Travaux Pratiques d'algorithmique n°4

Cours d'Informatique de Deuxième Année

—Deug SSM M3-MI—

Récursivité

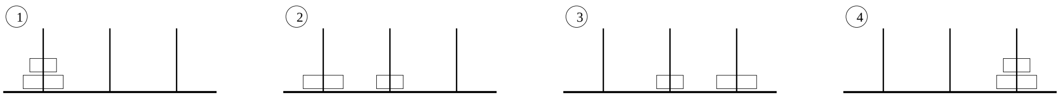
Au TP précédent, nous avons introduit la notion de récursivité sur des exemples simples. Nous allons maintenant appliquer cette méthode pour résoudre des problèmes plus complexes. Vous devrez présenter clairement les algorithmes, ainsi que le calcul de leur complexité.
Enfin, nous introduirons la notion de structure que nous généraliserons la séance suivante.

▸ **Exercice 1.** *On définit les nombres de Fibonacci par :*

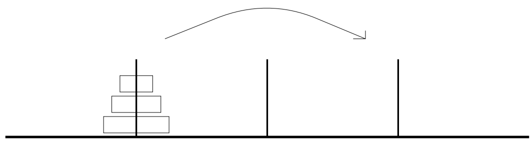
$$\begin{cases} F(0) = 0\ , \qquad F(1) = 1\ , \\ F(n) = F(n-1) + F(n-2) \quad \text{si } n \geq 2\ . \end{cases}$$

- *Calculer F(6) ;*
- *Programmer de façon récursive, la fonction* `int FiboR(int n)` *qui retourne le nième nombre de Fibonacci.*
- *Quel est le défaut de cette méthode ?*
- *Programmer une nouvelle fonction* `int FiboIt(int n)` *qui calcule le nième nombre de Fibonnaci de façon itérative. Celle-ci utilisera un tableau auxiliaire qui stockera les calculs intermédiaires.*
- *Evaluer la complexité en temps et en espace, de ce dernier algorithme.*
- *Quel est le défaut de cette méthode ?*
- *Finalement, programmer une nouvelle fonction* `int FiboItBis(int n)` *qui sera optimale en temps et en espace.*

▸ **Exercice 2.** *Nous cherchons à résoudre le problème des "tours de Hanoï". Ce problème consiste à déplacer* `n` *disques d'un piquet de départ à un piquet d'arrivée* *en utilisant un troisième piquet. La seule contrainte tient au fait qu'il est interdit de poser un disque sur un disque plus petit que lui.*



Résolution du problème pour $\mathtt{n} = 2$.



Exemple du problème pour $\mathtt{n} = 3$.

1. *résoudre "à la main" l'algorithme pour* `n` $= 3$ *;*
2. *en déduire un algorithme récursif de résolution du problème ;*
3. *écrire la fonction correspondant à cet algorithme : la fonction devra afficher la suite des déplacements des disques ; Elle retounera le nombre de déplacements nécessaires.* `int Hanoi(int n, char dep, char tmp, char arr)` *où* `n` *est la hauteur de la tour, et* `dep,tmp,arr` *sont les noms (un seul caractère) des trois piquets de départ, de transition, et d'arrivée.*
4. *donner enfin, la complexité en temps de votre algorithme.*

▸ **Exercice 3.** *Pour terminer cette séance, nous introduisons la notion de structure. La structure suivante permet de définir un tableau d'entiers ainsi que le nombre d'éléments du tableau.*

```
typedef struct tableau
{
  int tab[MAX];
  int nb_el;
} STab;
```

A partir de cette structure, programmer les fonctions suivantes :

- `int Est_Vide(STab t)` *qui indique si le tableau contenu dans la structure, est vide, (retourne 1 si vrai, 0 sinon) ;*
- `int Insere_Valeur(STab *t,int val)` *qui insère un élément de valeur* `val` *dans le tableau de la structure ;*
- `void Affiche_Tableau(STab t)` *;*
- `int Somme(STab t)` *qui retourne la somme des éléments du tableau ;*
- `int Enleve_Elem(STab *t,int ieme)` *qui supprime le k-ième élément du tableau.*