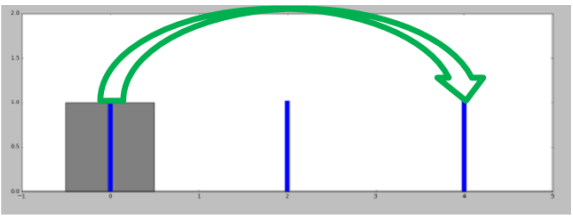
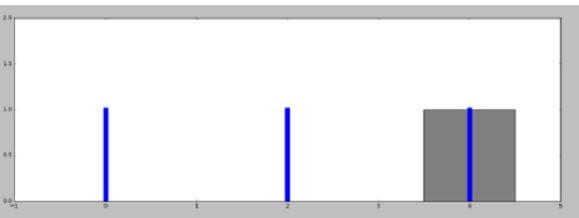
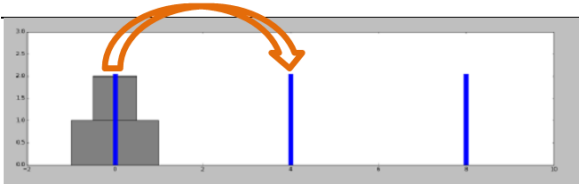
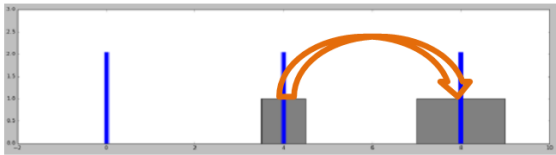
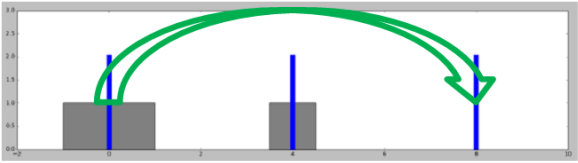

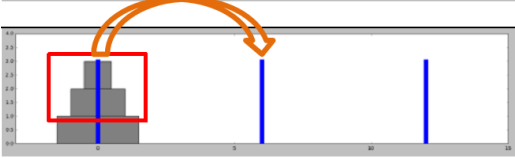
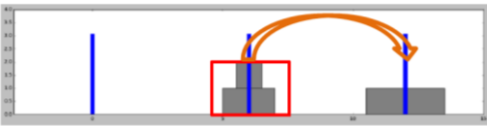
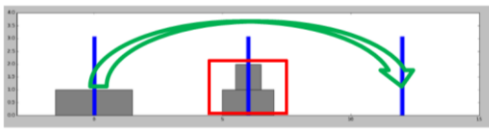
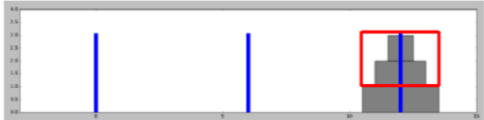
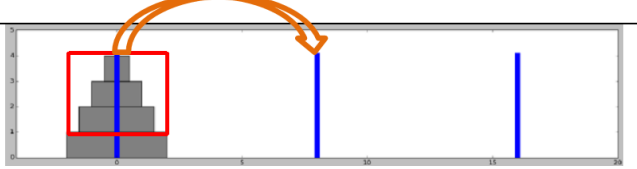
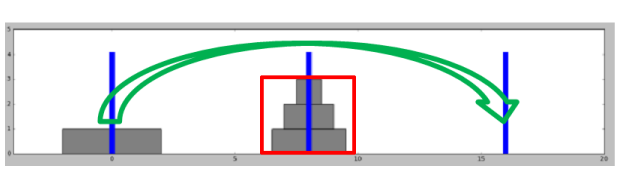
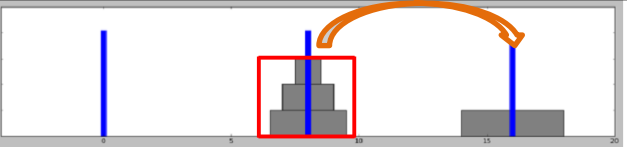
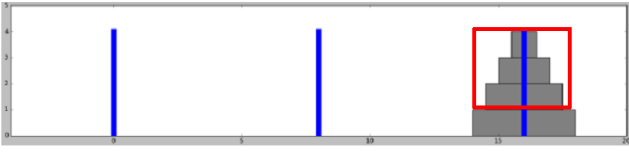


Remarque :

Ce TP fait suite au TP1 sur la manipulation des piles qui vous a conduit à résoudre le problème des tours de Hanoï à l’aide d’une méthode aléatoire. Nous allons maintenant voir qu’il est possible de résoudre ce problème avec le moins de coups possibles à l’aide de la récursivité vue en cours.

Le principe de la résolution récursive de ce problème est le suivant :

n	Principe de résolution	
1	<div>   </div> <p>→ Il suffit de déplacer l’étage de A sur C</p>	
2	<div>  <p>1. Déplacer l’étage supérieur de A sur B</p>  <p>3. Placer l’étage de B sur C</p> </div>	<div>  <p>2. Déplacer la base de la tour A en C</p>  <p>4. Tour de Hanoï déplacée</p> </div>
3	<div>  <p>1. Réussir à déplacer la petite tour à 2 étages sur B</p>  <p>3. Réussir à déplacer la petite tour à 2 étages de B sur C</p> </div>	<div>  <p>2. Déplacer la base de la tour A en C</p>  <p>4. Tour de Hanoï déplacée</p> </div>

n	Principe de résolution
4	<div>  <p>1. Réussir à déplacer la petite tour à 3 étages sur B</p> </div> <div>  <p>2. Déplacer la base de la tour A en C</p> </div> <div>  <p>3. Réussir à déplacer la petite tour à 3 étages de B sur C</p> </div> <div>  <p>4. Tour de Hanoï déplacée</p> </div>

On remarquera qu'à toute résolution des tours de Hanoï au rang n , il suffit de savoir déplacer une tour de Hanoï de taille $n-1$ sur la tour B au lieu de la déplacer sur la tour C (Hanoï au rang $n-1$).

Récupérer toutes les fonctions dont vous aurez besoins dans la correction du TP1, c'est-à-dire :

- `creer_pile()`
- `est_vide(p)`
- `empiler(p,x)`
- `depiler(p)`
- `creation_piles(n)`

Copier-coller dans votre code les fonctions disponibles dans le fichier « Affichage.py » afin d'afficher les tours de Hanoï.

Il suffit alors, à chaque fois que l'on veut afficher l'état des tours, de lancer : `f_animation(Piles)`

Remarque :

Pour que la solution du problème soit visible, une pause a été définie dans la fonction `f_animation(LP)`. A la dernière ligne est écrit : `plt.pause(0.5)`.

Plus l'argument est grand, plus la pause est longue. Vous pourrez ainsi jouer sur la rapidité d'exécution du code en changeant ce paramètre.

Exercice 1 : fonctions intermédiaires

Créer une fonction `deplacement(Piles,i,j)` qui déplace le sommet de la pile i vers la pile j , qui renvoie l'état de la variable **Piles** dans la console et qui affiche le résultat graphique.

Créer une fonction `intermediaire(i,j)` qui renvoie dans la liste $[0,1,2]$, le chiffre différent de i et j .

Exercice 2 : résolution récursive

Programmer une fonction `hanoi(n,dep,arr)` récursive qui, à tout rang n , déplace les n étages de la tour de départ vers la tour d'arrivée.

Au lancement de la **fonction** `hanoi(n,0,2)` le problème devrait être résolu quel que soit n .

Aide : Il suffit de compléter les lignes vides du code suivant, et le programme fonctionnera ;) Tout est dans votre réflexion !!!

```
def Hanoi(n, dep, arr):
    inter = intermediaire(dep, arr)
    if n == 1:
        ...
    else:
        ...
        ...
        ...
n = ...
Piles = creation_piles(n)
f_animation(Piles)
Hanoi(n, ...)
```

Si vous ne vous en sortez pas, il est possible de traiter par exemple les cas $n=1$, $n=2$, $n=3$, $n=4$ puis de voir que vous réalisez la même chose à chaque étape pour réduire ensuite la fonction.

Remarque : Vous pourrez essayer l'enchaînement affiché dans la console sur ce site :

<http://jeux.prise2tete.fr/tour-de-hanoi/tour-de-hanoi.php>