INFO2009 - Sessions d' supplémentaires

Chapitre 4

2025

1. (Examen de juin 2019)
   (a) Écrire une fonction C prenant en arguments deux entiers strictements positifs `n` et `f`, et calculant la multiplicité du facteur `f` dans `n`, c'est-à-dire le plus grand nombre entier `m` tel que $\texttt{f}^m$ divise `n`. Par exemple, si $n = 324$ et $f = 9$, on a $m = 2$ car $324 = 9^2.4$.
   (b) Par la méthode des invariants, démontrer que la valeur retournée par cette fonction est correcte.

2. (Examen de janvier 2021)
   (a) Écrire en langage C une fonction prenant en argument un nombre entier strictement positif `n`, et retournant le nombre de diviseurs de `n`. (Par exemple, cette fonction doit retourner 1 pour $n = 1$, 2 pour $n = 2$, et 8 pour $n = 24$.) On demande que l'implémentation de cette fonction soit raisonnablement efficace.
   (b) Déterminer les complexités en temps et en espace de la fonction obtenue au point (a).
   (c) Par la méthode des invariants, démontrer que la valeur retournée par la fonction obtenue au point (a) est correcte. Démontrer également que cette fonction se termine.

3. (Examen de janvier 2022)
   (a) Écrire en C une fonction prenant en argument un entier $n > 1$, et retournant le plus grand diviseur `d` de `n` tel que $d < n$. Par exemple, cette fonction appliquée à $n = 21$ doit retourner 7. On souhaite que l'implémentation de cette fonction soit raisonnablement efficace.
   (b) Déterminer les complexités en temps et en espace de la fonction obtenue au point (a).
4. (Examen de juin 2022)
   (a) Écrire en C une fonction prenant en argument un nombre entier non signé, et retournant la somme de tous les chiffres de son écriture décimale. Par exemple, pour le nombre 2022, cette fonction doit retourner 6.
   (b) Par la méthode des invariants, démontrer que la valeur retournée par cette fonction est correcte.

5. (Examen d'août 2022)
   (a) Écrire une fonction prenant en argument un nombre entier non signé et non nul, et retournant le nombre de zéros situés ła fin de son écriture décimale. Par exemple, pour le nombre 42000, cette fonction doit retourner 3. Pour le nombre 2022, elle doit retourner 0.
   (b) Par la méthode des invariants, démontrer que la valeur retournée par cette fonction est correcte.

6. (Examen de janvier 2023)
   L'*indicatrice d'Euler* est une fonction ϕ telle que pour tout nombre entier strictement positif n, $\phi(n)$ a pour valeur le nombre d'entiers m contenus dans l'intervalle $[1, n]$ tels que `pgcd(m,n)=1`, où `pgcd(m,n)` est le plus grand commun diviseur de m et de n.
   (a) Donner une implémentation de cette fonction, en supposant que l'on dispose d'une bibliothèque qui implémente déjà la fonction `pgcd`.
   (b) Calculer la complexité en temps de la fonction obtenue au point (a), en sachant que la complexité en temps de `pgcd(m,n)`, avec $m \leq n$, est $\mathcal{O}(\texttt{log}m)$.

7. (Examen d'août 2023)
   (a) Écrire une fonction calculant le plus petit diviseur strictement supérieur à 1 d'un nombre $n > 1$ donné. Par exemple, pour $n > 1$ donné. Par exemple, pour $n = 51$ et $n = 11$, la fonction doit respectivement retourner 3 et 11. On demande que l'implémentation de cette fonction soit raisonnablement efficace, c'est-à-dire que sa complexité en temps soit meilleure que $\mathcal{O}(n)$.
   (b) Calculer la complexité en temps de la fonction obtenue au point (a).

8. (Examen de janvier 2024)
   (a) Écrire une fonction `nfp` calculant le nombre de diviseurs premiers d'un nombre $n > 0$ donné. (Rappel : Un nombre $d > 0$ est dit premier s'il possède exactement deux diviseurs.)
   Par exemple, on doit avoir $\texttt{nfp}(150) = 3, \texttt{nfp}(7) = 1$ et $\texttt{nfp}(1) = 0$.
   Pour résoudre ce problème, on suppose que l'on dispose d'une fonction `premier(m)` déjà programmée qui retourne une valeur booléenne indiquant si son argument $m > 0$ est premier.
   La complexité en temps de la fonciton `premier(m)` est $\mathcal{O}(1)$.
   On souhaite que l'implémentation de la fonction `nfp(n)` soit raisonnablement efficace, c'est-à-dire que sa complexité en temps soit meilleure que $\mathcal{O}(n)$.
   (b) Calculer la complexité en temps de la fonction obtenue au point (a).

9. (Examen de juin 2019)
   (a) Décrire, le plus simplement possible, l'opération effectuée par la fonction C suivante.

```
unsigned f(unsigned v)
{
    int n;
    if (!v)
        return 0;

    n = v % 10 == 9 ? 1 : 0;

    return f(v / 10) + n;
}
```

   (b) Quelle est la complexité en temps de cette fonction ?
   (c) Écrire une fonction C réalisant exactement la même opération, mais sans effectuer d'appel récursif.

10. (Examen d'août 2022)
    (a) Calculer les complexités en temps et en espace de la fonction suivante, en expliquant votre raisonnement.

```
unsigned f(unsigned n)
    {
        if (n <= 1)
            return n;

        return f(n/2) + f(n % 2);
    }
```

    (b) Écrire une fonction qui calcule exactement la même opération, mais sans effectuer d'appel récursif.

11. (Examen d'août 2023)

(a) Expliquer, le plus simplement possible, ce que calcule la fonction suivante :

```
double f(double x, unsigned n)
{
    double a;

    if (!n)
        return x;

    a = f(x, n - 1);

    return a * a;
}
```

(b) Écrire une fonction réalisant exactement la même opération, mais sans effectuer d'appel récursif. *Note :* Il n'est pas permis d'utiliser des fonctions issues de la bibliothèque standard.

12. (Examen de juin 2024)
    (a) Décrire le plus simplement possible ce que calcule la fonction C suivante :

```
unsigned f(unsigned long n)
    {
        if (n == 0 || n % 2 != 0)
            return 0;

        return 1 + f(n / 2);
    }
```

    (b) Quelle est la complexité en espace de cette fonction ?
    (c) Écrire une fonction C réalisant exactement la même opération que la fonction du point (a) mais sans effectuer d'appel récursif.

13. (Examen d'août 2024)
    (a) Décrire le plus simplement possible ce que calcule la fonction C suivante :

```
unsigned f(unsigned i, unsigned j)
    {
        if (i < j)
            return f(j, i);

        if ( j == 0)
            return 0;

        return i + f(i, j - 1);
    }
```

    (b) Quelle est la complexité en espace de cette fonction ?
    (c) Écrire une fonction C réalisant exactement la même opération que la fonction du point (a), mais sans effecuter d'appel récursif.

14. (Examen de janvier 2025)
    (a) Décrire le plus simplement possible ce que clacule la fonction C suivante :

```
unsigned f(unsigned a, unsigned b)
    {
        if (a < b || b == 0)
            return 0;

        return 1 + f(a - b, b);
    }
```

    (b) Quelle est la complexité en espace de cette fonction ? (Justifier votre réponse.)
    (c) Écrire une fonciton réalisant exactement la même opération, mais sans effectuer d'appel récursif ni d'itération de boucle.