EXAMEN CORRIGE D'ALGORITHMIQUE, M1, Janvier 2014

Une page A4 recto-verso autorisée. Calculatrice, téléphone, ordinateur, lunettes Google interdits. N'écrivez aucun programme. Ecrivez lisiblement. Répondez aux questions dans l'ordre, en indiquant le numéro de chaque question.

Les réponses de ce corrigé sont bien plus détaillées que celles qui étaient attendues. Chacune des 8 premières questions a été notée sur $1/2$ point ; chacune des 16 autres questions a été notée sur 1 point.

1. Citez un problème indécidable en informatique.

Le problème de l'arrêt (d'une machine de Turing, en fait de tout programme).

2. Citez en un deuxième.

L'égalité de deux nombres réels, ou de deux nombres complexes, ou de deux fonctions (définies sur des domaines infinis). L'isomorphisme entre deux groupes infinis non commutatifs. Le 10 ième problème de Hilbert : est ce qu'une équation diophantienne a des solutions (en nombres entiers naturels, ou relatifs) ; en passant, la preuve de Matiyasevitch utilise la suite de Fibonacci.

3. Citez un problème informatique pour lequel aucun algorithme en temps polynomial n'est connu.

A part les problèmes indécidables, citons : 3-SAT ou MAX-SAT (satisfaire le maximum de clauses), la clique ou le stable max (ou de cardinal donné), un circuit ou un chemin hamiltonien, le sac à dos (knapsack), le voyageur de commerce, la coloration en $k \geq 3$ couleurs des sommets d'un graphe, et des centaines d'autres.

4. Citez en un deuxième.

5. Citez un algorithme optimal de tri, qui procède par comparaisons ; quel est l'ordre de grandeur du nombre de comparaisons effectuées pour trier n éléments ?

Le tri par fusion ("mergesort") est optimal, et effectue $O(\log n!) = O(n \log n)$ comparaisons entre les éléments qu'il faut ordonner.

6. Citez en un autre.

Le tri par tas ("heapsort"), ou une variante appelée "smoothsort", est aussi optimal. Les tris utilisant un arbre équilibré (les éléments sont insérés dans un arbre initialement vide, puis le plus petit élément est retiré de l'arbre, jusqu'à ce que l'arbre soit vide) sont aussi optimaux. Il existe de très nombreux arbres équilibrés (AVL, B-tree, etc). Le tri rapide ("quicksort") peut être en n^2 dans le pire des cas, mais est probablement en $O(n \log n)$.

7. En TP, nous avons affiché des courbes définies par leurs équations. Comment avons nous représenté ces équations ?

Par des "DAGs" ("Directed Acyclic Graph"). Ce sont des arbres où des noeuds peuvent être partagés : un noeud peut avoir plusieurs pères. Nous avons défini, en ocaml, le type dag ou expression comme suit :

```
type dag= Nb of float | Plus of (dag*dag)
        | Mult of  (dag*dag) | Var of string;;
```

ou bien (selon les groupes de TP) :

```
type dag= Nb of float | Plus of (dag*dag) | Mult of  (dag*dag) | X | Y;;
```

8. (suite) Comment avons nous fait pour décider que certains rectangles ne pouvaient pas être traversés par la courbe ?

Nous avons défini et utilisé une arithmétique d'intervalles ; par exemple $[a, b] + [a', b'] = [a + a', b + b']$. Pour tracer la courbe d'équation $f(x, y) = 0$, la fonction f était représentée par un DAG ou une expression, et $f(X, Y)$ était évaluée par l'arithmétique d'intervalles, avec X et Y deux intervalles, représentant donc une boîte. Quand 0 n'appartient pas à l'intervalle $f(X, Y)$, alors la courbe ne peut pas passer par la boîte (X, Y) car $f(x \in X, y \in Y)$ ne peut s'annuler. Sinon, la boite (X, Y) est coupée en 4, sauf quand elle est trop petite ; en ce cas, elle est affichée et remplie.

9. En TP, nous avons calculé les racines comprises entre 0 et 1 de polynômes univariés. Quel est le principe de la méthode que nous avons utilisée ?

Première méthode. Le polynôme $p(x)$ a été converti dans la base de Bernstein. Dans cette base, le plus petit coefficient et le plus grand coefficient donne un encadrement pour $p([0, 1])$. Quand cet intervalle contient 0, nous avons utilisé une méthode de subdivision : la méthode de Casteljau. Pour les racines plus grandes que 1, ou les racines négatives, nous avons utilisé un changement de variables.

Deuxième méthode. Nous avons aussi programmé la méthode de Newton, ou Newton-Raphson. Contrairement à la méthode précédente, elle ne fournit pas toutes les racines réelles. Dans les bons cas, elle converge vers une racine dépendant de la valeur initiale.

10. Vous disposez d'une arithmétique sur de grands entiers relatifs, qui fournit l'addition, la soustraction, le produit, le quotient, le modulo, la comparaison, la conversion à partir d'unechaîne de caractères, ou vers une chaîne de caractères ; mais elle ne fournit pas la partie entière de la racine carrée. Proposez un algorithme raisonnable (en temps polynomial avec le nombre de chiffres de a) pour calculer le grand entier $x \geq 0$ tel que $x^2 \leq a < (x + 1)^2$, avec a un grand entier donné. Ceci exclut l'algorithme qui essaie tous les entiers 0, 1, 2, ... jusqu'à trouver la solution.

Utiliser la méthode de Newton : $x_{n+1} = N(x_n) = (x_n + a/x_n)/2$, en arrondissant $N(x)$ sur l'entier supérieur ou égal ; autrement dit utiliser $N(x) =$

$\lceil (x + a/x)/2 \rceil$. Pour l'initialisation, calculez la première puissance de 4 qui est plus grande ou égale à a ; si c'est 4^k, alors partir de $x_0 = 2^k$. Calculer le point fixe x^* de N en partant de x_0. Puis tester $x^* - 1$, $x^* - 2$, etc jusqu'à trouver la solution ; x^* ne peut pas être beaucoup plus grand que la solution. En fait la solution se trouve dans l'intervalle $[\lfloor a/x^* \rfloor, x^*]$. Il est aussi possible de procéder par dichotomie dans cet intervalle.

11. (suite) Proposez un deuxième algorithme, de principe différent.

Procéder (uniquement) par dichotomie. L'intervalle initial est trouvé comme précédemment : si $4^{k-1} < a \leq 4^k$, alors $2^{k-1} \leq \lfloor \sqrt{a} \rfloor \leq 2^k$. Pour trouver k, l'algorithme trivial (essayer $4, 4^2, 4^3, \ldots$ ou diviser a par 4 itérativement jusqu'à obtenir 1 ou 0) suffit, puisqu'il aboutit en $O(\log_4 a)$ étapes.

12. En TP, nous avons utilisé la programmation dynamique pour résoudre deux problèmes. Citez un de ces problèmes.

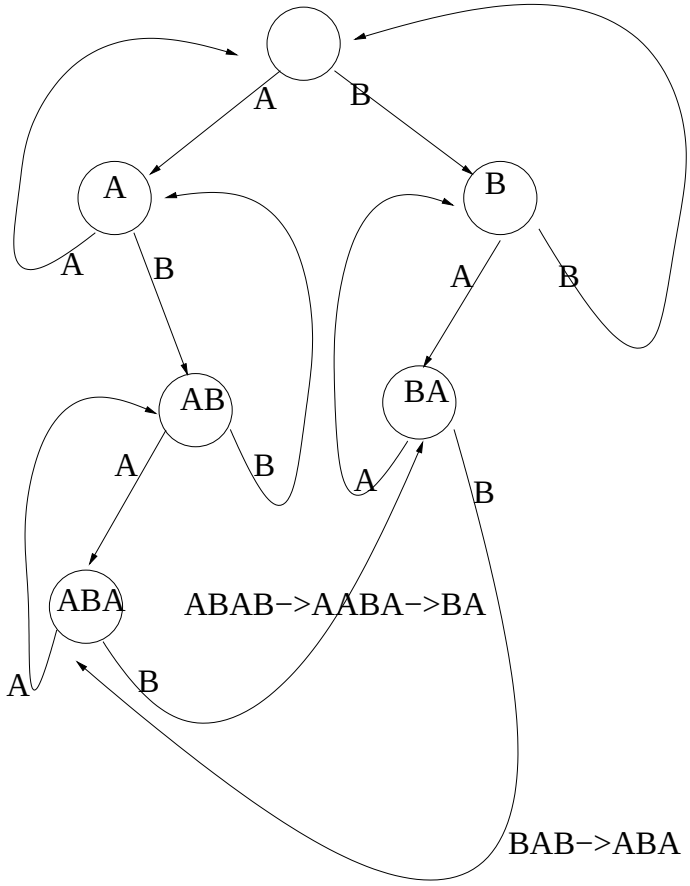
Le sac à dos avec des poids entiers.

13. (suite) Citez l'autre.

Le parenthésage optimal pour un produit de matrices (de tailles différentes). Un autre problème résoluble par programmation dynamique est la plus longue séquence commune à deux séquences données. Certains algorithmes de plus courts chemins peuvent être considérés comme de la programmation dynamique. Rappel : la programmation dynamique résout un problème d'optimisation combinatoire, en combinant les solutions optimales de sous problèmes de même nature ; la difficulté vient de ce qu'un problème donné se décompose de plusieurs façons ; il faut donc les tester toutes, et il faut éviter de recalculer plusieurs fois la même chose. Nous avons utilisé des tables de hachage pour régler ce problème. Le calcul des dates au plus tôt et au plus tard ne relève pas de la programmation dynamique (mais il est commode d'utiliser des tables de hachage ; ceci explique l'erreur de certains étudiants). Enfin, la programmation dynamique ne doit pas être confondue avec la programmation linéaire !

14. La méthode de Knuth-Bendix a terminé avec succès sur ces trois règles de ré-écriture : $AA \to \epsilon$; $BB \to \epsilon$; $BAB \to ABA$. On rappelle que ϵ est le mot vide. Listez les mots (formés des lettres A et B) irréductibles. Disposez les sous forme arborescente.

Les mots irréductibles sont $\epsilon, A, B, AB, BA, ABA$. Voici l'arbre, binaire ; le fils gauche du mot m est mA, le fils droit du mot m est mB ; la racine porte le mot vide ϵ. Les mots sont réduits par les règles de ré-écriture, ce qui explique que certains arcs "remontent". Par exemple, le fils droit du mot $m = ABA$ est $mB = ABAB = A(BAB) \to A(ABA) = (AA)BA \to BA$.



15. Vous disposez d'une balance à deux plateaux, de haute précision. Vous avez 3 pièces d'or ; l'une d'elles est fausse (exactemement une), et plus lourde. Comment, en une seule pesée, pouvez vous déterminer quelle est la pièce fausse ? Rédigez votre méthode avec soin.

Soient A, B, C les trois pièces. Vous posez A sur le plateau de gauche, et B sur le plateau de droite ; si (le poids de) A égale (le poids de) B, alors C est la pièce fausse ; sinon la balance penche et vous indique qui de A ou B est la plus lourde, et donc la fausse pièce.

16. (suite). Vous avez 3^k pièces d'or ; exactement une des 3^k pièces d'or est fausse et plus lourde. Comment, en k pesées, pouvez vous déterminer quelle est la pièce fausse ? Rédigez votre méthode avec soin.

Bien sûr, $k \geq 0$. Si $k = 0$, il n'y a qu'une pièce, et elle est fausse. Sinon, $k > 0$, et vous partitionnez l'ensemble P des 3^k pièces en trois ensembles A, B, C de tailles égales ($3^k/3 = 3^{k-1}$). Vous posez toutes les pièces de A sur le plateau

de gauche, et toutes les pièces de B sur le plateau de droite. Si (le poids de l'ensemble) A égale (le poids de l'ensemble) B, alors C contient la pièce fausse ; sinon la balance penche et vous indique qui de A ou B contient la pièce la plus lourde. Dans tous les cas, la taille de l'ensemble qui contient la fausse pièce a été divisée par 3. k pesées détermineront donc quelle est la pièce fausse parmi les 3^k pièces.

17. Déroulez les trois phases du tri par base ("radix sort") sur cet ensemble d'entiers : 888, 414, 148, 814, 881, 114, 118, 811, 844, 488, 814. N'utilisez que les trois tiroirs évidents.

Triez sur le chiffre des unités :
1 : 881, 811
4 : 414, 814, 114, 844, 814
8 : 888, 148, 118, 488
Concaténez : 881, 811, 414, 814, 114, 844, 814, 888, 148, 118, 488.
Triez sur le chiffre des dizaines :
1 : 811, 814, 114, 814, 118
4 : 844, 148
8 : 881, 888, 488
Concaténez : 811, 814, 114, 814, 118, 844, 148, 881, 888, 488.
Triez sur le chiffre des centaines :
1 : 114, 118, 148
4 : 488
8 : 811, 814, 814, 844, 881, 888
Concaténez : 114, 118, 148, 488, 811, 814, 814, 844, 881, 888. C'est bien correct.

18. Déroulez l'algorithme d'Euclide étendu (ou Bézout) pour $a = 100$ et $b = 46$ en remplissant un tableau comme ci-dessous. L'algorithme vu en cours calcule g le PGCD de a et b, ainsi que deux entiers relatifs ($\mathbb{Z}$) u et v. u et v sont tels que $au + bv = g = PGCD(a, b)$; on note $r = a \mod b$, et $q = \lfloor \frac{a}{b} \rfloor$ le quotient de a par b. Il n'y a qu'une seule réponse correcte.

a	b	r	q	g	u	v
100	46	8	2	2	6	-13
46	8	6	5	2	-1	6
8	6	2	1	2	1	-1
6	2	0	3	2	0	1
2	0	indéf	indéf	2	1	0

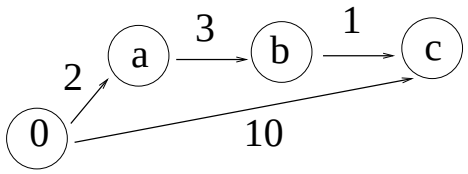
19. (suite) Quand vous remplissez les cases des colonnes u et v, en fonction du contenu de la ligne d'après ou d'avant, quelles formules utilisez vous ? Réponse suggérée : soient (g', u', v') le contenu de la ligne en dessous (ou au dessus ?). Alors $(g, u, v) = (g', ?, ?)$. La preuve n'est pas demandée. Il n'y a qu'une seule réponse correcte.

Nous savons que $g = g'$. Nous avons : $a = bq + r, au + bv = g, bu' + rv' = g$. Remplaçons a par sa valeur $bq+r$ dans $au+bv = g$. Nous obtenons $(bq+r)u+bv = g \Rightarrow b(qu + v) + ru = g$. Confrontons $b(qu + v) + ru = g$ et $bu' + rv' = g$; la solution évidente est $qu + v = u', u = v' \Rightarrow u = v', v = u' - qu = u' - qv'$. Donc

$$(g, u, v) = (g', v', u' - qv')$$

Certains étudiants ont remarqué que $g' = g, u = v'$, et donc $au + bv = g \Rightarrow v = (g - au)/b = (g - av')/b$. C'est correct aussi.

20. Posez le problème du plus court chemin entre le sommet étiqueté 0 et le sommet étiqueté c comme un problème de programmation linéaire, avec la méthode du potentiel (a, b, c sont les potentiels). Utilisez d'abord des inéquations, puis utilisez des variables d'écart.



Le problème de PL est :

$$a \leq 2, b \leq a + 3, c \leq b + 1, c \leq 10, \max c$$

On peut aussi maximiser $a+b+c$: ceci permettra de trouver tous les plus courts chemins. Attention : on maximise, ce qui peut être contre-intuitif. Ajoutons les variables d'écart, dans l'ordre s_1, s_2, s_3, s_4 :

$$a + s_1 = 2, b + s_2 = a + 3, c + s_3 = b + 1, c + s_4 = 10, \max a + b + c$$

Voici le tableau initial :

$$\begin{array}{rcl} \max & : & a + b + c \\ s_1 & = & 2 - a \\ s_2 & = & 3 + a - b \\ s_3 & = & 1 + b - c \\ s_4 & = & 10 - c \end{array}$$

21. (suite) Résolvez ce problème par la méthode du simplexe.

Augmentons la valeur de a ; il faut pivoter avec s_1 :

$$\begin{array}{rcl} \max & : & 2 - s_1 + b + c \\ a & = & 2 - s_1 \\ s_2 & = & 5 - s_1 - b \\ s_3 & = & 1 + b - c \\ s_4 & = & 10 - c \end{array}$$

Augmentons b ; il faut pivoter avec s_2 :

$$\begin{aligned} \max &: 7 - 2s_1 - s_2 + c \\ a &= 2 - s_1 \\ b &= 5 - s_1 - s_2 \\ s_3 &= 6 - s_1 - s_2 - c \\ s_4 &= 10 - c \end{aligned}$$

Augmentons c ; il faut pivoter avec s_3 :

$$\begin{aligned} \max &: 13 - 3s_1 - 2s_2 - s_3 \\ a &= 2 - s_1 \\ b &= 5 - s_1 - s_2 \\ c &= 6 - s_1 - s_2 - s_3 \\ s_4 &= 4 + s_1 + s_2 + s_3 \end{aligned}$$

C'est terminé : tous les coefficients de la fonction objectif sont négatifs. Vous pouvez vérifier visuellement sur le graphe que $a = 2, b = 5, c = 6$ est bien la solution.

Remarques : le tableau initial donne s_1, s_2, s_3, s_4. En calculant l'objectif $13 - 3s_1 - 2s_2 - s_3$ du tableau final, vous trouverez bien $a + b + c$. Les coefficients des variables (sauf dans la fonction objectif) valent tous -1, 0 ou 1, à cause de la totale unimodularité de la matrice sous jacente.

Il est possible d'exprimer ce problème comme un problème de flot 1 de coût minimum, et d'exprimer ce problème comme un problème de programmation linéaire ; mais ce n'était pas ce qui était demandé dans l'énoncé.

22. Quel problème résout la méthode appelée "union find" ?

Un programme crée des éléments initialement distincts $e_1, e_2, \ldots$ puis découvre ensuite que certains éléments sont en fait égaux. La méthode appelée "union find" résout ce problème de façon simple et efficace ; elle gère des classes d'équivalence.

23. Citez un problème ou un algorithme vu en cours qui utilise l'"union find".

Les composantes connexes d'un graphe. Une méthode de calcul de l'arbre couvrant optimum (min ou max). Le calcul des régions d'une image binaire (ou avec un faible nombre de couleurs).

24. Certains problèmes, bien que décidables, sont si difficiles que l'on ne sait même pas vérifier leur solution en temps polynomial. Citez en un.

La clique max. Le stable max. MAX-SAT. Le sac à dos. Le voyageur de commerce... En cas de réponse négative (il n'y a pas de solution), 3-SAT. Plus généralement, les problèmes NP quand il n'y a pas de solution (rappel : quand il y a une solution à un problème dans NP, il est possible de la vérifier en temps polynomial ; exemple : 3-SAT, clique ou stable de cardinal donné, chemin ou circuit hamiltonien, etc).