UNIVERSITÉ DE NANTES

QUATRIÈME SEMESTRE, 2015-2016

IUT - Département d'informatique

Devoir Surveillé - Compléments de mathématiques

Durée : quatre-vingts minutes / cent sept en cas de tiers temps

Exercice 1. *(Étude d'un algorithme)*
On appelle racine carrée entière d'un entier naturel n l'unique entier r vérifiant :

$$r^2 \leqslant n < (r+1)^2$$

a. Quelles sont les racines carrées entières des entiers de 5 à 15 ?

b. Voici un joli algorithme :

{ *Pre : n >= 0* }
Fonction Rac(n : *Entier naturel*) : *Entier naturel*
r = 0
{ *P0 :* r^2 *<= n* }
TantQue (r + 1)*(r + 1) <= n **Faire**
    { *P1 : P0 ET x* }
    r = r + 1
FinTantQue
{ *P2 : P0 ET y* }
Retourner z
{ *Post :* $Rac(n)^2$ *<= n* $\wedge$ *n <* $(Rac(n) + 1)^2$ }

i. Déterminez les conditions x et y et l'expression z.

ii. Démontrez la terminaison et la correction de cet algorithme.

iii. Déterminez sa complexité temporelle.

c. Proposez une fonction qui détermine la racine carrée entière avec une complexité logarithmique.

Exercice 2. (*Fang Tcheng*)
Voici pour mémoire une fonction étudiée en cours :

```
def triangle(self) :
        [r,c] = self.D
        m     = min(r,c)
        S     = self
        T     = zeros(r,c)
        while m > 0:
            NoLigne = 0
            while S[NoLigne, 0] == 0 and (NoLigne < m - 1):
                    NoLigne += 1
            S = S.swap(NoLigne,0)
            if S[0, 0] != 0:
                pivot = S[0,0]
                for k in range(1,m):
                    if S[k,0] != 0:
                        S = S.comb_lignes(pivot, -S[k,0],k,0)
            T = T.remplace_ligned(r - m,S.F)
            S = S.decoupe()
            m -= 1
        return T
```

a. Comment modifier ce code pour obtenir une méthode `det(self)` qui calcule le déterminant de la matrice `self` ? Vous indiquerez pourquoi `triangle` ne permet pas de calculer le déterminant directement puis vous préciserez les lignes à changer et proposerez des solutions.

b. Utilisez `triangle` pour déterminer une méthode `rang(self)` qui détermine le rang d'une matrice (3 lignes grand maximum).

Exercice 3. (*Géométrie ?*)
Une matrice est dite **orthogonale** si, et seulement si, sa transposée est égale à son inverse.

a. La matrice I_n est-elle orthogonale (Pour que M. .... ait au moins 1 point) ?

b. La matrice $A = \begin{pmatrix} \cos(t) & \sin(t) \\ -\sin(t) & \cos(t) \end{pmatrix}$ est-elle orthogonale pour tout $t \in \mathbb{R}$?

c. On considère deux réels t et t'. Calculez la partie réelle et la partie imaginaire de $e^{it} \times e^{it'}$ et déduisez-en que :

$$\cos(t+t') = \cos t \cos t' - \sin t \sin t'$$

Déterminez une formule similaire pour $\sin(t+t')$.

d. Déterminez une formule simple donnant A^k pour tout entier naturel k et démontrez votre résultat. Comment interpréter géométriquement le résultat obtenu ?

Exercice 4. (*It's more fun to compute*)
Inversez, si possible, la matrice suivante par la méthode de Gauß :

$$H = \begin{pmatrix} i & 2 & -3 \\ 0 & i & 2 \\ 0 & 0 & i \end{pmatrix}$$

On rappelle que i est le nombre vérifiant $i^2 = -1$

Exercice 5. (*Sagesse*)
Comment interpréter les résultats qui suivent ? Il peut y avoir plusieurs interprétations possibles selon les cas. Votre esprit de synthèse sera évalué.

a.
```
sage: var('a,b,c')
(a, b, c)
sage: M = matrix([[0,1,1,a],[1,0,1,b],[1,1,0,c]])
sage: M.echelon_form()
[ 1    0    0  -1/2*a + 1/2*b + 1/2*c]
[ 0    1    0   1/2*a - 1/2*b + 1/2*c]
[ 0    0    1   1/2*a + 1/2*b - 1/2*c]
```

b.
```
sage: M = matrix(QQ,[[0,1,1,1,0,0],[1,0,1,0,1,0],[1,1,0,0,0,1]])
sage: M.echelon_form()
[   1    0    0 -1/2  1/2  1/2]
[   0    1    0  1/2 -1/2  1/2]
[   0    0    1  1/2  1/2 -1/2]
```

c.
```
  sage: M = matrix([[1,2,3,a],[4,5,6,b],[7,8,9,c]])
sage: M.echelon_form()
[ 1  0 -1  0]
[ 0  1  2  0]
[ 0  0  0  1]
```

d.
```
sage: M = matrix(QQ,[[1,2,3,1,0,0],[4,5,6,0,1,0],[7,8,9,0,0,1]])
sage: M.echelon_form()
[   1    0   -1    0 -8/3  5/3]
[   0    1    2    0  7/3 -4/3]
[   0    0    0    1   -2    1]
```

e.
```
sage: B = matrix(QQ,[[1,1,1,1],[1,-1,-2,3],[2,3,5,0],[4,3,4,4]])
sage: B.echelon_form()
[   1    0    0  5/3]
[   0    1    0    0]
[   0    0    1 -2/3]
[   0    0    0    0]
```

f.
```
sage: B = matrix(QQ,[[1,1,1,1],[1,-1,-2,3],[2,3,5,0],[4,3,4,3]])
sage: B.echelon_form()
[1 0 0 0]
[0 1 0 0]
[0 0 1 0]
[0 0 0 1]
```

g.
```
sage: B = matrix(QQ,[[1,2,3,4,1],[4,5,6,2,2],[7,8,9,-1,4]])
sage: B.echelon_form()
[     1      0     -1      0 -17/3]
[     0      1      2      0  16/3]
[     0      0      0      1    -1]
```

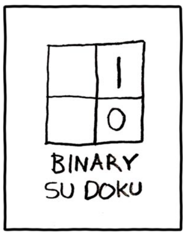

T.S.V.P.

h.

```
sage: B = matrix(CC,[[1j, 2, -3,1,0,0],[0,1j,2,0,1,0],[0,0,1j,0,0,1]])
sage: B.echelon_form()
[ 1  0  0  -1*I      2   -3 + 4*I]
[ 0  1  0     0   -1*I          2]
[ 0  0  1     0      0       -1*I]
```

Exercice 6. *(Loop)*
Quelle est la complexité de cet algorithme :

Exercice 7. *(Bonus)*
Remplir ce Sudoku (pour M. .... non, je ne citerai personne.)



Bon stage et que la force et la programmation fonctionnelle soient avec vous pour la suite de vos études !