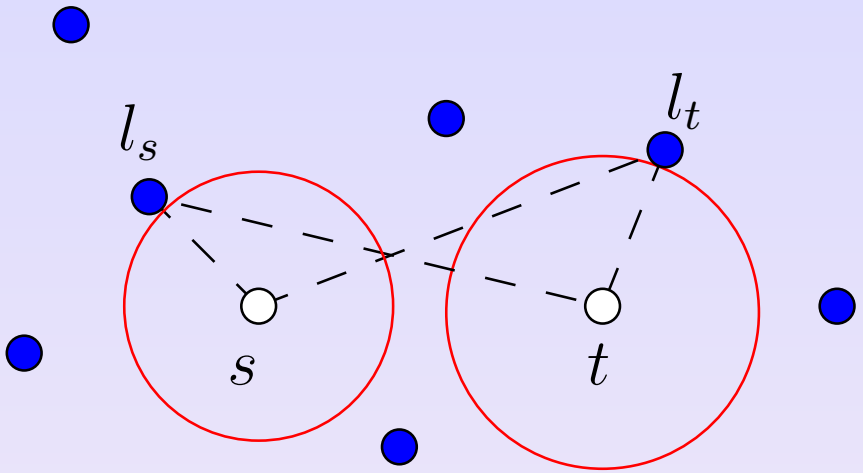
Si $t \in I(s)$, **alors** renvoyer $d(s,t)$

Distance $\hat{d}$ entre s and t



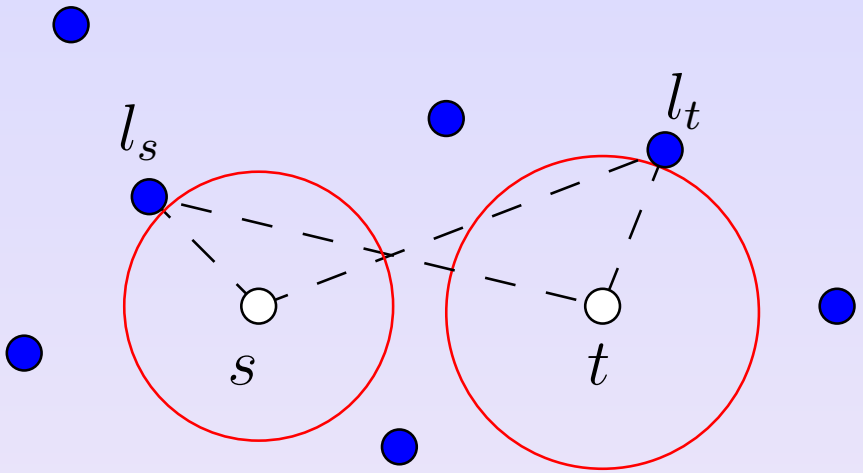
Si $t \in I(s)$, **alors** renvoyer $d(s,t)$

Distance $\hat{d}$ entre s and t



Si $t \in I(s)$, **alors** renvoyer $d(s,t)$
sinon renvoyer $\min\{d(s,l_s)+d(l_s),t),\ d(t,l_t)+d(l_t,t)\}$

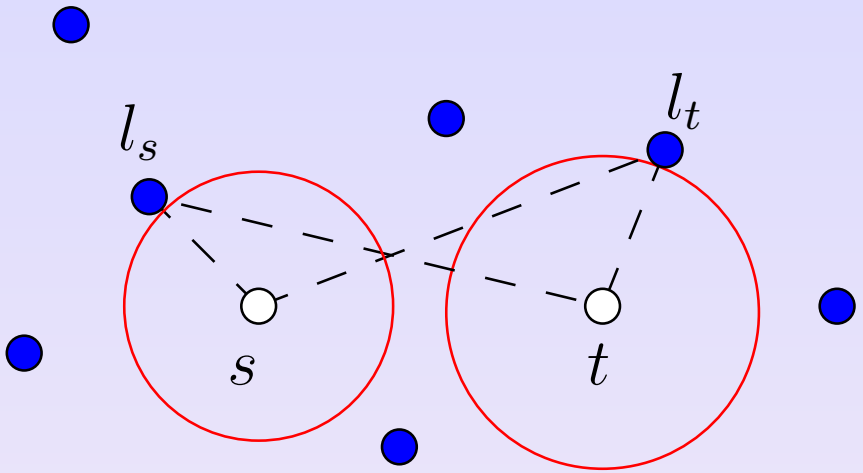
Distance $\hat{d}$ entre s and t



Si $t \in I(s)$, **alors** renvoyer $d(s,t)$
sinon renvoyer $\min\{d(s,l_s)+d(l_s),t),\ d(t,l_t)+d(l_t,t)\}$

[dictionnaire + table de hachage 2-level, temps de réponse ✔]

Distance $\hat{d}$ entre s and t



Si $t \in I(s)$, **alors** renvoyer $d(s,t)$
sinon renvoyer $\min\{d(s,l_s)+d(l_s),t),\ d(t,l_t)+d(l_t,t)\}$

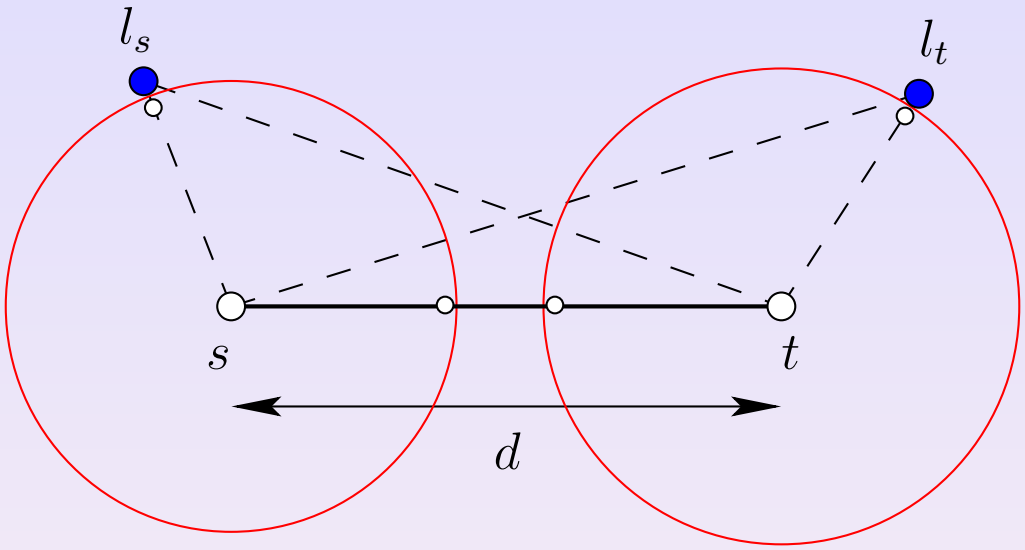
[étirement $2d+1$?]

☞ Si $t \notin I(s)$, alors $B_L(s) \cap B_L(t) = \varnothing$

[sinon $\exists w \in B_L(t) \cap B_L(s) \Rightarrow t \in C_L(w)$ et
$w \in B_L(s) \Rightarrow t \in I(s)$]

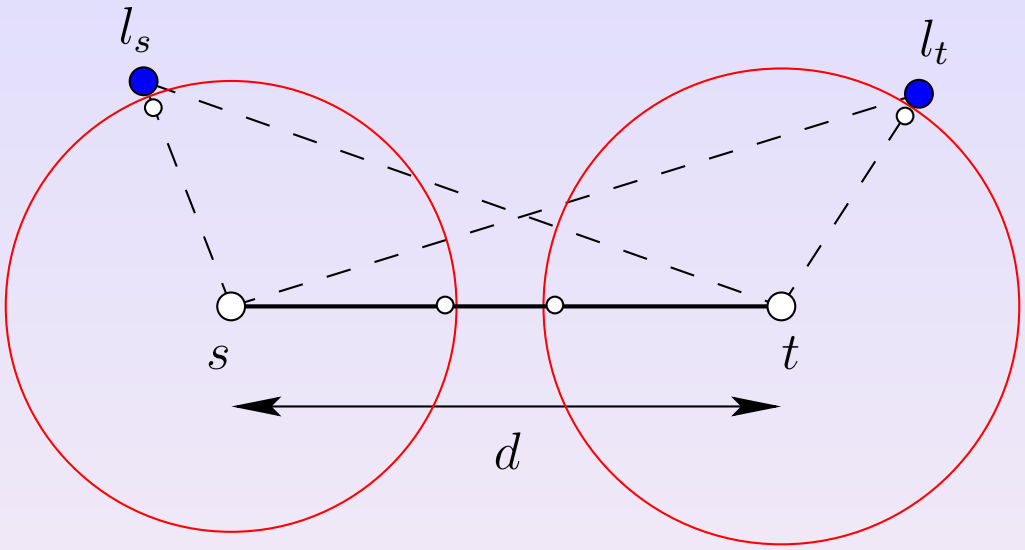
Si $t \notin I(s)$, c'est-à-dire $B_L(s) \cap B_L(t) = \varnothing$

Hypothèse : $d(s, l_s) \leqslant d(t, l_t)$



Si $t \notin I(s)$, c'est-à-dire $B_L(s) \cap B_L(t) = \varnothing$

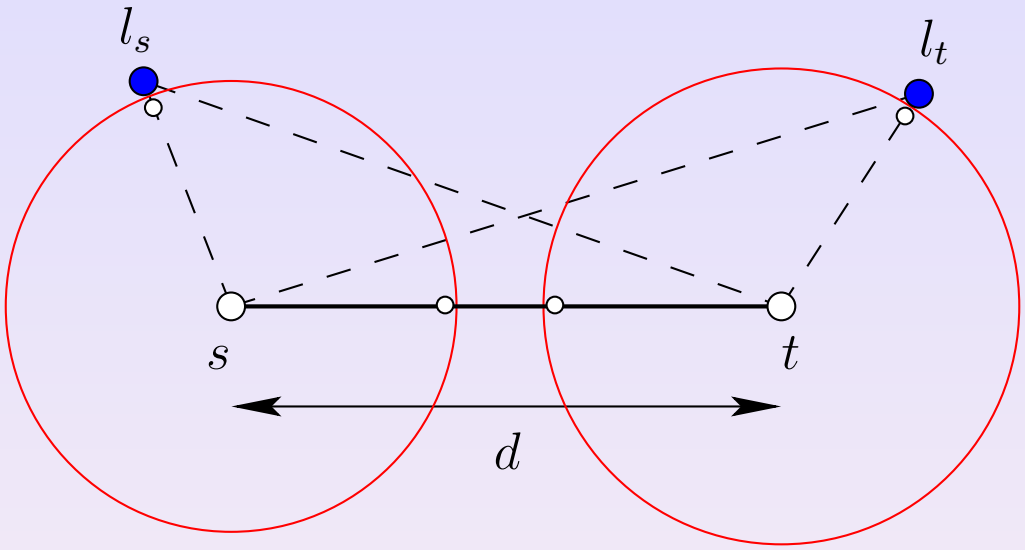
Hypothèse : $d(s, l_s) \leqslant d(t, l_t)$



$$\boxed{\text{☞ } [d(s, l_s) - 1] + 1 + [d(t, l_t) - 1] \leqslant d}$$

Si $t \notin I(s)$, c'est-à-dire $B_L(s) \cap B_L(t) = \varnothing$

Hypothèse : $d(s, l_s) \leqslant d(t, l_t)$

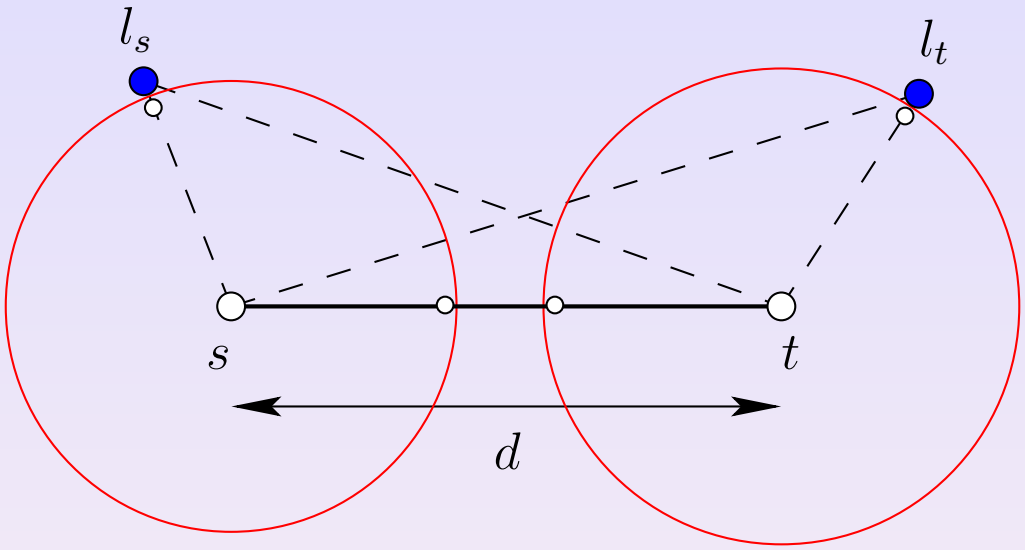


$$\boxed{\text{☞ } [d(s, l_s) - 1] + 1 + [d(t, l_t) - 1] \leqslant d}$$

$$\Rightarrow 2d(s, l_s) \leqslant d + 1$$

Si $t \notin I(s)$, c'est-à-dire $B_L(s) \cap B_L(t) = \varnothing$

Hypothèse : $d(s, l_s) \leqslant d(t, l_t)$



$$\boxed{\text{☞ } [d(s, l_s) - 1] + 1 + [d(t, l_t) - 1] \leqslant d}$$

$$\Rightarrow 2d(s, l_s) \leqslant d + 1$$

$$\Rightarrow \hat{d} \leqslant 2d(s, l_s) + d \leqslant 2d + 1$$

Conclusion, nouvelles directions

- Améliorer l'étirement et/ou l'espace. Établir des bornes inférieures fines est très difficile.

Conclusion, nouvelles directions

- Améliorer l'étirement et/ou l'espace. Établir des bornes inférieures fines est très difficile.
- Répondre à des requêtes plus complexes :

  $d(x, y, z)$ = distance entre x et y sans passer par z ?

  Et plus généralement construire des oracles supportant la suppression d'un ensemble arbitraire d'arêtes ou de nœuds.

Conclusion, nouvelles directions

- Améliorer l'étirement et/ou l'espace. Établir des bornes inférieures fines est très difficile.
- Répondre à des requêtes plus complexes :

  $d(x, y, z)$ = distance entre x et y sans passer par z ?

  Et plus généralement construire des oracles supportant la suppression d'un ensemble arbitraire d'arêtes ou de nœuds.
- Vers des algorithmes dynamiques

Conclusion, nouvelles directions

- Améliorer l'étirement et/ou l'espace. Établir des bornes inférieures fines est très difficile.
- Répondre à des requêtes plus complexes :

  $d(x, y, z)$ = distance entre x et y sans passer par z ?

  Et plus généralement construire des oracles supportant la suppression d'un ensemble arbitraire d'arêtes ou de nœuds.
- Vers des algorithmes dynamiques

Merci !