TD2 : décompte d'opérations et notations asymptotiques

Exercice 1 : Dénombrement avec les boucles

Pour chacune des fonctions, compter le nombre exact d'itérations de la boucle la plus interne réalisées en fonction de n. Lorsqu'il est trop difficile de calculer le nombre exact, proposer une fonction de n qui majore ce nombre. Donner la notation Θ de ce nombre d'itérations.

```python
def f1 (n):
  r = 0
  for i in range(1,n+1):
    for j in range(1,n+1):
      r = r + 3
  return r
```

```python
def f2 (n):
  r = 0
  for i in range(1,n+1):
    for j in range(1,i+1):
      r = r + 3
  return r
```

```python
def f3 (n):
  r = 0
  for i in range(5,n-5+1):
    for j in range(i-5,i+5+1):
      r = r + 3
  return r
```

```python
def f4 (n):
  r = 0
  for i in range(1,n+1):
    for j in range (1,n+1):
      for k in range (1,n+1):
        r = r + 3
  return r
```

```python
def f5 (n):
  r = 0
  for i in range(1,n+1):
    for j in range (1,i+1):
      for k in range(1,j+1):
        r = r + 3
  return r
```

```python
def f6 (n):
  r = 0
  i = n
  while i > 1:
    for j in range(1,i+1):
      r = r + 3
    i = i // 2
  return r
```

```python
def f7 (chaine1, chaine2):
  m = len(chaine1)
  n = len(chaine2)
  i=0
  for c in chaine1:
    if c in chaine2:
      i+=1
  return i
```

Exercice 2 :

Nous disposons de plusieurs algorithmes réalisant la même tache mais avec un nombre d'additions différent (colonne de gauche des tableaux).

Q 2.1 Indiquer le nombre d'opérations nécessaires à l'exécution de chaque algorithme pour les différentes tailles de données.

Complexité	$n = 10$	$n = 1\,000$	$n = 1\,000\,000$
n			
n^2			
n^3			
$\sqrt{n}$			
2^n			
$\log_2(n)$			
$n \log_2(n)$			

Q 2.2 Indiquer le temps nécessaire à l'exécution de chaque algorithme pour les différentes tailles de données en supposant que l'ordinateur effectue 10^9 opérations à la seconde.

Complexité	$n = 10$	$n = 1\,000$	$n = 1\,000\,000$
n			
n^2			
n^3			
$\sqrt{n}$			
2^n			
$\log_2(n)$			
$n \log_2(n)$			

Exercice 3 :

```python
def algo(data):
  n = len(data)
  i = 0
  swapped = True
  while i < n-1 and swapped:
    swapped = False
    for j in range(n-i-1):
      if data[j] > data[j+1]:
        data[j], data[j+1] = data[j+1], data[j]
        swapped = True
    i += 1
```

Q 3.1 Déroulez le fonctionnement de l'algorithme pour `data = [7, 3, 5, 2]`, en indiquant pour chaque changement de `data`, les valeurs de `i` et `j`, ainsi que sa nouvelle valeur.

Q 3.2 De manière générale, exprimez ce que serait un pire des cas et un meilleur des cas pour cet algorithme.

Q 3.3 Dans le pire des cas, quelle est la complexité de l'algorithme ?

Q 3.4 Dans le meilleur des cas, quelle est la complexité de l'algorithme ?

Exercice 4 : Complexités asymptotiques sur les piles et files

Reprendre les réponses aux questions des exercices 4 et 5 de la première fiche de TD. Donner la complexité asymptotique de ces opérations.

Exercice 5 :

```
1 """
2   CU: there exists p such that n = 2^p
3 """
4 def exo4 (n):
5   r = 0
6   while n != 0:
7     n = n // 2
8     r = r + 1
9   return r
```

Q 5.1 Calculer le nombre d'itérations effectuées en fonction de n.

Exercice 6 : Table redimensionnable

On considère une table avec redimensionnement dynamique : dès que la table a été intégralement remplie, on crée une nouvelle table de taille double et on y recopie toutes les éléments de la table précédente.

Imaginons qu'on réalise au total n insertions dans une table dont la taille était initialement 2^0.

C'est, par exemple, le principe derrière les `ArrayList` en Java

Q 6.1 Montrer que le coût en temps est de l'ordre de n malgré les redimensionnements et recopies nécessaires.

Q 6.2 En déduire qu'en moyenne la complexité de l'insertion est en $\mathcal{O}(1)$ pour les tables avec redimensionnement dynamique.

Exercice 7 : Application des définitions des bornes asymptotiques

Q 7.1 Montrer que :

1. $n^2 + 2n + 3 = \mathcal{O}(n^2)$,
2. $n^2 - 5n + 3 = \Omega(n)$,
3. $n^4 + n^2 = \Theta(n^4)$,
4. n^3 n'appartient pas à $\Theta(n^2)$.

Q 7.2 Donner et montrer un équivalent de la forme n^k des fonctions :

1. $f_1(n) = n^3 - n^2 + 2^{21}$
2. $f_2(n) = \sum_{i=1}^{n} i^2$

Exercice 8 : Vrai ou faux

On note respectivement c_m, c_p et c la complexité dans le meilleur des cas, le pire des cas et la complexité en général.

1. si $c_m(n) = \Omega(f(n))$ alors $c(n) = \mathcal{O}(f(n))$
2. si $c_m(n) = \Omega(f(n))$ alors $c(n) = \Omega(f(n))$
3. si $c_m(n) = \mathcal{O}(f(n))$ alors $c(n) = \Omega(f(n))$
4. si $c_p(n) = \mathcal{O}(f(n))$ alors $c(n) = \mathcal{O}(f(n))$
5. si $c_p(n) = \Omega(f(n))$ alors $c(n) = \mathcal{O}(f(n))$
6. si $c_p(n) = \Omega(f(n))$ alors $c(n) = \Omega(f(n))$