

# মার্জ সর্ট (Merge Sort)

অ্যালগরিদম : মার্জ সর্ট

ইনপুট : একটি অ্যারে A।

আউটপুট : সবগুলো ধাপের শেষে A অ্যারেটি সর্ট করা অবস্থায় থাকবে।

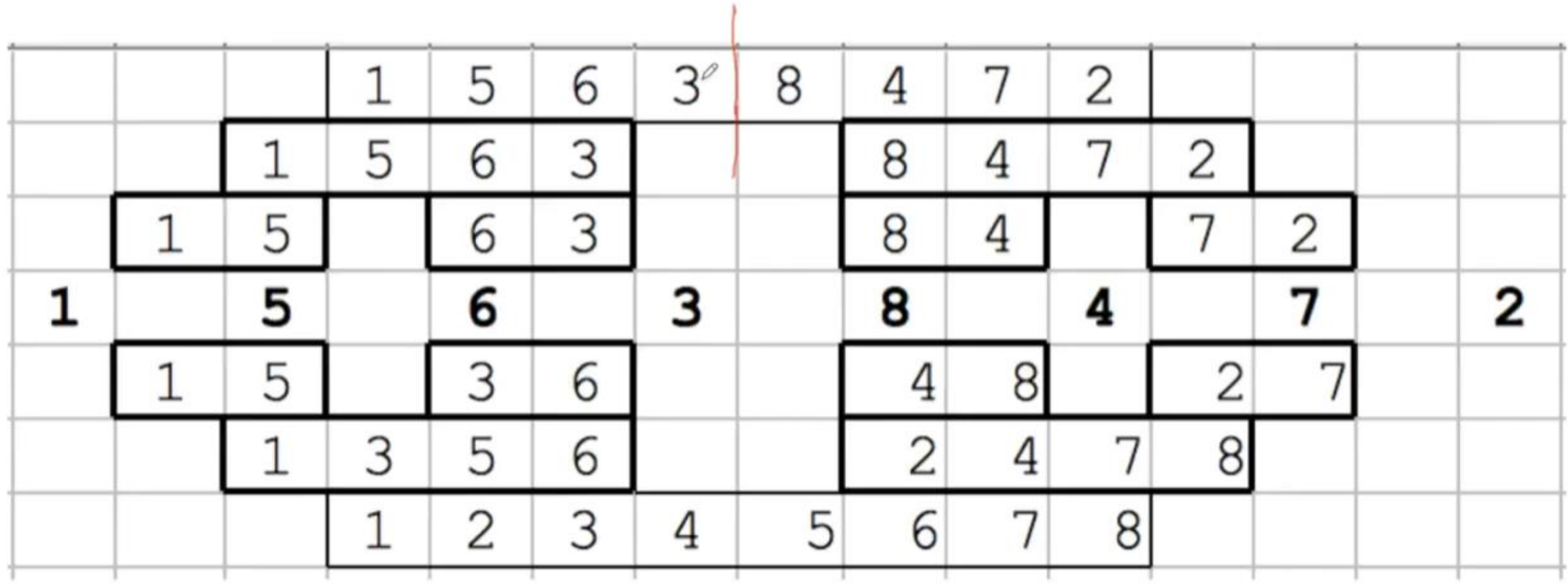
ধাপ 1: অ্যারেতে যদি একটি বা তার চেয়ে কম উপাদান থাকে, তাহলে সেই অ্যারেটি ইতিমধ্যে সর্ট করা।

ধাপ 2: A অ্যারের প্রথম অর্ধেক উপাদান left নামক অ্যারেতে রাখি। left অ্যারেকে মার্জ সর্ট করি।

ধাপ 3: A অ্যারের দ্বিতীয় অর্ধেক উপাদান right নামক অ্যারেতে রাখি। right অ্যারেকে মার্জ সর্ট করি।

ধাপ 4: left অ্যারে ও right অ্যারে মার্জ করি।

# মার্জ সর্ট (Merge Sort)



# মার্জ সর্ট (Merge Sort)-এর কমপ্লেক্সিটি

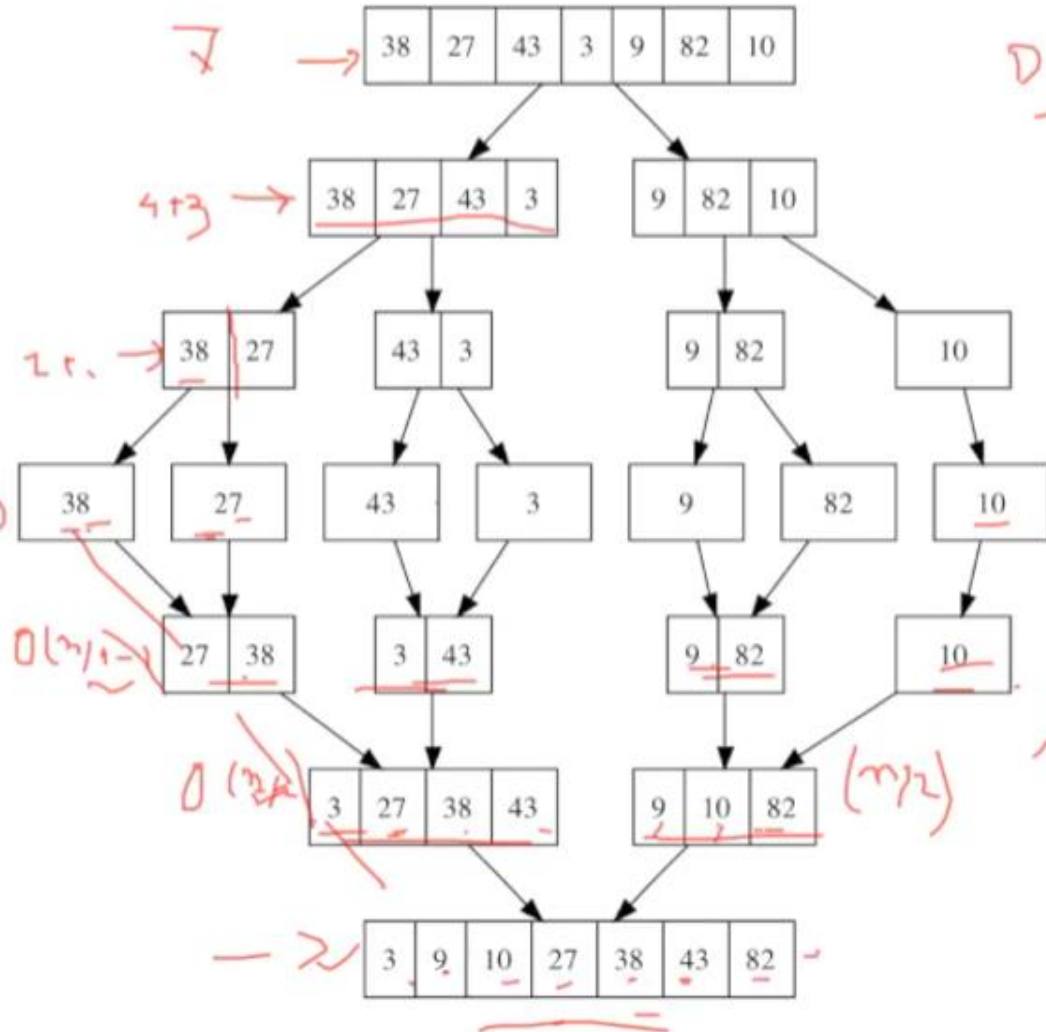


DIVIDE AND CONQUER

space  $\rightarrow$  100  
 $L, R$   
 $50, 50$   
 $O(n)$   
 $O(\frac{n}{2} + \frac{n}{2})$

$O(n) \times \log_2 n$   
 $O(n \log_2 n)$

$2^3 = 8$   $\log_2 8 = 3$   
 $T(n) = 2 \times T(n/2) + n$



# মার্জ সর্ট (Merge Sort)



```
void merge_sort(int A[], int left, int right)
{
    if (left >= right) {
        return;
    }

    int mid = left + (right - left) / 2;

    merge_sort(A, left, mid);
    merge_sort(A, mid+1, right);

    merge(A, left, mid, right);
}
```

Handwritten annotations in red:

- Below the parameter `A[]`, there is a handwritten `A[] = {5}` with an arrow pointing to the `5`.
- Below the `return;` statement, there is a handwritten `0`.
- Below the `mid` calculation, there is a handwritten `0. 0` with arrows pointing to the `0` and `0` in the expression.
- At the top, there are handwritten `0` and `n-1` with arrows pointing to the `left` and `right` parameters respectively.