INTRO TO DIVIDE + CONQUER

Learning Goals

- Describe divide + conquer structure [D&C2]
- Create recurrence relation for runtime of D&C [D&C1]
- Use tree formula to solve recurrence relation [D&C1]

Divide & Conquer Structure [D&C 2]

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
```

← Base Case

* When too small to divide further? *

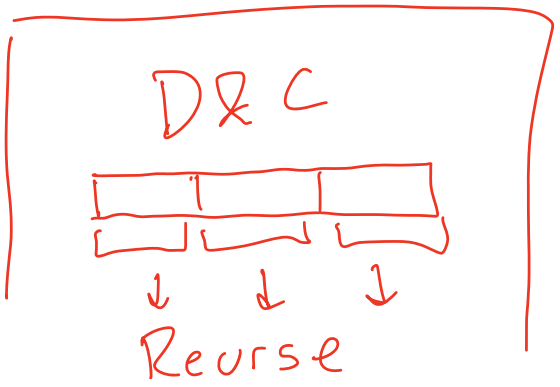
← Preprocessing Step

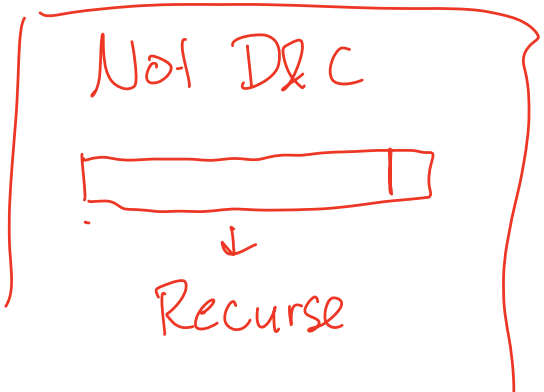
Divide input into equal sized pieces + recurse on each piece

* How could solve whole problem given soln to parts? *

← Combine solns from each recursive call

$\lfloor \frac{n}{2} \rfloor$





Runtime Analysis [D&C 1]

① Identify variable that determines input size.

② $T(n)$ = runtime on input size n

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
```

$T(n) = \begin{cases} \text{———} & \text{Base Case} \\ \text{———} & \text{Recurrence} \end{cases}$

Create recurrence relation for $T(n)$

$O(1)$

$T(\frac{n}{2})$ $T(\frac{n}{2})$

$O(1)$ ← $O(1)$

$\times O(n)$

$O(1)$ $O(n)$

$O(n)$

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

MergeSort[A[1:n/2]]
MergeSort[A[1:n/2]]

$T(\frac{n}{2}) + T(\frac{n}{2}) + O(1) + O(n)$

$2T(\frac{n}{2}) + O(n)$

Tree Formula [D&C 1]

$\log_2 4 \Rightarrow 2^x = 4$

$a > b^d$

$$T(n) = \begin{cases} O(1) & \text{if } n \le 1 \\ 2T\left(\frac{n}{2}\right) + O(n) & \text{else } (n \ge 2) \end{cases}$$

(2 crossed out, replaced by 4)

$c = 1$, $a = 2$, $b = 2$, $d = 1$, $n^1 = n$

$a = \cancel{2}\ 4$

$b^d = 2^1 = 2$

$T(n) = O(n \log_2 n)$

$T(n) = O(n^{\log_2 4}) = O(n^2)$

If

$$T(n) = \begin{cases} O(1) & \text{if } n \le c \\ aT\left(\frac{n}{b}\right) + O(n^d) & \text{else} \end{cases}$$

for c, a, b, d constants

$$\Rightarrow T(n) = \begin{cases} O(n^d \log_b n) & \text{if } a = b^d \\ O(n^d) & \text{if } a < b^d \\ O(n^{\log_b a}) & \text{if } a > b^d \end{cases}$$

What is the runtime of Wacky MergeSort?

A $O(n)$

B $O(n \log n)$

C $O(n^2)$ (circled)

D $O(n^4)$

$$4T\left(\frac{n}{2}\right) = T(2n)$$

$$4O\left(\frac{n^2}{2}\right) = O(2n^2) = O(n^2)$$