Calculabilité

TP11

Y. Deville
C-H. Bertrand Van Ouytsel - V. Coppé - A. Gerniers
N. Golenvaux - M. Parmentier

Mai 2021

Questions du test

Les affirmations suivantes sont-elles vraies ?

1. Soient deux problèmes de décision A et B. Si $A \leq_p B$, alors on a automatiquement $A \leq_a B$ et $A \leq_f B$.

   Réponse : Vrai
   $A \leq_p B$ signifie que A est *polynomialement réductible* à B. Il s'agit d'un cas particulier plus restrictif de $\leq_f$ où l'on impose que la fonction f soit de complexité temporelle polynomiale. Précédemment, nous avions égalemment vu que $A \leq_f B$ impliquait $A \leq_a B$.

2. Soient X et Y deux problèmes de décisions tels que $X \leq_f Y$. Si Y est dans P, alors X aussi.

   Réponse : Faux
   $X \leq_f Y$ nous indique qu'il existe une fonction totale calculable f telle que $x \in X \Leftrightarrow f(x) \in Y$ mais on ne connait pas la complexité de la fonction f. Si celle-ci prend un temps exponentiel par exemple (donc non-polynomial) alors X n'appartient pas forcément à P.

Questions du test

Les affirmations suivantes sont-elles vraies ?

3. Soient X et Y deux problèmes de décisions tels que $X \leq_p Y$. Si X est dans P, alors Y aussi.

   Réponse : Faux
   Y peut être dans NP car $X \in P \subset NP$ (exemple : $X =$ déterminer si un nombre est pair, $Y =$ SAT)

4. Soient X et Y deux problèmes de décisions tels que $X \leq_p Y$. Si Y est dans P, alors X aussi.

   Réponse : Vrai
   La réduction $\leq_p$ a été définie dans cette optique. Si Y est dans P et qu'il existe une fonction totale calculable en un temps polynomial f telle que $x \in X \Leftrightarrow f(x) \in Y$, X est également dans P. Il suffit d'appliquer cette fonction f à l'élément de X qui nous intéresse et ensuite d'utiliser l'algorithme de décision polynomial de Y. Cela se fera bien en un temps polynomial.

Questions du test

Les affirmations suivantes sont-elles vraies ?

5. Soient X et Y deux problèmes de décisions tels que $X \leq_p Y$. Si X est NP-complet, alors Y aussi.

   Réponse : Faux
   Y peut être NP-difficile ou non calculable (exemple : $X =$ SAT et $Y =$ HALT)

6. Soient X et Y deux problèmes de décisions tels que $X \leq_p Y$. Si X est NP-complet et Y est NP, alors Y est NP-complet.

   Réponse : Vrai
   Si X est un problème NP-complet (par rapport à $\leq_p$), cela signifie qu'il appartient à NP mais aussi que tout problème appartenant à NP peut-être réduit à X. Or nous avons ici que $X \leq_p Y$, Y est donc nécessairement NP-complet sinon X ne pourrait pas être réduit à Y.

Questions du test

7. Le théorème de Cook affirme que SAT est NP-complet. Désignons par SATpair le problème quasi-identique de satisfaction d'une formule propositionnelle mais qui ne peut posséder qu'un nombre pair de variables propositionnelles. Le problème SATpair est-il aussi compliqué que le problème SAT ? Formulé autrement : le problème SATpair est-il NP-complet ?

   Réponse : Vrai
   SAT $\leq_p$ SATpair (il suffit de rajouter une variable bidon s'il y a un nombre impair de variables) et SATpair $\in NP$, donc SATpair est NP-complet

Questions du test

L'affirmation suivante est-elle vraie ?

8. Soient X, Y et Z, trois problèmes de décision. Si $X \leq_p Y$ et $Y \leq_p Z$, peut-on affirmer avec certitude que $X \leq_p Z$?
   Nous supposons que, dans ces réductions polynomiales, les tailles des outputs des fonctions de complexité temporelle polynomiale f sont bornées par les complexités de f.)

   Réponse : Vrai
   La réduction polynomiale $\leq_p$ est une relation transitive. En effet, s'il existe une fonction f calculable de manière polynomiale telle que $x \in X \Leftrightarrow f(x) \in Y$ et une fonction g calculable de manière polynomiale telle que $y \in Y \Leftrightarrow g(y) \in Z$, il existe une fonction h (comme $f \circ g(x) = f(g(x))$ grâce à la suppposition sur la taille de l'output) calculable en un temps polynomial telle que $x \in X \Leftrightarrow h(x) \in Z$.

Questions du test

L'affirmation suivante est-elle vraie ?

9. Il existe une infinité de problèmes NP-complets.

   Réponse : Vrai
   Notons SATMin$N$, le problème SAT pour les formules qui contiennent au moins N variables propositionnelles différentes (SATMin2 correspond donc aux problèmes SAT pour les formules qui contiennent au moins 2 variables propositionnelles différentes, SATMin3 pour les formules qui contiennent au moins 3 variables propositionnelles différentes et ainsi de suite). Il existe une infinité de ces problèmes SATMin.

   Or, nous avons que $\forall N$ SATMin$N \leq_p$ SAT (car SATMinN est un cas particulier de SAT) et SAT $\leq_p$ SATmin$N$ (car si une expression en contient pas suffisament de variables, il suffit de lui ajouter des variables bidons, ex : $\wedge$ Bidon1 $\wedge$ Bidon2...)

Questions du test

L'affirmation suivante est-elle vraie ?

10. $P = NP$

    Réponse : La personne qui peut répondre et le prouver gagne 1 000 000 USD !

Question 1 du TP

Is there a problem that can be solved in polynomial time in Java, but always takes exponential time with a Turing machine ? Explain.

Question 1 du TP

Is there a problem that can be solved in polynomial time in Java, but always takes exponential time with a Turing machine ? Explain.

Réponse : Non, la différence est au plus d'ordre polynomiale. Il s'agit d'une thèse, car cela n'a jamais été démontré, mais on n'a jamais trouvé de contre-exemple jusqu'ici.

Question 2 du TP

For the following problems A and B find if $A \leq_p B$ and justify :

1. $A(S, x)$ is the problem of knowing if the maximum element of a set of integers S is greater than a value x.
   $B(S, x)$ is the problem of knowing if the minimum element of a set of integers S is lower than a value x.

2. $A(S_1, S_2)$ is the problem of knowing if $\sum_{x \in S_1} x = \sum_{x \in S_2} x$.
   $B(S)$ is the problem of knowing if $\sum_{x \in S} x = 0$.

3. $A(S, y)$ is the problem of knowing if there is a subset $S^* \subseteq S$ such that $\sum_{x \in S^*} x = y$.
   $B(S, y)$ is the problem of knowing if $\sum_{x \in S} x = y$.

Question 2 du TP

For the following problems A and B find if $A \leq_p B$ and justify :

1. $A(S, x)$ is the problem of knowing if the maximum element of a set of integers S is greater than a value x.
   $B(S, x)$ is the problem of knowing if the minimum element of a set of integers S is lower than a value x.

   Réponse :
   Soit la fonction $f(S, x)$ qui multiplie tous les éléments de S et x par -1. Il existe un programme qui calcule cette fonction dont la complexité est dans $O(n)$).
   On peut décider A en utilisant un programme qui décide B :
   $P_A(S, x) \equiv \texttt{return } P_B\big(f(S, x)\big)$

Question 2 du TP

For the following problems A and B find if $A \leq_p B$ and justify :

2. $A(S_1, S_2)$ is the problem of knowing if $\sum_{x \in S_1} x = \sum_{x \in S_2} x$.
   $B(S)$ is the problem of knowing if $\sum_{x \in S} x = 0$.

   Réponse :
   Soit la fonction $f(S_1, S_2)$ qui renvoie un nouvel ensemble S tel que $S = \bigcup_{x \in S_1} \{x\} \bigcup_{x \in S_2} \{-x\}$.
   Il existe un programme qui calcule cette fonction dont la complexité est dans $O(n))$.
   On peut calculer A en utilisant un programme qui calcule B :
   $P_A(S_1, S_2) \equiv \texttt{return } P_B\big(f(S_1, S_2)\big)$

Question 2 du TP

For the following problems A and B find if $A \leq_p B$ and justify :

3. $A(S, y)$ is the problem of knowing if there is a subset $S^* \subseteq S$ such that $\sum_{x \in S^*} x = y$.
   $B(S, y)$ is the problem of knowing if $\sum_{x \in S} x = y$.

   Réponse :
   Soit la fonction $f(S, y)$ qui renverrait un sous-ensemble de S tel que la somme de ses éléments soit égal à y s'il en existe un.
   On peut calculer A en utilisant un programme qui calcule B :
   $P_A(S, y) \equiv \texttt{return } P_B\big(f(S, y), y\big)$.
   Cela dit, pour trouver un tel sous-ensemble, la fonction f devrait naïvement essayer toutes les combinaisons possibles de sous-ensembles de S, ce qui la rend de complexité du programme qui calcule cette fontion de manière triviale non-polynomiale. Jusqu'à preuve du contraire (si $P = NP$), il n'existe pas de programme avec une complexité polynomiale qui calcule cette fonction. En effet, le problème A est en fait un problème NP-complet.

Question 3 du TP

Assume you have a polynomial time algorithm to decide the satisfiability of a propositional formula. Describe how to use this algorithm to find satisfying values for the variables in polynomial time. What kind of reduction is it ? ($\leq_a$, $\leq_f$ or $\leq_p$?)

Question 3 du TP

Assume you have a polynomial time algorithm to decide the satisfiability of a propositional formula. Describe how to use this algorithm to find satisfying values for the variables in polynomial time. What kind of reduction is it ? ($\leq_a$, $\leq_f$ or $\leq_p$?)

Réponse : On utilise une réduction polynomiale.

$assign \leftarrow \{\}$
if $\neg P_{SAT}(F)$ **then**
  return null
for $x \in Vars(F)$ **do**
  if $P_{SAT}(F|_{assign \cup \{(x,\text{true})\}})$ **then**
    $assign \leftarrow assign \cup \{(x, \text{true})\}$
  else
    $assign \leftarrow assign \cup \{(x, \text{false})\}$
return $assign$

Question 4 du TP

Which of the following can we infer from the fact that the Hamiltonian cycle (HC)[1] problem is NP-complete, if we assume that $P \neq NP$? What would your answer be if $P = NP$? Justify.

1. There does not exist a deterministic algorithm that solves the HC problem in polynomial time.
2. There exists an algorithm that solves the HC problem in polynomial time, but no one has been able to find it yet.
3. The HC problem is not in P.
4. The problem of sorting an array can be reduced in polynomial time to the HC problem.
5. All non-deterministic algorithms for the HC problem run in polynomial time.

---

1. The Hamiltonian cycle problem is the problem consisting of deciding if a given graph accepts at least one cycle that visits all vertices exactly once.

Question 4 du TP

Which of the following can we infer from the fact that the Hamiltonian cycle (HC) problem is NP-complete, if we assume that $P \neq NP$?

1. There does not exist a deterministic algorithm that solves the HC problem in polynomial time.
   Réponse : Vrai, sinon $P = NP$.
2. There exists an algorithm that solves the HC problem in polynomial time, but no one has been able to find it yet.
   Réponse : Faux, sinon $P = NP$.
3. The HC problem is not in P.
   Réponse : Vrai, sinon $P = NP$.
4. The problem of sorting an array can be reduced in polynomial time to the HC problem.
   Réponse : Vrai, car $P \subset NP$ et HC est NP-complet.
5. All non-deterministic algorithms for the HC problem run in polynomial time.
   Réponse : Faux, on peut écrire un algorithme moins efficace !

Question 4 du TP

Which of the following can we infer from the fact that the Hamiltonian cycle problem (HC) is NP-complete, if we assume that $P = NP$?

1. There does not exist a deterministic algorithm that solves the HC problem in polynomial time.
   Réponse : ~~Vrai, sinon $P = NP$~~. Faux
2. There exists an algorithm that solves the HC problem in polynomial time, but no one has been able to find it yet.
   Réponse : ~~Faux, sinon $P = NP$~~. Vrai
3. The HC problem is not in P.
   Réponse : ~~Vrai, sinon $P = NP$~~. Faux
4. The problem of sorting an array can be reduced in polynomial time to the HC problem.
   Réponse : Vrai, car $P \subset NP$ et HC est NP-complet.
5. All non-deterministic algorithms for the HC problem run in polynomial time.
   Réponse : Faux, on peut écrire un algorithme moins efficace !

Question 5 du TP

The Traveling Salesman (TS) problem is the problem of finding a cycle in a graph with strictly positive weights that touches every vertex such that the sum of the weights on the cycle do not exceed a given bound. The Hamiltonian Cycle (HC) problem is the problem of finding, in a graph, a cycle that touches every vertex exactly once. Prove that TS is NP-complete by using the fact that HC is NP-complete.

Question 5 du TP

Réponse : Soit $G = (V, E)$ un graphe. Le problème du voyageur de commerce peut être formulé comme un problème de décision : $TS(G, b) \Leftrightarrow \exists cycle \in G$ tel que $cost(cycle) \leq b$.

Pour prouver que TS est NP-complet, il faut prouver que :

i. TS $\in NP$

Les solutions possibles du problème TS sont les cycles de G passant par chaque $v \in V$ dont la longueur est au plus $2|E|$. On a donc un domaine fini qui peut être généré de manière non-déterministe en un temps polynomial. On peut vérifier une solution (i.e. calculer le coût total du cycle et vérifier s'il est $\leq b$) en un temps polynomial. $\Longrightarrow$ TS $\in NP$

ii. HC $\leq_p$ TS

Question 5 du TP

Réponse : Soit $G = (V, E)$ un graphe. Le problème du voyageur de commerce peut être formulé comme un problème de décision : $TS(G, b) \Leftrightarrow \exists cycle \in G$ tel que $cost(cycle) \leq b$.

Pour prouver que TS est NP-complet, il faut prouver que :

i. TS $\in NP$

ii. HC $\leq_p$ TS

Soit $f(G)$ qui assigne un coût de 1 à toutes les arrêtes dans E. Alors on peut calculer HC en utilisant un programme qui calcule TS : $P_{HC} \equiv \texttt{return}\ P_{TS}\big(f(G), |V|\big)$

Rem. : Formellement, le problème TS est défini pour un graphe complet. Pour être complet, la fonction $f(G)$ doit donc aussi ajouter des arrêtes avec un coût ∞ entre chaque paire de nœuds non connectés dans G.

Challenges

1. Provided TS is NP-complete, prove HC is NP-complete.
2. Vrai ou faux : l'intersection de NP et $coNP$ est vide.
3. Prove that $P = NP$ iff $P = coNP$
4. Prove that if there exists A that is NP-complete wrt $\leq_p$ and such that $A \in coNP$, then $NP = coNP$.

Note : $coNP$ est la classe des ensembles dont le **complémentaire** est dans NP.

Challenges

1. Provided TS is NP-complete, prove HC is NP-complete.

   Réponse :

   i. HC $\in NP$

   On peut générer un chemin au hasard dont la longueur est $|V|$ en un temps polynomial (si un cycle existe, au moins une exécution donnera ce cycle).

   ii. TS $\leq_p$ HC

   On définit $f(G, b)$ qui transforme G en un graphe *dirigé* G' de la manière suivante :

   - Initialement, on prend $G' = (V' = V, E' = \emptyset)$.
   - On choisit un nœud a, et on rajoute dans E' une arête dirigée partant de a vers chaque nœud qui peut être rejoint par un chemin de coût $\leq b$ dans G.
   - Pour tout autre nœud x, on rajoute dans E' une arête dirigée de x vers tout nœud y pour lequel il existe dans G un chemin (ou un cycle si $y = a$) qui part de a et rejoint y en passant par x avec un coût $\leq b$.

   Alors $P_{TS}(G, b) \equiv \texttt{return}\ \ P_{HC}\big(f(G, b)\big)$

Challenges

2. Vrai ou faux : l'intersection de NP et $coNP$ est vide.

   Réponse : Faux
   $NP \cap coNP \supseteq P$

3. Prove that $P = NP$ iff $P = coNP$

   Réponse : On sait que $P = coP$. Donc :

$$\begin{aligned} P &= NP \\ coP &= NP \\ cocoP &= coNP \\ P &= coNP \end{aligned}$$

   Rem. : on a donc aussi $P = NP \Leftrightarrow NP = coNP$.

Challenges

4. Prove that if there exists A that is NP-complete wrt $\leq_p$ and such that $A \in coNP$, then $NP = coNP$.

   Réponse : L'existence d'un tel A implique que $NP \subseteq coNP$. Prouvons que cela implique également que $coNP \subseteq NP$. Soit $X \in coNP$. Alors $\bar{X}$ est dans NP. Puisque $NP \subseteq coNP$, $\bar{X}$ est aussi dans $coNP$. Cela signifie que le complémentaire de $\bar{X}$, c'est-à-dire X lui-même, est dans NP. Conclusion : $NP = coNP$.