# Theoretical time complexity

## Merge sort

$T(1) = 0$

$$T(n) = 2T(\frac{n}{2}) + n \\ T(n) = 2[2T(\frac{n}{4}) + \frac{n}{2}] + n = 4T(\frac{n}{4}) + 2n \\ T(n) = 4[2T(\frac{n}{8}) + \frac{n}{4}] + 2n = 8T(\frac{n}{8}) + 3n \\ \vdots \\ T(n) = nT(\frac{n}{n}) + n\log n = nT(0) + n\log n \\ T(n) = n\log n$$

$\therefore \text{Constant} = 1$

To find $O(g(n))$,

$T(n) \leq cg(n);\ n > n_0$

$\therefore \text{Time complexity} = O(n\log n)$

## Quick sort

$T(1) = 0$

$T(n) = T(p) + T(n - p - 1) + n$

Average case:

$$T(n) = 2T(\frac{n}{2}) + n \\ T(n) = 2[2T(\frac{n}{4}) + \frac{n}{2}] + n = 4T(\frac{n}{4}) + 2n \\ T(n) = 4[2T(\frac{n}{8}) + \frac{n}{4}] + 2n = 8T(\frac{n}{8}) + 3n \\ \vdots \\ T(n) = nT(\frac{n}{n}) + n\log n = nT(0) + n\log n \\ T(n) = n\log n$$

$\therefore \text{Constant} = 1$

To find $O(g(n))$,

$T(n) \leq cg(n);\ n > n_0$

$\therefore$ Time complexity $= O(n \log n)$

## Heap sort

Heap sort uses heap insert and heap delete

Heap insert (1 element):

$T_1(0) = 0$

$$\begin{aligned} T_1(n) &= T_1(\tfrac{n}{2}) + 1 \\ T_1(n) &= T_1(\tfrac{n}{4}) + 2 \\ &\vdots \\ T_1(n) &= T_1(0) + \log n \\ T_1(n) &= \log n \end{aligned}$$

Heap delete (1 element):

$T_2(0) = 0$

$$\begin{aligned} T_2(n) &= T_2(\tfrac{n}{2}) + 1 \\ T_2(n) &= T_2(\tfrac{n}{4}) + 2 \\ &\vdots \\ T_2(n) &= T_2(0) + \log n \\ T_2(n) &= \log n \end{aligned}$$

Heap sort:

$$\begin{aligned} T(n) &= n \cdot T_1(n) + n \cdot T_2(n) \\ T(n) &= 2n \log n \end{aligned}$$

$\therefore$ Constant $= 2$

To find $O(g(n))$,

$T(n) \leq cg(n);\ n > n_0$

$\therefore$ Time complexity $= O(n \log n)$