MergeSort

Manoj Gupta

September 11, 2016

Consider the following algorithm which merges two sorted array A_1 and A_2 of size n_1 and n_2. For this, we maintain two counters c_1 and c_2 which is initialized to 1. We move through these two array in increasing order picking up the minimum element at each step.

make a new array A of size n_1 and n_2;
$c_1 \leftarrow 1$;
$c_2 \leftarrow 1$;
$c \leftarrow 1$;
while $c_1 \leq n_1$ *and* $c_2 \leq n_2$ **do**
    if $A_1[c_1] \leq A_2[c_2]$ **then**
        $A[c] \leftarrow A_1[c_1]$;
        $c \leftarrow c + 1$;
        $c_1 \leftarrow c_1 + 1$;
    end
    else
        $A[c] \leftarrow A_2[c_2]$;
        $c \leftarrow c + 1$;
        $c_2 \leftarrow c_2 + 1$;
    end
end
while $c_1 \leq n_1$ **do**
    $A[c] \leftarrow A_1[c_1]$;
    $c \leftarrow c + 1$;
    $c_1 \leftarrow c_1 + 1$;
end
while $c_2 \leq n_2$ **do**
    $A[c] \leftarrow A_2[c_2]$;
    $c \leftarrow c + 1$;
    $c_2 \leftarrow c_2 + 1$;
end

Figure 1: MERGE(A_1, A_2) :Merging two sorted arrays

We now design an algorithm that uses the above merge algorithm to sort an array. This is a recursive algorithm which firsts partitions the array into two equal parts and invoke mergesort on these two sub-array. Assume that mergesort sorts these sub-arrays, then using our procedure MERGE$(\cdot, \cdot)$, we can merge these two sorted sub-arrays to get a single sorted array of size n.

if $n = 1$ **then**
    return $A[1]$;
end
else
    $A_1 \leftarrow$ MERGESORT$(A[1 \ldots n/2])$;
    $A_2 \leftarrow$ MERGESORT$(A[n/2 + 1 \ldots n])$;
    $A_3 \leftarrow$ MERGE(A_1, A_2);
    return A_3;
end

Figure 2: MERGESORT$(A[1 \ldots n])$

We now see the run of our MERGESORT$(A[1 \ldots n])$.

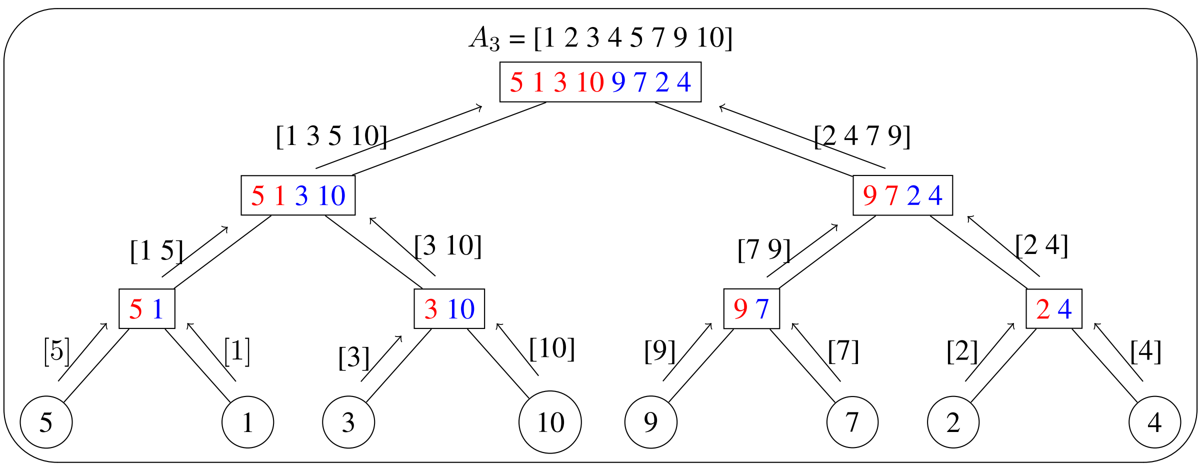

Figure 3: The run of MergeSort on the array $A = [5, 1, 3, 10, 9, 7, 2, 4]$

1 Proof of Correctness

We first prove that MergeSort sort n elements. To this end, we first prove the following simple lemma:

Lemma 1.1. *Given two sorted arrays A_1 and A_2 of size n_1 and n_2 respectively, MergeSort(A_1, A_2) correctly merges the two array to find output a single sorted array.*

Proof. The proof is by induction. Assume that MergeSort(A_1, A_2) correctly sorts two array if $n_1 + n_2 \leq 2$. There are three sub cases in the base case: (1) If all the elements are in array A_1, then the second while loop in MERGE will sort the array. (2) Similarly if all the elements are in A_2, then the third while loop will sort all the elements. (3) Else each array has one element each and MERGE will choose the minimum in the first while loop and put it in the auxillary array A_3, and then it will put the second element, thus completing the merge correctly.

Assume that MergeSort(A_1, A_2) correctly merges two array of size $n_1 + n_2 - 1$. We will now prove that MergeSort(A_1, A_2) merges array of size $n_1 + n_2$ respectively.

Consider the first comparison in the mergesort when the first element of A_1, say a is compared with the first element of A_2, say b. Without loss of generality, assume that $a < b$. Then a is the first element in the auxiliary array. Note that a is also the minimum element in A_1 and A_2. So, a has found it correct place in

A_3. Now we increase the counter for A, c_1 by 1. Thus we now have two arrays, A_1 whose effective size is $n_1 - 1$ and A_2 whose size if n_2. By induction, MERGE will correctly merge these array and put all the elements after a. So, MERGE(A_1, A_2) correctly merges array A_1 and A_2. $\square$

Lemma 1.2. MERGESORT(A) *sorts array A of size n.*

Proof. Again we will prove by induction on n. For the base case, assume that $n = 1$, thus A is trivially sorted. Using strong induction hypothesis, assume that MERGESORT(A) sorts an array of size $1, 2, \ldots, n-1$. Consider MERGESORT$(\cdot)$ on an array of size n. The procedure first divides the array into two parts of size $\approx n/2$. Using induction hypothesis, we claim that MERGESORT$(\cdot)$ will sort these two array of size $n/2$. And using Lemma 1.1, we claim that MERGE$(\cdot, \cdot)$ will correctly merge these two sorted arrays to give a single sorted array. $\square$

2 Running Time

We calculate the running time of MERGESORT using recurrence relation. One can show that the running time of MERGE is $O(n)$ if the cumulative size of two arrays is n. If $T(n)$ is the time taken by mergesort to sort an array of size n, then we can write $T(n)$ as follows:

$$T(n) = T(n/2) + T(n/2) + cn$$
$$T(2) = c$$

Note that MERGESORT first divides the array into two parts of roughly equal size. The time taken by MERGESORT to sort these two parts is $T(n/2)$. This is because of our assumption that the time taken by MERGESORT to sort an array of size n is $T(n)$. We will also make assumption to make our calculations simpler. Specifically, we will assume that n is a power of 2. This is mean that $n/2^i$ is always an interger (till $i = \log n$). The last equation equation above shows the time taken to merge the two sorted array. We will now solve this recurrence relation using a method called **repeated substitution**. In this method, we will repeatedly substitute the value of $T()$.

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

It is now easy to see the k^{th} substitution will lead to the following equality: $T(n) = 2^kT(n/2^k) + kcn$. However, this process cannot go on. We know that when each array is of size 1, $T(2) = c$. Thus, $n/2^k$ will be 2, when $n = 2^{k+1}$ or $k = \log n - 1$. Thus $T(n) = 2^{\log n - 1}T(2) + \log n(cn) = cn/2 + cn \log n = O(n \log n)$. Thus, the running time of MERGESORT is $O(n \log n)$.

There is one more simple method to solve the above recurrence relation. In this method, we first guess the answer to the recurrence relation and then prove it by induction. Since, we already know that $T(n) = O(n \log n)$, there is nothing to guess here. So let us assume that $T(n) \leq cn \log n$.

Proof by Induction

For base case, $T(2) \leq c2 \log 2 = 2c$ which is greater than c. Using strong induction hypothesis, assume that $T(i) \leq ci \log i$ for $i = 1, 2, \ldots, n-1$. We will now prove that $T(n) \leq cn \log n$. To this end, we will use the recurrence relation and apply induction hypothesis on $T(n/2)$. We know that $T(n) = 2T(n/2) + cn \leq 2c(n/2)\log(n/2) + cn = cn(\log n - 1) + cn = cn \log n$ $\square$.

We also note that MERGESORT will always take a time of $O(n \log n)$ on any array independent of the contents of the array. This is due to the fact that MERGE always takes $O(n)$ time to merger two arrays whose cumulative size is $O(n)$.