1 Worst-case analysis of Mergesort

Assume that the number of elements to be sorted

$$n = 2^k.$$

In such a case,

$$k = \log_2 n$$

and the recurrence relation (1) with the initial condition (2) is

$$W(n) = 2W(\frac{n}{2}) + (n-1) \tag{1}$$

$$W(1) = 0 \tag{2}$$

The most straightforward way of solving it is to guess its solutoin by unfolding the recurrence relation (1), and then by proving by induction that it is a solution, indeed.

$$W(n) = 2W(\frac{n}{2}) + (n-1) =$$

(substitute $\frac{n}{2}$ for n in the recurrence relation (1) and apply to $W(\frac{n}{2})$)

$$= 2(2W(\frac{\frac{n}{2}}{2}) + (\frac{n}{2} - 1)) + (n-1) =$$

$$= 4W(\frac{n}{4}) + (n-2) + (n-1) =$$

(substitute $\frac{n}{4}$ for n in the recurrence relation (1) and apply to $W(\frac{n}{4})$)

$$= 4(2W(\frac{n}{8}) + (\frac{n}{4} - 1)) + (n-2) + (n-1) =$$

$$= 8W(\frac{n}{8}) + (n-4) + (n-2) + (n-1) =$$

$$= 2^3W(\frac{n}{2^3}) + (n-2^2)) + (n-2^1) + (n-2^0) =$$

$$= ... =$$

(keep doing this until you get 2^k in front of $W()$)

$$= 2^kW(\frac{n}{2^k}) + (n-2^{k-1})) + (n-2^{k-2}) + ... + (n-2^0) =$$

(use $2^k = n$ and introduce $\sum$ notation)

$$= nW(1) + \sum_{i=0}^{k-1}(n - 2^i) =$$

(use $W(1) = 0$ and split the sum)

$$= n0 + (nk - \sum_{i=0}^{k-1} 2^i) =$$

$$= nk - 2^k + 1 =$$

(use $k = \log_2 n$)

$$= n \log_2 n - n + 1.$$

Now, we shall prove that function

$$w(n) = n \log_2 n - n + 1$$

is the solution of the original recurrence relation (1) with initial condition (2), indeed.

Let's first rewrite function w using $n = 2^k$. We have:

$$w(2^k) = 2^k \log_2 2^k - 2^k + 1 = 2^k k - 2^k + 1 = 2^k(k-1) + 1,$$

that is,

$$w(2^k) = 2^k(k-1) + 1. \tag{3}$$

In particular, substututing $k-1$ for k,

$$w(2^{k-1}) = 2^{k-1}(k-2) + 1. \tag{4}$$

Let us verify that the function w satisfies (1) for every $n = 2^k$, that is,

$$w(2^k) = 2w(\frac{2^k}{2}) + (2^k - 1). \tag{5}$$

The left-hand side of (5) is:

$$L = w(2^k) =$$

(by (3))

$$= 2^k(k-1) + 1.$$

The right-hand side of (5) is:

$$R = 2w(\frac{2^k}{2}) + (2^k - 1) =$$

$$2w(2^{k-1}) + (2^k - 1) =$$

(by (4))

$$= 2(2^{k-1}(k-2) + 1) + (2^k - 1) =$$

$$= 2^k(k-2) + 2) + (2^k - 1) =$$

$$= 2^k(k-1) + 1.$$

So, $L = R$, and the (5) is satisfied.

Now, we verify that the function w defined by (3) satisfies (2) as well.

$$w(1) = w(2^0) = 2^0(0-1) + 1 = 0 - 1 + 1 = 0.$$

So, (2) is satisfied.

This completes the proof. □

Obviously, the solution is unique: you can write a recursive Java program that unambiguously computes $W(n)$ for each $n = 2^k, k = 0, 1, 2, ...$, since by (1),

$$W(2^{k+1}) = 2W(2^k) + 2^{k+1} - 1.$$

Exercise Do it!