MCAC 302: Design and Analysis of Algorithms

Neelima Gupta

ngupta@cs.du.ac.in

January 12, 2021

Replace = with leftarrow

Input: Arrays A and B of size n and m respectively
Output: Merged Sorted Array C

$i = 1,\ j = 1,\ k = 1;$
while $i \leq n$ && $j \leq m$ **do**
  if $L[i] \leq R[j]$ **then**
    $A[k] = L[i];\ i = i + 1;$
  else
    $A[k] = R[j];\ j = j + 1;;$
  end
  $k = k + 1$
end
while $i \leq n$ **do**
  $A[k] = L[i];\ i = i + 1;\ k = k + 1;$
end
while $j \leq m$ **do**
  $A[k] = L[j];\ j = j + 1;\ k = k + 1;$
end

Algorithm 1: Merge(A, B, C)

Time Complexity of Merge

For every element copied in the output array, we spend constant amount to time. Thus total time is $O(n + m)$: the size of the array.
best case #comparisons =?

Time Complexity of Merge

For every element copied in the output array, we spend constant amount to time. Thus total time is $O(n+m)$: the size of the array.
best case #comparisons =? $\min\{n, m\}$
worst case #comparisons =?

Time Complexity of Merge

For every element copied in the output array, we spend constant amount to time. Thus total time is $O(n+m)$: the size of the array.
best case #comparisons =? $\min\{n, m\}$
worst case #comparisons =? $n+m$

Time Complexity of Merge

For every element copied in the output array, we spend constant amount to time. Thus total time is $O(n+m)$: the size of the array.
best case #comparisons =? $\min\{n, m\}$
worst case #comparisons =? $n+m$
best case time?

Time Complexity of Merge

For every element copied in the output array, we spend constant amount to time. Thus total time is $O(n+m)$: the size of the array.
best case #comparisons =? $\min\{n, m\}$
worst case #comparisons =? $n + m$
best case time?
worst case time?

Time Complexity of Merge Sort

Let $T(n)$ be the time to sort n elements using merge sort then,

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

$$= O(n \log n)$$

Space Complexity of Merge Sort

Is it in-place?

Space Complexity of Merge Sort

Is it in-place?
How much space?