Entraînement : calcul de complexité

Pour tous les exercices, la grille d'évaluation est la suivante.

A (20)	Toutes les complexité ont été données de façon correcte. La notation O a été bien utilisée.
B (16)	Les complexités sont toutes correctes mais quelques imprécisions sur la notation O.
C (11)	Entre 1 et 3 erreurs sur les complexités, mais la classe de complexité (logarithmique, linéaire /polynomiale ou exponentielle) est respectée.
D (8)	Entre 1 et 3 erreurs dont des erreurs de classe de complexité.
E (1)	4 erreurs ou plus

Exercice 1.
Donner la complexité de chacun des algorithmes suivants (sans justification). Chaque algorithme prend en entrée un entier n. La réponse sera donnée en utilisant la notation O.

```
Algo_a :
    c <- 0
    Pour i allant de 1 à n:
        Pour j allant de 1 à i:
            c <- c + 1

Algo_b :
    c <- 0
    Pour i allant de 1 à n:
        Algo_a(i)

Algo_c :
    c <- 0
    Tant que c < n:
        c <- c + 1000
.
```

```
Algo_d :
    c <- 0
    Tant que c*c < n:
        c <- c+1

Algo_e :
    c <- 1
    Tant que c < n :
        c <- c*2

Algo_f :
    f <- 1
    c <- 0
    Tant que c < n:
        i <- 0
        Tant que i < f:
            i <- i+1
        f <- f+i
        c <- c+1
```

Exemple de réponses et note finale associée.

Remarque : parfois certaines cases sont laissées vide si la réponse n'a pas été donnée (il vaut mieux avouer qu'on ne sait pas pas plutôt que d'écrire quelque chose de complètement faux).

La première ligne donne les réponses correctes attendues

a	b	c	d	e	f	note
$O(n^2)$	$O(n^3)$	$O(n)$	$O(\sqrt{n})$	$O(\log(n))$	$O(2^n)$	A
$O(\frac{n(n-1)}{2})$	$O(n^3)$	$O(\frac{n}{1000})$	$O(\sqrt{n})$	$O(\log(n))$	$O(2^n)$	B
$O(\frac{n(n-1)}{2})$	$O(n^2)$	$O(\frac{n}{1000})$	$\sqrt{n}$	$\log(\frac{n}{2})$		C
$O(n^2)$	$O(n^3)$	$O(n)$	$O(\frac{n}{2})$			C
$O(n^2)$	$O(n^3)$	$O(n)$	$O(\sqrt{n})$	$O(\frac{n}{2})$		D
$O(n^2)$	$O(n)$		$O(\log(n))$	$O(\log(n))$	$O(\log(n))$	E

Exercice 2.
Donner la complexité de chacun des algorithmes suivants (sans justification). Chaque algorithme prend en entrée un entier n.

```
Algo_a :
    c <- 0
    Pour i allant de 1 à n:
        Pour j allant de 1 à i:
            Pour k allant de 1 à j:
                c <- c + 1

Algo_b :
    c <- 0
    Pour i allant de 1 à n:
        c <- c + 1

    Pour i allant de 1 à n:
       c <- c + 1

.
```

```
Algo_c :
    c <- 1
    d <- 0
    Pour i allant de 1 à n:
        c <- 2*c
        Pour j allant de 1 à c:
            d <- d+1

Algo_d :
    c <- n
    Tant que c > 0:
        c <- c / 2

Algo_e :
    c <- 1
    Tant que c < n:
        c <- c + 2
```

```
Algo_f :
    c <- 0
    Tant que c < n:
        Algo_b(c)
        c <- c + 1
```

Exemple de réponses et note finale associée.

Remarque : parfois certaines cases sont laissées vide si la réponse n'a pas été donnée (il vaut mieux avouer qu'on ne sait pas pas plutôt que d'écrire quelque chose de complètement faux).

La première ligne donne les réponses correctes attendues

a	b	c	d	e	f	note
$O(n^3)$	$O(n)$	$O(2^n)$	$O(\log(n))$	$O(n)$	$O(n^2)$	A
$O(n^3)$	$O(2n)$	$O(2^n)$	$O(\log(n))$	$O(\frac{n}{2})$	$O(n^2)$	B
$O(n^3)$	$O(n^2)$	$O(2^n)$	$O(\log(n))$	$O(2n)$	$O(n^2)$	C
$O(n^3)$	$O(n)$			$O(n)$	$O(n^2)$	C
$O(n^3)$	$O(n)$	$O(n^2)$	$O(\frac{n}{2})$	$O(n)$	$O(n^2)$	D
$O(n^3)$	$O(n^2)$			$O(n)$	$O(n)$	E

Exercice 3 (Partiel 2017-18)**.**
Donner la complexité de chacun des algorithmes suivants (sans justification). Chaque algorithme prend en entrée un entier n.

```
Algo_a :
    c <- 0
    Pour i allant de 1 à n:
        Pour j allant de 1 à i:
            c <- c + 1

        Pour j allant de i à n:
            c <- c + 1

Algo_b :
    c <- 1
    Tant que c < n:
       c <- c*2
.
```

```
Algo_c :
    c <- 1
    Tant que c < n:
        c <- c + 2

Algo_d :
    c <- n
    Tant que c > 0:
        c <- c / 2

Algo_e :
    Pour i allant de 1 à n:
        Si i*i > n:
            Retourner i
```

```
Algo_f :
    c <- 0
    Pour i allant de 1 à n:
        Pour j allant de 1 à n:
            Pour k allant de 1 à n:
                Pour l allant de 1 à n:
                    c <- c + 1
```

Exemple de réponses et note finale associée.
La première ligne donne les réponses correctes attendues

a	b	c	d	e	f	note
$O(n^2)$	$O(\log(n))$	$O(n)$	$O(\log(n))$	$O(\sqrt{n})$	$O(n^4)$	A
$O(n^2)$	$O(\log(n))$	$O(\frac{n}{2})$	$O(\log(n))$	$O(\sqrt{n})$	$O(n^4)$	B
$O(n^2)$	$O(\log(n))$	$O(n)$	$O(\log(n))$	$O(n)$	$O(n^4)$	C
$O(n^3)$	$O(\log(n))$	$O(n)$	$O(\log(n))$	$O(\sqrt{n})$	$O(n^4)$	C
$O(n^3)$	$O(\log(n))$	$O(n)$	$O(n)$	$O(\sqrt{n})$	$O(n^4)$	D
$O(n^2)$	$O(\frac{n}{2})$	$O(n)$	$O(\log(n))$	$O(\sqrt{n})$	$O(n^4)$	D
$O(n^2)$	$O(\log(n))$	$O(n)$	$O(\log(n))$	$O(n)$	$O(4^n)$	D

Question donnée au partiel 1 2017-2018, résultats obtenus :

A	B	C	D	E
85%	0%	0%	15%	0%

Exercice 4 (Partiel 2018-2019)**.**
Donner la complexité de chacun des algorithmes suivants (sans justification). Chaque algorithme prend en entrée un entier positif n.

```
Algo_a :
    c <- 0
    Pour i allant de 1 à n:
        c <- c+2

Algo_b :
    c <- 1
    Tant que c < n:
       c <- c*2

Algo_c :
    c <- 0
    Pour i allant de 1 à n:
        c <- c * c
        Pour j allant de 1 à n:
                c <- c * 2
```

```
Algo_d :
    c <- 1
    Pour i allant de 1 à n:
        c <- c + 1
        Pour j allant de 1 à i:
            c <- c + 1

Algo_e :
    Tant que n > 0:
        n <- n//2

Algo_f :
    c <- 0
    Pour i allant de 1 à n:
        Pour j allant de 1 à i:
            Pour k allant de 1 à j:
                c <- c + 1
```

Exemple de réponses et note finale associée.

La première ligne donne les réponses correctes attendues

a	b	c	d	e	f	note
$O(n)$	$O(\log(n))$	$O(n^2)$	$O(n^2)$	$O(\log(n))$	$O(n^3)$	A
$O(\frac{n}{2})$	$O(\log(n))$	$O(n^2)$	$O(n^2)$	$O(\log(n))$	$O(n^3)$	B
$O(n)$	$O(\log(n))$	$O(n^2)$	$O(n^3)$	$O(\log(n))$	$O(n^3)$	C
$O(n)$	$O(\log(n))$	$O(n^2)$	$O(n^2)$	$O(\log(n))$	$O(n^2)$	C
$O(n)$	$O(n)$	$O(n^2)$	$O(n^2)$	$O(\log(n))$	$O(n^3)$	D

Question donnée au partiel 1 2018-2019, résultats obtenus :

A	B	C	D	E
60%	0%	18%	22%	0%