Aix*Marseille université

Domaine Sciences et Technologies
L3 Info, parcours Maths-info

Algo. avancée et th. des graphes : TP 1
Code UE : S04IN5B2

Année 2020-21

2-SAT

Le travail se déroule en binôme. Sources partielles :

```
git clone https://gitlab.com/algographes/tp1.git
```

Le problème 2-SAT est un problème de décision : étant donnée une formule 2-SAT P donnée, il s'agit de répondre à la question "est-ce que la formule P est satisfiable ou non ? ". C'est un cas particulier du problème SAT qui est NP-complet, mais pour le problème 2-SAT il existe des approches *polynomiales*. Vous allez implémenter une approche de ce type. Les étapes pour la réalisation du projet se suivent et chacune dépends des étapes précédentes. Vous devez donc traiter les étapes les unes après les autres sans en sauter aucune et en commençant par la première. Vous avez 4 semaines pour réaliser ce TP, qui sera évalué téléversé pour évaluation sur la page AMETICE du cours.

Le problème 2-SAT

Un problème 2-SAT est défini par un ensemble fini de clauses (on parle de *forme normale conjonctive*). Chaque clause est constituée de 2 littéraux et chaque littéral est une variable ou la négation d'une variable. Voici par exemple un problème 2-SAT constitué de 4 clauses, dans lequel apparaissent les 3 variables x, y et z :

$$\{x \vee \neg y, \neg x \vee z, x \vee z, \neg y \vee \neg z\}$$

La formule est satisfiable s'il existe une affectation des variables x, y et z avec les valeurs **V** (vrai) et **F** (faux) qui satisfait toutes les clauses (pour qu'une clause soit satisfaite il suffit qu'un de ses littéraux soit vrai). Par exemple l'affectation $x \leftarrow \mathbf{V}$, $y \leftarrow \mathbf{F}$, $z \leftarrow \mathbf{V}$, satisfait la formule ci-dessus : la première clause est satisfaite puisque x est vrai, la 2è clause parce que z est vrai, la 3è parce que x et z sont vrais, et la 4è parce que y a la valeur faux.

Déterminer si une formule SAT dans le cas général est satisfiable est très difficile (le problème est NP-complet). Dans le cas particulier où les clauses sont de longueur 2 il existe des approches efficaces (linéaires en le nombre de clauses). La plus connue est la suivante :

1. construire le *graphe des implications*,
2. calculer les composantes fortement connexes du graphe des implications,
3. si une composante contient à la fois un littéral et son opposé, la formule est insatisfiable ; dans le cas contraire la formule est satisfiable.

Vous allez implémenter cette approche.

Graphe des implications

Dans le problème 2-SAT chaque clause est de longueur 2, c'est à dire de la forme

$$l_1 \vee l_2$$

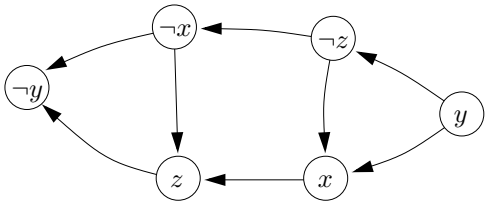
où l_1 et l_2 sont des littéraux. Cette clause est équivalente à

$$(\neg l_1 \Rightarrow l_2) \wedge (\neg l_2 \Rightarrow l_1)$$

Le graphe des implications représente les implications qui apparaissent dans cette formule. Plus formellement, si V est l'ensemble des variables figurant dans la formule, et C l'ensemble de ses clauses, le graphe des implications $G = (S, A)$ est un graphe orienté défini par :

$$S = \{x \mid x \in V\} \cup \{\neg x \mid x \in V\},$$
$$A = \{(\neg l_1, l_2), (\neg l_2, l_1) \mid (l_1 \vee l_2) \in C\}.$$

Notez donc que $|S| = 2 \times |V|$ et $|A| = 2 \times |C|$. Pour l'exemple précédent le graphe des implications serait :



Calcul des composantes fortement connexes

Dans un graphe orienté, deux sommets u et v sont *fortement connectés* s'il existe un chemin orienté de u vers v et vice versa. C'est une relation d'équivalence et l'ensemble des sommets du graphe peut être partitionné en composantes fortement connectées, sous-ensembles de sommets dans lesquels entre deux sommets quelconques il existe un chemin orienté. L'algorithme de Kosaraju (linéaire en le nombre d'arcs) construit les composantes fortement connexes d'un graphe donné. Le calcul s'effectue en deux temps :

étape 1 : parcours en profondeur du graphe des implications G ; on mémorisera les dates de fin de traitement des sommets,

étape 2 : parcours en profondeur du graphe transposé de G noté G^t (le graphe G dans lequel les arcs sont inversés) ; chaque fois, pour lancer l'exploration, on choisira le sommet non visité dont la date de fin de traitement à l'étape 1 est la plus grande. Les arbres construits lors de ce parcours sont les composantes fortement connexes du graphe G.

Vous trouverez des explications plus détaillées à la fin de ce document.

Ce que vous devez faire

Dans l'ordre :

1. lire les clauses depuis un fichier (téléchargez le fichier exemple formule-2-sat.txt),
2. construire le graphe des implications G,
3. calculer les composantes fortement connexes, et pour cela vous devrez construire le graphe transposé G^t.
4. déterminer si aucune composante ne contient à la fois un littéral et son opposé.

Les formules que vous aurez à traiter sont de la forme suivante (c'est un exemple) :

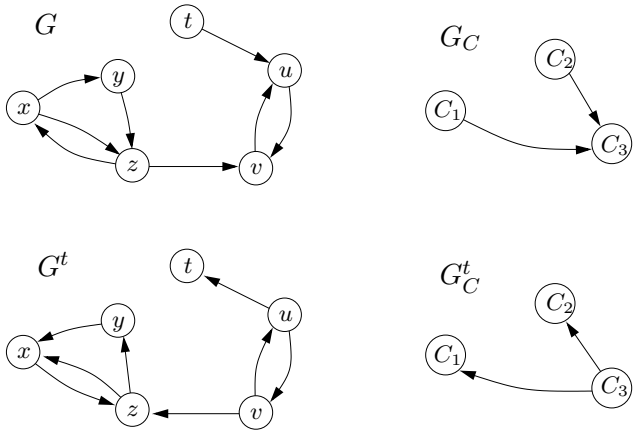
```
c Commentaire
p cnf 5 3
1 -5 4 0
-1 5 3 4 0
-3 -4 0
```

La première ligne (commençant par 'c') est un commentaire. La deuxième ligne indique le nombre de variables (ici il y a 5 variables, qui sont implicitement numérotées de 1 à 5) et le nombre de clauses (dans l'exemple il y a 3 clauses). Chaque clause est définie par ses littéraux (x ou $-x$ selon le signe du littéral) et se termine par la valeur 0.

Les graphes seront représentés avec des listes d'adjacence (un tableau indexé sur les sommets contenant pour chaque sommet la liste chainée de ses voisins). Vous utiliserez les définitions et fonctions mises à votre disposition sur gitlab :

`git clone https://gitlab.com/algographes/tp1.git`

Composantes fortement connexes.
On note G_C le graphe des composantes de G (ses sommets sont les composantes f.c., et pour tout arc (u, v) de G, si u et v ne sont pas dans la même composante, alors G_C contient l'arc (C_u, C_v)), et G_C^t le graphe des composantes f.c. de G^t.



1. G et G_t ont les mêmes composantes fortement connexes,
2. toute composante f.c. *initiale* de G (que des arcs sortants, les composantes C_1 et C_2 dans l'exemple) est une composante f.c. *finale* de G_t (aucun arc sortant, dans le graphe G_C^t aucun arc ne part des sommets C_1 et C_2),
3. les graphes G_C et G_C^t sont acycliques, donc G_C^t a au moins une composante *finale*,
4. le parcours en profondeur dans G_t à partir d'un sommet s d'une de ses composantes *finale* C découvre les sommets de C et uniquement ceux-là,
5. soit v_0 le sommet dont la visite s'est terminée en dernier lors du parcours en profondeur du graphe G ($fin[v]$ est la date de fin de traitement la plus grande, noté $\mathtt{t}_v^s$ dans le cours), v_0 apparaît dans une composante *initiale* de G (et donc dans une composante finale de G^t).

Pourquoi v_0 est dans une composante initiale. Remarquons tout d'abord que v_0 est le premier sommet de C découvert lors du parcours en profondeur de G. Supposons que la composante C contenant v_0 n'est pas une composante initiale. Il existe alors une composante C' telle que G contient un arc d'un sommet de C' vers un sommet de C. On note v le premier sommet de C' découvert lors du parcours en profondeur de G. Soit v est découvert avant v_0, et v_0 serait un descendant de v (th. du chemin blanc) et on aurait $fin[v_0] < fin[v]$. Soit v_0 est découvert avant v, les sommets de C' ne pouvant pas être visités à partir de sommets de C (si c'était le cas G_C^t contiendrait les ars (C, C') et (C', C), et C et C' formeraient une seule composante), v sera découvert après la fin du traitement des sommets de C, et on aurait $fin[v_0] < debut[v] < fin[v]$.