Learning Goals

- Describe divide + conquer structure ✓
- Describe methods for D & C for
  - proving correctness
  - analyzing time complexity (runtime) ←

Exit Tickets

Divide & C

MergeSort

Input : Integer array A of length n
Output: Sorted array

```
// Base Case
if n == 1 then
    return A;
end
// Divide and Conquer
A_1 = MergeSort(A[1 : n/2]);
A_2 = MergeSort(A[n/2 + 1 : n]);
// Combine
p_1 = p_2 = 1;
for i=1 to n do
    if A_1[p_1] < A_2[p_2] then
        A[i] = A_1[p_1];
        p_1++;
    else
        A[i] = A_2[p_2];
        p_2++;
    end
end
```

Divide & Conquer Structure

MergeSort

Input : Integer array A of length n
Output: Sorted array

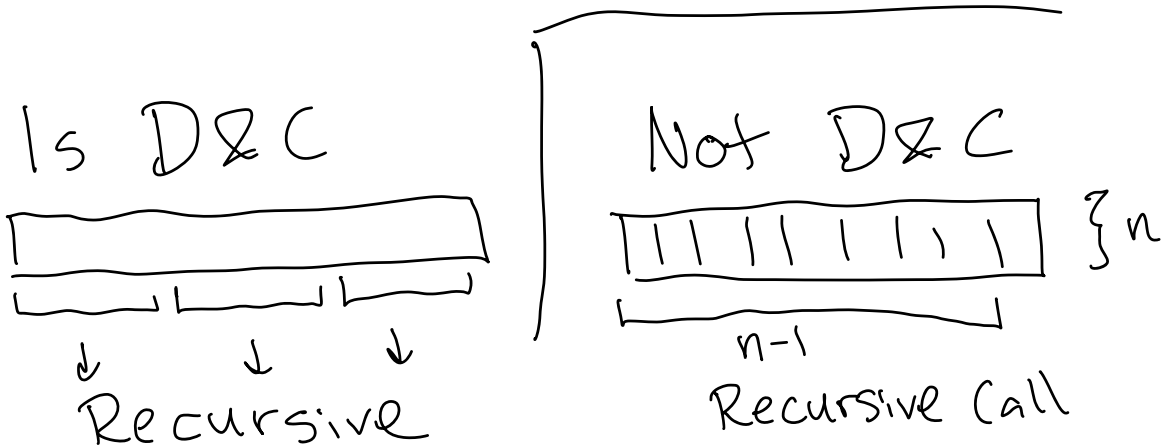
```
// Base Case
1  if n == 1 then
2  |  return A;
3  end
// Divide and Conquer
4  A_1 = MergeSort(A[1 : n/2]);
5  A_2 = MergeSort(A[n/2 + 1 : n]);
// Combine
6  p_1 = p_2 = 1;
7  for i=1 to n do
8  |  if A_1[p_1] < A_2[p_2] then
9  |  |  A[i] = A_1[p_1];
10 |  |  p_1++;
11 |  else
12 |  |  A[i] = A_2[p_2];
13 |  |  p_2++;
14 |  end
15 end
```

← Base case

Preprocessing (none in Mergesort)

← Divide input into equal sized pieces + recursively solve each piece

← Combine solutions from each recursive call to get ultimate solution



MergeSort

Input : Integer array A of length n
Output: Sorted array

```
// Base Case
if n == 1 then
    return A;
end
// Divide and Conquer
A_1 = MergeSort(A[1 : n/2]);
A_2 = MergeSort(A[n/2 + 1 : n]);
// Combine
p_1 = p_2 = 1;
for i=1 to n do
    if A_1[p_1] < A_2[p_2] then
        A[i] = A_1[p_1];
        p_1++;
    else
        A[i] = A_2[p_2];
        p_2++;
    end
end
```

How to prove correct?

Strong Induction → See review videos

(other options?)

How to analyze runtime?

Recurrence Relation

$T(n)$ = runtime on input of size n

$$T(n) = \begin{cases} O(1) & \text{if } n \leq 1 \\ 2T(n/2) + O(n) & \text{else} \end{cases}$$

↓ Solve

$T(n) = O(n \log n)$