**EJERCICIO 1**

Hay varias secuencias de incremento conocidas, entre las cuales se encuentran:

**Secuencia de Tokuda:**

* Fórmula: ceiling( (9 \* (9/4)^n – 4) / 5)
* Ejemplo de secuencia: 1, 4, 9, 20, 46, 103, 233

**Secuencia de Hibbard:**

* Fórmula: 2^k - 1
* Ejemplo de secuencia: 1, 3, 7, 15, 31

**Secuencia de Papernov y Stasevich:**

* Fórmula: 2^k + 1
* Ejemplo de secuencia: 1, 3, 5, 9, 17, 33, 65

**EJERCICIO 2**

Seleccionar el algoritmo adecuado es crucial para reducir la complejidad temporal del Quicksort. Hay diferentes métodos para elegir el pivote:

1. Elegir un número aleatorio del conjunto con una complejidad de O(1).
2. Tomar tres elementos aleatorios y seleccionar el valor medio.

Ambos métodos pueden tener un peor tiempo de O(n^2) dependiendo del orden de la secuencia, siendo el primer método más propenso a ello.

1. Recorrer todo el arreglo para hallar el pivote ideal con una complejidad de O(n), asegurando que Quicksort tenga una complejidad de O(n\*logn), aunque esto implica cálculos adicionales.

Las librerías de lenguajes modernos suelen tomar ambos extremos del arreglo y elegir el mayor como pivote.

**EJERCICIO 3**

Primero, se debe ordenar ambos conjuntos. El algoritmo es similar al Quicksort. Consiste en dos bucles while: el bucle exterior recorre el conjunto mayor y el bucle interior recorre el conjunto menor. Mientras el número en el bucle interior sea menor que el del conjunto mayor, se itera. Si la condición no se cumple, se verifica con un if si los valores son iguales. Si no lo son, el bucle mayor itera de nuevo, y el bucle interior comienza donde quedó la iteración anterior. Si el bucle menor termina, los conjuntos son disjuntos. Si en la verificación del if los valores son iguales, se hace un break, indicando que no son disjuntos. La complejidad del algoritmo es O(m\*n) en el peor caso si los conjuntos son disjuntos. En el mejor de los casos, al no ser disjuntos, el tiempo será menor.