Université Paris Diderot Année 2020-2021

L3 M1 - Bio Info

Algorithmique

Exercice 1 : Listes chaînées

On rappelle qu'une liste chaînée est la donnée d'une chaîne de maillons. Plus précisément, un *maillon* est une structure avec un champ `data` représentant une donnée stockée, et un champ `next` qui est un pointeur (éventuellement `null`) vers un autre maillon. Une liste chaînée est alors une structure contenant un pointeur `head` vers le premier maillon d'une chaîne finie de maillons.

1. Écrire une procédure récursive permettant de calculer la taille d'une liste. Quelle est sa complexité en fonction de la taille de la liste ?
2. Écrire une procédure récursive permettant d'accéder à l'élément à une position i donnée d'une liste. Quelle est sa complexité en fonction de i ?
3. Écrire des procédures récursives permettant d'ajouter et de supprimer un élément à une liste à une position i donnée. Quelles sont leurs complexités en fonction de i ?
4. Écrire une procédure récursive permettant de concaténer deux listes. Quelle est la complexité de cette procédure en fonction de la taille des deux listes ?
5. Écrire une procédure permettant d'inverser une liste. Cette procédure pourra utiliser une procédure auxilliaire prenant en argument des pointeurs vers deux maillons. Quelle est la complexité de cette procédure en fonction de la taille de la liste ?
6. Comparer ces complexités avec celles des mêmes opérations pour les tableaux.

Exercice 2 : Quelques récursions supplémentaires

1. Écrire une procédure récursive permettant de déterminer si un mot est un palindrome, c'est-à-dire un mot qui est égal à son miroir (par exemple "radar"). On représentera un mot comme un tableau de caractères.
2. Écrire une procédure récursive permettant d'obtenir la décomposition en base 2 d'un entier naturel n.

Exercice 3 : Exponentiation rapide pour Fibonacci

On note $(u_n)_n$ la suite de Fibonacci, c'est-à-dire la suite vérifiant $u_0 = 0$, $u_1 = 1$ et $u_{n+2} = u_{n+1} + u_n$ pour tout $n \geq 0$. Soit $v_n = \begin{pmatrix} u_n \\ u_{n+1} \end{pmatrix} \in \mathbb{R}^2$ pour tout $n \geq 0$.

1. Décrire une matrice $M \in \mathbb{R}^{2\times 2}$ vérifiant $v_{n+1} = Mv_n$ pour tout $n \geq 0$.
2. Pour tout $n \geq 0$, à quoi correspond le vecteur $M^n \begin{pmatrix} 0 \\ 1 \end{pmatrix}$?
3. Décrire des algorithmes itératif et récursif permettant de calculer M^n en $\Theta(n)$.
4. Décrire un algorithme permettant de calculer M^n en $\Theta(\log(n))$ opérations.
5. En déduire un algorithme pour calculer u_n en $\Theta(\log(n))$ opérations.

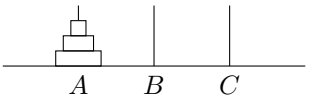
Remarque : cette méthode s'adapte pour calculer les termes de toute suite $(u_n)_n$ dont la formule de récurrence est de la forme $u_{n+k} = a_{k-1}u_{n+k-1} + \cdots + a_0 u_n$.

Exercice 4 : Tours de Hanoï

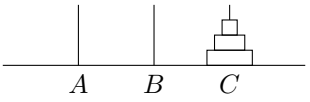
Le jeu des tours de Hanoï consiste à déplacer n disques de diamètres différents d'une "tour", notée A, à une autre tour, notée C, en utilisant une tour intermédiaire, notée B. Un mouvement d'un disque d'une tour à une autre est autorisé si

- le disque est au sommet de la tour de départ,
- la tour d'arrivée est vide ou le disque à son sommet a un diamètre supérieur à celui que l'on souhaite déplacer,
- le disque est placé au sommet de la tour d'arrivée.

Initialement, les n disques sont tous sur la tour A avec un diamètre décroissant avec la hauteur. Le but du jeu est de déterminer la séquence de mouvements à faire pour mettre tous ces disques sur la tour C. Dans le cas $n = 3$, on part de la configuration



et l'on souhaite arriver à la configuration



en ne faisant que des mouvements autorisés.

1. Résoudre le jeu à la main dans le cas $n = 3$ et $n = 4$. Quelle est la configuration des tours lorsque l'on déplace le disque de plus gros diamètre ?
2. En déduire un algorithme pour n général. Plus précisément, écrire un algorithme récursif sous la forme d'une fonction `SOLVE_HANOI(from,to,`n`)` qui produira les mouvements à effectuer pour déplacer les n premiers disques de la tour `from` vers la tour `to` en supposant que toutes les tours sauf `from` sont vides ou avec des disques de diamètres supérieurs.
3. Calculer le nombre de mouvements nécessaires pour un n fixé et en déduire la complexité de l'algorithme.