Análisis de Complejidad de Heap Sort

Análisis detallado de la función de tiempo $T_p(n)$ para Heap Sort

$$T_p(n) = t_1 + t_2 + \sum_{i=1}^{n-1} \left(t_3 + t_4 + \sum_{j=0}^{\log i} (t_5 + t_6 + t_7) + t_8 \right) + t_3$$

$$T_p(n) = t_{123} + \sum_{i=1}^{n-1} \left(t_{3458} + \sum_{j=0}^{\log i} t_{567} \right)$$

$$T_p(n) = t_{123} + \sum_{i=1}^{n-1} t_{3458} + \sum_{i=1}^{n-1} \sum_{j=0}^{\log i} t_{567}$$

$$T_p(n) = t_{123} + (n-1)t_{3458} + \sum_{i=1}^{n-1} (\log i + 1) t_{567}$$

$$T_p(n) = t_{123} + (n-1)t_{3458} + t_{567} \sum_{i=1}^{n-1} (\log i + 1)$$

$$T_p(n) = t_{123} + (n-1)t_{3458} + t_{567} \left(\sum_{i=1}^{n-1} \log i + (n-1) \right)$$

$$T_p(n) = t_{123} + (n-1)t_{3458} + t_{567} \left(\log((n-1)!) + (n-1) \right)$$

Usando la aproximación de Stirling:

$$\log((n-1)!) \approx (n-1)\log(n-1) - (n-1)$$

Sustituyendo:

$$T_p(n) \approx t_{123} + (n-1)t_{3458} + t_{567} \cdot ((n-1)\log(n-1))$$

Forma cuadrática aproximada

$$T_p(n) = c_1 n \log n + c_2 n + c_3$$

Donde:

$$c_1 = t_{567}$$
$$c_2 = t_{3458}$$
$$c_3 = t_{123}$$

Conclusión

La complejidad temporal del algoritmo *Heap Sort* es:

$$\boxed{T(n) = \mathcal{O}(n \log n)}$$

Esta complejidad se mantiene igual en los tres casos: **mejor**, **promedio** y **peor**, ya que la estructura del heap obliga a realizar una operación de `heapify` de costo logarítmico tras cada extracción, sin importar el orden inicial de los elementos.