Exercice 1 (2.5 points) *Mettez sous la forme* $\Theta(n^\alpha \log^\beta n)$ *les expressions suivantes (sans démonstration)*

1. $\frac{\log_b n}{\log_a n}$
2. $a^{\log_b n}$
3. $\sum_{i=1}^{i=n} \frac{1}{i}$
4. $\log(n!)$
5. $\sum_{i=1}^{n} i^c$

où $a, b, c \geq 2$ *sont des entiers constants.*

Correction. *[* $5 \times$ *0.5 = 2.5 points]*

1. $\frac{\log_b n}{\log_a n} = \Theta(1)$
2. $a^{\log_b n} = \Theta(n^{\log_b a})$
3. $\sum_{i=1}^{i=n} \frac{1}{i} = \Theta(\log n)$
4. $\log(n!) = \Theta(n \log n)$
5. $\sum_{i=1}^{n} i^c = \Theta(n^{c+1})$

Exercice 2 (4 points) *Démontrez le 3 de l'exercice 1 en précisant les constantes utilisées.*

Correction. *[1 point]* Il existe un entier p tel que $2^p \leq n < 2^{p+1}$. Ainsi, puisque

$$\frac{1}{2^p} + \ldots + \frac{1}{8} + \frac{1}{4} + \frac{1}{4} + \frac{1}{2} \leq \sum_{i=1}^{n} \frac{1}{i} \leq \frac{1}{1} + \frac{1}{2} + \frac{1}{2} + \frac{1}{4} + \frac{1}{4} + \frac{1}{4} + \frac{1}{4} + \frac{1}{8} + \ldots + \frac{1}{2^p}$$

on a: *[1 point]*

$$\frac{p}{2} = \sum_{i=1}^{p} 2^{i-1} 2^{-i} \leq \sum_{i=1}^{n} \frac{1}{i} \leq \sum_{i=0}^{p} 2^i 2^{-i} = p + 1$$

[1 point] De plus, $2^{p+1} \leq 2n$ implique $p+1 \leq \log_2(2n) = \log_2 2 + \log_2 n = 1 + \log_2 n$. D'où $p+1 \leq 2 \log_2 n$ pour $\log_2 n \geq 1$ c'est-à-dire, $n \geq 2^1 = 2$. Donc $p + 1 = O(\log n)$.

[1 point] De plus, $\frac{n}{2} < 2^p$ implique $p \geq \log_2 \frac{n}{2} = \log_2 n - 1$. D'où $p \geq \frac{1}{2} \log_2 n$ pour $\frac{1}{2} \log_2 n \geq 1$ c'est-à-dire, $n \geq 2^2 = 4$. Donc $p = \Omega(\log n)$.

Exercice 3 (4 points) *Démontrez que le code suivant renvoie le plus grand diviseur commun des deux entiers quelconques, mais non tous nuls,* a *et* b*, en traitant les cas* $a > b$ *et* $a \leq b$*.*

```
def e(a,b):
        if a*b==0:
                return a+b
        else:
                return e(b,a%b)
```

Correction. *[0.5 point]* Soit $e(a, b)$ la valeur renvoyée par le code. Si $ab = 0$, l'algorithme renvoie $e(a, b) = a + b = (a \text{ ou } b) = pcdg(a, b)$ car $a = 0$ ou $b = 0$ (exclusivement). Supposons $ab > 0$.

1. *[0.5 point]* Si $a > b$, alors $a = qb + r$ et l'algorithme renvoie $e(b, r)$.
2. Si $a \leq b$, alors
   - *[0.5 point]* Si $a = b$, alors $a = 1.b + 0$ et l'algorithme renvoie $e(b, 0) = b = pgcd(a, b)$.
   - *[0.5 point]* Si $a < b$, alors $a = 0.b + a$ et l'algorithme renvoie $e(b, a)$.

Il suffit donc de montrer que $pcgd(a, b) = pgcd(b, r)$ avec $a = bq + r$ et $0 \leq r < b$.

[2 points] Pour cela, il suffit de montrer que l'ensemble des diviseurs communs d de a et b est égal à l'ensemble des diviseurs communs d de b et r. En effet, si $\exists k, \ell$ entiers tels que $a = kd$ et $b = \ell d$, alors $r = a - qb = (k - q\ell)d$. De plus, si $\exists \ell, m$ entiers tels que $b = \ell d$ et $r = md$, alors $a = qb + r = (q\ell + m)d$.

Exercice 4 (5,5 points) *Soient $P_1 = (A_{11} + A_{12})B_{22}$, $P_2 = A_{11}(B_{12} - B_{22})$, $P_3 = (A_{21} + A_{22})B_{11}$, $P_4 = A_{22}(-B_{11} + B_{21})$, et $P_5 = (A_{11} + A_{22})(B_{11} + B_{22})$ les 5 premiers produits récursifs de l'algorithme de Strassen calculant le produit $C = AB$ des deux matrices carrées A, B. On se réfèrera à ces matrices avec les indices $i, j \in \{1, 2\}$, $k \in \{1, 2, 3, 4, 5\}$, et avec $X \in \{A, B\}$.*

1. *Donnez l'expression des quatre sous-matrices C_{ij} en fonction des matrices X_{ij}.*
2. *Donnez les deux expressions de sous-matrices C_{ij} en fonction des matrices P_k représentées par*

$$\begin{pmatrix} \cdot & \cdot & \cdot & \cdot \\ + & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & + & \cdot & \cdot \end{pmatrix} = \begin{pmatrix} \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ + & + & \cdot & \cdot \end{pmatrix} + \begin{pmatrix} \cdot & \cdot & \cdot & \cdot \\ + & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ - & \cdot & \cdot & \cdot \end{pmatrix}$$

$$\begin{pmatrix} \cdot & \cdot & + & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & + \\ \cdot & \cdot & \cdot & \cdot \end{pmatrix} = \begin{pmatrix} \cdot & \cdot & + & + \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \end{pmatrix} + \begin{pmatrix} \cdot & \cdot & \cdot & - \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & + \\ \cdot & \cdot & \cdot & \cdot \end{pmatrix}$$

3. *Donnez les deux expressions de combinaisons des X_{ij} en fonctions des matrices P_k représentées par*

$$\begin{pmatrix} + & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & + \\ \cdot & - & \cdot & + \end{pmatrix} = \begin{pmatrix} + & \cdot & \cdot & + \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ + & \cdot & \cdot & + \end{pmatrix} + \begin{pmatrix} \cdot & \cdot & \cdot & - \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & + \\ \cdot & \cdot & \cdot & \cdot \end{pmatrix} - \begin{pmatrix} \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ + & + & \cdot & \cdot \end{pmatrix}$$

$$\begin{pmatrix} + & \cdot & - & \cdot \\ + & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & + \end{pmatrix} = \begin{pmatrix} + & \cdot & \cdot & + \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ + & \cdot & \cdot & + \end{pmatrix} + \begin{pmatrix} \cdot & \cdot & \cdot & \cdot \\ + & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ - & \cdot & \cdot & \cdot \end{pmatrix} - \begin{pmatrix} \cdot & \cdot & + & + \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \end{pmatrix}$$

4. *Donnez la représentation et l'expression formelle des produits P_6 et P_7 manquants.*
5. [BONUS hors-barème] *Donnez la représentation et l'expression formelle des quatre sous-matrices C_{ij} en fonctions de $P_1, P_2, \ldots, P_7$.*
6. *Donnez le code python d'un algorithme* `mult(A,B)`*, où* `A,B` *sont deux tableaux à deux dimensions, qui calcule le produit de deux matrices carrées de A, B de taille n, en appliquant la définition*

$$C = \left(\sum_{k=1}^{n} a_{ik} b_{kj} \right)_{\substack{1 \leq i \leq n \\ 1 \leq j \leq n}} \quad \text{avec} \quad A = \left(a_{ij}\right)_{\substack{1 \leq i \leq n \\ 1 \leq j \leq n}} \quad \text{et } B = \left(b_{ij}\right)_{\substack{1 \leq i \leq n \\ 1 \leq j \leq n}}$$

7. *Donnez l'expression récursive du temps d'exécution $T(n)$ de l'algorithme de Strassen.*
8. *Expliquez en quoi l'algorithme de Strassen améliore* `mult(A,B)`

Correction.

1. *[0.5 points]* $C_{11} = A_{11}B_{11} + A_{12}B_{21}$, $C_{12} = A_{11}B_{12} + A_{12}B_{22}$, $C_{21} = A_{21}B_{11} + A_{22}B_{21}$, et $C_{22} = A_{21}B_{12} + A_{22}B_{22}$.

2. *[0.5 points]* $C_{12} = P_1 + P_2$ et $C_{21} = P_3 + P_4$

3. *[0.5 points]* $A_{11}B_{11} - A_{12}B_{22} + A_{22}(B_{21} + B_{22}) = P_5 + P_4 - P_1$, et $A_{11}(B_{11} + B_{12}) - A_{21}B_{11} + A_{22}B_{22} = P_5 + P_2 - P_3$

4. *[1 point]* $P_6 = (A_{12} - A_{22})(B_{21} + B_{22})$

$$\begin{pmatrix} \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & + & \cdot & - \\ \cdot & + & \cdot & - \end{pmatrix}$$

et $P_7 = (-A_{11} + A_{21})(B_{11} + B_{21})$

$$\begin{pmatrix} - & \cdot & + & \cdot \\ - & \cdot & + & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \end{pmatrix}$$

5. *[1 point]* On obtient $C_{11} = P_6 + P_5 + P_4 - P_1$

$$\begin{pmatrix} + & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & + & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \end{pmatrix} = \begin{pmatrix} + & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & + \\ \cdot & - & \cdot & + \end{pmatrix} + \begin{pmatrix} \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & + & \cdot & - \\ \cdot & + & \cdot & - \end{pmatrix}$$

et $C_{22} = P_7 + P_5 + P_2 - P_3$,

$$\begin{pmatrix} \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & + & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & + \end{pmatrix} = \begin{pmatrix} + & \cdot & - & \cdot \\ + & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & + \end{pmatrix} + \begin{pmatrix} - & \cdot & + & \cdot \\ - & \cdot & + & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \end{pmatrix}$$

et on avait déjà $C_{12} = P_1 + P_2$, $C_{21} = P_3 + P_4$.

6. *[1 point]*

```
def mult(A,B):
        n = len(A)
        C = [[0 for i in range(n)] for j in range(n)]
        for i in range(n):
                for j in range(n):
                        for k in range(n):
                                C[i][j] += A[i][k] * B[k][j]
        return C
```

7. *[0.5 point]* On peut choisir une constante $n_0 \geq 1$ telle que

$$T(n) = \begin{cases} \Theta(1) & \text{si } n \leq n_0 \\ 7T(n/2) + \Theta(n^2) & \text{si } n > n_0 \end{cases}$$

8. *[0.5 point]* Il s'ensuit que $T(n) = \Theta(n^{\log_2 7}) < \Theta(n^3)$ qui est la complexité de `mult(A,B)`.

Exercice 5 (6 points) *Soient un entier* `n` *et un tableau* `X` *de* n *entiers en variables globales.*

1. *Donnez le code d'une fonction récursive* `count(i)` *telle que, pour tout* $i = 1, \ldots, n$*, l'appel à* `count(n-i)` *affiche tous les différents tableaux* `X` *avec*

   (a) `X[:n-i]` *inchangé, et*

   (b) `X[n-i:]` *qui sont les expressions en base* 7 *des entiers de* $0, \ldots, 7^i - 1$.

2. *Donnez le code d'une fonction récursive* `permut(i)` *telle que l'appel à* `permut(0)` *affiche les* $n!$ *permutations de* $\{1, 2, \ldots, n\}$ *sous le format de tableaux (par exemple 123, 132, 213, 231, 312, 321 pour* $n = 3$*).*

3. *Donnez le code python d'un algorithme qui résout le problème du voyageur de commerce, à savoir, étant donné* n *villes et les* $\binom{n}{2}$ *distances entre deux villes, calculer le trajet le plus court passant par toutes les villes (sans revenir au point de départ).*

Correction.

1. *[1,5 points]*

```
def count(i):
        for j in range(7):
                X[i]=j
                if i==n-1:
                        print X
                else:
                        count(i+1)
```

2. *[1,5 points]*

```
U=[0 for i in range(n)]
def permut(i):
        for j in range(1,n+1):
                if U[j-1]==0 :
                        X[i]=j
                        U[j-1]=1
                        if i==n-1:
                                print X
                        else:
                                permut(i+1)
                        U[j-1]=0
```

3. *[1,5 points]* Il faut d'abord créer une fonction `distance(X)` calculant la distance parcourue en fonction de l'ordre `X` considéré, et une variable global `min` initialisée à une borne supérieure, par exemple la somme des $\binom{n}{2}$ distances. Ensuite, il faut créer `voyage(i)` une modification de `permut(i)` qui met à jour `min` en appelant `distance(X)` si le test `if i==n-1` est vrai.

   [1 point] Soit `D[i][j]` la distance entre les villes i et j.

```
def distance(X):
        d=0
        for i in range(n-1):
                d += D[X[i]][X[i+1]]
        return d
```

   [0,5 point]

```
def voyage(i):
        for j in range(n):
                if U[j]==0 :
                        X[i]=j
                        U[j]=1
                        if i==n-1:
                                if distance(X)<min:
                                        min=distance(X)
                        else:
                                voyage(i+1)
                        U[j]=0
```