Merge Sort

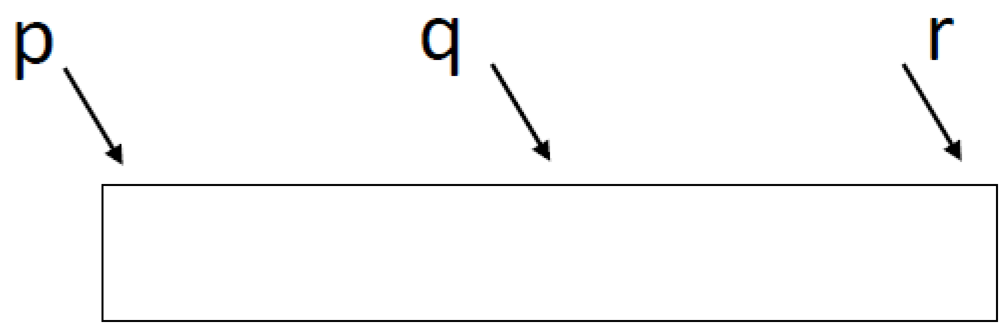
Some Slides from Jerome (MacGill University) and Shang-Hua Teng

Students are required to read from Book too

Dr Muhammad Atif Tahir

Merge Sort

```
MERGE-SORT(A,p,r)
   if p < r then
        q=(p+r)/2
        MERGE-SORT(A,p,q)
        MERGE-SORT(A,q+1,r)
        MERGE(A,p,q,r)
```



Starting by calling Merge-Sort(*A*,*1*,*n*)

Precondition:

Array A has at least 1 element between indexes p and r ($p \leq r$)

Postcondition:

The elements between indexes p and r are sorted

Merge Sort (Merge method)

- MERGE-SORT calls a function MERGE(A,p,q,r) to merge the sorted subarrays of A into a single sorted one
- The proof of MERGE can be done separately, using loop invariants

MERGE (A,p,q,r)

Precondition: A is an array and p, q, and r are indices into the array such that p <= q < r. The subarrays A[p.. q] and A[q +1.. r] are sorted

Postcondition: The subarray A[p..r] is sorted

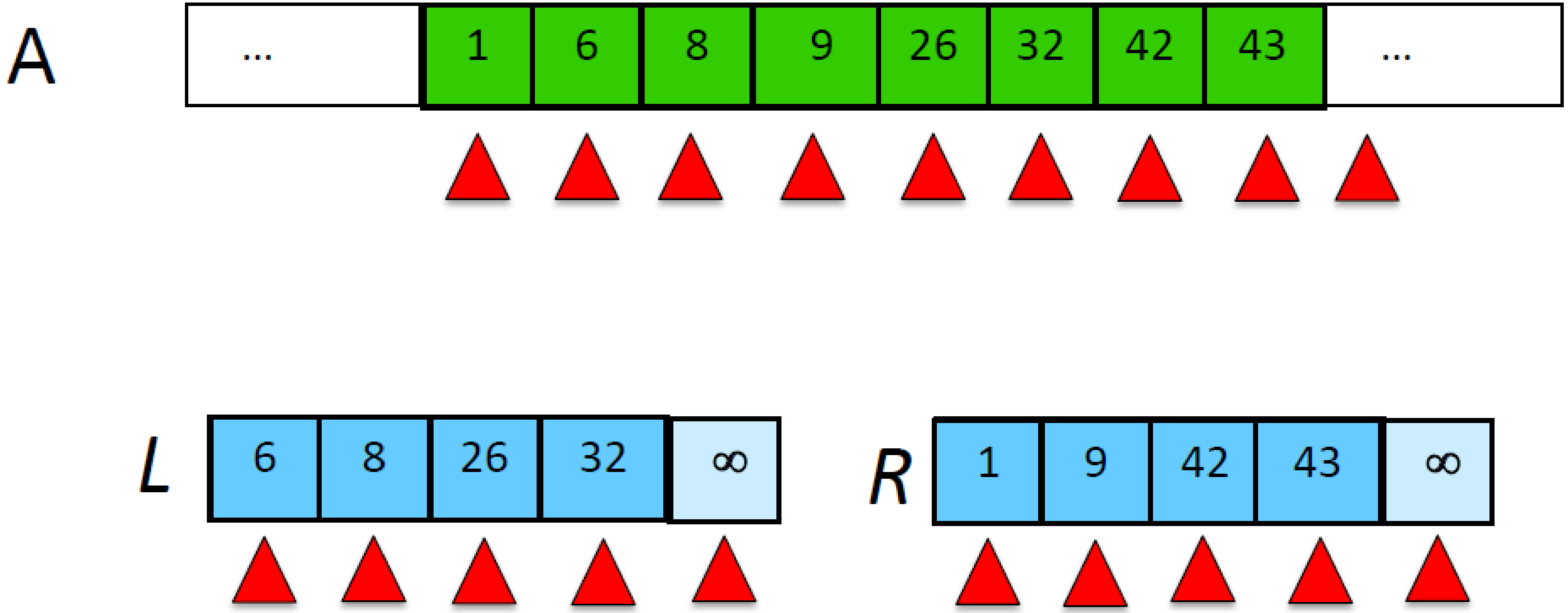
Procedure Merge

```
Merge(A, p, q, r)
n1 ← q − p + 1
n2 ← r − q
    for i ← 1 to n1
        do L[i] ← A[p + i − 1]
    for j ← 1 to n2
        do R[j] ← A[q + j]
    L[n1+1] ← ∞
    R[n2+1] ← ∞
    i ← 1
    j ← 1
    for k ←p to r
        do if L[i] ≤ R[j]
            then A[k] ← L[i]
                 i ← i + 1
            else A[k] ← R[j]
                 j ← j + 1
```

Input: Array containing sorted subarrays $A[p..q]$ and $A[q+1..r]$.

Output: Merged sorted subarray in $A[p..r]$.

Merge/combine – Example



Idea: The lists L and R are **already sorted**.

Correctness of MergeArray

- Loop-invariant
  - At the start of each iteration of the **for** loop, the subarray $A[1:k-1]$ contains the $k-1$ smallest elements of $L[1:s+1]$ and $R[1:t+1]$ in sorted order. Moreover, $L[i]$ and $R[j]$ are the smallest elements of their arrays that have not been copied back to A

Inductive Proof of Correctness

- **Initialization:** (the invariant is true at beginning)

  prior to the first iteration of the loop, we have $k = 1$, so that $A[1,k-1]$ is empty. This empty subarray contains $k-1 = 0$ smallest elements of L and R and since $i = j = 1$, $L[i]$ and $R[j]$ are the smallest element of their arrays that have not been copied back to A.

Inductive Proof of Correctness

- **Maintenance:** (the invariant is true after each iteration)

  assume $L[i]$ <= $R[j]$, the $L[i]$ is the smallest element not yet copied back to A. Hence after copy $L[i]$ to $A[k]$, the subarray $A[1..k]$ contains the k smallest elements. Increasing k and i by 1 reestablishes the loop invariant for the next iteration.

Inductive Proof of Correctness

- **Termination:** (loop invariant implies correctness)

  At termination we have k = s+t + 1, by the loop invariant, we have A contains the k-1 (s+t) smallest elements of L and R in sorted order.

Running Time of Merge-Sort

- Running time as a function of the input size, that is the number of elements in the array *A*.
- The Divide-and-Conquer scheme yields a clean recurrences.
- Assume T(n) be the running time of merge-sort for sorting an array of n elements.
- For simplicity assume n is a power of *2*, that is, there exists k such that $n = 2^k$.

Recurrence of T(*n*)

- T(1) = 1
- for n > 1, we have

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

$$T(n) = \begin{cases} 1 & \text{if } n = 1 \\ 2T(n/2) + cn & \text{if } n > 1 \end{cases}$$