Contando singulares

Un número natural se dice *singular* cuando todos sus dígitos, excepto el más significativo (su primer dígito), son diferentes de dicho dígito más significativo. Por ejemplo, 56767 es singular, ya que 6767 no contiene el 5. Sin embargo, 1321 no es singular, ya que 321 contiene el 1.

Debemos diseñar un algoritmo recursivo eficiente que cuente la cantidad de números naturales *singulares* menores que un n dado ($n \geq 0$).

Trabajo a realizar

Debe diseñarse el algoritmo recursivo pedido, completando los apartados indicados entre comentarios en el archivo `plantilla.cpp` que se proporciona como apoyo. Debe implementarse, además, el algoritmo. El punto de entrada al mismo será la función `num_singulares_menoresque`. Si se considera necesario, deberá definirse e implementarse una generalización adecuada, y definir el algoritmo pedido como una inmersión de dicha generalización. Aparte de llevar a cabo el diseño del algoritmo, e implementar el mismo, deberás determinar justificadamente su complejidad, definiendo y resolviendo las recurrencias necesarias (la resolución se llevará a cabo utilizando los patrones de recurrencias genéricas discutidos en clase).

Programa de prueba

Se proporcionan dos versiones alternativas (puede elegirse una u otra comentando o descomentando la definición `#define DOM_JUDGE` que se encuentra al comienzo del archivo):

- Si DOM_JUDGE está definido, el programa lee por la entrada estándar enteros representables como valores del tipo `long long` (enteros que utilizan, como mínimo, 64 bits). Todos los enteros leídos serán no negativos, excepto el último, que será -1 y marcará el final de los casos de prueba. Cada vez que lee un caso de prueba, el programa invoca a `num_singulares_menoresque` e imprime el resultado. A continuación, se muestra algunos ejemplos de entrada / salida:

Entrada	Salida
8 46 5728 193451785320 -1	8 42 4247 64121611726

- Si DOM_JUDGE no está definido, el programa genera aleatoriamente 1000 números y compara el resultado del algoritmo con una implementación *naif* del mismo. Este modo de ejecución es útil para someter al algoritmo a una prueba de estrés. Si se genera un número para el cuál la implementación recursiva y la implementación *naif* discrepan, el programa termina y muestra el valor para el que el algoritmo falla. Para activar este modo basta comentar la línea `#define DOM_JUDGE`

El archivo completo debe entregarse a través del juez en línea de la asignatura.