Exercices sur la récursivité

Exponentiation rapide

1. Écrire une fonction récursive $Fpow$ qui prend en argument deux entiers a, n et calcule a^n à l'aide des formules :
$$\text{si } n = 2p : \ a^n = (a^p)^2$$
$$\text{si } n = 2p+1 : \ a^n = a\,(a^p)^2$$

2. Évaluer sa complexité temporelle.

Partages d'un entier

Un partage en p d'un entier n est une liste $[x_1, ..., x_p] \in (\mathbb{N}^*)^p$ vérifiant :
$$\begin{cases} x_1 + x_2 + ... + x_p = n \\ x_1 \geq x_2 \geq ... \geq x_p \end{cases}$$

1. Écrire une fonction récursive $partage$ qui prend en argument 2 entiers n, p et retourne la liste des partages en p de n. Indication : comment construire un partage commençant par i ?

2. Écrire une fonction *temps* qui calcule le temps de partage. Pour cela on utilisera la fonction *time.clock* .

3. Écrire une fonction *grafp* qui prend en argument un entier n, et trace le graphe de la fonction *temps*$(n, .)$. Écrire une fonction *grafn* analogue qui trace le graphe de *temps*$(., p)$ sur $[p, p + 20]$.

Dédoublonnage

On dispose d'une liste L contenant certains objets en plusieurs exemplaires.

1. Ecrire une *fonction pure* récursive $sd1$ qui prend L en argument, et retourne la liste L' constituée des mêmes objets que L mais en simple exemplaire.

2. Ecrire une *procédure pure* $sd2$ qui fait la même chose que $sd1$.

3. Evaluer les complexités temporelle et spatiale de $sd1$ et $sd2$.

4. Ecrire une fonction de comparaison temporelle de $sd1$ et $sd2$, calculant le temps de dédoublonnage d'une liste de n entiers aléatoires entre 0 et 9. Regarder pour $n \in \{900, 950, 1000\}$.

5. Ecrire une fonction récursive $sd3$ qui prend en argument une liste L, et retourne la liste des couples $(item, nbre)$ où les $item$ sont les éléments distincts de L, et $nbre$ est leur nombre d'occurrence dans L.

Fibonacci ultime

1. Prouver par récurrence que la suite de Fibonacci u vérifie :
$$\forall p \in \mathbb{N}\ ,\ \begin{cases} u_{2p} = u_p^2 + 2u_p u_{p-1} \\ u_{2p+1} = u_p^2 + u_{p+1}^2 \end{cases}$$

2. Ecrire une fonction récursive $fibultim$ qui prend en argument un entier n, et retourne u_n en utilisant les formules précédentes et un dictionnaire.
Un dictionnaire est un ensemble d'éléments du type $x : y$. Ici l'initialisation sera $dico = \{0 : 0, 1 : 1\}$, et pour compléter le dictionnaire après avoir calculé $y = fib(x)$ on fait $dico[x] = y$.

3. Inclure à cette fonction un compteur d'appels, et comparer la performance de $fibultim$ à celle de la fonction vue en cours.

Jeu de Nim

Deux joueurs A et B sont devant une table, sur laquelle on dispose n allumettes. A joue et enlève 1, 2 ou 3 allumettes. B joue ensuite et fait de même. Celui qui enlève la dernière allumette a perdu.

1. Quelles sont les valeurs de n pour lesquelles A peut jouer de telle sorte que B perde au tour suivant ?

2. Quelles sont les valeurs de n pour lesquelles A peut jouer de telle sorte que quelle que soit la manière de jouer de B au tour suivant, B perde au tour d'après ? Faire un raisonnement récursif et en conclure quelles sont les valeurs de n pour lesquelles A est sûr de gagner.

3. Ecrire une fonction $Ajoue$ qui prend en argument un entier n, et retourne le nombre d'allumettes restantes après que A ait joué. On suppose que A joue de façon à assurer sa victoire si n est favorable, sinon il joue au hasard.

4. Ecrire une fonction $Bjoue$ qui fait de même, ie assure la victoire de B si $Ajoue(n)$ est favorable.

5. Ecrire une fonction récursive Nim qui prend en argument un entier n, et imprime une suite de phrases du type : " X joue et il reste p allumettes " et se terminant par " X a perdu. "

6. Le jeu de Marienbad est analogue mais se joue avec plusieurs tas de $n_1, ..., n_p$ allumettes. Chaque joueur prend alternativement au moins 1 allumette (il n'y a pas de maximum) dans un seul des tas. Ecrire des fonctions $AjoueM$ et $BjoueM$ utilisant des stratégies aléatoires, et une fonction récursive $Marienbad$ qui simule la partie.