Time complexity of merge sort

Krzysztof Bartoszek

October 7, 2010

Algorithm 1 merge_sort(list)

if length(list) == 1 then
    return list
else
    A = merge_sort(first half of list)
    B = merge_sort(second half of list)
    C = merge(A, B) return C
end if

We will analyze the time complexity of the above algorithm. Define by $a_n$ as the time needed to sort a list of $2^n$ elements. The time complexity of the algorithm can be described by the following recursion,

$$a_n = 2a_{n-1} + c_1 2^n$$

$$a_0 = c_0.$$

We need to solve this recursion to find an explicit dependence of the time on $n$ and we will do this via its generating function $A(x)$.

$$A(x) = \sum_{n=0}^{\infty} a_n x^n = c_0 + \sum_{n=1}^{\infty} (2a_{n-1} + c_1 2^n)x^n = c_0 + \sum_{n=1}^{\infty} 2a_{n-1}x^n + \sum_{n=1}^{\infty} c_1 2^n x^n = c_0 + 2x \sum_{n=0}^{\infty} a_n x^n + c_1 2x \frac{1}{1-2x} = c_0 + 2x A(x) + \frac{2c_1 x}{1-2x},$$

provided $|x| < 0.5$. This gives us that

$$(1 - 2x) A(x) = c_0 + \frac{2c_1 x}{1-2x} + \frac{2c_1 x}{(1-2x)^2} = \frac{c_0 - c_1}{1-2x} + \frac{c_1}{(1-2x)^2}.$$

Using the formulas given in the lecture for the generating functions of different sequences,

$$A(x) = \frac{c_0 - c_1}{1-2x} + \frac{c_1}{(1-2x)^2} = \left( c_0 - c_1 \right) \sum_{n=0}^{\infty} \binom{n+1}{1} (2x)^n = \sum_{n=0}^{\infty} (n+1)(2x)^n = \sum_{n=0}^{\infty} 2^n (c_0 + c_1 n) x^n.$$
We therefore have that the formula for the sequence is,

\[ a_n = (c_0 + c_1 n)2^n \approx c_1 n 2^n = O(n2^n). \]

Now let \( t_k \) be the time needed to sort \( k = 2^n \) elements,

\[ t_k = a_n = a_{\log_2 k} = c_1 k \log k = O(k \log k). \]

Now for a general \( k > 8 \) (we don’t want to worry about small \( k \)s which would cause problems in the argumentation below), let \( n_k := \min\{n > 3 : 2^{n-1} \leq k \leq 2^n\}, \) i.e. \( 2^{n_k-1} \leq k \leq 2^{n_k} \). We can bound the time complexity to sort a list of \( k \) elements by the time needed to sort \( 2^{n_k} \) elements which is \( O(2^{n_k} \log 2^{n_k}) \). Now we bound the time for \( k \) from the bottom and above,

\[
\begin{align*}
2^{n_k-1} \log 2^{n_k-1} &< k \log k < 2^{n_k} \log 2^{n_k} \\
2^{n_k-1} \log 2^{n_k-1} &< k \log k < 2^{n_k} \log 2^{n_k} < 2 \cdot 2^{n_k-1} \log 2^{n_k-1} \\
2^{n_k-1} \log 2^{n_k-1} &< k \log k < 2 \cdot 2^{n_k-1} \log 2^{n_k-1} < 4k \log k \in O(k \log k),
\end{align*}
\]

and as we are interested in getting complexity in terms of \( O(\cdot) \) we get that the complexity of the merge sort algorithm is \( O(k \log k) \) (we assumed \( k > 8 \) but we don’t worry about small \( k \)).