TD Invariants

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.

▷ Résolu par Raminia

```
f( )
i ← 0
Tant que i ≤ n faire
    f( )
    i ← i + 1
Fin tant que
```

Figure 1. Premier algorithme

▷ Résolu par Simon P.

```
i ← 0
Tant que i² ≤ n faire
    f( )
    f( )
    Pour j ∈ ⟦1, n⟧ faire
        f( )
    Fin pour
    i ← i + 1
Fin tant que
```

Figure 2. Second algorithme

▷ Résolu par Raminia

```
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
```

Figure 3. Troisième algorithme

▷ Résolu par Ewen

```
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 ← ⌊i/2⌋
Fin tant que
```

Figure 4. Quatrième algorithme

II. Avec de la récusivité

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.

▷ Résolu par Ewen & Carla

```
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)
```

Figure 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
  $h(n/2)$

Figure 6. Second algorithme