Merge Sort Time Complexity Proof

`n`개의 정수에 대해 병합정렬하는데 걸리는 시간을 `T(n)` 이라고 하자.

이때, 병합정렬은

1. 좌측 절반 정렬 => $T(n/2)$
2. 우측 절반 정렬 => $T(n/2)$
3. 좌/우측 정렬결과 합치기(병합) => n

으로 구성되어 있다.

$$\begin{aligned} T(n) &= T(n/2) + T(n/2) + n \\ &= 2 \times T(n/2) + n \end{aligned}$$

절반을 정렬하는데 걸리는 시간을 아래 식으로 표현 가능

$$\begin{aligned} T(n/2) &= T(n/4) + T(n/4) + n/2 \\ &= 2 \times T(n/2^2) + n/2 \end{aligned}$$

이때, $T(n/2)$를 치환하여 $T(n)$을 다시 작성

$$\begin{aligned} T(n) &= 2 \times T(n/2) + n \\ &= 2 \times \{2 \times T(n/2^2) + n/2\} + n \\ &= 2^2 \times T(n/2^2) + 2n \end{aligned}$$

이번에는 절반의 절반을 정렬하는데 걸리는 시간을 식으로 표현 ($T(n/4) = T(n/2^2)$)

$$\begin{aligned} T(n/2^2) &= T(n/8) + T(n/8) + n/4 \\ &= 2 \times T(n/2^3) + n/2^2 \end{aligned}$$

한번더 $T(n/2^2)$를 치환하여 $T(n)$을 재작성

$$\begin{aligned} T(n) &= 2^2 \times T(n/2^2) + 2n \\ &= 2^2 \times \{2 \times T(n/2^3) + n/2^2\} + 2n \\ &= 2^3 \times T(n/2^3) + 3n \end{aligned}$$

재귀식

위와 같이 i번 재귀를 반복할 경우, 아래와 같이 표현 가능

$$T(n) = 2^i \times T(n/2^i) + in$$

병합정렬의 기저조건(base case)은 `n == 1`이다. 만약 i번째 재귀함수에서 기저 조건에 도달했다면, 식은 아래와 같다.

$$T(n) = 2^i \times T(1) + in$$

이때, i와 n의 관계를 아래와 같이 재정의 할 수 있다.

$$\begin{aligned} &n/2^i = 1 \\ &n = 2^i \\ &i = \log_2 n \end{aligned}$$

모든 i를 $log_2 n$으로 치환

$$T(n) = 2^{log_2 n} \times T(1) + nlog_2 n$$

기저조건인 $T(1)$의 시간복잡도는 1 이다.

$$T(1) = 1$$

정리

$$\begin{aligned} T(n) &= 2^{log_2 n} \times 1 + nlog_2 n \\ &= n + nlog_2 n \end{aligned}$$

시간복잡도

그러므로 병합정렬의 시간복잡도 big-O 는 $O(n + nlog_2 n) = O(nlog_2 n)$이다.