

Agrégation de mathématiques, option informatique
Algorithmique
Complément sur algorithmes sur les graphes

1^{er} décembre 2014

Exercice 1

2-SAT

Le problème 3-SAT est NP-complet. Dans cet exercice, nous allons montrer que si on se restreint aux formules qui sont des *formes normales conjonctives* d'ordre 2, comme par exemple $(x_1 \vee x_2) \wedge (\neg x_1 \vee x_3) \wedge (x_2 \vee \neg x_4) \wedge \dots$ alors le problème de satisfiabilité est dans P.

On définit le problème 2-SAT de la manière suivante :

entrée : une formule $\phi(x_1, \dots, x_n)$ sous forme normale conjonctive d'ordre 2

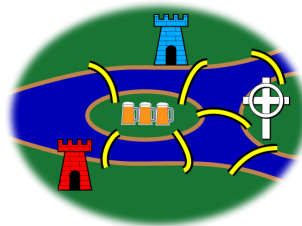
sortie : oui ssi il existe une valuation pour $x_1 \dots x_n$ qui rende ϕ vraie

1. En observant que la formule $x \vee y$ est équivalente à $\neg x \rightarrow y$, donner un problème équivalent sur les graphes orientés résoluble en temps polynomial.
2. Conclure.

Exercice 2

chemin eulérien

Un chemin eulérien dans un graphe non orienté est un chemin qui passe une et une seule fois par toutes les arêtes.



1. Rappelez la caractérisation d'existence d'un chemin eulérien dans un graphe non-orienté.
2. Écrire un algorithme qui calcule un chemin eulérien dans un graphe orienté s'il en existe un.

Exercice 3

caractérisation des arbres

Cormen

1. Montrer que les assertions suivantes sont équivalentes :
 1. G est un arbre (i.e. connexe et acyclique) ;
 2. Deux sommets quelconques de G sont reliés par une chaîne simple unique ;
 3. G est connexe mais, si l'on enlève une arête, le graphe n'est plus connexe ;
 4. G connexe et $|A| = |S| - 1$;
 5. G acyclique et $|A| = |S| - 1$;
 6. G est acyclique mais, si l'on ajoute une arête, le graphe contient un cycle.

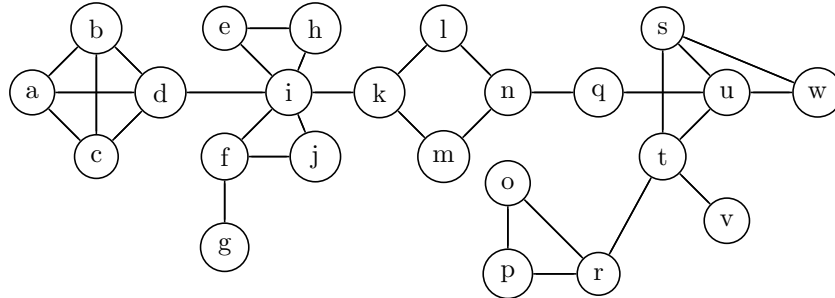
Cormen, p. 574

Soit $G = (V, E)$ un graphe non orienté connexe. Dans G :

- un *point d'articulation* est un sommet dont la suppression déconnecte le graphe ;
- un *isthme* est une arête dont la suppression déconnecte le graphe ;
- une *composante 2-connexe* est un ensemble maximal connexe de sommets tel que le sous-graphe induit par ces sommets n'a pas de point d'articulation.

On notera $T = (V, F)$ un arbre de parcours de G obtenu par parcours en profondeur et $deb(v)$ le début de visite de v dans la numérotation associée.

1. Sur l'exemple ci-dessous, déterminer les points d'articulation, les isthmes et les composantes 2-connexes.



2. Expliquer pourquoi dans le parcours en profondeur d'un graphe non orienté et connexe on n'obtient que deux sortes d'arcs : les arcs de l'arbre et les arcs retour.
3. Montrer que la racine de T est un point d'articulation si et seulement si elle a au moins deux fils dans T .

Dans la suite, on dira que x est descendant de y pour dire que x est descendant de y dans T . Par convention, « descendant » est pris au sens large : on a aussi x est un descendant de x .

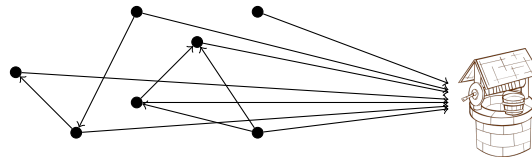
4. Soit x un sommet de T distinct de la racine. Prouver que x est un point d'articulation si et seulement s'il existe un fils y de x dans l'arbre tel qu'il n'existe pas d'arc retour de la forme (z, t) où z est un descendant de y et t est un ancêtre propre de x .

On définit la fonction $auDessous$, de V dans \mathbb{N} par

$$auDessous(x) = \min(\{deb(x)\} \cup A(x))$$

où $A(x) = \{deb(z) \mid (y, z) \text{ est un arc retour avec } y \text{ descendant de } x\}$.

5. Donner un algorithme en $O(|V| + |E|)$ de calcul de la fonction $auDessous$.
6. Montrer comment calculer les points d'articulation en $O(|V| + |E|)$.
7. Prouver qu'une arête est un isthme si et seulement si elle n'appartient à aucun cycle élémentaire.
8. Montrer comment calculer les isthmes en $O(|V| + |E|)$.
9. Considérer le graphe H dont les sommets sont les composantes 2-connexes de G et tel que deux composantes sont adjacentes si et seulement si elles ont au moins un sommet en commun. Que pouvez vous dire de ce graphe ?



On considère un graphe orienté $G = (S, A)$ où les arcs sont décrits à l'aide d'une matrice d'adjacence $A = (A_{i,j})_{i,j \in S}$. On suppose que $A_{i,i} = 0$ pour tout $i \in S$. Un puits est un sommet $s \in S$ ssi $\sum_{t \in S} A_{t,s} = n - 1$ et $\sum_{t \in S} A_{s,t} = 0$.

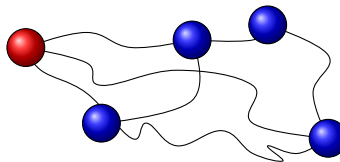
1. Écrire un algorithme en $O(|S|)$ qui donne le puits d'un graphe G si G possède un puits et qui retourne \square sinon. (indice : qu'apprend-on de la valeur de $A_{i,j}$?)

On suppose que $A_{i,j}A_{j,i} = 0$ pour tout $i, j \in S$. Malheureusement, le graphe est stockée sur un serveur très lent et on dispose une instruction *telecharger*(i, j) qui donne la valeur de $A[i, j]$.

2. Montrer qu'il existe un algorithme où il y a au plus $3n - \lceil \log_2 n \rceil - 3$ appels à *telecharger*. (faites un tournoi avec ' i bat j ' quand $A[i, j] = 0$)

Exercice 6

Plus court chemin et programmation linéaire



Considérons le problème des plus courts chemins à partir d'une source :

entrée : un graphe pondéré positivement $G = (S, p)$ et un sommet s ;
 sortie : $(\delta_t)_{t \in S}$ où δ_t est la distance d'un plus court chemin de s vers t ou $+\infty$ si il n'y a pas de chemin de s vers t

À une instance G, s du problème des plus court chemins à partir d'une source, on considère l'instance de programmation linéaire suivante :

maximiser $\sum_{t \in S} d_t$
 sous les contraintes :
 — $d_s = 0$;
 — $d_v \leq d_t + p(t, v)$ pour tout $t, v \in S$.

1. Montrer qu'une solution $(d_t)_{t \in S}$ du programme linéaire est telle que $\delta_t = d_t$ pour tout $t \in S$.

Exercice 7**Des probabilités pour montrer l'existence !**

On considère un graphe non orienté $G = (S, A)$. On appelle coupe une partition (S_1, S_2) de l'ensemble des sommets S . La taille de la coupe (S_1, S_2) est le nombre d'arêtes qui relient un sommet de S_1 à un sommet de S_2 .

On s'intéresse au problème de la coupe maximale

entrée : un graphe non orienté G

sortie : la taille maximum d'une coupe

Le problème de décision associé est NP-complet (admis). On s'intéresse ici à écrire un algorithme qui calcule une coupe de taille $\geq \frac{|A|}{2}$.

1. Écrire un algorithme 'naïf' qui parcourt les sommets un à un et qui les met avec une probabilité $\frac{1}{2}$ dans S_1 et sinon dans S_2 .
2. En calculant l'espérance de la taille de coupe, montrer qu'il existe une coupe de taille $\geq \frac{|A|}{2}$.
3. Écrire l'algorithme qui maximise l'espérance à chaque fois.

Exercice 8**Fermeture réflexive et transitive****Handbook of Computer Science, Algorithms and complexity**

1. Soit un graphe orienté $G = (S, A)$ représentée par des listes d'adjacence. Supposons que les sommets sont $S = \{1, \dots, n\}$. Montrer comment trier les listes d'adjacence par ordre croissant en temps $O(|S| + |A|)$?

Considérons un graphe orienté acyclique $G = (S, A)$. L'objectif est de calculer la clôture réflexive et transitive $G^* = (S, A^*)$ de G .

2. En utilisant un tri topologique, et en supposant que les listes d'adjacence sont triées par ordre croissant, écrire un algorithme qui calcule $A^*(i)$ pour $i = n$ jusqu'à 1. Pour chaque i , on parcourt la liste $A(i)$.

Soit $G^- = (S, A^-)$ le graphe appelée réduction transitive obtenue à partir de G et en supprimant tout arc $x \rightarrow y$ s'il existe un chemin non direct de x à y .

3. Adapter l'algorithme et prouver que sa complexité temporelle est en $O(|S||A^-| + |S| + |A|)$.

Après, considérons un graphe orienté $G = (S, A)$ qui peut éventuellement contenir des cycles. L'objectif est de calculer la clôture réflexive et transitive $G^* = (S, A^*)$ de G .

4. Montrer que l'on peut calculer la clôture réflexive et transitive en $O(|S| + |A^*| + |S_C||E_C^-|)$ où le graphe quotient des composantes fortement connexes est $G_C = (S_C, E_C)$ et où E_C^- est l'ensemble des arcs dans la réduction transitive de G_C .