Aix*Marseille université

Domaine Sciences et Technologies
Portail 1 René Descartes

Introduction à l'informatique : TD 3
Code UE : S14IN1I1

Année 2022–2023

Algorithmes – correction et complexité

Exercice 1 (Correction d'un algorithme)

1. Chacun des algorithmes suivants admet une erreur. Déterminez-la. Le paramètre d'entrée `n` est un entier positif.

```
Fonction algo1(d n: entier): entier
    b: entier;
Début
    b := 1;
    Tant que (n != 0) faire
        b := 2 * b;
        n := n - 2;
    Finfaire
    Retour b;
Fin
```

```
Fonction algo2(d n: entier): entier
    a, b: entier;
Début
    b := 1;
    a := 1;
    Tant que (a < n) faire
        b := 2 + a;
    Finfaire
    Retour b;
Fin
```

2. Écrivez une fonction qui prend en paramètres deux entiers `x` et `y` et qui retourne le maximum de `x` et `y`. Notons que l'entier m est le maximum de x et y s'il s'agit de l'unique entier tel que $m \geq x$ et $m \geq y$ et ($m = x$ ou $m = y$).
   Montrez que votre algorithme termine et qu'il est correct.
3. On veut écrire une fonction qui calcule la factorielle d'un entier positif `n` passé en entrée. L'algorithme suivant admet une erreur.

```
Fonction factorielle(d n: entier): entier
    i, fact: entier;
Début
    fact := 1;
    i := 0;
    Tant que (i < n) faire
        fact := fact * i;
        i := i + 1;
    Finfaire
    Retour fact;
Fin
```

   (a) Déterminez l'erreur et corrigez-la.

   (b) Déroulez l'exécution de `factorielle(4)` en la présentant sous la forme d'un tableau permettant de suivre l'évolution des variables instruction par instruction. En particulier, vous indiquerez les valeurs des variables `i`, et `fact` à la fin de chaque itération de la boucle. L'itération 0 indiquera les valeurs de ces variables avant la première itération.
   Quelle est la valeur de `fact` à la fin de l'exécution ?

   (c) Montrez que votre algorithme est correct.

Exercice 2 (Complexité d'un algorithme)

1. Déterminez le nombre d'affectations effectuées par chacun des algorithmes suivants pour $i = 3$. Généralisez ce résultat pour tout $i \in \mathbb{N}$.

```
Fonction algo1(d i: entier): entier
    a, b, n: entier;
Début
    n := 2^i;
    a := 1;
    b := 0;
    Tant que (a < n) faire
        b := 2 * b;
        a := a + 1;
    Finfaire
    Retour b;
Fin
```

```
Fonction algo2(d i: entier): entier
    a, b, n: entier;
Début
    n := 2^i;
    a := 1;
    b := 0;
    Tant que (a < n) faire
        b := 2 * b;
        a := a * 2;
    Finfaire
    Retour b;
Fin
```

```
Fonction algo3(d i: entier): entier
    a, b, n: entier;
Début
    n := 2^i;
    a := 1;
    b := 0;
    Tant que (n != 1) faire
        b := n * b;
        n := n / 2;
    Finfaire
    Retour b;
Fin
```

```
Fonction algo4(d i: entier): entier
    a, b, j, k: entier;
Début
    a := 1;
    b := 0;
    Pour j de 0 à i-1 faire
        Pour k de 0 à i-1 faire
            b := 2 * b;
        Finfaire
    Finfaire
    Retour b;
Fin
```

2. Quel algorithme est le plus lent ? Quel le plus rapide ?

3. On considère maintenant l'algorithme suivant, qui prend en entrée un entier n strictement positif et renvoie soit *vrai*, soit *faux* :

```
Fonction mystère(d n: entier): booléen
    a, i, j: entier;
Début
    i := 1;
    a := 1;
    Tant que (i < n) faire
        a := a + 1;
        i := 2 * i;
    Finfaire
    Si (a mod 2 = 0) alors
        Pour j de 0 à n-1 faire
            a := a + 5;
        Finfaire
    Finsi
    Retour a;
Fin
```

   (a) Donnez deux valeurs de n telles que la condition `a mod 2 = 0` est évaluée à `vrai` et à `faux` respectivement.

   (b) Expliquez pourquoi cet algorithme termine sur toute entrée.

   (c) Évaluez le nombre d'affectations de l'algorithme en fonction de n *dans le pire des cas*, puis simplifiez l'expression avec la notation « grand O ».

   (d) Évaluez le nombre d'affectations de l'algorithme en fonction de n *dans le meilleur des cas*, puis simplifiez l'expression avec la notation « grand O ».