Université de Bordeaux — **4TMA801U – Master**
Mathématiques — Année 2019–2020

Corrigé du DS du 11 mars 2020

Barème. Les deux questions demandant d'écrire du code Sage sont notées sur 7 points, le reste de l'exercice 1 sur 5 points, et le reste de l'exercice 2 sur 8 points. Le détail est indiqué à chaque question. L'exercice 1 avait pour l'essentiel déjà été traité en TD.

Exercice 1 – [Algorithme d'Euclide classique]

1) (3.5 pts)

```
def EuclideClassique(A,B):
    #On indique par _im1, _i, _ip1 les quantités au rangs i-1, i, i+1
    #Au début i=1
    [U_im1, V_im1, R_im1] = [1, 0, A]
    [U_i, V_i, R_i] = [0, 1, B]
    i=1

    while (R_i != 0): #ou alors while not (R_i == 0):
        [Q,R] = divmod(R_im1, R_i)
        U_ip1 = U_im1 - Q*U_i
        V_ip1 = V_im1 - Q*V_i
        i = i+1
        #Il faut maintenant décaler les indices
        [U_im1, V_im1, R_im1] = [U_i, V_i, R_i]
        [U_i, V_i, R_i] = [U_ip1, V_ip1, R_ip1]

    #On a maintenant R_i=0, on retourne le résultat du rang i-1
    return([R_im1, U_im1, V_im1])
```

2) (1 pt) L'algorithme d'Euclide classique est quadratique, sa complexité est de $O(n^2)$ opérations algébriques.

3) (3 pts) On procède par récurrence. Le résultat est vrai au rang $i = 0$ car

$$\begin{pmatrix} U_0 & V_0 \\ U_1 & V_1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = I_2.$$

Supposons-le vrai au rang i. Alors, vu l'algorithme,

$$\begin{pmatrix} U_{i+1} & V_{i+1} \\ U_{i+2} & V_{i+2} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & -Q_{i+1} \end{pmatrix} \begin{pmatrix} U_i & V_i \\ U_{i+1} & V_{i+1} \end{pmatrix}.$$

Cela montre la deuxième égalité au rang $i + 1$. Pour la première, il suffit de remarquer que

$$\begin{pmatrix} U_{i+1} & V_{i+1} \\ U_{i+2} & V_{i+2} \end{pmatrix} \begin{pmatrix} A \\ B \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & -Q_{i+1} \end{pmatrix} \begin{pmatrix} U_i & V_i \\ U_{i+1} & V_{i+1} \end{pmatrix} \begin{pmatrix} A \\ B \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & -Q_i \end{pmatrix} \begin{pmatrix} R_i \\ R_{i+1} \end{pmatrix} = \begin{pmatrix} R_{i+1} \\ R_{i+2} \end{pmatrix}.$$

4) (1 pt) Si $1 \leqslant i \leqslant t$, alors R_{i+1} est obtenu comme reste d'une division euclidienne par R_i, donc $\deg(R_{i+1}) > \deg(R_i)$. La suite $(\deg(R_i))_{i \leqslant i \leqslant t+1}$ est donc strictement décroissante.

On a de plus $\deg(B) \leqslant \deg(A)$, donc la suite des $\deg(R_i)$ est en fait décroissante à partir de $i = 0$. Elle commence à $n \geqslant n/2$ et termine à $-\infty < n/2$, l'existence de j en découle. Il est de plus unique, car sinon cela contradirait la décroissance de la suite.

Exercice 2 – [Algorithme d'Euclide rapide]

1) (1 pt) Selon la question 3 de l'exercice 1, on a

$$M(A,B)\begin{pmatrix}A\\B\end{pmatrix} = \begin{pmatrix}U_t & V_t\\ U_{t+1} & V_{t+1}\end{pmatrix}\begin{pmatrix}A\\B\end{pmatrix} = \begin{pmatrix}R_t\\R_{t+1}\end{pmatrix} = \begin{pmatrix}\operatorname{pgcd}(A,B)\\0\end{pmatrix}.$$

2) (3.5 pts)

```
def EuclideRapide(A,B):
    M1 = dpgcd(A,B)
    Rj = M1[0,0] * A + M1[0,1] * B
    Rjp1 = M1[1,0] * A + M1[1,1] * B
    if (Rjp1 == 0):
        return(M1)
    [Q, Rjp2] = divmod(Rj, Rjp1)
    M2 = M1.parent()[0,1,1,-Q]          #selon l'indication
    if (Rjp2 == 0):
        return(M2*M1)
    M3 = EuclideRapide(Rjp1, Rjp2)      #appel récursif
    return (M3*M2*M1)
```

3) (1 pt) Par définition de j à la question 4 de l'exercice 1, on a $\deg(R_{j+1}) < n/2$. Comme R_{j+2} est le reste d'une division euclidienne par R_{j+1}, on a $\deg(R_{j+2}) < \deg(R_{j+1}) < n/2$.

4) (2 pts) Sur des entrées de degré au plus n, l'algorithme ci-dessus effectue au plus un appel à `dpgcd` en degré n, un nombre constant d'additions et multiplications en degré n, une division euclidienne en degré n, et un appel récursif en degré au plus $\lfloor n/2 \rfloor$ par la question précédente. Comme le coût de la division euclidienne est $O(T(n))$, il existe une constante C telle que

$$\forall n \geqslant 1, \quad E(n) \leqslant H(n) + E(\lfloor n/2 \rfloor) + CT(n).$$

5) (3 pts) Comme $T(n) = O(H(n))$, il existe une constante C' telle que

$$\forall n \geqslant 1, \quad E(n) \leqslant C'H(n) + E(\lfloor n/2 \rfloor).$$

Soit $n \geqslant 1$. On montre par récurrence que pour tout entier $k \geqslant 0$,

$$E(n) \leqslant C' \sum_{i=0}^{k} H(\lfloor n/2^i \rfloor) + E(\lfloor n/2^{k+1} \rfloor).$$

En particulier si k est choisi tel que $2^k \leqslant n < 2^{k+1}$, alors

$$E(n) \leqslant C' \sum_{i=0}^{k} H(\lfloor n/2^i \rfloor)$$

puisque l'on peut prendre $E(0) = 0$.

On utilise maintenant l'hypothèse de surlinéarité sur H :

$$E(n) \leqslant C' \sum_{i=0}^{k} \frac{1}{2^i} H(n) \leqslant 2C'H(n).$$

On a donc bien $E(n) = O(H(n))$.

6) (1 pt) On montre la validité de l'algorithme 2 par récurrence sur n. Supposons qu'il renvoie un résultat correct sur toutes les entrées de degré strictement inférieur à n ; on suppose également que l'algorithme `dpgcd` est correct. Supposons enfin, pour simplifier, que R_{j+1} et R_{j+2} sont non nuls. Alors on peut vérifier, ligne à ligne, que les R_i de l'algorithme 2 coïncident avec ceux de l'algorithme 1 et que

$$M_1 = \begin{pmatrix} 0 & 1 \\ 1 & -Q_j \end{pmatrix} \cdots \begin{pmatrix} 0 & 1 \\ 1 & -Q_1 \end{pmatrix},$$

$$M_2 = \begin{pmatrix} 0 & 1 \\ 1 & -Q_{j+1} \end{pmatrix},$$

$$M_3 = \begin{pmatrix} 0 & 1 \\ 1 & -Q_t \end{pmatrix} \cdots \begin{pmatrix} 0 & 1 \\ 1 & -Q_{j+2} \end{pmatrix} \qquad \text{(par hypothèse de récurrence).}$$

Ainsi $M_3 M_2 M_1 = M(A, B)$ par la question 3 de l'exercice 1. Le raisonnement est similaire si R_{j+1} ou R_{j+2} est nul.