Divide-and-conquer

- mergesort [von Neumann '45]

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

Goal: rearrange elements of A so $A[1] \leq A[2] \leq \ldots \leq A[n]$

1) Divide A into two subarrays of ~equal size.
2) Recursively sort the subarrays.
3) Merge sorted subarrays.

Input:	S	O	R	T	I	N	G	E	X	A	M	P	L
Divide:	S	O	R	T	I	N	G	E	X	A	M	P	L
Recurse Left:	I	N	O	R	S	T	G	E	X	A	M	P	L
Recurse Right:	I	N	O	R	S	T	A	E	G	L	M	P	X
Merge:	A	E	G	I	L	M	N	O	P	R	S	T	X

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]):
  if n > 1
      m ← ⌊n/2⌋
      MERGESORT(A[1..m])        ⟨⟨Recurse!⟩⟩
      MERGESORT(A[m+1..n])      ⟨⟨Recurse!⟩⟩
      MERGE(A[1..n], m)
```

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

  for k ← 1 to n
      A[k] ← 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.

($k=1$, $i=1$, & $j=m+1$) proves lemma

Convention: $A[n+1..n]$ denotes an empty array.

"last" iteration of $k=n+1$ does nothing

Assume for any assignments $k'>k$, $i'\geq i$, & $j'\geq j$ 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
$\Rightarrow \min\{A[i..m] \cup A[j..n]\}$
$= A[i]$ ✓

similar if $i > m$

else if $A[i] < A[j]$
$\Rightarrow$ min must be $A[i]$

else min is $A[j]$

so $B[k]$ gets smallest element

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

Theorem: MergeSort sorts $A[1..n]$.

Assume alg sorts arrays of length $k<n$.

If $n \le 1$, we do nothing; array is already sorted!

RF sorts $A[1..m]$ & $A[m+1...n]$ (fewer elements). Merge merges.

Quicksort [Hoare '59]:

1) Choose an arbitrary <u>pivot</u> element from the array.
2) Partition array into three subarrays: elements < pivot
   pivot
   elements >
3) Recursively sort first & 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	$\boxed{P}$	L
Partition:	A	G	O	E	I	N	L	M	$\boxed{P}$	T	X	S	R
Recurse Left:	A	E	G	I	L	M	N	O	$\boxed{P}$	T	X	S	R
Recurse Right:	A	E	G	I	L	M	N	O	$\boxed{P}$	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):
  swap A[p] ↔ A[n]
  ℓ ← 0                    ⟨⟨#items < pivot⟩⟩
  for i ← 1 to n−1
      if A[i] < A[n]
          ℓ ← ℓ+1
          swap A[ℓ] ↔ A[i]
  swap A[n] ↔ A[ℓ+1]
  return ℓ+1
```

Lomuto

Partition: Takes the index p of the pivot. Returns new index of pivot

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

if $A[i] \geq A[n]$, $A[1..\ell]$ unchanged

$A[\ell+1..i]$ has one new member

Divide-and-conquer:

1) <u>Divide</u> given instance into <u>independent, smaller</u> instances of same problem.

2) <u>Delegate</u> each smaller instance to Recursion Fairy.

3) <u>Combine</u> solutions to smaller instances into a solution for given instance.

If given instance is small use a trivial/brute force alg.