10.6
Merge Sort

Sorting

Input Given an array of n elements

Goal Rearrange them in ascending order

Merge Sort [von Neumann]

MergeSort

1. Input: Array $\boldsymbol{A}[1 \ldots \boldsymbol{n}]$

$\boldsymbol{A\ L\ G\ O\ R\ I\ T\ H\ M\ S}$

Merge Sort [von Neumann]

MergeSort

1. Input: Array $A[1 \ldots n]$

   $$A\ L\ G\ O\ R\ I\ T\ H\ M\ S$$

2. Divide into subarrays $A[1 \ldots m]$ and $A[m+1 \ldots n]$, where $m = \lfloor n/2 \rfloor$

   $$A\ L\ G\ O\ R \qquad I\ T\ H\ M\ S$$

Merge Sort [von Neumann]

MergeSort

1. Input: Array $A[1 \ldots n]$

   $$A \; L \; G \; O \; R \; I \; T \; H \; M \; S$$

2. Divide into subarrays $A[1 \ldots m]$ and $A[m+1 \ldots n]$, where $m = \lfloor n/2 \rfloor$

   $$A \; L \; G \; O \; R \qquad I \; T \; H \; M \; S$$

3. Recursively **MergeSort** $A[1 \ldots m]$ and $A[m+1 \ldots n]$

   $$A \; G \; L \; O \; R \qquad H \; I \; M \; S \; T$$

Merge Sort [von Neumann]

MergeSort

1. Input: Array $A[1 \dots n]$

   $$A\ L\ G\ O\ R\ I\ T\ H\ M\ S$$

2. Divide into subarrays $A[1 \dots m]$ and $A[m+1 \dots n]$, where $m = \lfloor n/2 \rfloor$

   $$A\ L\ G\ O\ R \qquad I\ T\ H\ M\ S$$

3. Recursively **MergeSort** $A[1 \dots m]$ and $A[m+1 \dots n]$

   $$A\ G\ L\ O\ R \qquad H\ I\ M\ S\ T$$

4. Merge the sorted arrays

   $$A\ G\ H\ I\ L\ M\ O\ R\ S\ T$$

Merge Sort [von Neumann]

MergeSort

1. Input: Array $A[1 \dots n]$

   $$A\ L\ G\ O\ R\ I\ T\ H\ M\ S$$

2. Divide into subarrays $A[1 \dots m]$ and $A[m+1 \dots n]$, where $m = \lfloor n/2 \rfloor$

   $$A\ L\ G\ O\ R \qquad I\ T\ H\ M\ S$$

3. Recursively **MergeSort** $A[1 \dots m]$ and $A[m+1 \dots n]$

   $$A\ G\ L\ O\ R \qquad H\ I\ M\ S\ T$$

4. Merge the sorted arrays

   $$A\ G\ H\ I\ L\ M\ O\ R\ S\ T$$

Merging Sorted Arrays

1. Use a new array ***C*** to store the merged array
2. Scan ***A*** and ***B*** from left-to-right, storing elements in ***C*** in order

A G L O R ***H I M S T***

A G H I L M O R S T

Merging Sorted Arrays

1. Use a new array C to store the merged array
2. Scan A and B from left-to-right, storing elements in C in order

$$A\ G\ L\ O\ R \qquad H\ I\ M\ S\ T$$

$$A\ G\ H\ I\ L\ M\ O\ R\ S\ T$$

Merging Sorted Arrays

1. Use a new array C to store the merged array
2. Scan A and B from left-to-right, storing elements in C in order

$$A\ G\ L\ O\ R \qquad H\ I\ M\ S\ T$$

$$A\ G\ H\ I\ L\ M\ O\ R\ S\ T$$

Merging Sorted Arrays

1. Use a new array ***C*** to store the merged array
2. Scan ***A*** and ***B*** from left-to-right, storing elements in ***C*** in order

A G L O R ***H I M S T***

A G H I L M O R S T

Merging Sorted Arrays

1. Use a new array C to store the merged array
2. Scan A and B from left-to-right, storing elements in C in order

$$A\ G\ L\ O\ R \qquad H\ I\ M\ S\ T$$
$$A\ G\ H\ I\ L\ M\ O\ R\ S\ T$$

Merging Sorted Arrays

1. Use a new array $\boldsymbol{C}$ to store the merged array
2. Scan $\boldsymbol{A}$ and $\boldsymbol{B}$ from left-to-right, storing elements in $\boldsymbol{C}$ in order

$$\boldsymbol{A\ G\ L\ O\ R} \qquad \boldsymbol{H\ I\ M\ S\ T}$$
$$\boldsymbol{A\ G\ H\ I\ L\ M\ O\ R\ S\ T}$$

3. Merge two arrays using only constantly more extra space (in-place merge sort): doable but complicated and typically impractical.

Formal Code

```
MERGESORT($A[1..n]$):
  if $n > 1$
      $m \leftarrow \lfloor n/2 \rfloor$
      MERGESORT($A[1..m]$)
      MERGESORT($A[m+1..n]$)
      MERGE($A[1..n], m$)
```

```
MERGE($A[1..n], m$):
  $i \leftarrow 1$; $j \leftarrow m+1$
  for $k \leftarrow 1$ to $n$
      if $j > n$
          $B[k] \leftarrow A[i]$; $i \leftarrow i+1$
      else if $i > m$
          $B[k] \leftarrow A[j]$; $j \leftarrow j+1$
      else if $A[i] < A[j]$
          $B[k] \leftarrow A[i]$; $i \leftarrow i+1$
      else
          $B[k] \leftarrow A[j]$; $j \leftarrow j+1$

  for $k \leftarrow 1$ to $n$
      $A[k] \leftarrow B[k]$
```

THE END

...

(for now)