INFO2009-Introduction à l'informatique

Aide à la réussite - Session n°9 - Récursivité

Bertrand Alexis

Université de Liège

2023

Rappel

Récursivité

Une fonction est récursive si elle s'appelle elle-même. Des fonctions sont récursives si plusieurs fonctions s'appellent mutuellement.

Attention

Comme dans les boucles, on peut provoquer des récursions infinies.

Éviter la récursion infinie

- Donner des arguments modifiés avant chaque appel (comme pour les boucles)
- Cas de base pour sortir l'appel de la pile

Complexité en espace

La complexité en espace d'une fonction consiste à regarder la place que la fonction utilise et a créé.

Ce qu'on doit regarder

- Les arguments
- Les variables créées et invoquées
- L'endroit où le code qui invoque la fonction doit reprendre son exécution dès que cette fonction se termine

Règle simplifiée

- On utilise une case mémoire lorsqu'une variable est créée
- On utilise n cases mémoires lorsqu'un tableau de taille n est créé
- On utilise une case mémoire lorsque, en récursivité, on ajoute un appel récursif à la pile d'exécution

Exercices

1.
   - Écrire une fonction C prenant en arguments deux entiers strictement 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 f^m divise n. Par exemple, si $n = 324$ et $f = 9$, on a $m = 2$ car $324 = 9^2.4$.
   - Par la méthode des invariants, démontrer que la valeur retournée par cette fonction est correcte

2.
- 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.
- Déterminer les complexités en temps et en espace de la fonction obtenue au point précédent
- Par la méthode des invariants, démontrer que la valeur retournée par la fonction obtenue est correcte. Démontrer également que cette fonction se termine

3.
- 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
- Déterminer les complexités en temps et en espace de la fonction obtenue

4. ▶ 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);
}
```

▶ Écrire une fonction qui calcule exactement la même opération, mais sans effectuer d'appel récursif

5. ▶ 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;
}
```

- ▶ Quelle est la complexité en temps de cette fonction ?
- ▶ Écrire une fonction C réalisant exactement la même opération, mais sans effectuer d'appel récursif

6. L'indicatrice d'Euler est une fonction ϕ telle que pour tout nombre entier strictement positif n, ϕ(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.
   - Donner une implémentation de cette fonction, en supposant que l'on dispose d'une bibliothèque qui implémente déjà la fonction pgcd
   - Calculer la complexité en temps de la fonction obtenue, en sachant que la complexité en temps de pgcd(m, n), avec m $\leq$ n, est $\mathcal{O}$(log m)