Problema Suma 15

Suma 15

Consiste en Acomodar los números enteros del 1 al 9 sin repetir, uno en cada casilla, de tal manera que todas las filas, columnas y diagonales sumen 15.

Una solución a este problema sería to

4	9	2
3	5	7
8	1	6

El problema consiste entonces en determinar cuántas formas en total hay

Suma 15

Para resolver este problema, y en general para cualquiera, hay que preguntarse ¿Cuántas posibles soluciones hay?

R/. 9! = 362.880

Una vez determinado este asunto la siguiente pregunta es to ¿una solución mediante búsqueda exhaustiva es factible?

Si la respuesta es afirmativa, ¿cómo diseñar entonces un algoritmo?

Suma 15

Para resolver este problema, y en general para cualquiera, hay que preguntarse ¿Cuántas posibles soluciones hay?

R/. 9! = 362.880

Una vez determinado este asunto la siguiente pregunta es to ¿una solución mediante búsqueda exhaustiva es factible?

Si la respuesta es afirmativa, ¿cómo diseñar entonces un algoritmo?

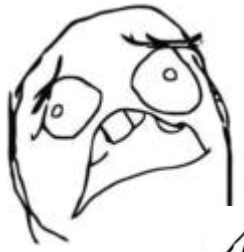
```
for h = 1 to 9 to:
     c_h = h
count = 0
for i1 = 0 to 8:
     for i2 = 0 to 8:
          for i3 = 0 to 8:
               …
                    for i9 = 0 to 8:
                         if valid(i) then process(c, i)
print count

valid(c):
     if i_j ≠ i_k para todo j, k con j ≠ k:
          return true
     else
          return false

process(c):
     if c_i1+ c_i2+ c_i3=15 and c_i4+ c_i5+ c_i6=15 and c_i7+ c_i8+ c_i9=15 and
        c_i1+ c_i4+ c_i7=15 and c_i2+ c_i5+ c_i8=15 and c_i3+ c_i6+ c_i9=15 and
        c_i1+ c_i5+ c_i9=15 and c_i3+ c_i5+ c_i7=15:
          count++
}
```

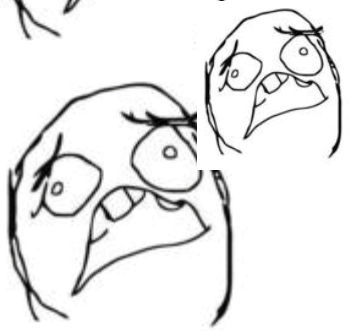
¿Cuál es el orden de complejidad de este algoritmo?

$O(9^9)$



¿Y si no se tratara de una matriz de 3x3 sino en general de *N*x*N*?

$O(N^{2N^2})$



¿Cómo procesar únicamente los 9! Candidatos y evitar el uso de la función *valid()*?

```
count = 0
for i = 1 to 9:
     c_i = i
do:
     if true:
          process(c)
while nextPermutation(c)

process(c):
     if c_i1+ c_i2+ c_i3=15 and c_i4+ c_i5+ c_i6=15 and ...
```

Permutaciones de un arreglo

En el anterior, así como en muchos otros problemas, se requiere encontrar todas las permutaciones de un determinado arreglo. Una alternativa en Java para ello es la siguiente:

```
void permute(int[] array, String res, int s) {
        if (s == array.length) {
            //Procesar la permutación almacenada en el String res
            //Ejemplo: System.out.println(res);
        } else {
            for (int i = 0; i < array.length; i++) {
                // visited es un arreglo de booleanos, definido globalmente, del mismo
                // tamaño del arreglo a permutar, e inicialmente con valores de false
                if (visited[i])
                     continue;
                visited[i] = true;
                permute(array, res + array[i] + " ", s+1);
                visited[i] = false;
            }
        }
}
```

Permutaciones de un arreglo

Mientras que en Python sería:

```
from itertools import permutations
array = [1, 2, 3, 4, 5]
p = permutations(array)
for c in p:
   process(c) #Por ejemplo print(c)
```

```
(1, 2, 3, 4, 5)
(1, 2, 3, 5, 4)
(1, 2, 4, 3, 5)
...
(5, 4, 3, 2, 1)
```