Divide-and-conquer
- mergesort [von Neumann '45]

Given array $A[1..n]$ of comparable objects (numbers)


1) Divide $A$ into two subarrays of equal size.
2) Recursively sort the subarrays.
3) Merge sorted subarrays.
merge: 1) first element of $A[1..n]$ should be smaller of the subarrays' first elements
2) recursively merge what remains of both subarrays
**MergeSort**\(A[1..n]\)

\[ m \leftarrow \lfloor n/2 \rfloor \]

\[ \text{MergeSort}(A[1..m]) \quad \] \text{\textit{(Recurse!)}}

\[ \text{MergeSort}(A[m+1..n]) \quad \] \text{\textit{(Recurse!)}}

\[ \text{Merge}(A[1..n], m) \]

\[ i \leftarrow 1; \ j \leftarrow m + 1 \]

\[ \text{for } k \leftarrow 1 \text{ to } n \]

\[ \begin{align*}
& \text{if } j > n \\
& \quad B[k] \leftarrow A[i]; \ i \leftarrow i + 1 \\
& \text{else if } i > m \\
& \quad B[k] \leftarrow A[j]; \ j \leftarrow j + 1 \\
& \text{else if } A[i] < A[j] \\
& \quad B[k] \leftarrow A[i]; \ i \leftarrow i + 1 \\
& \text{else} \\
& \quad B[k] \leftarrow A[j]; \ j \leftarrow j + 1 \\
\end{align*} \]

\[ \text{for } k \leftarrow 1 \text{ to } n \]

\[ A[k] \leftarrow B[k] \]

**Lemma:** Merge merges sorted subarrays \(A[1..m] + A[m+1..n]\) into one sorted array of their elements.

Fix indices \(k, i, j\) just before some iteration of the big for loop. I claim we put \(A[i..m] + A[j..n]\) into \(B[k..n]\) in sorted order.
\( k = 1, \ i = 1, \ s + j = m + 1 \) proves lemmas.

Convention: \( A[\ldots, n] \) denotes an empty array.

"last" iteration of \( k = n + 1 \) does nothing.

Assume for any assignments \( k' > k, \ i' \geq i, \ s + j' \geq j \) immediately, before a later iteration we merge \( A[i', m] + A[j', n] \) into \( B[k', n] \).

Note: we have \( n - k' + 1 < n - k + 1 \) iterations remaining in assumption.
If $k = n + 1$, our zero iterations leave $B[k..n] = B[n+1..n]$ sorted.

O.w.

if $j > n$, then $A[j..n]$ is empty

\[ \Rightarrow \min \{ A[i..m] \cup A[j..n] \} = A[i] \] \checkmark

similar is $i > m$

else is $A[i] \leq A[j]$

\[ \Rightarrow \min \text{ must be } A[i] \]

else $\min \text{ is } A[j]$

so $B[k]$ gets smallest element
If we set \( B[k] \leftarrow A[i] \)
we need to merge \( A[i+1..m] \) and \( A[j..n] \) into \( B[k+1..n] \).
and we do so by IH.
similar if \( B[k] \leftarrow A[j] \)

**Theorem: MergeSort sorts**

Assume alg sorts arrays of length \( k < n \).

If \( n = 1 \), we do nothing: array is already sorted!

RF sorts \( A[1..m] \) and \( A[m+1..n] \) (fewer elements). Merge merges,
Quicksort [Hoare '59]:

1) Choose an arbitrary pivot element from the array.

2) Partition array into three subarrays: elements < pivot, pivot, elements >

3) Recursively sort first and third subarrays.

---

Input: S O R T I N G E X A M P L
Choose a pivot: S O R T I N G E X A M P L
Partition: A G O E I N L M P T X S R
Recurse Left: A E G I L M N O P T X S R
Recurse Right: A E G I L M N O P R S T X
QuickSort(A[1..n]):
if (n > 1)
    Choose a pivot element A[p]
r ← Partition(A, p)
QuickSort(A[1..r − 1])  ⟨(Recurse!)⟩
QuickSort(A[r + 1..n])  ⟨(Recurse!)⟩

Partition(A[1..n], p):
l ← 0  ⟨(#items < pivot)⟩
for i ← 1 to n − 1
    if A[i] < A[n]
        ℓ ← ℓ + 1
return ℓ + 1

\textbf{Claim:} After \textit{ith} iteration of Partition's loop, \( A[1..l] \)
is all \( < A[n] \) (the pivot), \( A[l+1..i] \) is all \( \geq A[n] \). \textit{pivot}

if \( A[i] \geq A[n] \), \( A[1..l] \) unchanged \( \downarrow \)
\( A[l+1..i] \) has one new
Divide-and-conquer:

1) Divide given instance into independent, smaller instances of same problem.

2) Delegate each smaller instance to Recursion Fairy.

3) Combine solutions to smaller instances into a solution for given instance. If given instance is small, use a trivial/brute force alg.