Travaux Pratiques d'algorithmique n°4

Cours d'Informatique de Deuxième Année

—DEUG Sciences S3-MI—

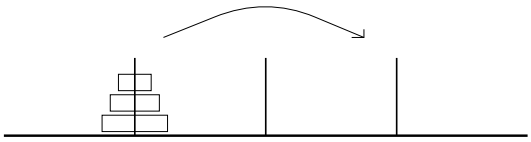
Récursivité

Dans cette séance, nous approfondissons la notion de récursivité à travers deux exemples : le problème des tours de Hanoï et le tri par fusion. Nous nous intéressons en particulier aux calculs de complexité de chaque algorithme implémenté.

▸ Exercice 1.

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* $\mathtt{n} = 3$;
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 retournera 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.*
5. *On suppose maintenant que la pile de disques de départ est constituée de disques de n tailles différentes, le nombre de disques par taille étant de p. Écrire la fonction* `Hanoi2` *qui résout le problème des "tours de Hanoï" pour une pile de disques ainsi constituée.*

▸ **Exercice 2.** *Le but de cet exercice est d'implémenter la méthode du tri par fusion. Cette méthode permet de fusionner et de trier deux suites triées dans l'ordre croissant. Soient* $a_0, ..., a_n$ *et* $b_0, ..., b_m$ *deux suites triées dans l'ordre croissant. Pour obtenir la suite interclassée* $c_0, ..., c_{n+m+1}$ *des* a_i *et* b_j*, il suffit de parcourir simultanément les deux suites* $a_0, ..., a_n$ *et* $b_0, ..., b_m$ *et de poser* $c_k = min(a_i, b_j)$ *où* $k = i + j$*.*

Donc pour trier un tableau `t` *de réels, on divise le tableau en deux partie, on trie les deux parties et on interclasse les deux parties triées.*

1. *Écrire une fonction itérative* `void Fusion(float tab[],float fusion[],int g,int d)` *qui à partir du tableau* `tab` *dont les sous-tableaux* $\{tab_g, ..tab_{\frac{g+d}{2}}\}$ *et* $\{tab_{\frac{g+d}{2}+1}, ..., tab_d\}$ *sont triés dans l'ordre croissant, écrit dans le tableau* `fusion` *le tableau* `tab` *trié.*

2. *Écrire une fonction* `TriFusion` *qui trie un tableau* `t` *de réels par la méthode donnée ci-dessus.*

Donner la complexité en temps de votre algorithme.