Complexité - Solutions

Entraînement 1

```python
def division_euclidienne(a, b):
    q = 0
    r = a
    while r >= b :
        r = r - b
        q = q + 1
    return (q, r)
```

On note (r_n) la suite des valeurs que prend la variable r, on a donc $r_0 = a$.

À chaque itération de la boucle, la valeur de r diminue de b donc (r_n) est arithmétique de raison $-b$ ainsi $r_n = a - nb$.

La sortie de la boucle a lieu lorsque $r < b$ c'est à dire pour le plus petit n tel que : $a - nb < b \Leftrightarrow \dfrac{a}{b} - 1 < n$. On en déduit que n vérifie $n \leqslant \dfrac{a}{b} < n + 1$ ainsi $n = \left\lfloor \dfrac{a}{b} \right\rfloor$. Il vient :

$$C = \mathrm{O}\left(\frac{a}{b}\right)$$

Attention, il arrive que la complexité dépendent de plusieurs paramètres.

Entraînement 2

```python
def est_parfait2(n):
    ''' Déterminer si n >= 2 est parfait '''
    S = 1
    k = 2
    while k**2 < n:
        if n%k == 0:
            S += k + n//k
        k += 1
    if k**2 == n:
        S += k
    return S == n
```

On note (k_i) la suite des valeurs que prend la variable k, on a donc $k_0 = 2$.

À chaque itération de la boucle, la valeur de k est incrémentée d'où $k_i = 2 + i$.

La sortie de la boucle a lieu lorsque $k^2 > n$ c'est à dire pour le plus petit i tel que : $i \geqslant \sqrt{n} - 2$. On en déduit que :

$$C = \mathrm{O}\left(\sqrt{n}\right)$$

Ici, il ne fallait compter le calcul du reste de de la division euclidienne que pour un temps constant.

Dans les faits, la division euclidienne se fait par un algorithme qui est en $O(ln(a/b))$ où a est le dividende et b le diviseur.

Entraînement 3

```python
def est_croissante(L):
  ''' Teste si la liste non vide L est croissante au sens large '''
    i = 1
    while i < len(L) and L[i - 1] <= L[i]:
        i += 1
    return i == len(L)
```

Dans le meilleur des cas $L[1] < L[0]$ est la boucle n'est exécutée qu'une fois. On a dans ce cas $C = \mathrm{O}(1)$.

Dans le pire des cas la liste est triée dans l'ordre croissant et la boucle est alors exécutée n fois. On a dans ce cas $C = \mathrm{O}(n)$.

Entraînement 4

Attention à vos récurrences, qui doivent être rédigées sans bluffer l'hérédité.

Le code ci-dessous doit être étudier en **détails** par les étudiants qui n'ont pas réussis la question.

```python
def prod_mat(A, B):
    ''' Calcule le produit des matrices A et B carres de taille 2'''
    C = [[0, 0], [0, 0]]
    for i in range(2):
        for j in range(2):
            C[i][j] = sum(A[i][k]*B[k][j] for k in range(2))
    return C


def fibo_exp_rap(n):

    P = [[1, 0], [0, 1]]
    Q = [[0, 1], [1, 1]]

    while n != 0:
        if n%2 == 1:
            P = prod_mat(P, Q)
        Q = prod_mat(Q, Q)
        n = n // 2

    return P[1][0]
```