Les Tours de Hanoï
Année 2018 – 2019

Elèves de troisième : Goyhetche Manon, Maniet Mathys, Sourdeval Titouan

Elèves de seconde : Capdeville Ambre**,** Cassou-Leins Lilian, Garcia Eloïse, Delannoy Beverly, Tresmontan Maïlie

Encadrés par : Arriau Cathy, Barneix Chantal, Billard Marie, Goyhetche Alain

Établissements : Collège Gaston Fébus, Orthez / Lycée Gaston Fébus, Orthez

Chercheur : M Cresson Jacky, Université de Pau et des Pays de l'Adour.

1. Présentation du sujet

Durant cette année, lors des ateliers Maths en Jeans, un groupe de troisièmes et un de secondes ont cherché à résoudre le jeu de la tour de Hanoï.
Ce jeu est constitué de trois colonnes posées sur un socle. Une pyramide faite de disques, empilés du plus grand (base) au plus petit (sommet) est placée sur une colonne.
Le but du jeu est de déplacer la pyramide sur la colonne opposée en ne déplaçant qu'un disque à la fois. Il est interdit de poser un disque de diamètre plus grand sur un disque plus petit. (1)

1. Annonce des conjectures et résultats obtenus

Dans cet article, nous vous présenterons, dans un premier temps, la solution du groupe de troisièmes, qui après divers essais, ont dégagé une stratégie de résolution puis l'ont programmé à l'aide de Scratch. Puis, dans un second temps, nous vous présenterons la solution des secondes, programmée en Python basée sur un raisonnement récursif et leur démonstration déterminant le nombre optimal de coups en fonction du nombre de disques.

2. Texte de l'article :

Premières observations :

Méthode (premiers essais)

Dans un premier temps, nous nous sommes familiarisés avec le jeu et avons commencé à résoudre le problème avec peu de disques (3 puis 4). Nous tentions ensuite de déterminer le minimum de coups possibles. Afin d'essayer de trouver une stratégie, nous avons nommé les disques de 1 (le plus petit) à n (le plus grand) et nous avons appelé les colonnes $col1$ (colonne de départ / gauche) , $col2$ (colonne du milieu) et $col3$ (colonne d'arrivée / droite).
Nous avons commencé à noter les disques qui se déplacent et avons trouvé des règles de résolution.
Nous avons enfin essayé d'appliquer ces méthodes de résolution à des pyramides ayant plus de disques (5, 6 , etc.) et avons remarqué qu'elles fonctionnaient.

Nos règles :

- *Première règle* : ***La première règle concerne uniquement le premier coup du jeu.***

Si le nombre de disques avec lequel on joue **est impair**, alors le **disque 1** ira sur la colonne **de droite** ($col3$), **Sinon**, le **disque 1** ira sur la colonne **du milieu** ($col2$)

- *Seconde règle* :

Le **disque 1** ne va **jamais** sur un **disque impair**.

- *Troisième règle* :

Le **disque 1** va de **préférence** sur un **disque pair**, **sinon**, il ira sur la **colonne vide**.
Remarques : Cette règle ne pourrait s'appliquer s'il y avait deux disques pairs aux sommets ou deux colonnes vides. Le premier cas ne s'est jamais présenté (mais nous ne savons pas le démontrer), le second cas ne se présente qu'au début de la partie.

- *Quatrième règle*

Aucun disque ne joue deux fois de suite et le disque 1 joue une fois sur 2.
Remarques : La première partie de la règle permet de diminuer le nombre de coups. La seconde partie nous sera extrêmement utile pour la « symétrie » puis pour la programmation.

La symétrie :

Lors de nos recherches, nous avons noté les disques déplacés et nous avons remarqué des « symétries » :

Nombres de disques	**Disques déplacés**
2	1 2 1
3	1 2 1 3 1 2 1
4	1 2 1 3 1 2 1 4 1 2 1 3 1 2 1

Nous pouvons donc remarquer que pour chaque partie :

Le nom des disques déplacés forme un palindrome.(2)

Le plus grand disque est toujours situé à la médiane de la série.(3)

On peut facilement deviner la série suivante. Il suffit de copier la précédente, d'y ajouter à la fin le disque le plus grand de cette nouvelle partie, puis de copier à nouveau la série précédente.

Programmation :

Structure générale du programme (Scratch)

Nous avons tout d'abord matérialisé les colonnes par trois listes appelées $col1, col2, col3$. Après que le programme ait demandé au joueur le nombre de disques avec lequel il souhaite jouer, les numéros des disques s'afficheront dans la liste de gauche, $col1$, le disque 1 étant au sommet de la liste.

Afin de savoir où « déplacer le disque », nous avons créé une variable appelée niveau pour chaque colonne que nous adapterons à chaque mouvement. Si la colonne est vide, alors le niveau sera égal à $n+1$ où n est le nombre total de disques.

- La première règle du jeu étant utilisée une seule fois, nous avons créé un lutin qui lui est dédié (nommé lutin « 1er coup »).

Comme la quatrième règle dit que le disque 1 joue une fois sur deux, nous avons créé deux lutins :

- Le lutin lorsque le « 1 ne joue pas », il s'occupera des déplacements de tous les disques autres que 1.
- Le lutin lorsque le « 1 joue » qui s'occupera uniquement des mouvements du disque 1.

Premier Lutin  ***« 1er coup »***

Nous utilisons la fonction pour savoir si n(nombre total de disques) est pair. Si c'est le cas, le disque 1 ira sur la colonne du milieu ($col2¿$, sinon il ira sur la colonne de droite ($col3$)

A la fin de l'exécution des instructions, le lutin envoie un message au lutin « 1 ne joue pas ».

Second Lutin « 1 ne joue pas »

Lorsqu'il reçoit un message qui lui est destiné, le programme débute.

Le programme principal (annexe 1) vérifie en premier lieu si le jeu est terminé : Lorsque le disque 1 est au plus haut niveau de la colonne 3. Si c'est le cas, alors le jeu s'arrête. Sinon, le programme détermine où se trouve le disque 1.

Trois cas sont alors possibles :

- Si le disque 1 est au sommet de la colonne 1, alors le programme appellera le bloc « 2-3 » qui déterminera la colonne de départ (2 ou 3) et celle d'arrivée.
- Si le disque 1 est au sommet de la colonne 2, alors le programme appellera le bloc « 1-3 »
- Si le disque 1 est au sommet de la colonne 3, alors le programme appellera le bloc « 1-2 »

Explication d'un bloc : (annexe 2)

La première partie du bloc « 2-3 » (annexe 3) sert à déterminer si une des deux colonnes est vide. Dans ce cas, le programme regarde plus précisément laquelle des deux est vide pour y déplacer le disque du haut de l'autre colonne.

Sinon, la seconde partie du bloc « 2-3 » (annexe 4) entre en jeu et compare le plus haut disque de chaque colonne et déplacera le plus petit sur l'autre colonne.

Lorsque le disque est déplacé, le programme envoie un message au troisième lutin.

Troisième lutin « 1 joue »

Le programme principal de ce lutin regarde tout d'abord où se situe le disque 1 pour connaitre la colonne de départ et appelle ensuite des sous-programmes spécifiques à chaque position du disque 1 (blocs « $1n$ », « $2n$ », «$3n$ »)

Explication d'un bloc :

La première partie du bloc (annexe 5) sert à déterminer si l'élément placé au sommet d'une des deux colonnes est impair. Si c'est le cas, le programme regarde plus précisément laquelle afin de déplacer le disque 1 sur l'autre colonne. *Attention, il est impossible que les disques des sommets des trois colonnes soient impairs, mais nous n'avons pas su le démontrer.*

Sinon, la seconde partie du bloc (annexe 6) détermine où est la colonne vide ($niveau=n+1$) et déplacera le disque 1 sur l'autre colonne. *Attention, dans ce cas, il y aura une colonne vide et l'autre colonne aura à son sommet un disque pair, mais nous n'avons encore une fois pas su le démontrer.*

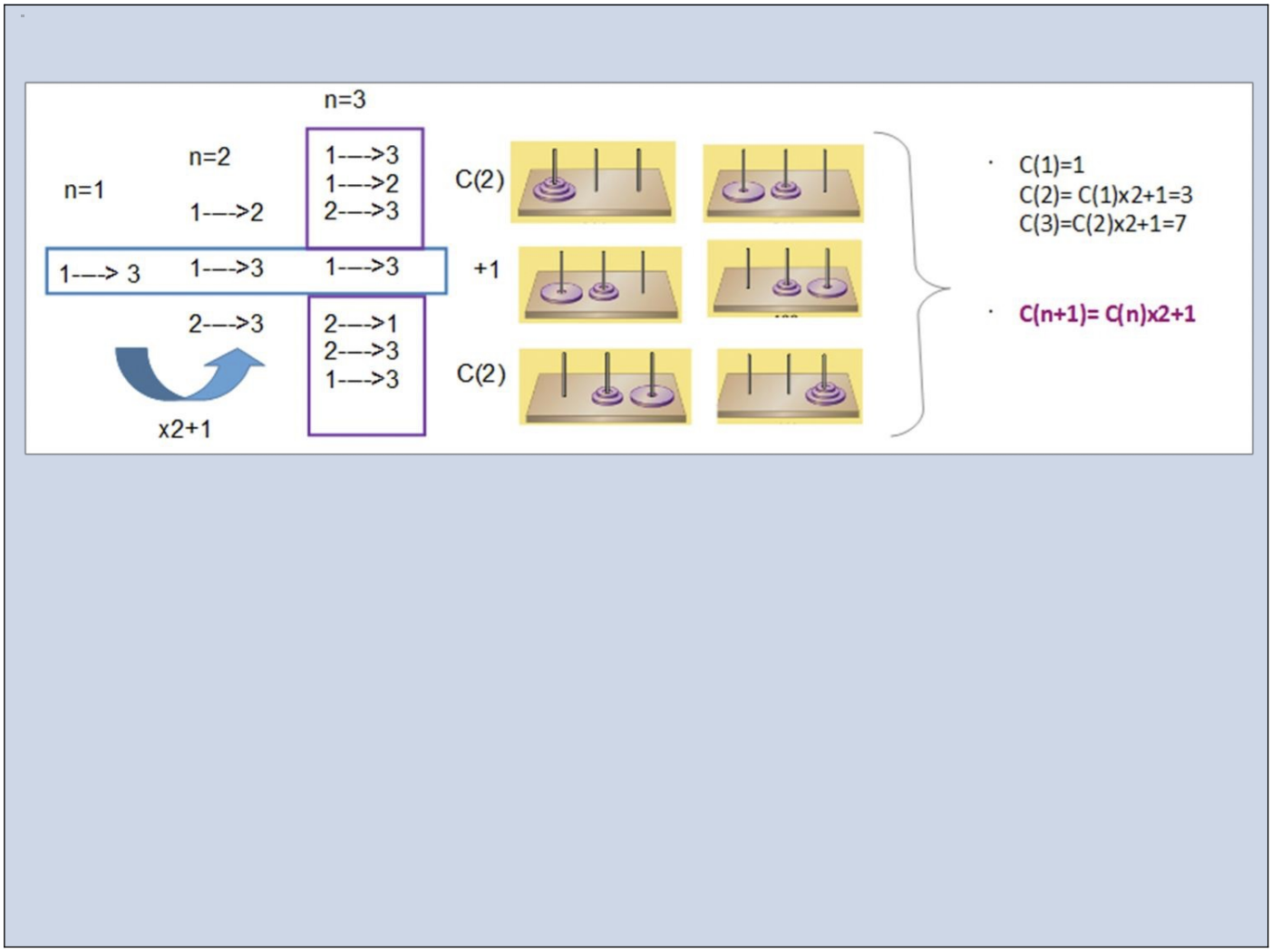
A la fin de ce programme, le lutin envoie un message au lutin « 1 ne joue pas ».

Vous pouvez tester le programme en utilisant ce lien : https://scratch.mit.edu/projects/317795918

- ***Solution des secondes***

Premières observations :

Après plusieurs parties, on a listé les coups des parties gagnantes pour un certain nombre de disques, noté « *n* », c'est grâce à ces listes que l'on a pu démontrer une formule permettant, à partir du nombre de coup optimal pour *n* disques, d'obtenir le nombre de coups optimal pour *n*+1 disques.



En multipliant le nombre de coups pour n disques par 2 et en lui ajoutant 1, on trouve le nombre de coups optimal pour n+1 disques. Ici, C(n) est le nom que l'on donne au nombre de coups optimal pour n disques.(4)

De ces listes on dégage également, un schéma qui se répète, dans chaque liste de coups, on remarque des blocs de trois coups tous séparés par un déplacement intermédiaire.

En nommant les tiges respectivement Départ, Intermédiaire et Arrivée, chaque bloc correspond aux déplacements suivants :

Départ ⟶ Intermédiaire
Départ ⟶ Arrivée
Intermédiaire ⟶ Arrivée

Nous retenons alors ce schéma sous le nom de « DIDAIA".

C'est en se basant sur ce schéma que l'on obtiendra au final le programme simplifié en Python.

Y a t-il un lien direct entre le nombre de disques et le nombre de coups optimal ? :

n	1	2	3	4	...	n	n+1
C(n)	1	3	7	15		C(n)	C(n+1)

En nous intéressant aux puissances de 2, nous avons pu dégager une deuxième formule : $C(n)=2^n-1$, elle permet de trouver le nombre de déplacements minimal pour n disques sans passer par le nombre de coups précédent. On peut démontrer cette formule grâce à un raisonnement par récurrence.

Pour illustrer en quoi consiste le raisonnement par récurrence on considère un escalier aux marches infinies. Les marches portent chacune un numéro entier allant de 1 à n ; le principe est alors le suivant :

- on montre que l'on sait monter sur la 1ère marche (portant le n°1)
- on imagine ensuite que l'on se trouve sur une marc e portant un numéro n
- on montre alors que l'on peut passer à la marche suivante portant le numéro $n + 1$ n° n

Ainsi donc, si on sait monter sur la 1ère marche et passer d'une marche quelconque à la suivante alors on pourra aller de la 1ère à la 2ième marche, de la 2ième à la 3ième et ainsi de suite et donc monter l'escalier.

Démonstration de : $C(n)=2^n-1$ (5)

Comme dit auparavant $C(n+1)$ disques correspond à $C(n+1)=C(n)\times 2+1$

- On vérifie que la formule $2^n - 1$ convient pour 1 disque, c'est-à-dire pour $n = 1$

  Pour $n = 1$ il y a 1 déplacement et $2^1 - 1 = 1$ Donc la formule convient pour 1 disque.

- On suppose qu'elle convient pour un nombre n de disques et donc que : $C(n) = 2^n - 1$

  Il faut montrer qu'elle reste vraie pour 1 disque de plus et donc que: $C(n+1) = 2^{n+1} - 1$

  On part de ce que l'on sait : $C(n+1)=C(n)\times 2+1$ et on remplace $C(n)$ par $2^n - 1$

  Ainsi: $C(n+1) = (2^n - 1)\times 2 + 1 = 2\times 2^n - 2 + 1 = 2^{n+1} - 1$

En résumé: $2^n - 1$ est juste pour 1 disque et fonctionne quand on passe de n disques à 1 de plus.
Donc on peut considérer qu'elle est juste pour autant de disques que l'on veut.

Vers un programme simplifié

Avant de créer le programme, on a travaillé sur les fonctions informatiques en langage Python.
On a commencé par des exercices du genre :

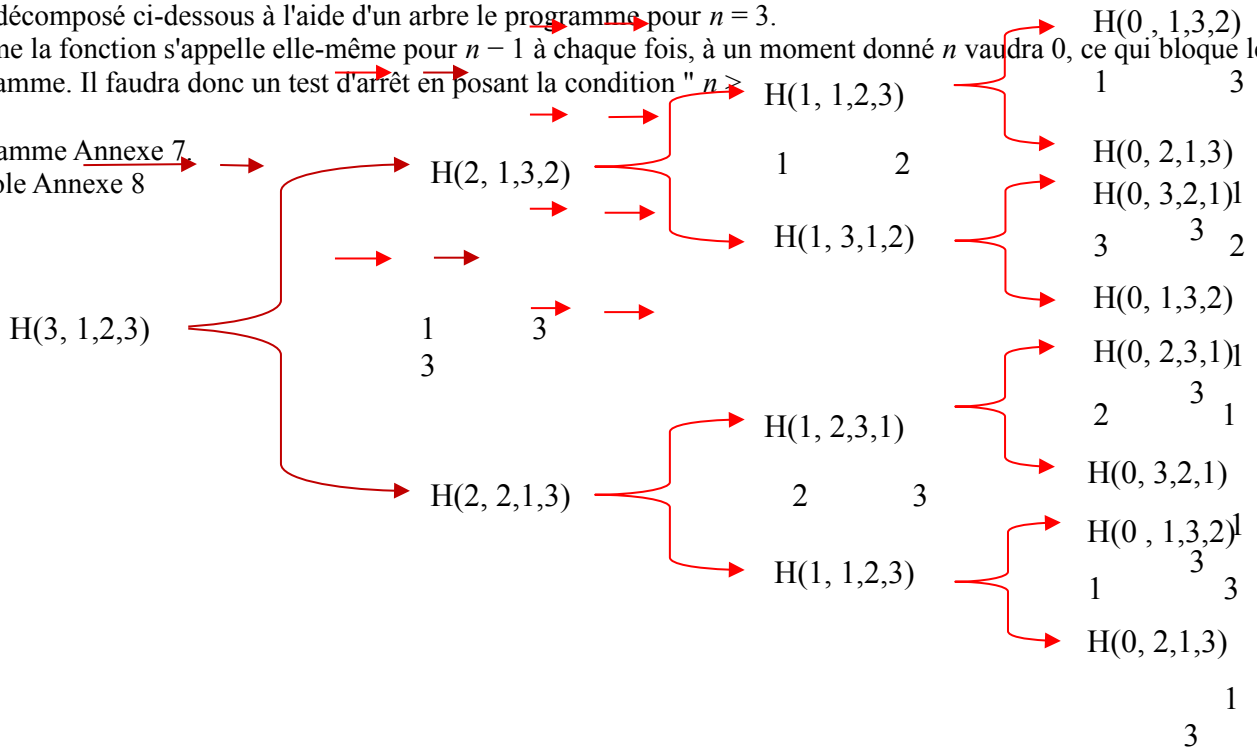
1) Déterminer si un triangle est rectangle ou non.
2) Calculer factorielle de n
3) Dire si un mot est un palindrome
4) Construction géométrique

On a établi un programme récursif qui a l'avantage d'être très court.

Pour cela on crée une fonction H avec 4 paramètres : n le nombre de disques, 1 pour la tige de départ, 2 pour la tige intermédiaire et 3 pour la tige d'arrivée. Soit : H(n , 1 , 2 , 3)
Déplacer n disques consiste à déplacer $n - 1$ disques de la tige de départ(1) sur la tige intermédiaire (2), puis le n-ième sur la tige d'arrivée, et amener les $n - 1$ de l'intermédiaire (2) à l'arrivée (3).

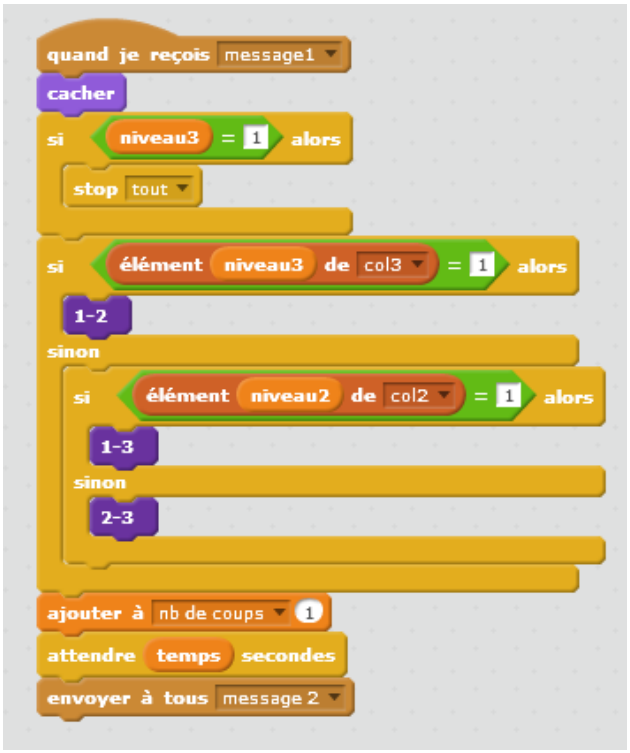
On a décomposé ci-dessous à l'aide d'un arbre le programme pour $n = 3$.
Comme la fonction s'appelle elle-même pour $n - 1$ à chaque fois, à un moment donné n vaudra 0, ce qui bloque le programme. Il faudra donc un test d'arrêt en posant la condition " $n \geq$

Programme Annexe 7
Console Annexe 8

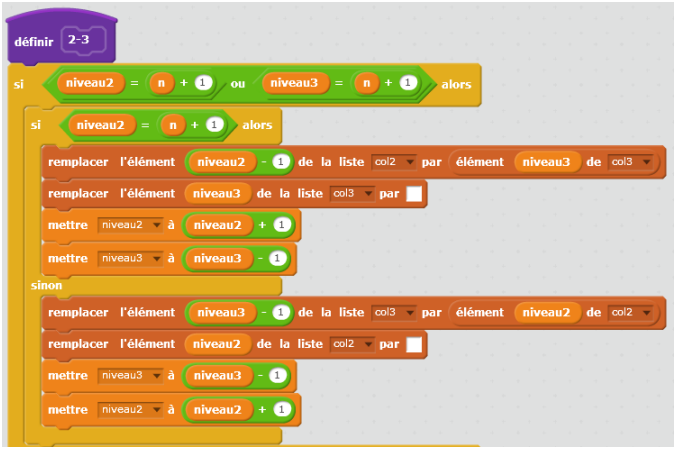


Annexe 1

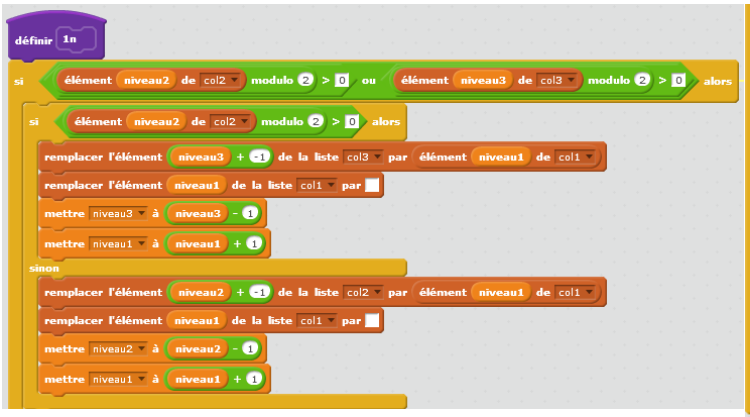
Annexe 2



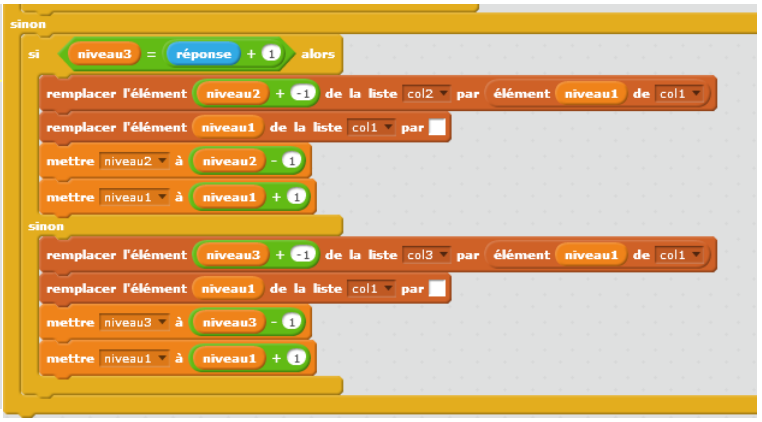
Annexe 3



Annexe 4



Annexe 5 :



Annexe 6

Annexe 7

```
1 def hanoi (n,a,b,c):
2     if n>0:
3         hanoi(n-1,a,c,b)
4         print(a,"--->",c)
5         hanoi(n-1,b,a,c)
6
7
```

Annexe 8

pour 3 disques:

```
>>> hanoi(3,1,2,3)
1 ---> 3
1 ---> 2
3 ---> 2
1 ---> 3
2 ---> 1
2 ---> 3
1 ---> 3
>>> hanoi(3,"gauche","milieu","droite")
gauche ---> droite
gauche ---> milieu
droite ---> milieu
gauche ---> droite
milieu ---> gauche
milieu ---> droite
gauche ---> droite
>>>
```

Notes d'éditions

(1) « colonne extrémale » deviendra « la colonne de gauche »

(2) Un palindrome est un ensemble de caractères dont l'ordre est le même qu'on le lise de droite à gauche ou de gauche à droite

(3) milieu de la série

(4) Le résultat annoncé, qui peut être résumé par C(n+1)=2 C(n)+1, n'est pas justifié. Pour déplacer n+1 disques, il faut déjà déplacer les n disques au-dessus et avoir une place vide pour le (n+1)ième disque (cela correspond à un minimum de C(n) déplacements), puis on déplace le plus grand disque en position finale (1 déplacement) et enfin on déplace à nouveau les n autres disques (au minimum C(n) déplacements).

(5)Dans la démonstration de : C(n) = 2^n-1, il faut ajouter « pour tout entier n strictement positif

NOTES

Note 1 (fin du 2e paragraphe)

Note 2 (section 3, dans la partie « La symétrie ») après « palindrome », ajouter une définition d'un palindrome :
« Un palindrome est un ensemble de caractères dont l'ordre est le même qu'on le lise de droite à gauche ou de gauche à droite »

Note 3 (section 3, dans la partie « La symétrie », 3e ligne) le mot « médiane » est mal utilisé, le bon mot est simplement milieu. Je propose de mettre en note uniquement « milieu ».

Note 4 (section 3 solution des secondes, premières observations), après l'image en haut de la page 4, première phrase : « Le résultat annoncé, qui peut être résumé par C(n+1)=2 C(n)+1, n'est pas justifié. Pour déplacer n+1 disques, il faut déjà déplacer les n disques au-dessus et avoir une place vide pour le n+1^e disque (cela correspond à un minimum de C(n) déplacements), puis on déplace le plus grand disque en position finale (1 déplacement) et enfin on déplace à nouveau les n autres disques (au minimum C(n) déplacements). »

Note 5 (en haut de la page 5) Dans la démonstration de : C(n) = 2^n-1, il faut ajouter « pour tout entier n strictement positif »