Mini-Projet de Mathématiques Discrètes

Partie 1.

1- Dans notre algorithme récursif, on déplace d'abord n-1 disques vers la tour intermediaire (U_{n-1} déplacements), puis le n éme disque sur la tour de destination (1 déplacement), et enfin les n-1 disques sur la tour destination (encore une fois U_{n-1} déplacements).

Le nombre de déplacements réalisé par notre algorithme peut donc se calculer ainsi :

$$U_0 = 0$$
$$U_{n+1} = 2 \times U_n + 1$$

2- Démontrons que U_n est donné par la formule $U_n = 2^n - 1$

Par récurrence :

- initialisation : $U_0 = 0 = 2^0 - 1$
- Supposons U_n vraie, (sachant que $U_{n+1} = 2 \times U_n + 1$)

$$U_n = 2^n - 1$$
$$U_{n+1} = 2(2^n - 1) + 1$$
$$U_{n+1} = 2^{n+1} - 2 + 1$$
$$U_{n+1} = 2^{n+1} - 1$$

3 et 4 - Cet algorithme est optimal car il déplace une seule fois le plus gros disque à chaque appel récursif. Pour pouvoir déplacer le plus gros disque, il n'y a pas d'autre solution que de d'abord déplacer tous les disques au dessus vers la tour intermediaire, et de mettre le plus gros sur la tour finale.

On recommence ensuite en inversant la tour de départ et la tour intermediaire. Il est donc impossible de résoudre le problème en faisant moins de coups, et toute solution mettant le même temps à déplacer tous les disques devra s'y prendre de cette manière.

5- Supposons que dans l'algorithme optimal on ne soit pas obligé de déplacer le plus petit disque un coup sur 2 :

- Soit on le déplace 2 fois de suite, ce qui peut revenir à utiliser 2 coups pour ne rien changer à la situation ou à utiliser 2 coups pour effectuer un deplacement faisable en un seul : Dans les 2 cas cela ne peut pas correspondre à l'Algorithme optimal
- Soit on ne le touche pas pendant 2 coup, ce qui revient à déplacer un disque dans un sens, puis à lui faire faire exactement le même déplacement dans l'autre sens : encore une fois incompatible avec l'algorithme optimal

Il est donc impossible de réaliser l'algorithme optimal sans déplacer le plus petit disque exactement un coup sur 2.

7- Dans notre algorithme itératif, on dépace le petit disque un coup sur 2 dans le sens $A \rightarrow B \rightarrow C$ si on a un nombre de disque pair (et dans le sens inverse pour un nombre de disque impair).

On déplace d'abord le plus petit disque un coup, puis on entre dans une boucle qui réalise le seul mouvement possible ne concernant pas le petit disque et ensuite fait bouger le petit disque d'un cran dans le bon sens.

On s'arrête quand on a réalisé autant de déplacements que le nombre minimal de déplacements possible (Ce qui montre également que cet algorithme itératif est optimal).

8- Le plus gros disque subira un seul déplacement($D(1)=2^0=1$) : celui qui le mettra à la base de la tour de destination.

Le second plus gros en subira le double($D(2)=2^1=2$) : celui qui le mettra à la base de la tour intermédiaire et celui qui le mettra sur le plus gros sur la tour de destination.

En raisonnant ainsi sur l'algorithme décrit plus haut, en en déduit que $D(n)=2^{n-1}$.

On peut ainsi calculer $U_n=D(n)+D(n-1)+...+D(1)$

$$U_n=\sum_{i=1}^{n} D(i)$$

9- Cet algorithme commence par déplacer n-1 disques de la tour de départ vers la tour destination, puis on déplace le plus gros disque d'un cran vers la destination (donc sur la tour intermediaire). Ensuite, on libère la tour destination en déplacant les n-1 disque qui s'y trouvent vers la tour de départ et on met le plus gros dique sur la tour de destination. Pour finir on à juste à recommencer avec les n-1 disques se trouvant sur la tour de départ.

10- Si on tire directement Vn des algorithmes du code, on obtient :

$$\begin{matrix} V_0=0 \\ V_n=2\times V_{n-1}+U_{n-1}+2 \end{matrix} \quad \text{avec} \quad \begin{matrix} U_0=0 \\ U_n=2\times V_{n-1}+1 \end{matrix}$$

En Simplifiant un peu on obtient l'équation de récurrence suivante :

$$V_n=2\times V_{n-1}+2\times V_{n-2}+3$$
$$avec \quad U_0=0 \quad et \quad U_1=2$$

La résolution de cette équation de récurrence nous donne :

$$V_n=\frac{1}{6}\left(\left(3-2\sqrt{3}\right)\left(1-\sqrt{3}\right)^n+\left(3+2\sqrt{3}\right)\left(1+\sqrt{3}\right)^n-6\right)$$

11- On a vu que dans cet algorithme on déplace uniquement 2 fois le disque le plus gros de chaque appel, ce qui est le nombre de déplacements minimum (avec la contrainte ajoutée).

De plus, on se retrouve à la fin d'un appel dans la situation initiale avec un disque en moins et on recommence. Il n'est donc pas possible de résoudre ce problème en faisant moins de mouvements que ce que fait cet algorithme.