ESIR1 / ALGC mardi 20 mai 2014
Contrôle continu 30 mn

Ce qui suit n'est qu'une indication des réponses attendues, et pas des phrases à répéter à l'identique. D'une façon générale, relire N fois exactement la même phrase fait douter qu'elle ait été comprise. Le but de ce qui suit est aussi de vous aider à mieux répondre à des questions en général (par exemple un entretien d'embauche).

Je répète que vous devez montrer que vous avez compris la question, et que vous savez y répondre. Savoir y répondre c'est aussi employer les mots du domaine. Pour un élève cuisinier il faut savoir distinguer émincer, ciseler, détailler, effiler, etc, et pas se contenter de dire couper. Pour un élève marin il vaut mieux dire choquer l'écoute pour virer à bâbord plutôt que lâcher la corde pour aller à gauche. Je vois mal pourquoi on serait moins exigeant pour un élève ingénieur que pour un élève cuisinier. Savoir répondre c'est aussi être suffisamment précis. Pour un élève marin dire que devant une bouée Sud il vire pour l'éviter ne me rassure pas ; je veux vraiment savoir de quel côté il veut passer. Pour vous, dire qu'on va faire une réduction ne me rassure pas plus ; je veux vraiment savoir laquelle vous voulez faire. Là encore, je vois mal pourquoi on serait moins exigeant pour un élève ingénieur que pour un élève marinier.

1. **Pour quelle raison fondamentale tous les problèmes ne peuvent-ils pas être décidables ?**

Parce que il y a plus de problèmes que d'algorithmes. Plus précisément, une conséquence des théorèmes de Cantor est que les problèmes forment un ensemble non-dénombrable, et les algorithmes un ensemble dénombrable. Je vous assure que ça tient dans une case. Plusieurs d'entre vous y sont arrivés en toute simplicité.

Ne pas dire qu'il y a potentiellement une infinité de problèmes, et seulement un nombre fini d'algorithmes connus. C'est vrai, mais sans intérêt. Quand un problème est indécidable ça veut dire qu'aucun algorithme, connu ou pas encore connu, ne peut le résoudre. C'est une propriété intrinsèque du problème, et ça n'a rien à voir avec l'état de la connaissance.
Ne pas non plus faire référence au théorème de Rice. Celui-ci nous donne une condition suffisante pour qu'un algorithme qui prend en paramètre des programmes soit indécidable, mais il ne parle que de ça.

2. **Pour un algorithme, qu'est ce que ne pas avoir un coût supportable ?**

On a discuté cette idée au début du cours pour arriver à la conclusion qu'il fallait mettre la barre entre polynomial et exponentiel.

Dire qu'un coût insupportable est une complexité trop importante ou un coût beaucoup trop grand est juste tautologique. Je ne vous demande pas la couleur du cheval blanc d'Henri IV !

3. **Comment appelle-t-on un problème de $\mathcal{NP}$ qui est au moins aussi difficile que tout problème de $\mathcal{NP}$?**

C'est un problème $\mathcal{NP}$-complet.

Il y a eu presque 100 % de bonnes réponses. Cela me ravit, mais m'étonne aussi. Pourquoi une telle réussite à cette question et pas aux autres ?

4. **Comment prouve-t-on en général qu'un problème est comme à la question 3 ?**

En réduisant un problème connu pour être $\mathcal{NP}$-complet (ex. SAT, mais un autre problème $\mathcal{NP}$-complet peut être plus pratique ; cela dépend du problème cible) en ce problème. Préciser le sens de la réduction est hyper important ; dans un sens on répond à la question, dans l'autre on n'apprend rien.

Ne pas dire qu'on doit comparer le problème avec tous les autres de $\mathcal{NP}$! Il y a d'un côté la définition, ici elle suggère de comparer un problème de $\mathcal{NP}$ avec tous les autres, et de l'autre des méthodes de preuves, et on a vu que la méthode de la réduction est un bon cheval de bataille, à condition de l'enfourcher dans le bon sens. C'est très souvent que la définition la plus simple n'est pas la plus opérationnelle, et c'est souvent l'objectif de la science que de mettre des méthodes opérationnelles en face de définitions qui ne le sont pas.

5. **Soit deux problèmes A et B, où A est connu pour être polynomial. Comment prouver la polynomialité de B en s'appuyant sur celle de A ?**

En réduisant B vers A. Voir remarques précédentes. Se rappeler que les bonnes nouvelles vont de droite à gauche, et que être polynomial est une bonne nouvelle (voir question 2).

6. **Qu'est-ce qui permet à la représentation ROBDD d'être souvent efficace pour résoudre SAT ?**

C'est le fait que les algorithmes qui opèrent sur les ROBDD sont polynomiaux en la taille du ROBDD.

Beaucoup ont répondu à une question d'efficacité par un argument de commodité, dans le genre il est facile de reconnaître qu'un ROBDD n'est pas réduit à la feuille FAUX. Certains ont même dit que cela se voyait au premier regard ! Mais un algorithme n'a pas notre regard global, il ne voit presque rien. C'est d'ailleurs pour cela que le principe « Agir local pour un résultat global » est si important. Imaginez-vous lire un texte sur une page quadrillée, mais ne pouvoir voir qu'un carreau à la fois ; c'est tout le challenge de l'algorithmique.
Ne pas dire non plus que les ROBDD sont commodes car leurs feuilles sont forcément VRAI ou FAUX. C'est le cas aussi des arbres de décision, qui pourtant ne sont ni commodes ni efficaces.

7. **Qu'est-ce qui fait que la représentation ROBDD n'est pas toujours efficace ?**

C'est le fait que même si les algorithmes qui opèrent sur les ROBDD sont polynomiaux avec la taille du ROBDD, celle-ci peut être exponentielle.

Certains m'ont parlé de la hauteur du ROBDD, mais celle-ci est fixée, c'est le nombre de variables ; un niveau par variable. Par contre, la largeur est importance, c'est elle qui peut être exponentielle.

8. **Quel est le principe d'une mémo-fonction ?**

C'est une fonction qui stocke les réponses aux précédents appels pour y répondre à coût constant. C'est une stratégie proche de celle d'un cache.

9. **Quel est le principe de l'algorithme DPLL ?**

Rechercher une affectation des variables d'une formule en leur affectant successivement les valeurs VRAI et FAUX. Si on s'en tenait là le coût serait certainement exponentiel, mais on ajoute à cela de repérer des circonstances particulières pour éviter d'explorer certaines combinaisons. Noter que l'exploration pourrait se faire même sans utiliser la forme FNC. Celle-ci ne devient utile que pour bien caractériser les circonstances particulières de façon qu'elles soient reconnaissables par un algorithme qui, je le répète, ne dispose pas de notre vision globale.

10. **Supposons que $C_P = \Omega(g)$, où C_P est la complexité d'un problème P. Supposons aussi que $C_A = O(f)$, où C_A est la complexité d'un algorithme A qui résout P. Rayer celles des affirmations suivantes qui sont fausses.**

1. **$C_P = \Omega(g + c)$, où c est une constante.** Vrai, car on n'est pas à une constante près avec les ordres de grandeur.
2. **Si $g = \Omega(h)$ alors $C_P = \Omega(h)$.** Vrai, car ces relations sont transitives.
3. **$C_A = \Omega(g)$.** Vrai. Pour répondre à ces questions, pas besoin de calculs compliqués. Faites-vous un petit dessin. Truc = Omega(Machin) ? Alors Truc est au-dessus de Machin. Bidule = GrandO(Bazar) ? Alors Bazar est au-dessus de Bidule. Foo est un algorithme qui résout Bar ? Alors Foo, qui n'est qu'un des algorithmes qui résolvent Bar, et pas forcément le meilleur, est au-dessus de Bar. Vous dessinez les enveloppes des ensembles Omega(…) et GrandO(…), et vous regardez ce qui se passe. Et vous, vous pouvez juger globalement de la situation, pas comme un algorithme.
4. **$g = O(f)$.** Vrai**.**
5. **$\Omega(f)$ est inclus dans $\Omega(g)$.** Vrai**.**
6. **Si $C_A = \Theta(f)$ alors $C_P = \Theta(g)$.** Faux, pour déduire un Théta(…), il faut montrer une domination dans les deux sens, mais ici on n'a rien qui permet de le montrer**.**
7. **$C_P = \Theta(f)$.** Faux**.**
8. **f est une fonction exponentielle.** Faux, rien ne permet de dire une chose pareille. C'est comme si je vous demandais l'âge du capitaine.

Je ne devrais pas avoir à le dire, mais quand la consigne est rayer, il faut rayer, et pas entourer ou colorier en rose. Certaines de vos façons de vous tirer des balles dans le pied me dépassent complètement !

Je vous souhaite un bon stage, et vous encourage à prendre au sérieux mes remarques dont la portée n'est vraiment pas limitée au contrôle continu d'un cours d'algorithmique.