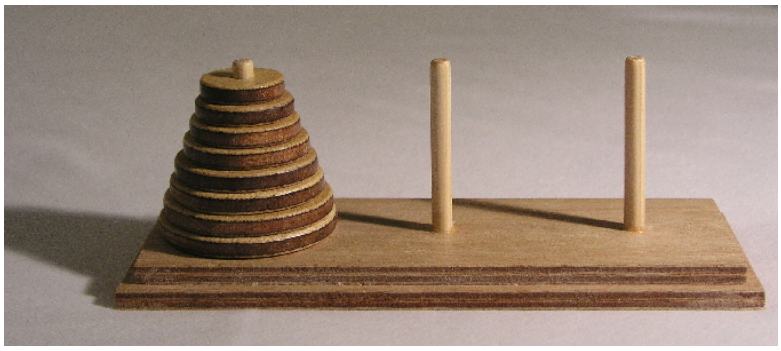
# Ejercicio 1

Torres de Hanoi: hay que mover N discos de la torre A a la torre B. Pero debemos mover con las siguientes restricciones:

1. Los discos se mueven de forma individual.
2. No puede haber un disco mas ancho sobre otro menos estrecho
3. Solo podemos mover el disco que se encuentra en la parte mas alta de la torre



Diseñar el algoritmo que permita resolver el problema.

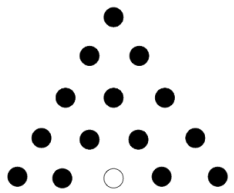
## Solución

Muchas soluciones en la web:

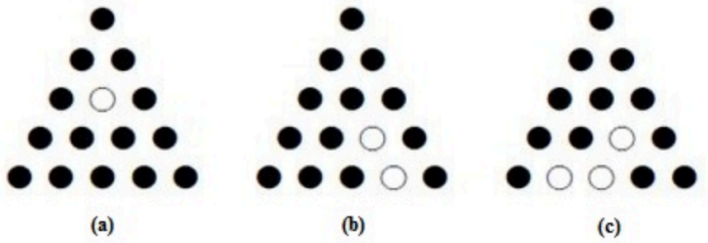
http://es.wikipedia.org/wiki/Torres_de_Hanói
http://puntocomnoesunlenguaje.blogspot.com.es/2012/04/torres-de-hanoi.html

# Ejercicio 2

Puzle de fichas: esto es un juego de puzle que se juega con un tablero de 15 posiciones distribuidas en un triangulo equilátero. Inicialmente hay fichas en todas las posiciones menos en una. Un movimiento debe hacerse haciendo saltar una ficha sobre otra contigua, de manera que la ficha que salta vaya a un hueco y la ficha saltada se retira y deja hueco. También queda un hueco en la posición que ocupaba la ficha que salta. El objetivo es dejar el menor número de fichas hasta que no se puedan mover mas fichas.



Pongamos un ejemplo de dos movimientos. Partiendo de (a) movemos la 4ª ficha de la fila 5ª, saltando sobre la 3ª ficha de la fila 4ª y la pasamos al hueco de la 2ª ficha de la 2ª fila, esto nos lleva a (b). De (b) pasamos a (c) saltando la ficha 2ª sobre la 3ª de la fila 5ª.




Diseñar el algoritmo que permita resolver el problema.

# Solución

Lo primero es tener claro la estructura de datos con la que representamos el problema. Lo representaremos con un array de dos dimensiones, en el que la primera fila tiene una columna, la segunda fila dos columnas, … Los índices de este array de dos dimensiones son:

```
                0,0
              1,1   1,0
            2,0   2,1   2,2
          3,0   3,1   3,2   3,3
        4,0   4,1   4,2   4,3   4,4
     5,0   5,1   5,2   5,3   5,4   5,5
6,0   6,1   6,2   6,3   6,4   6,5   6,6
```

Hay seis tipos de saltos posibles. Si suponemos la ficha de 4,2, la podremos mover a: 4,0 (caso A), 4,4 (caso B), 2,0 (caso C), 6,2 (caso D), 2,2 (caso E) y 6,4 (caso F). Cada caso varía sus coordenadas del origen al destino, y de la ficha saltada, de forma diferente. Esas variaciones, saltando desde la coordenada i,j son:

| Caso | destino | salto |
|---|---|---|
| caso A | i,j-2 | i,j-1 |
| caso B | i,j+2 | i,j+1 |
| caso C | i-2,j-2 | i-1,j-1 |
| caso D | i+2,j | i+1,j |
| caso E | i-2,j | i-1,j |
| caso F | i+2,j+2 | i+1,j+1 |

El algoritmo recursivo podría ser:

```
boolean resolver()
  saltos <- calcularPosiblesSaltos() // para todos los sitios donde hay ficha
                                     // ver si se puede hacer el salto A,B,C,D,E,F
  Si saltos está vacío
    huecos <- calcularHuecos
```

```
  Si huecos > maxHuecos
    return true // esta es la mejor solución encontrada hasta ahora
  sino
    return false // esta solución no es mejor que otra encontrada antes

Para todos los saltos de saltos
  deshacer los últimos saltos dados pero guardar el mejor
  dar este salto
  Si resolver() es falso
    deshacer el salto
    restaurar el que era mejor
  sino
    fijar este salto como mejor // maxHuecos=huecos
```