L'accessibilité pour un graphe non orienté est dans L

Un article d'Omer Reingold

Arthur Dumas et Igor Martayan

9 novembre 2021

Introduction

- L'accessibilité pour un graphe orienté est un problème NL-complet
- Qu'en est-il pour un graphe non orienté (USTCON) ?
- D'après le théorème de Savitch, on dispose d'un algo en espace $\log^2 |G|$
- En 2004, Reingold montre que ce problème est dans L

Aperçu de l'algorithme

Algorithme

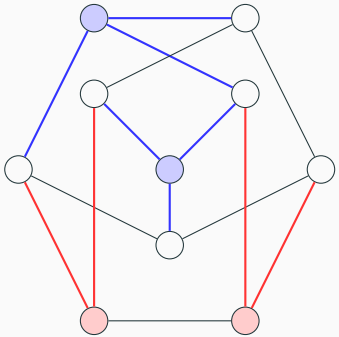
1. transformer G en un graphe expanseur G_E par itérations successives
2. énumérer les chemins de taille logarithmique dans G_E
3. chercher si l'un de ces chemins relie s à t

Les graphes expanseurs, c'est quoi ?

Graphe expanseur

G est un graphe (D, K, A)-expanseur si :

1. G est D-régulier : tout sommet est de degré D
2. G est (K, A)-connecté :
   pour $S \subset V$ de taille au plus K,
   il y a au moins $A \cdot |S|$ arêtes reliant S à $V \setminus S$



$D = 3, K = 2, A = 2$

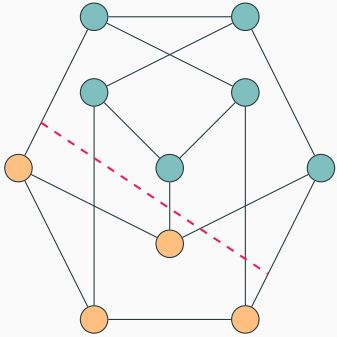
Propriétés sympathiques

Connexité

Si G est $\left(\frac{|V|}{2}, \varepsilon\right)$-connecté (avec $\varepsilon > 0$),
alors G est un graphe connexe.

Chemins de taille logarithmique

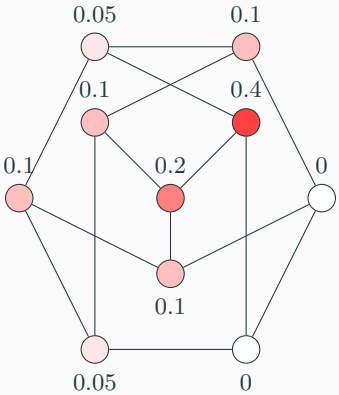
Si G est un $\left(D, \frac{|V|}{2}, 1+\varepsilon\right)$-expanseur ($\varepsilon > 0$),
alors toute paire de sommets est reliée par un chemin de taille $\mathcal{O}(\log |V|)$.



On suppose ici que G est D-régulier.
Soit M la matrice d'adjacence de G divisée par D.

Lien entre valeur propre et expanseur

Si $\lambda_2(M) < 1$, alors G est $\left(\frac{|V|}{2}, 1+\varepsilon\right)$-connecté.

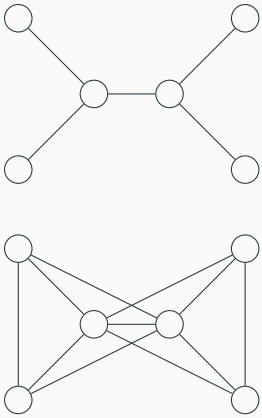


Transformation en expanseur

Puissances d'un graphe

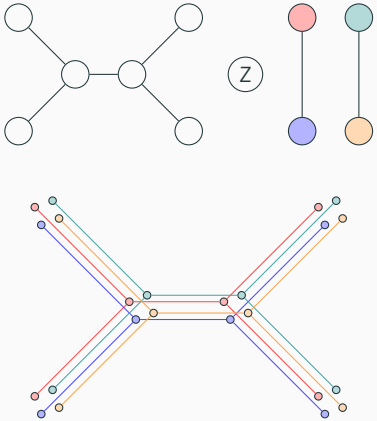
Pour chaque sommet s de G, on relie s à tous les sommets de distance au plus p.

- $\oplus$ on améliore la connectivité du graphe
- $\ominus$ on augmente beaucoup le nombre d'arêtes



Produit zig-zag

Si $\begin{cases} \{g_1, g_2\} \in E_G \\ \{h_1, h_2\} \in E_H \\ \{h_2, h_3\} \in E_H \end{cases}$ alors $\left\{ \begin{bmatrix} g_1 \\ h_1 \end{bmatrix}, \begin{bmatrix} g_2 \\ h_3 \end{bmatrix} \right\} \in E_{G ⓩ H}$



Transformation principale

Entrée : G un graphe D^{16}-régulier et H un graphe D-régulier de D^{16} sommets
Sortie : G_l où :

- $l = \mathcal{O}(\log |V|)$ suffisamment grand (en fonction de D et $|V|$)
- $G_0 = G$
- Pour $1 \leqslant i \leqslant l$, $G_i = (G_{i-1} \textcircled{z} H)^8$

Pourquoi c'est dans Logspace ?

Pour représenter $G = (V, E)$, un graphe D-régulier, on numérote les arêtes de chaque sommet :

$$\begin{array}{ccc} V \times [\![D]\!] & \longrightarrow & V \times [\![D]\!] \\ (v, i) & \longmapsto & (w, j) \end{array}$$

si l'arête (v, w) est la i-ème du sommet v
et la j-ème du sommet w

Transformation principale

Entrée : G un graphe D^{16}-régulier et H un graphe D-régulier de D^{16} sommets (donnés par leur rotation map)
Sortie : (La rotation map de) G_l

Il faut "écrire" les $((v, a), Rot_{G_l}(v, a))$ où $(v, a) \in V \times [\![D^{16}]\!]^l$.
La taille de (v, a) est en $\mathcal{O}(\log |V|)$.
Quant à $Rot_{G_l}(v, a)$...

$v \in V$ (taille $\mathcal{O}(\log|V|)$), $a_0, a_1, ..., a_l \in [\![D^{16}]\!]$, $\forall i, a_i = k_i^1 k_i^2 ... k_i^{16}$, (taille $\mathcal{O}(1)$ pour chaque a_i) On va évaluer G_i. Si $i = 0$, il suffit de lire sur l'entrée. Sinon, on met à jour les variables pour suivre la définition de la transformation : Pour $j = 1$ jusqu'à 16 faire :

- $a_{i-1}, k_i^j \leftarrow Rot_H(a_{i-1}, k_i^j)$
- Si j est impair: $(v, a_0 ... a_{i-1}) \leftarrow Rot_{G_{i-1}}((v, a_0 ... a_{i-2}), a_{i-1})$
- Si $j = 16$: $k_i^1 ... k_i^{16} \leftarrow k_i^{16} ... k_i^1$

Profondeur de l'arbre de récursion en $l + 1 = \mathcal{O}(\log|V|)$. Les opérations faites sont en espace $\mathcal{O}(\log|V|)$. Donc cette transformation se fait en espace $\mathcal{O}(\log|V|)$

Opérations préliminaires

Écrire la rotation map de H, un $(D, \frac{D^{16}}{2}, 1+\varepsilon)$-expanseur (où D est une constante, et ε assez grand).
Ce graphe est déterminé à l'avance, donc cela utilise une quantité constante de mémoire.

Créer la rotation map de G_{reg}, un graphe D^{16}-régulier :
On remplace chaque sommet par un cycle de taille $|V|$.

- $Rot(v, w, 1) = Rot(v, w + 1[|V|], 2)$, $Rot(v, w, 2) = Rot(v, w - 1[|V|], 1)$
- $Rot(v, w, 3) = Rot(w, v, 3)$ s'il existe une arête entre v et w dans G
  $Rot(v, w, 3) = Rot(v, w, 3)$ sinon
- $\forall i \in [\![3, |V|]\!], Rot(v, w, i) = Rot(v, w, i)$

- $Rot(v, w, 1) = Rot(v, w + 1[|V|], 2)$, $Rot(v, w, 2) = Rot(v, w - 1[|V|], 1)$
- $Rot(v, w, 3) = Rot(w, v, 3)$ s'il existe une arête entre v et w dans G
  $Rot(v, w, 3) = Rot(v, w, 3)$ sinon
- $\forall i \in [\![3, |V|]\!], Rot(v, w, i) = Rot(v, w, i)$

Pour chaque sommet (u, v) l'arête 3 est dans la matrice d'adjacence de G, et pas besoin d'écrire les autres arêtes

Conclusion

Conclusion

Algorithme

1. transformer G en un graphe expanseur G_E
2. énumérer les chemins (de taille logarithmique) dans G_E
3. chercher si l'un de ces chemins relie s à t

USTCON est dans Logspace

Références :

- Omer Reingold, *Undirected connectivity in log-space*, JACM, 2008
- *http://theory.stanford.edu/~liyang/teaching/projects/undirected-connectivity-is-in-L.pdf*
- *https://stanford.edu/~styopa/pdfs/expanders.pdf*