J1MI2013: Algorithmes et Programmes: feuille 9

Récursivité

Travaux dirigés

Exercice 1. Les fonctions `f1` et `f2` sont définies comme suit :

```
int f1(int n){
  if (n == 0){
    return 1;
  }
  return f1(n) + f1(n);
}
```

```
int f2(int n){
  return f2(n-1) + f2(n-1);
}
```

1. Que se passe-t-il à l'appel de `f1(5)` ? et de `f2(5)` ?

On modifie ces fonctions, en écrivant les 3 versions suivantes `f`, `g` et `h` :

```
int f(int n){
  if (n <= 0){
    return 1;
  }
  return f(n-1) + f(n-1);
}
```

```
int g(int n){
  if (n <= 0){
    return 1;
  }
  return 2 * g(n-1);
}
```

```
int h(int n){
  if (n <= 0){
    return 1;
  }
  int tmp = h(n/2)
  if (n%2 == 0){
    return tmp * tmp;
  }
  return 2 * tmp * tmp;
}
```

2. Les fonctions `f`, `g` et `h` calculent la même fonction mathématique. Laquelle ? Justifiez la réponse.
3. Dessinez l'arbre des appels (comme celui vu en cours pour la suite de Fibonacci) pour `f(4)`.

On s'intéresse au **temps d'exécution** et à l'**espace mémoire** consommés par ces fonctions. Une mesure approximative du temps d'exécution est donnée soit par le nombre d'appels récursifs, soit par le nombre d'opérations arithmétiques effectuées. Une mesure de l'espace consommé est donnée par la taille maximale prise par la pile d'exécution.

4. Pour chacune des fonctions `f`, `g` et `h`, répondez aux questions suivantes :
   a. Estimez l'ordre de grandeur du nombre d'appels récursifs quand on appelle la fonction avec l'argument `n`. Même estimation pour le nombre d'opérations arithmétiques effectuées lors de cet appel.
   b. Estimez l'ordre de grandeur de la hauteur maximale de la pile d'exécution quand on appelle la fonction avec l'argument `n`.
   c. En supposant qu'un ordinateur fait un milliard d'opérations arithmétiques par seconde, donnez un ordre de grandeur du temps nécessaire à la fonction pour s'évaluer quand `n` vaut 100.

Exercice 2. Écrire une version récursive de la fonction `bool appartient(int x, int t[], int n)` qui renvoie `true` si le tableau `t` contient la valeur `x` parmi ses premiers `n` éléments, et renvoie `false` sinon.

Exercice 3. Écrire une version récursive des fonctions `quotient`, `reste` et `sommeChiffres` pour lesquelles on a utilisé des algorithmes itératifs dans la feuille de TD 3.