Langace C - Travaux Dirigés n° 4

— Licence Informatique —

▶ Exercice 1 (Recherche Dichotomique)
Écrire une fonction `dicho` qui effectue la recherche dichotomique d'un enier `x` dans un tableau de `N` entiers triés par ordre croissant. Cette fonction renverra l'indice de la case où a été trouvé `x` et `-1` sinon. Pour créer le tableau trié on utilisera la fonction d'insertion d'un élément vue au Td 2.

▶ Exercice 2 Décomposition Binaire
Écrire une fonction `binaire` qui affiche les bits de la décomposition binaire d'un entier positif dans l'ordre de poids décroissant.

▶ Exercice 3 (Exponentiation logarithmique)
Écrire une fonction `power` qui calcule x^n pour un premier paramètre flottant x et un paramètre entier n. Essayer d'avoir un nombre de mutiplications proportionnel au logarithme de n. On pourra utiliser de façon récursive les formules

$$\begin{aligned} x^{2n} &= (x^n)^2 \\ x^{2n+1} &= x^{2n} \times x \end{aligned}$$

▶ Exercice 4 (Nombres de Fibonacci)
Écrire une fonction récursive `fibo` qui calcul le n-ième nombre de Fibonacci F_n. On rappelle que les nombres de Fibonacci sont définis par

$$F_0 = 1 \quad F_1 = 1 \quad \text{et} \quad F_{n+2} = F_{n+1} + F_n$$

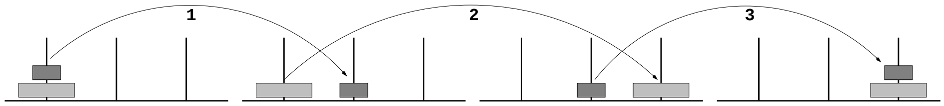
Quelle est le nombre d'appels récursifs nécessaire pour calculer F_n. On pourra commencer par faire calculer ce nombre pour les premières valeurs de n afin de deviner une formule.

▶ Exercice 5 (Nombres de Fibonacci 2)
Le but de cet exercice est de calculer les nombres de Fibonacci avec un nombre logarithmique d'appels récursifs. Écrire une fonction `fiborec` qui calcule le couple (F_n, F_{n-1}). Cette fonction calcule récursivement le couple (F_p, F_{p-1}) pour $p = n/2$ puis utilise les formules suivantes

$$\begin{aligned} F_{2n+1} &= 2F_nF_{n-1} + F_n^2 \\ F_{2n} &= F_n^2 + F_{n-1}^2 \\ F_{2n-1} &= 2F_nF_{n-1} - F_{n-1}^2 \end{aligned}$$

Ensuite appeler la fonction `fiborec` à l'aide d'une fonction `fibolog`.

▶ Exercice 6 (Tours de Hanoï)
Le problème des *Tours de Hanoï* consiste à déplacer n disques d'un piquet de départ à un piquet d'arrivée en utilisant seulement un troisième piquet. La seule contrainte tient au fait qu'il est interdit de poser un disque sur un autre disque plus petit. L'exemple graphique ci-dessous illustre la situation pour $n = 2$.



1. Résoudre *à la main* le problème pour $n = 3$.
2. En déduire un algorithme récursif pour un n quelconque.
3. Ecrire une fonction récursive `Hanoi` qui renvoie le nombre de déplacements effectués.