

Una de las utilidades de la recursividad es la búsqueda de todas las soluciones posibles, por ejemplo la búsqueda de todas las combinaciones de números.

## CALCULAR COMBINACIONES CON RECURSIVIDAD

Con el siguiente código puedo combinar con repeticiones elementos de una lista indicando de qué longitud quiero dichas combinaciones.

```
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

public class App{

    static List<String> combinar(int longitud, List<String> numeros){
        List<String> result= new ArrayList<>();
        if(longitud==0){
            return result;//resultado vacío
        }
        if(longitud==1){
            return numeros;
        }
        List<String> sublista= combinar(longitud-1,numeros);
        for(String numero:numeros){
            for(String numeroCombinado:sublista){
                result.add(numero+numeroCombinado);
            }
        }
        return result;
    }

    public static void main(String[] args) {
        List<String> numeros=Arrays.asList("0","1","2");

        for(int longitud=1;longitud<4;longitud++){
            System.out.println("combinaciones de 0, 1 y 2 con longitud "+longitud);
            List<String> result=combinar(longitud,numeros);
            for(String s:result){
                System.out.println(s);
            }
        }

        System.out.println("combinaciones de 8 y 5 con longitud 4");
        numeros=Arrays.asList("8","5");
        for(String s:combinar(4,numeros)){
            System.out.println(s);
        }
    }
}
```

Observando la salida que produce, puedo entender el funcionamiento del método recursivo.

combinaciones de 0, 1 y 2 con longitud 1

0

1

2

combinaciones de 0, 1 y 2 con longitud 2

00

01

02

10

11

12

20

21

22

combinaciones de 0, 1 y 2 con longitud 3

000

001

002

010

011

012

020

021

022

100

101

102

110

111

112

120

121

122

200

201

202

210

211

212

220

221

222

combinaciones de 8 y 5 con longitud 4

8888

8885

8858

8855

8588

8585

8558

8555

5888

5885

5858

5855

5588

5585

5558

5555

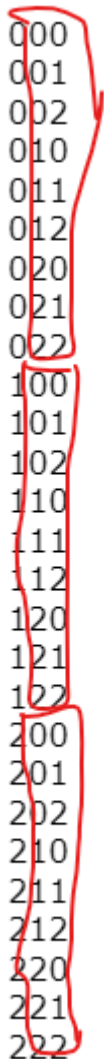
Aunque ya está a la vista el código java del método recursivo, nos esforzamos en buscar "la frasecita recursiva" donde el

*Combinar*  $m$  numeros en combinaciones de de longitud  $n$  consiste en:

*concatenar* cada uno de los  $m$  números con el resultado de *combinar* los  $m$  números con longitud  $-1$

Observa que, como ya sabemos, para hacer la definición recursiva, el término definido **combinar** se incluye en la propia.

por ejemplo para los números 0, 1, 2 observamos como las combinaciones de longitud 3 consiste efectivamente en concatenar el 0, 1 y 2 con las combinaciones de estos números pero de longitud 2



000  
001  
002  
010  
011  
012  
020  
021  
022  
100  
101  
102  
110  
111  
112  
120  
121  
122  
200  
201  
202  
210  
211  
212  
220  
221  
222

El caso base será cuando la longitud vale 1 que devuelve directamente la lista de números.

### **Ejercicio U6\_B6C\_1: Combinaciones sin recursividad.**

Normalmente, cuando el método recursivo en su interior tiene sólo un caso recursivo suele haber también una solución "fácil" con bucles, piensa por ejemplo en el ejemplo de factorial. En cambio, cuando existen varias llamadas recursivas, como por ejemplo en los típicos métodos de los árboles, la versión no recursiva es más difícil.

En este caso, podemos entonces aspirar a una solución con bucle. Se pide que escribas el método `combinar()`, qué es un método auxiliar que ayuda a nuestro método `combinaciones()` a resolver el problema. El método `combinaciones()` podría ser cómo sigue y queda como ejercicio escribir el método `combinar()`

```
import java.util.ArrayList;  
import java.util.Arrays;  
import java.util.List;
```

```

public class App {

    //falta método combinar, que es en este caso un método NO RECURSIVO

    static List<String> combinaciones(int longitud, List<String> numeros) {
        // en principio result almacena la combinación de longitud 1
        // es decir, es la copia de la lista numeros
        // List<String> result= numeros; MAL, así modificaré el original
        List<String> result = new ArrayList<>(numeros);
        for (int i = 0; i < longitud - 1; i++) { // longitud-1 porque ya inicializamos con 1 combinacion
            result = combinar(result, numeros);
        }
        return result;
    }

    public static void main(String[] args) {
        List<String> numeros = Arrays.asList("0", "1", "2");

        // suponemos longitud >0
        for (String combinacion : combinaciones(3, numeros))
            System.out.println(combinacion);
    }
}

```

### Ejercicio U6\_B6C\_2: De vuelta a la recursividad, combinaciones sin repetición.

Se trata ahora modificar el código recursivo anterior de tal forma que para el main anterior la salida sea la siguiente

```

combinaciones sin repetición de 0, 1 y 2 con longitud 1
0
1
2
combinaciones sin repetición de 0, 1 y 2 con longitud 2
01
02
10
12
20
21
combinaciones sin repetición de 0, 1 y 2 con longitud 3
012
021
102
120
201
210
combinaciones sin repetición de 8 y 5 con longitud 4

```

Observa el último resultado (para 8 y 5), no imprime nada ya que no es posible con sólo dos números y sin repetirlos, listas de longitud 4