Portail Descartes
Introduction à la science info : TD 6
Algorithmes récursifs

amU Faculté des sciences Aix Marseille Université

Exercice 1 (Exponentiation) L'exponentiation d'un nombre réel x par un entier naturel n est définie par

$$x^n = \prod_{i=1}^{n} x = \underbrace{x \times \cdots \times x}_{n \text{ fois}}$$

1. À partir de cette définition, donnez une définition mathématique par récurrence de x^n en fonction de x^{n-1}. Quel est le cas de base ?
2. À partir de cette définition, décrivez un algorithme récursif `power(x, n)` en Python pour calculer x^n (sans utiliser l'opération d'exponentiation `**` ou `pow` de Python).
3. Montrez que votre algorithme `power(x, n)` termine sur tous nombre x et entier naturel n.

Exercice 2 (Suite de Fibonacci) La suite de Fibonacci a 0 comme 0-ième terme, 1 comme premier terme et tout terme successif est la somme des deux termes précédents. Voici donc les premiers termes de cette suite :

$$0 \quad 1 \quad 1 \quad 2 \quad 3 \quad 5 \quad 8 \quad 13 \quad 21 \quad 34 \quad 55 \quad 89 \quad 144 \quad 233 \quad 377 \quad 610 \quad \cdots$$

1. Donnez une définition mathématique par récurrence du n-ième terme f_n de la suite de Fibonacci.
2. À partir de cette définition, donnez un algorithme récursif `fib(n)` qui calcule f_n.
3. Justifiez la terminaison de votre algorithme `fib(n)` pour tout entier naturel n.

Exercice 3 (Maximum d'un tableau) Soit A un tableau non vide de nombres. Alors, on peut calculer le maximum de A en comparant son premier élément et le maximum des autres éléments, sauf si A a un seul élément, et dans ce cas on prend tout simplement le premier élément comme maximum.

— Par exemple, pour le tableau $A_1 = [7, 1, 3, 2, 4]$ on compare 7 avec le maximum de $[1, 3, 2, 4]$ et on en déduit que le maximum de A_1 est 7.
— En revanche, pour $A_2 = [2, 1, 1, 5, 7]$ on compare 2 avec le maximum de $[1, 1, 5, 7]$ et cette fois-ci c'est le deuxième terme qui nous donne le maximum de 7 pour A_2.
— Si $A_3 = [6]$, alors il suffit de prendre 6 comme maximum.

1. Écrivez un algorithme récursif `max_array(A, i)` qui renvoie le maximum du sous-tableau de A allant de la position i à la position $n-1$ (comprises) pour $0 \leq i \leq n-1$ (où n est la longueur du tableau A), en implémentant le raisonnement ci-dessus.
2. Justifiez la terminaison de `max_array(A, i)` pour tout tableau non vide A et $0 \leq i \leq n-1$.

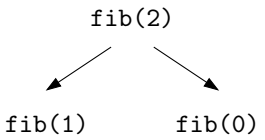
Exercice 4 (Encore l'exponentiation) La définition d'exponentiation x^n de l'exercice 1 demande un nombre de multiplications proportionnel à l'exposant n, soit $O(n)$ multiplications. Il est possible de calculer le résultat en beaucoup moins de multiplications avec un raisonnement différent. Remarquez que si $y = x^{\lfloor n/2 \rfloor}$, alors

— si n est pair, on a $x^n = y^2$;
— si n est impair, on a $x^n = x \times y^2$.

1. Écrivez un algorithme récursif `fast_power(x, n)` qui renvoie x^n en exploitant cette réflexion (sans utiliser l'opération d'exponentiation `**` ou `pow` de Python).
2. Justifiez la terminaison de `fast_power(x, n)` pour tout nombre x et pour tout entier naturel n.
3. Combien d'appels récursifs demande le calcul de `fast_power(x, n)` ? Combien de multiplications on exécute, dans le meilleur et le pire des cas, en fonction de n ?

Exercice 5 (Encore Fibonacci) L'algorithme récursif `fib(n)` proposé dans l'exercice 2 a un défaut que les autres algorithmes de cette planche n'ont pas. Comme il y a deux appels récursifs, parfois on répète plusieurs fois les mêmes calculs dans le premier et le deuxième appel. Par exemple, `fib(5)` demande de calculer `fib(4)` et `fib(3)`, mais le calcul de `fib(4)` demande de recalculer `fib(3)`.

1. Voici l'arbre de récursion de `fib(2)` avec tous les appels à `fib` exécutés (sans inclure la table des valeurs des variables), où on ne répète pas de calculs :



   Dessinez de la même façon les arbres de récursion de `fib(3)`, `fib(4)` et `fib(5)`. Combien de fois on exécute le calcul de `fib(1)` dans les trois cas ?

2. Soit x_n le nombre de fois qu'on exécute le calcul de `fib(1)` dans le calcul de `fib(n)`, pour tout $n \geq 0$. Trouvez une expression mathématique par récurrence pour x_n et tirez vos conclusions.