

TD Invariants n°1

Calcul de complexité

I. Avec des boucles

Dans cet exercice, on demande de compter le nombre d'appels à la fonction f dans les algorithmes ci-dessous. On l'exprimera sous forme *exacte*, puis on donnera un grand O plus simple.

```

f()
i ← 0
Tant que  $i \leq n$  faire
  | f()
  | i ← i + 1
Fin tant que

```

▷ Résolu par Raminia

Fig. 1. Premier algorithme

```

i ← 0
Tant que  $i^2 \leq n$  faire
  | f()
  | f()
  | Pour  $j \in \llbracket 1, n \rrbracket$  faire
  | | f()
  | Fin pour
  | i ← i + 1
Fin tant que

```

▷ Résolu par Simon P.

Fig. 2. Second algorithme

```

i ← 0
Tant que  $2^i < n$  faire
  | j ← i
  | Tant que  $j < n$  faire
  | | f() j ← j + 1
  | Fin tant que
  | i ← i + 1
Fin tant que

```

▷ Résolu par Raminia

Fig. 3. Troisième algorithme

```

i ← n
Tant que i > 0 faire
  j ← 0
  f()
  Tant que j < n faire
    f()
    k ← j
    Tant que k < n faire
      f()
      k ← k + 1
    Fin tant que
    j ← j + 1
  Fin tant que
  i ← ⌊ $\frac{i}{2}$ ⌋
Fin tant que

```

▷ Résolu par Ewen

Fig. 4. Quatrième algorithme

II. Avec de la récursivité

Dans cet exercice, on demande de compter le nombre d'appels à la fonction f dans les algorithmes ci-dessous, pour *chacune* des fonctions (g , h , et k). On l'exprimera uniquement sous forme d'un grand O simple.

Remarques. Ces algorithmes ne terminent pas forcément mais, passé un certain instant, les appels à la fonction f s'arrêtent. Attention, la variable n n'est pas forcément un entier.

```

Procédure g(n)
  i ← 1
  Tant que i < n faire
    f()
    i ← i + 2
  Fin tant que
  g(n/2)
  g(n/2)
  g(n/2)

```

▷ Résolu par Ewen & Carla

Fig. 5. Premier algorithme

▷ Résolu par Simon P. & Raminia

Procédure $h(n)$ $i \leftarrow 1$ **Tant que** $i < n$ **faire** $f()$ $i \leftarrow i + 1$ **Fin tant que** $k(n)$ $k(n)$ **Procédure** $k(n)$ $i \leftarrow 2$ **Tant que** $i < n$ **faire** $f()$ $i \leftarrow i^2$ **Fin tant que** $h(n/2)$ **Fig. 6.**

Second algorithme