INFO31 - Examen 2de session – juin 2014 - correction

Notations pour les questions 1 et 2

- le modulo (reste de la division entière de a par b) sera noté a%b, variable r
- le résultat de la division entière de a par b sera noté $\lfloor \frac{a}{b} \rfloor$ variable q

Question 1 Déroulez l'algorithme d'Euclide pour calculer le PGCD de 165 et 75

Rappel condition : `a>=0 et b>=0 a>=b`
arrêt : `pgcd(a,0) =a`
base de la récursion : `pgcd(a,b) =pgcd(a,a%b)`

Calcul pgdc(165,75)=pgcd(75,15) car 165=75*2+15
pgcd(75,15)= pgcd(15,0) car 75=15*5+0 : c'est le cas d'arrêt
donc pgcd(165,75)=pgcd(75,15)=15

Questions 2 et 3

2- Déroulez l'algorithme d'Euclide étendu (ou Bézout) pour a = 165 et b = 75 en remplissant un tableau comme ci-dessous. Dans la dernière ligne, b est nul et a est le PGCD cherché. L'algorithme vu en cours calcule g le PGCD de a et b, ainsi que u et v. u et v sont tels que au + bv = g = P GCD(a, b) ; on note r = a mod b, et q = ⌊ a ⌋ le quotient de a par b. Il n'y a qu'une seule réponse correcte.

3- (suite) Soient a ∈ N et b ∈ N deux entiers naturels donnés. Soit une solution (u, v), donnée elle aussi, du problème de Bézout : donc au + bv = PGCD(a, b). Mais cette solution (u, v) n'est pas minimale : il existe une autre solution (u′ , v ′) telle que au′ + bv ′ = PGCD(a, b) et le vecteur (u′ , v ′) est plus court que (u, v). Proposez un algorithme pour trouver une solution plus petite que (u, v), s'il en existe. Attention : vous devez utiliser (u, v), pas seulement a, b.

Rappel Bezout(a,b) renvoie trois valeurs (u, v et pgcd)
condition : `a>=0 et b>=0 a>=b`
arrêt : bezout(a,0) renvoie `u=1, v=0, pgcd=a`
base de la récursion : u', v' et pgcd' reçoivent les résultats de Bezout(a,a%b)
et Bezout renvoie

```
u = v'
v=u'- b/a * v'
pgcd=pgcd'
```

Au tableau demandé sont ajoutées :

- une colonne contenant le niveau 1 pour l'appel initial, 2 pour le suivant, etc.
- une colonne pour la vérification de la propriété au+bv=pgcd

Avec la notation en niveaux : $u_3=1$ et $v_3=0$ et $u_{n-1}=v_n$ et $v_{n-1}=u_n-q_{n-1}v_n$

Etape 1 : remplir les premières colonnes du tableau depuis le niveau 1 (arrêt pour b=0)

niveau	a	b	r=a%b	q	g=pgcd(a,b)	u	v	au+bv
1	165	75	15	2				
2	75	15	0	5				
3	15	**0**						

Etape 2 : à l'arrêt :

- a contient le pgcd qui peut être reporté partout
- **u vaut 1 et v vaut 0** (sur cet exemple u_3 et v_3)

niveau	a	b	r=a%b	q	g=pgcd(a,b)	u	v	au+bv
1	165	75	15	2	15			
2	75	15	0	5	15			
3	**15**	0			15	**1**	**0**	

Etape 3: remplir les dernières colonnes du tableau depuis le niveau 3 vers le niveau 1

niveau	a	b	r=a%b	q	g=pgcd(a,b)	u	v	au+bv
1	165	75	15	2	15	1	0-2*1 =-2	
2	75	15	0	5	15	0	1-0 =1	
3	15	0			15	1	0	

… et vérifier :

niveau	a	b	r=a%b	q	g=pgcd(a,b)	u	v	au+bv
1	165	75	15	2	15	1	-2	165*1-2*75=15
2	75	15	0	5	15	0	1	75*0+15=15
3	15	0			15	1	0	15*1+0*0=15

Question 4 et 5

4- Vous avez 3 pièces d'or, une balance à deux plateaux, et vous savez qu'exactement une des pièces sur les trois est fausse : elle est plus légère. Expliquez comment vous détectez la pièce fausse en une seule pesée.

5- (suite) Vous avez 3k pièces d'or, une balance à deux plateaux, et vous savez qu'exactement une des pièces est fausse : elle est plus légère. Expliquez comment vous détectez la pièce fausse en un minimum de pesées.

Le traitement est récursif (sur le nombre de pièces).

A chaque étape le nombre de pièces est divisée par trois (on retombe sur 3 puisqu'il y a au départ 3^k pièces).

Le **cas d'arrêt** correspond à la question 3 : deux pièces sont placées sur la balance (une pièce sur chaque plateau). Si la balance n'est pas équilibrée, elle indique la pièce fausse (la plus légère). Si la balance est équilibrée, les deux pièces sur la balance ont le même poids, la fausse pièce est donc celle qui n'est pas sur la balance.

Pour 3^k pièces avec $k > 1$, on partage les pièces en trois paquets de 3^{k-1} pièces chacun. On procède avec chaque paquet comme avec les pièces (on place deux paquets sur la balance, un sur chaque plateau), etc. Ceci permet de trouver lequel des trois paquets contient la fausse pièce. On recommence alors sur ce paquet (qui ne contient que 3^{k-1} pièces).

Question 6 Quel est le nom de la méthode utilisée pour résoudre "le problème des reines" et "le compte est bon" ?

La méthode de recherche avec backtrack (retour-arrière) dans l'ensemble de toutes les solutions possibles.

Question 7 Quel est l'ordre de grandeur du nombre minimum de produits matriciels nécessaires

pour calculer M^n, où M est une matrice carrée, et n un entier naturel ?
L'ordre de grandeur est en log(n) car l'exposant peut-être divisé par deux en utilisant la propriété :

$$M^{2k} = (M \times M)^k$$

Rappel : la méthode à utiliser est celle de la multiplication rapide définie par :

- les cas d'arrêt :$M^0 = I$ et $M^1 = M$
- le traitement récursif proprement dit est décrit pour les valeurs paires et impaires de l'exposant :
  - pour $M^{2k} = (M \times M)^k$: multiplication de matrice et division par 2 de l'exposant pour l'appel récursif
  - pour $M^{2k+1} = M \times M^{2k}$: multiplication de matrice et diminution de 1 de l'exposant pour l'appel récursif

Tous les deux appels récursifs au plus, l'exposant est divisé par 2, donc il y a au plus :

$2 \log_2 n = O(\log n)$

appels récursifs et multiplications de matrices.

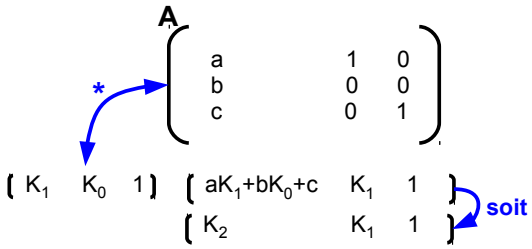
Question 8 Proposez une méthode rapide pour calculer Kn , où K0 , K1 sont donnés, et Kn = a Kn−1 + b Kn−2 + c quand n > 1, avec des valeurs connues pour a , b , c. Indication : utilisez une matrice de taille 3 × 3 et l'algorithme de puissance rapide.
DOMINIQUE, j'ai renommé a et b les constantes a1 et a2 pour « alléger » les notations … dis-moi si cela te pose problème. ???
Comme pour le calcul de la suite de Fibonacci, il est possible d'écrire :

$$\begin{pmatrix} K_{n+2} & K_{n+1} & 1 \end{pmatrix} = \begin{pmatrix} K_n & K_{n-1} & 1 \end{pmatrix} * \begin{pmatrix} a & 1 & 0 \\ b & 0 & 0 \\ c & 0 & 1 \end{pmatrix}$$

et

$$\begin{pmatrix} K_{n+2} & K_{n+1} & 1 \end{pmatrix} = \begin{pmatrix} K_n & K_{n-1} & 1 \end{pmatrix} * \begin{pmatrix} a & 1 & 0 \\ b & 0 & 0 \\ c & 0 & 1 \end{pmatrix}^n$$

En effet :



d'où un calcul possible sous la forme $(K^{n+2} \quad K^{n+1} \quad 1) = (K^n \quad K^{n-1} \quad 1)A^n$
avec la multiplication rapide des matrices définie par :

$A^0 = 1$
$A^1 = I$, la matrice identité
$A^{2k} = (A^2)^k$ pour les puissances paires et $A^{2k+1} = A * (A^2)^k$ pour les puissances impaires

Question 9 Les paysans russes réduisaient la multiplication de deux entiers naturels à des sommes (dont des doublements), et des divisions par deux : l'un des multiplicandes est implicitement décomposé en base 2. Donnez les formules récursives correspondantes pour a × b ; vous supposerez que c'est a qui est décomposé en base 2.
Cas d'arrêt : a=1 (le produit est alors b)
A chaque étape, si a est >1 :

- on fait un appel récursif sur `a/2` et `b*2`
- si a est impair : on ajoute `a` au résultat de l'appel récursif

D'où :

- russe(1,b)=b
- pour a>1 et a impair : russe (a,b)=a+russe(a/2,b*2)
- pour a>1 et pair : russe (a,b)=russe(a/2,b*2)

Question 10 Définissez le problème SAT en 3 lignes au plus. Un algorithme rapide pour le résoudre est-il connu ?

DOMINIQUE, je ne sais pas à quels niveaux de formalisation et précision les étudiants doivent répondre à cette question de cours...

Un solveur SAT est un programme qui décide automatiquement si une formule de logique propositionnelle est satisfaisable .

Ou ...

Problème SAT : étant donnée une formule CNF (en forme normale conjonctive) du calcul propositionnel à n variables, existe-t-il une valuation des n variables qui satisfait la formule ?

C'est un problème NP-COMPLET (il est donc décidable mais difficile).
Faut-il citer le théorème de Cook ?

Question 11 Un compilateur peut-il décider si deux fonctions sont équivalentes (deux fonctions sont équivalentes si elles donnent les mêmes résultats pour tous les arguments possibles) ?
Non, il s'agit d'un problème indécidable.

Question 12 Soit l'équation f (x) = x2 − 4 = 0. Définissez la fonction de Newton N(x) associée. Quel sera le point fixe de N en partant de x0 = 4 ? Pour x ∈ [1/2, 5], dessinez la courbe (x, y = N (x)), la droite d'équation x = y, et les premières étapes de la méthode de Newton, en partant de x0 = 4, et de x0 = 1. Pour cela vous calculerez (sans calculette) les valeurs numériques de N(0), N(1), N(2), N(3), N(4), N(5).

La fonction de Newton associée à $f(x)$ est définie par $N(x)=x-\frac{f(x)}{f'(x)}$

soit $N(x)=x-\frac{x^2-4}{2x}=\frac{x^2+4}{2x}$

Tableau de valeurs de N(x) :

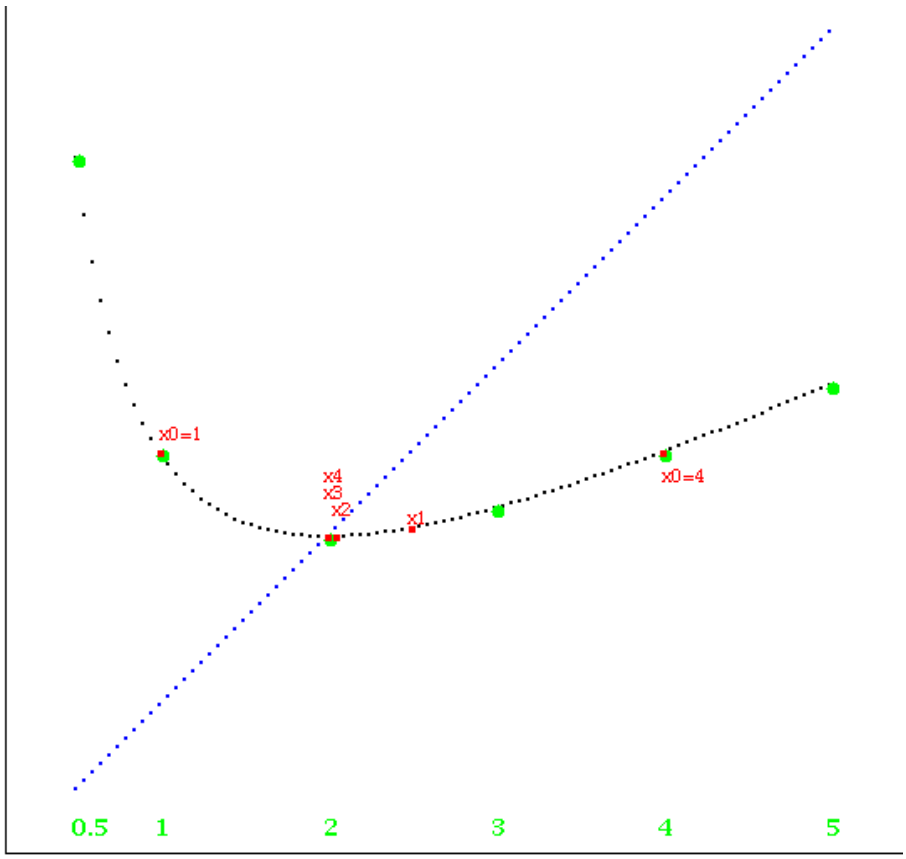
x	0 (positif)	1/2	1	**2**	3	4	5
N(x)	+∞	17/4 = 4,25	5/2 = 2,5	**2**	13/6 = 2,2	5/2 = 2,5	29/10 = 2,9

2 est un point fixe de N car N(2)=2

Suite $x_n=N(x_{n-1})$:

x	1 ou 4	2,5	2,05	2,0006
N(x)	2,5	2,05	2,0006	2,0000009

En partant de x0=1 ou x0=4, x_n décroît vers 2.



Sur la figure ci-dessus :

- la courbe N(x) est en noir avec les points du tableau de valeurs en vert,
- la droite y=x est en bleu,
- les points de la suite $x_n=N(x_{n-1})$ à partir de $x_0=1$ et $x_0=4$ sont en rouge.

Question 13 Un arbre binaire, représentant une expression arithmétique, est donné ; ses feuilles portent des nombres ou des noms de variables ; ses noeuds portent des noms d'opérations : +, ×. Décrivez un algorithme pour convertir cet arbre en une expression postfixe. Rappel : 1, 2, 3, ×, + est une expression postfixe, dont l'évaluation donne 1 + 2 × 3 = 7.

Pour passer une expression arithmétique d'un format arbre binaire à un format postfixe, il faut parcourir l'arbre binaire en profondeur en traitant les éléments dans l'ordre filsgauche, filsdroit, racine.

Notations

Pour un arbre binaire a, nous utilisons `fg(a)`, `fd(a)` et `rac(a)` pour accéder respectivement à son sous-arbre gauche, sous-arbre droit et à l'élément stocké sur sa racine (sous forme de chaîne de caractère). Une fonction booléenne `estVide(a)` permet de savoir si un arbre est vide.

La concaténation sur les chaînes de caractères est notée par un point.

Algorithme

```
postfixe(a) :
   // en paramètre l'arbre contenant l'expression mathématiques
   // le résultat est la chaîne de caractères
   //                 contenant l'expression postfixe
   // HYPOTHESE : l'arbre représente une expression correcte
   si estVide(a)
   alors res ← ""
   sinon resg ← postfixe(fg(a))
            resd ← postfixe(fd(a))
            res ← resg." ".resd." ".rac(a)
   fsi
   résultat res
```

Question 14 (suite). Une expression postfixe est représentée par une liste d'éléments ; chaque élément est soit un nombre, soit un nom de variable, soit un nom d'opération (+, ×). Décrivez un algorithme pour convertir cette expression postfixe en arbre binaire. Indication : vous pouvez vous inspirer de l'algorithme d'évaluation d'une expression postfixe.

Pour passer une expression arithmétique d'un format postfixe à un format arbre binaire, nous avons besoin d'une pile contenant des arbres binaires (qui sont les sous-arbres de l'arbre final).

Notations

Sur les arbres binaires, nous utilisons créer(`ch`, `fg`, `fd`) qui renvoie un arbre binaire ayant à la racine la chaîne de caractère `ch`, et comme fils gauche et fils droit les arbres `fg` et `fd`, respectivement. Un arbre vide sera noté `arbreVide`.

Sur les piles nous disposons d'une opération `empiler(p,a)` qui ajoute l'arbre `a` en sommet de pile et d'une opération `sommet(p)` qui renvoie l'arbre placé au sommet de la pile et qui dépile. Une opération `pileVide()` permet d'initialiser une pile vide.

Sur les listes, nous disposons d'une fonction `tete(l)` qui récupère le premier élément de la liste et d'une fonction `suite(l)` qui récupère la liste privée de son premier élément. Une fonction booléenne `listeVide(l)` permet de savoir si une liste est vide.

Algorithme

```
arbre(l) :
   // en paramètre la liste contenant l'expression mathématiques
   // le résultat est l'arbre construit
   // HYPOTHESE : la liste représente une expression correcte
   p ← pileVide()
   tant que listeVide(l) est faux
   faire elem ←  tete(l)
         l ← suite(l)
         selon cas
               cas  operateur(elem)
                     ad ← sommet(p)
                     ag ← sommet(p)
                     a ← creer(elem,ag,ad)
                     empiler(p,a)
               cas variable(elem) ou nombre(elem)
                     a ← creer(elem,arbreVide,arbreVide)
                     empiler(p,a)
         fcas
   ftq
   res ← sommet(p)
   résultat res
```