Décidabilité et classes de complexité

Quentin Fortier

December 5, 2024

Décidabilité

Définition

Un algorithme est un programme fini (suite d'instructions) qui prend une entrée finie.
On suppose que cet algorithme s'exécute sur un ordinateur avec une quantité illimité de mémoire.
Sur une entrée, un algorithme peut :

- renvoyer un résultat en temps fini
- ne pas renvoyer de résultat, soit parce qu'il ne termine pas (boucle infinie...), soit parce qu'il plante (dépassement de tableau...)

Décidabilité

Définition

Une machine universelle est un programme U prenant en entrée le code source d'un programme f et un argument x, et simulant l'exécution de f sur x. Autrement dit :

- si $f(x)$ renvoie un résultat y en temps fini, alors $U(f, x)$ renvoie y en temps fini ;
- si $f(x)$ diverge en temps fini, $U(f, x)$ diverge en temps fini ;
- si $f(x)$ ne termine jamais, $U(f, x)$ ne termine jamais.

<u>Exemple</u> : l'interpréteur OCaml (utop) est une machine universelle pour les programmes OCaml.

Décidabilité

Définition

Un problème de décision est un couple (I, I^+) tel que :

- I est l'ensemble des instances du problème
- $I^+ \subset I$ est l'ensemble des instances positives du problème

On peut aussi définir un problème sous forme d'une question binaire sur une instance.

Décidabilité

Exemples de problèmes de décision :

SAT

- Instance : une formule logique φ
- Question : φ est-elle satisfiable ?

$I = \{\varphi \mid \varphi \text{ formule logique}\}$, $I^+ = \{\varphi \mid \varphi \text{ satisfiable}\}$

APPARTIENT

- Instance : un mot w et un automate A
- Question : $w \in L(A)$?

$I = \{(w, A) \mid w \text{ mot et } A \text{ automate}\}$, $I^+ = \{(w, A) \mid w \in L(A)\}$

Décidabilité

Définition

Un problème de décision (I, I^+) est dit décidable s'il existe un algorithme A qui :

- prend une instance $i \in I$ du problème en entrée
- renvoie `true` en temps fini si i est une instance positive ($i \in I^+$)
- renvoie `false` en temps fini si i est une instance négative ($i \notin I^+$)

Sinon, le problème est dit indécidable.

Décidabilité

Remarques :

- A doit terminer en temps fini sur toutes les instances.
- Pour montrer qu'un problème est décidable, il suffit d'exhiber un algorithme qui le décide. Montrer qu'il est indécidable est a priori plus difficile : il faut montrer qu'aucun algorithme ne peut le résoudre.
- Quand on s'intéresse à la décidabilité d'un problème, seule l'existence d'un algorithme compte : sa complexité n'a aucune importance. Il n'est donc pas non plus nécessaire de préciser les structures de données utilisées, tant que celles-ci sont calculables.
- Si l'ensemble des instances positives est de cardinal fini, le problème est trivialement décidable : il suffit d'énumérer toutes les instances positives et tester si l'une d'entre elles est égale à l'entrée.

Décidabilité

Exercice

Montrer que le problème SAT est décidable.

Décidabilité

Définition

Une fonction $f : E \to F$ est calculable s'il existe un algorithme A qui, pour tout élément $x \in E$, termine en temps fini et renvoie $f(x)$.

Décidabilité

ARRET

- Instance : le code source d'un programme f et un argument x
- Question : f termine-t-il sur l'entrée x ?

Décidabilité

ARRET

- Instance : le code source d'un programme f et un argument x
- Question : f termine-t-il sur l'entrée x ?

En OCaml, cela revient à écrire une fonction
`arret : string -> string -> bool` qui termine sur toute entrée et telle que si `f` est une fonction et `x` une chaîne de caractères,
`arret f x` renvoie `true` si $f(x)$ termine, `false` sinon.

Décidabilité

ARRET

- Instance : le code source d'un programme f et un argument x
- Question : f termine-t-il sur l'entrée x ?

En OCaml, cela revient à écrire une fonction `arret : string -> string -> bool` qui termine sur toute entrée et telle que si `f` est une fonction et `x` une chaîne de caractères, `arret f x` renvoie `true` si $f(x)$ termine, `false` sinon.

Théorème

Le problème de l'arrêt est indécidable.

Décidabilité

Définition

On dit qu'un problème de décision $P_1 = (I_1, I_1^+)$ se réduit à un problème de décision $P_2 = (I_2, I_2^+)$, noté $P_1 \leq P_2$, s'il existe une fonction calculable $f : I_1 \longrightarrow I_2$ telle que :
$\forall i \in I_1, i \in P_1 \Longleftrightarrow f(i) \in P_2$.

Autrement dit : P_1 se réduit à P_2 si un algorithme pour résoudre P_1 permet de résoudre P_2.

Décidabilité

Exercice

Montrer que CONNEXE $\leq$ ACCESSIBLE.

CONNEXE

- Instance : un graphe G
- Question : G est-il connexe ?

ACCESSIBLE

- Instance : un graphe $G = (S, A)$ et deux sommets $s, t \in S$
- Question : existe-t-il un chemin de s à t dans G ?

Décidabilité

Théorème

Soient P_1 et P_2 deux problèmes de décision tels que $P_1 \leq P_2$. Alors :

- Si P_1 est indécidable, alors P_2 est indécidable.
- Si P_2 est décidable, alors P_1 est décidable.

Méthode pour montrer qu'un problème P_1 est indécidable :

1. Supposer par l'absurde que P_1 est décidable par une fonction f_1.
2. En déduire une fonction f_2 qui décide un problème P_2 connu comme indécidable (par exemple ARRET).
3. Conclure que P_2 est indécidable.

Exercice

Montrer que le problème ARRET-VIDE est indécidable.

ARRET-VIDE

- Instance : une fonction f
- Question : f termine-t-il sur l'entrée vide ?

Classes de complexité

Définition

La taille $|x|$ d'une instance x d'un problème est le nombre de bits nécessaires pour la coder.

Classes de complexité

Définition

La taille $|x|$ d'une instance x d'un problème est le nombre de bits nécessaires pour la coder.

Remarques :

- Un entier n est codé en base 2, donc sa taille est $\log_2(n)$. On pourrait aussi le coder en unaire ce qui donnerait une taille n, mais ce n'est pas « raisonnable ».
- On s'intéresse seulement à l'ordre de grandeur de la taille (O(...)).

Classes de complexité

Définition

La taille $|x|$ d'une instance x d'un problème est le nombre de bits nécessaires pour la coder.

Remarques :

- Un entier n est codé en base 2, donc sa taille est $\log_2(n)$. On pourrait aussi le coder en unaire ce qui donnerait une taille n, mais ce n'est pas « raisonnable ».
- On s'intéresse seulement à l'ordre de grandeur de la taille (O(...)).

Exemples :

- La taille d'un entier n est $\log_2(n)$.
- Un ensemble de p entiers dans $[\![1, n]\!]$ a une taille de $p \log_2(n)$.
- Un graphe à n sommets représenté par une matrice d'adjacence a une taille de n^2.

Classes de complexité : P

Définition

La classe P est l'ensemble des problèmes de décision qui admettent un algorithme de complexité polynomiale en la taille de l'entrée (c'est-à-dire O(n^k) pour une constante k, où n est la taille de l'entrée).

Classes de complexité : P

Définition

La classe P est l'ensemble des problèmes de décision qui admettent un algorithme de complexité polynomiale en la taille de l'entrée (c'est-à-dire O(n^k) pour une constante k, où n est la taille de l'entrée).

Définition (HP)

La classe EXP est l'ensemble des problèmes de décision qui admettent un algorithme de complexité exponentielle en la taille de l'instance (c'est-à-dire $O(2^{n^k})$ pour une constante k, où n est la taille de l'entrée).

<u>Remarque</u> : P $\subset$ EXP.

Classes de complexité : P

Exemple :

PGCD

- Instance : entiers a, b, d
- Question : d est-il le PGCD de a et b ?

On peut calculer le PGCD de a et b en utilisant l'algorithme d'Euclide en complexité $O(\log_2(a) + \log_2(b))$, polynomiale en la taille de l'entrée.

Exemple :

PREMIER

- Instance : un entier n
- Question : n est-il premier ?

On peut énumérer les entiers de 2 à $\sqrt{n}$ pour tester si n est divisible par l'un d'entre eux, en complexité $O(\sqrt{n})$.
Ceci est polynomial en n mais exponentielle en la taille $\log_2(n)$ de n (car $\sqrt{n} = 2^{\frac{\log_2(n)}{2}}$), ce qui montre seulement PREMIER $\in EXP$.

Classes de complexité : P

Soit $G = (S, A)$ un graphe et $G' = (S', A')$ un sous-graphe de G (c'est-à-dire $S' \subset S$ et $A' \subset A$).
On dit que G' est une clique de G si G' est complet, c'est-à-dire que pour tout couple de sommets $u, v \in G'$, il existe une arête entre u et v.

Exercice

1. Soit k un entier fixé. Montrer que k-CLIQUE $\in$ P.
2. Montrer que CLIQUE $\in$ EXP.

k-CLIQUE

- Instance : un graphe G
- Question : G contient-il une clique de taille k ?

CLIQUE

- Instance : un graphe G et un entier k
- Question : G contient-il une clique de taille k ?

Classes de complexité : NP

Définition

Un problème de décision (I, I^+) appartient à la classe NP s'il existe :

- un algorithme de décision A prenant en entrée un couple (x, c) où $x \in I$
- un polynome Q

tels que :

- A s'exécute en temps polynomial en $|x| + |c|$
- $\forall x \in I$, $x \in I^+ \Leftrightarrow \exists c, |c| \leq Q(|x|), A(x, c) = \texttt{true}$

Remarques :

- A est appelé vérificateur.
- c est appelé certificat et peut-être un entier, un ensemble de valeurs...
- NP (*Nondeterministic Polynomial*) ne veut pas dire « non polynomial ».

Classes de complexité : NP

Intuitivement : on montre qu'un problème P_1 appartient à NP en trouvant, pour chaque instance positive x de P_1, un certificat c de taille polynomiale en $|x|$ permettant de vérifier en temps polynomial en $|x| + |c|$ que x est une instance positive de P_1.

Classes de complexité : NP

Exercice

Montrer que les problèmes suivants appartiennent à NP :

CLIQUE

- Instance : un graphe G et un entier k
- Question : G contient-il une clique de taille k ?

SAT

- Instance : une formule logique φ en forme normale conjonctive
- Question : φ est-elle satisfiable ?

co-FACTORISATION

- Instance : deux entiers n et p
- Question : n ne possède t-il aucun diviseur d tel que $1 < d < p$?

(En admettant que PREMIER $\in$ P)

Classes de complexité : NP

Théorème

P $\subset$ NP.

Remarque : La question « P $=$ NP ? » est un des problèmes ouverts les plus célèbres en informatique.

Classes de complexité : NP

Théorème

P $\subset$ NP.

Remarque : La question « P $=$ NP ? » est un des problèmes ouverts les plus célèbres en informatique.

Exercice

Montrer que NP $\subset$ EXP.

Classes de complexité : Réduction

Définition

On dit qu'un problème de décision $P_1 = (I_1, I_1^+)$ se réduit polynomialement à un problème de décision $P_2 = (I_2, I_2^+)$, noté $P_1 \leq_p P_2$, s'il existe une fonction calculable en complexité polynomiale $f : I_1 \longrightarrow I_2$ telle que : $\forall i \in I_1, i \in P_1 \Leftrightarrow f(i) \in P_2$.

Intuitivement : P_1 se réduit polynomialement à P_2 si un algorithme pour résoudre P_2 permet de résoudre P_1 en complexité polynomiale. Pour cela, il faut pouvoir transformer une instance de P_1 en une instance de P_2 en complexité polynomiale.

Classes de complexité : Réduction

Exercice

Montrer que STABLE $\leq_p$ CLIQUE.

STABLE

- Instance : un graphe G et un entier k
- Question : G contient-il un ensemble stable de taille k ?

CLIQUE

- Instance : un graphe G et un entier k
- Question : G contient-il une clique de taille k ?

Exercice

Montrer que $\leq_p$ est réflexive et transitive.

Classes de complexité : Réduction

Théorème

Soient P_1 et P_2 deux problèmes de décision tels que $P_1 \leq_p P_2$.

- Si $P_2 \in$ P alors $P_1 \in$ P.
- Si $P_2 \in$ NP alors $P_1 \in$ NP.

Classes de complexité : NP-complétude

Définition

Soit P_1 un problème de décision.

- P_1 est dit NP-difficile si $\forall P_2 \in$ NP, $P_2 \leq_p P_1$.
- P_1 est dit NP-complet si P_1 est NP-difficile et $P_1 \in$ NP.

Intuitivement : un problème est NP-difficile s'il est au moins aussi difficile à résoudre que tous les problèmes de NP.

Classes de complexité : NP-complétude

Théorème

Supposons $P_1 \leq_p P_2$.

- Si P_1 est NP-difficile alors P_2 est NP-difficile.
- Si P_1 est NP-complet alors P_2 est NP-complet.

Classes de complexité : NP-complétude

Exercice

Soit P_1 un problème NP-difficile.
Montrer que si $P_1 \in$ P alors P $=$ NP.

Remarque : Il est conjecturé que P $\neq$ NP, donc qu'il est impossible de résoudre un problème NP-complet en temps polynomial.

Classes de complexité : NP-complétude

Théorème de Cook-Levin (admis)

SAT est NP-complet.

Classes de complexité : NP-complétude

Méthode pour montrer qu'un problème P_1 est NP-complet :

1. Montrer que $P_1 \in$ NP.
2. Montrer que $P_2 \leq_p P_1$, où P_2 est un problème NP-complet connu (par exemple SAT).

Classes de complexité : NP-complétude

Théorème

3-SAT est NP-complet.

k-SAT

- Instance : une formule logique φ en forme normale conjonctive (FNC) avec au plus k littéraux par clause.
- Question : φ est-elle satisfiable ?

Classes de complexité : NP-complétude

Théorème

3-SAT est NP-complet.

k-SAT

- Instance : une formule logique φ en forme normale conjonctive (FNC) avec au plus k littéraux par clause.
- Question : φ est-elle satisfiable ?

Remarques :

- On en déduit que k-SAT est NP-complet pour $k \geq 3$, car 3-SAT $\leq_p$ k-SAT.
- Par contre 1-SAT et 2-SAT sont dans P (voir X-ENS 2016)

Classes de complexité : NP-complétude

Exercice

Transformer $\varphi = (x \wedge y) \vee \neg z$ en formule 3-SAT comme dans la preuve précédente (transformation de Tseytin).

Classes de complexité : NP-complétude

Exercice

1. Montrer que STABLE $\in$ NP.
2. Pour $\varphi = \bigwedge_{k=1}^{p} C_k$ une instance de 3-SAT, on définit $G_\varphi = (S, A)$ où :
   • S contient un sommet par littéral, autant de fois qu'il apparaît dans φ.
   • A contient une arête entre deux sommets s'ils sont dans la même clause ou s'ils sont la négation l'un de l'autre.
   Dessiner G_φ si $\varphi = (\overline{x} \vee y \vee \overline{z}) \wedge (x \vee \overline{y} \vee z) \wedge (x \vee y \vee z) \wedge (\overline{x} \vee \overline{y})$.
3. Montrer que si G_φ contient une clique de taille p alors φ est satisfiable.
4. Montrer que si φ est satisfiable alors G_φ contient une clique de taille p. Conclure.
5. Montrer que CLIQUE est NP-complet.

Classes de complexité : NP-complétude

Exercice

Notons $P_1 =$ CYCLE-HAMILTONIEN et $P_2 =$ CHEMIN-HAMILTONIEN.

1. Montrer que P_1 est NP-complet, en trouvant une réduction de 3-SAT à P_1.
2. Montrer que P_2 est NP-complet.

CYCLE-HAMILTONIEN

- Instance : un graphe orienté G
- Question : G contient-il un cycle hamiltonien, c'est-à-dire un cycle passant une et une seule fois par chaque sommet ?

CHEMIN-HAMILTONIEN

- Instance : un graphe G orienté
- Question : G contient-il un chemin hamiltonien, c'est-à-dire un chemin passant une et une seule fois par chaque sommet ?