Mini-projet - LI311
Autour des Tours de Hanoï

L'objectif de ce mini-projet est de concevoir et d'évaluer la complexité de divers algorithmes pour résoudre des jeux mathématiques. On fera appel à un algorithme de graphe vu en cours et en TD, ainsi qu'à des algorithmes dédiés. Ce mini-projet sera également l'occasion de découvrir comment il est possible de travailler avec des graphes de grande (voire très grande) taille, en ne les représentant pas explicitement en mémoire.

1. Introduction

Dans le tome 3 de ses Récréations mathématiques (parues en 1892), Édouard Lucas relate que "N. Claus de Siam a vu, dans ses voyages pour la publication des écrits de l'illustre Fer-Fer-Tam-Tam, dans le grand temple de Bénarès, au-dessous du dôme qui marque le centre du monde, trois aiguilles de diamant, plantées dans une dalle d'airain, hautes d'une coudée et grosses comme le corps d'une abeille. Sur une de ces aiguilles, Dieu enfila au commencement des siècles, 64 disques d'or pur, le plus large reposant sur l'airain, et les autres, de plus en plus étroits, superposés jusqu'au sommet. C'est la tour sacrée du Brahmâ. Nuit et jour, les prêtres se succèdent sur les marches de l'autel, occupés à transporter la tour de la première aiguille sur la troisième, sans s'écarter de règles immuables imposées par Brahmâ. Quand tout sera fini, la tour et les brahmes tomberont, et ce sera la fin du monde !".

2. Les explorateurs et les adorateurs

Un groupe d'explorateurs a découvert un document indiquant l'emplacement précis du grand temple de Bénarès. Une mission internationale est alors lancée afin de rejoindre ce site sacré. Le temple est située sur une île au milieu d'un lac. Pour rallier l'île, les n explorateurs, accompagnés de n adorateurs de Brahmâ, doivent emprunter une barque (que tout le monde est capable de manœuvrer) de capacité k (autrement dit, au maximum k personnes peuvent embarquer à la fois). À aucun moment, ni sur la rive de départ, ni sur l'île, ni sur l'embarcation, il ne doit y avoir un groupe d'explorateurs en minorité par rapport à un groupe d'adorateurs car sinon ces derniers pourraient tenter de les convertir, et le destin des malheureux serait alors voué à rester cloîtrés dans le temple de Bénarès jusqu'à la fin de leurs jours. Il s'agit donc ici de concevoir un algorithme pour faire passer sur l'île les n explorateurs et les n adorateurs en un nombre minimum de traversées.

La résolution de ce problème peut se voir comme la recherche d'une chaîne dans un graphe particulier, appelé graphe des configurations. Un sommet de ce graphe représente une configuration du problème (c'est-à-dire une description complète de la situation courante : nombre d'explorateurs et d'ado-

rateurs sur chaque rive, et position de la barque) et une arête une traversée permettant de passer d'une configuration à l'autre. Le triplet (n_E, n_A, p) défini par le nombre d'explorateurs n_E qui sont sur l'île, le nombre d'adorateurs n_A qui sont sur l'île et la position p de la barque ($p = I$ si la barque se situe sur la rive de l'île, $p = \overline{I}$ si la barque se situe sur la rive de départ) caractérise complètement une configuration. En effet, si n_E explorateurs et n_A adorateurs se trouvent sur l'île, alors $n - n_E$ explorateurs et $n - n_A$ adorateurs se trouvent sur la rive de départ. Dans le graphe, on ne considère que les configurations valides. Une configuration est dite valide si aucun explorateur n'est en situation d'être converti par les adorateurs. Une arête relie deux configurations valides si une traversée valide (c'est-à-dire telle qu'aucun explorateur n'est en situation d'être converti sur la barque, et que la capacité de la barque n'est pas dépassée) permet de passer d'une configuration à l'autre. Le sommet initial correspond à $(0, 0, \overline{I})$ et le sommet but est (n, n, I). Pour $n = 3$, le sommet initial est relié entre autres au sommet $(0, 2, I)$ car la traversée de deux adorateurs est valide (voir figure 1) : sur les deux rives et sur la barque, aucun explorateur n'est en situation d'être converti.

Question 1. Indiquer une séquence de traversée valides (plus précisement, la séquence correspondante de configurations valides) pour passer de la configuration initiale à la configuration but pour $n = 3$ et $k = 2$.

Dans la suite de cette section, on suppose n et k quelconques ($k < 2n$).

Question 2. Définir l'ensemble des configurations valides voisines de la configuration initiale.

Question 3. L'objet de cette question est d'identifier les conditions sur n_E et n_A que doivent respecter les configurations valides. On distingue trois cas. Le premier est $n_E = 0$. Quels sont les deux autres cas ?

Question 4. L'objet de cette question est d'identifier les conditions sur n_E, n_A, n'_E, n'_A pour que les configurations valides (n_E, n_A, I) et $(n'_E, n'_A, \overline{I})$ soient voisines (autrement dit, que la traversée de (n_E, n_A, I) à $(n'_E, n'_A, \overline{I})$, ou inversement, soit valide). On distingue deux cas. Le premier est le suivant : $n_E - n'_E = 0$ et $0 < n_A - n'_A \leq k$. Quel est l'autre cas ?

Question 5. On vise dans cette question à déterminer une séquence minimale de traversées pour que les n explorateurs et les n adorateurs atteignent tous l'île sans qu'aucun explorateur ne soit converti par des adorateurs. Autrement dit, on cherche à déterminer dans le graphe des configurations une plus courte chaîne en nombre d'arêtes de $(0, 0, \overline{I})$ à (n, n, I). On note N le nombre de sommets et M le nombre d'arêtes dans le graphe des configurations. Quel algorithme en $O(N + M)$ vu en cours et en TD permet de déterminer une plus courte chaîne en nombre d'arêtes dans un graphe ? On appellera *Traversee* la procédure consistant à appliquer cet algorithme dans le graphe des configurations.

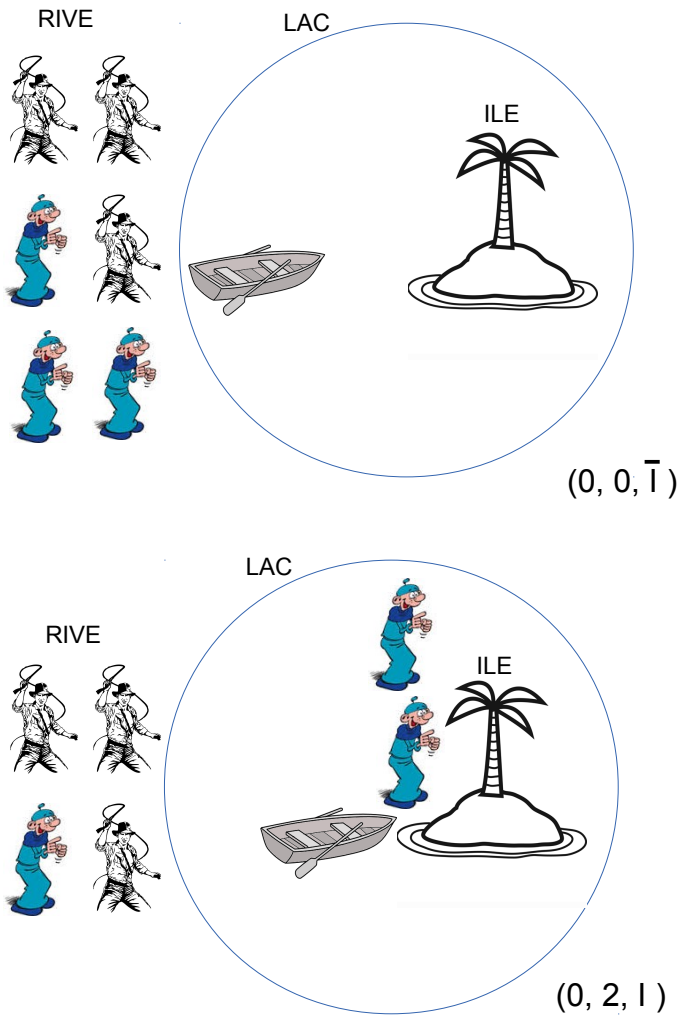


FIGURE 1 – Un exemple de traversée valide dans un problème à 3 explorateurs et 3 adorateurs (Indiana Jones figure un explorateur, Disciple un adorateur).

Question 6. Exprimer N en fonction de n. Si on suppose que la capacité de la barque est une *constante*, quelle est la complexité de *Traversee* en fonction de n ?

Question 7. Implémenter *Traversee* sans représenter explicitement le graphe des configurations en mémoire (on n'utilisera donc pas de représentation par matrice d'adjacence ou par listes d'adjacence) : à chaque fois qu'il sera nécessaire de déterminer les voisins d'un sommet, on fera appel aux conditions identifiées dans la question 4 pour générer les configurations voisines. Par exemple, pour $n = 3$ et $k = 2$, si le sommet courant visité par *Traversee* est $(0, 1, \overline{I})$ alors les configurations voisines à visiter sont $(1, 1, I)$, $(0, 2, I)$ et $(0, 3, I)$.

Pour mémoriser les configurations visitées, on utilisera le principe suivant. On fait correspondre à chaque configuration un numéro d'identification entre 1 et N. On définit ensuite un tableau de $\lceil \frac{N}{8} \rceil$ octets. Ce tableau d'octets peut être interprété comme un tableau de N booléens (on n'utilise pas directement un tableau de booléens car cela prendrait beaucoup de place en mémoire). Le bit d'indice i de ce dernier tableau est égal à 1 si la configuration numérotée i a été visitée, 0 sinon.

À l'aide d'expérimentations numériques où on aura désactivé l'affichage de la solution (ou toute forme d'écriture dans un fichier), tracer la courbe du temps CPU d'exécution de l'algorithme implémenté en fonction du nombre n d'explorateurs, la capacité de la barque étant fixée à $k = 4$. Il s'agira de générer un fichier de points qui sera utilisé pour générer une courbe à l'aide d'un outil graphique tel que `gnuplot` ou `xgraphic` pour lesquels une petite documentation est en ligne sur le site web de l'UE : `http://goo.gl/6yxw2R` (pour gnuplot), `http://goo.gl/Yf8w9b` (pour xgraphic). Comparer la croissance de cette courbe avec la complexité théorique.

3. Les Tours de Hanoï

Une fois arrivés sur l'île, la mission confiée aux explorateurs est de découvrir la date au plus tôt à laquelle aura lieu la fin du monde. Rappelons que le problème des Tours de Hanoï consiste à déplacer des disques de diamètres différents d'un piquet de départ à un piquet d'arrivée en s'aidant d'un piquet intermédiaire, tout en respectant les règles suivantes (édictées par Brahmâ) :

- on ne peut déplacer plus d'un disque à la fois,
- on ne peut placer un disque que sur un autre disque plus grand que lui ou sur un piquet vide.

On suppose que cette dernière règle est également respectée dans la configuration de départ. Le piquet de départ (resp. d'arrivée) est le piquet de gauche (resp. droite).

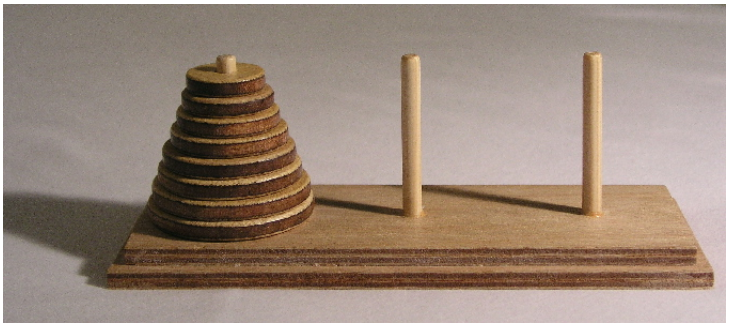


FIGURE 2 – Un exemple à 8 disques.

3.1. Le graphe de Hanoï

Nous avons vu en TD un algorithme de résolution depuis la configuration de départ. Néanmoins, lorsque les explorateurs arrivent sur place, les prêtres sont bien sûr déjà attelés à leur tâche depuis une durée indéfinie. La mission qui incombe aux explorateurs est donc d'évaluer le nombre de déplacements nécessaires à la résolution du problème depuis une configuration *quelconque*. L'informaticien du groupe propose d'adapter au problème des Tours de Hanoï la méthode de résolution déjà employée pour résoudre le problème précédent. Il s'agit donc dans un premier temps de définir un graphe des configurations pour le problème des Tours de Hanoï. On caractérise une configuration par un tableau $p[1\ldots\text{n}]$ où $p[d] \in \{1, 2, 3\}$ correspond au numéro du piquet sur lequel est placé le disque d. Une configuration est voisine d'une autre si l'une peut être obtenue depuis l'autre par le déplacement d'un seul disque. Le graphe des configurations pour n disques est noté G_n (graphe de Hanoï). Les graphes G_1, G_2 et G_3 sont représentés sur la figure 3.

Question 8. Combien y a-t-il de sommets dans le graphe des configurations G_n en fonction de n ?

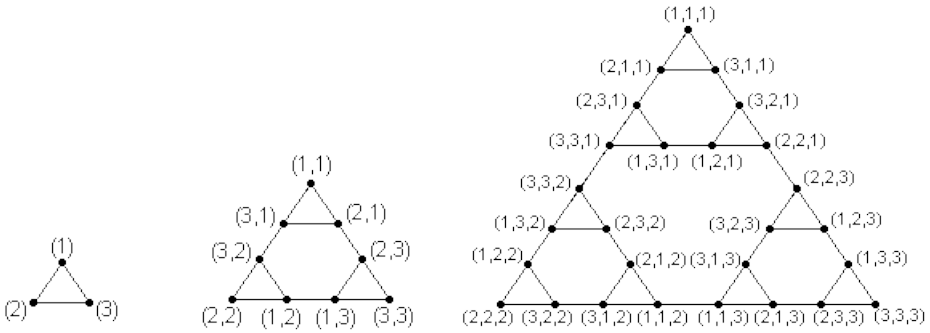


FIGURE 3 – Les graphes G_1, G_2 et G_3.

Question 9. Par récurrence sur n, déterminer le nombre d'arêtes dans le graphe des configurations G_n en fonction de n.

Rappel : soit $(u_k)_{k\in\mathbb{N}}$ une suite géométrique de raison $q \neq 1$ (i.e., une suite telle que $u_{k+1} = q \times u_k$), on a : $\sum_{k=m}^{n} u_k = u_m \frac{1-q^{n-m+1}}{1-q}$.

3.2. Une première approche de résolution

Question 10. Dans cette question, on considère le même algorithme que pour le problème précédent, cette fois-ci appliqué au graphe de Hanoï afin de déterminer une séquence minimale de déplacements de disques jusqu'à résolution depuis une configuration quelconque. On appellera *Hanoi1* cet algorithme. Encore une fois, on générera "à la volée" les voisins d'une configuration dans le graphe de Hanoï. Pour représenter une configuration, on utilise un triplet de piles (une pile de disques par piquet), afin d'identifier en temps constant les déplacements de disques possibles depuis une configuration. Quelle est la complexité de *Hanoi1* en fonction du nombre n de disques ?

Question 11. Implémenter *Hanoi1*. À l'aide d'expérimentations numériques où on aura désactivé l'affichage de la solution (ou toute forme d'écriture dans un fichier), tracer la courbe du temps moyen d'exécution de l'algorithme implémenté en fonction du nombre n de disques. Pour ce faire, pour chaque valeur de n, on calculera le temps moyen d'exécution pour 50 configurations de départ tirées aléatoirement. Comparer la croissance de cette courbe avec la complexité théorique.

3.3. Une deuxième approche de résolution

On se propose maintenant de développer un autre algorithme pour résoudre ce même problème de recherche d'une séquence minimale de déplacements de disques jusqu'à résolution depuis une configuration quelconque. Dans ce

but, on va adapter l'algorithme récursif vu en TD pour pouvoir trouver une séquence minimale depuis une position quelconque.

Question 12. Implémenter la procédure *Hanoi* vue en TD, rappelée ci-dessous, où :

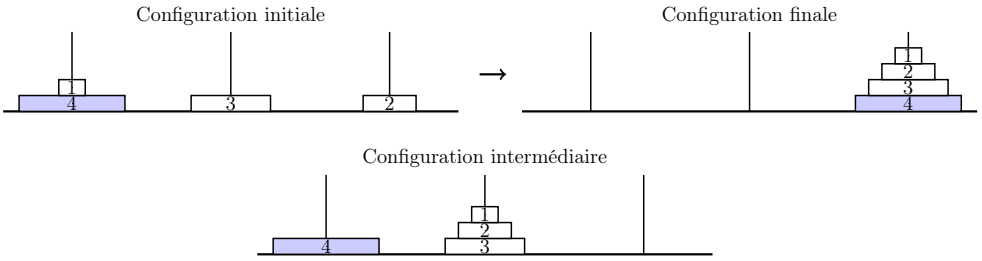
- n : nombre de disques utilisés ;
- *6 - piquet1 - piquet2* : numéro du piquet intermédiaire (par exemple, si *piquet1* $= 1$ et *piquet2* $= 3$, alors *6 - piquet1 - piquet2* $= 6-1-3 = 2$) ;
- $Hanoi(n, origine, destination)$: déplace n disques du piquet *origine* au piquet *destination* en passant par le piquet $6-origine-destination$;
- $Bouger(origine, destination)$: déplace le disque se trouvant en haut de la pile du piquet *origine* vers le piquet *destination*.

```
procédure Hanoi(n, piquet1, piquet2 : entiers)
  si n>0 alors
    Hanoi(n-1, piquet1, 6 - piquet1 - piquet2) ;
    Bouger(piquet1, piquet2) ;
    Hanoi(n-1, 6 - piquet1 - piquet2, piquet1) ;
  fin si
```

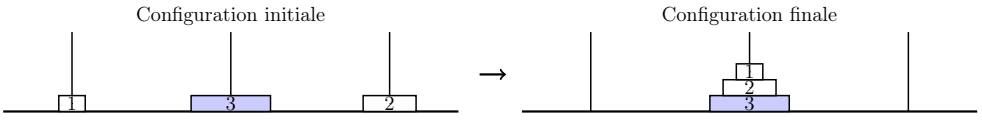
Question 13. Pour résoudre le problème des Tours de Hanoï depuis une configuration quelconque, on se propose de décomposer le problème de départ en n sous-problèmes $P(d)$ portant sur les disques 1 à d (pour d variant de 1 à n, le sous-problème sur n disques étant le problème de départ). Un exemple de décomposition en sous-problèmes pour un problème à 4 disques est donné sur la figure 4. Pour passer de la configuration initiale à la configuration finale (sous-problème $P(4)$), il est nécessaire de passer par la configuration intermédiaire indiquée sur la figure 4(a), où les disques 1 à 3 sont empilés sur le piquet 2. Plus généralement, pour pouvoir déplacer le disque d de *piquet1* à *piquet2* dans $P(d)$, il est nécessaire que les disques 1 à $d-1$ soient empilés sur le piquet 6 - *piquet1* - *piquet2*. On appelle *configuration intermédiaire* d'un sous-problème $P(d)$ une telle configuration. En ignorant le disque d, la configuration intermédiaire de $P(d)$ devient la configuration finale de $P(d-1)$. On définit deux tableaux $p_i[1 \dots n]$ et $p_f[1 \dots n]$ où :

- $p_i[d]$ indique la position du disque d dans la configuration initiale du sous-problème $P(d)$, qui n'est rien d'autre que la position de d dans la configuration initiale de $P(n)$;
- $p_f[d]$ indique la position du disque d dans la configuration finale du sous-problème $P(d)$; cette position dépend de $p_i[d+1]$ et $p_f[d+1]$.

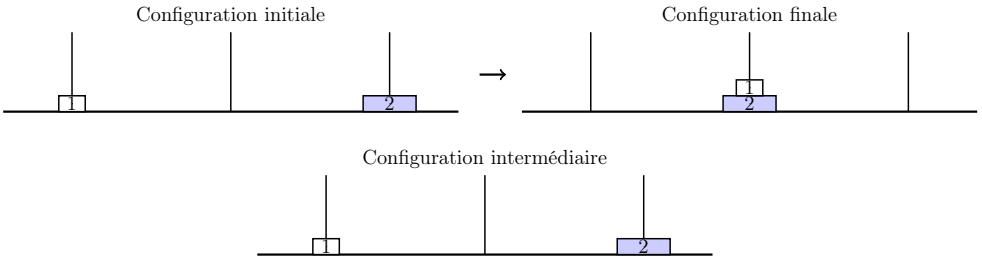
Les valeurs de $p_i[d]$ et $p_f[d]$ ($d \in \{1, \dots, 4\}$) pour l'exemple de la figure 4 sont indiquées dans le tableau 1. Donner la formule de récurrence qui permet de déterminer $p_f[d]$ en fonction de $p_i[d+1]$ et $p_f[d+1]$, en distinguant le cas $p_i[d+1] = p_f[d+1]$ du cas $p_i[d+1] \neq p_f[d+1]$. On indiquera également la valeur $p_f[n]$ d'initialisation de la récurrence.



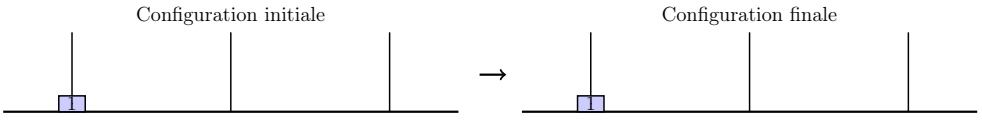
(a) Sous-problème $P(4)$



(b) Sous-problème $P(3)$



(c) Sous-problème $P(2)$



(d) Sous-problème $P(1)$

FIGURE 4 – Un exemple de décomposition en sous-problèmes pour 4 disques.

sous-problème	disque	position initiale	position finale
$P(4)$	$d = 4$	$p_i[4] = 1$	$p_f[4] = 3$
$P(3)$	$d = 3$	$p_i[3] = 2$	$p_f[3] = 2$
$P(2)$	$d = 2$	$p_i[2] = 3$	$p_f[2] = 2$
$P(1)$	$d = 1$	$p_i[1] = 1$	$p_f[1] = 1$

TABLE 1 – Valeurs de $p_i[d]$ et $p_f[d]$ pour l'exemple de la figure 4.

Question 14. Les opérations nécessaires pour résoudre intégralement le problème de la figure 4 sont indiquées sur la figure 5 : elles correspondent à la résolution incrémentale des sous-problèmes $P(d)$ pour d croissant de 1 à n. On remarque qu'aucune opération n'est nécessaire pour passer de la configuration finale de $P(2)$ à celle de $P(3)$ car les disques 1 à 3 sont déjà à la bonne position dans la configuration finale de $P(2)$. Si $p_i[d] = p_f[d]$, il n'y donc aucune opération à réaliser pour passer de la configuration finale de $P(d-1)$ à la configuration finale de $P(d)$. Si $p_i[d] \neq p_f[d]$, identifier les opérations à réaliser pour passer de la configuration finale de $P(d-1)$ à la configuration finale de $P(d)$. On fera en particulier appel à la procédure *Hanoi*.

Question 15. En utilisant les réponses aux questions 12 et 13, en déduire un algorithme de résolution (qu'on appellera *Hanoi2*) du problème des Tours de Hanoï à n disques depuis une configuration quelconque. Quelle est la complexité de *Hanoi2* ?

Question 16. Implémenter *Hanoi2*. À l'aide d'expérimentations numériques où on aura désactivé l'affichage de la solution (ou toute forme d'écriture dans un fichier), tracer la courbe du temps moyen d'exécution de l'algorithme implémenté en fonction du nombre n de disques. Pour ce faire, pour chaque valeur de n, on calculera le temps moyen d'exécution pour 50 configurations de départ tirées aléatoirement. Comparer la croissance de cette courbe avec la complexité théorique.

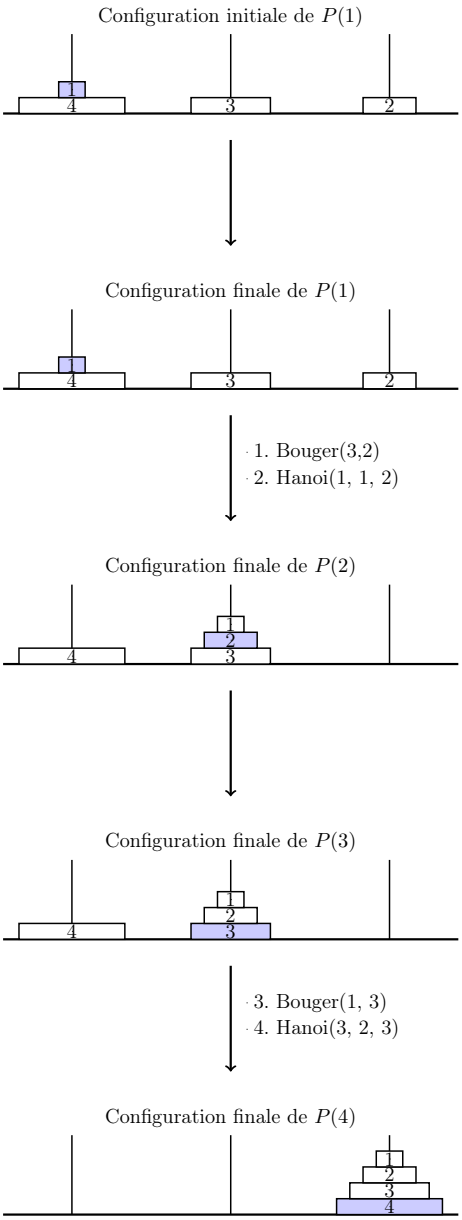


FIGURE 5 – Résolution de l'exemple.

3.4. Calcul du nombre de déplacements de disques

Constatant l'impossibilité de déterminer une séquence optimale de mouvements pour 64 disques (de par sa taille rédhibitoire), les explorateurs décident de concevoir un algorithme qui ne calculera pas la séquence optimale proprement dite, mais uniquement le nombre de déplacements de disques qu'elle comporte (ce qui est suffisant pour déterminer le temps minimum restant avant la fin du monde).

Question 17. Proposer une adaptation de l'algorithme *Hanoi2* qui accomplit cette tâche (on appellera *Hanoi3* cet algorithme). Quelle est la complexité de *Hanoi3* ?

Question 18. Les explorateurs observent la configuration suivante : $p_i[d] = 1$ pour $d = 1, \ldots, 10$, $p_i[11] = 3$, $p_i[d] = 2$ pour $d = 12, \ldots, 18$, $p_i[d] = 1$ pour $d = [19, \ldots, 40]$, $p_i[d] = 3$ pour $d = 41, \ldots, 63$, $p_i[d] = 2$ pour $d = 64$. En supposant qu'un déplacement de disque prend une seconde (les prêtres usent de la Force pour déplacer les disques) et qu'une année comporte 31 536 000 secondes, combien de temps au minimum reste-t-il avant la fin du monde si la légende dit vrai ?

4. Organisation et dates

Le travail est à effectuer par binôme du même groupe. Le choix du langage de programmation est libre.

Les projets doivent être rendus le **20 novembre 2013** au plus tard **par mail** à votre chargé de TD. Votre livraison sera constituée d'une archive `tar.gz` qui comportera les sources du programme, un fichier README détaillant comment compiler et exécuter le programme, et un rapport (un fichier au format `pdf`) avec les réponses aux différentes questions.

Le plan du rapport suivra le plan du sujet. Pour les questions théoriques, on fournira les preuves. Pour les questions portant sur l'implémentation, on décrira *brièvement* le principe des algorithmes implémentés, ainsi que les structures de données utilisées, et on citera les portions clés du code. Enfin, pour les expérimentations numériques demandées, on présentera les courbes obtenues et on les commentera.

Une **soutenance** est prévue lors de la semaine du 25 novembre 2013.