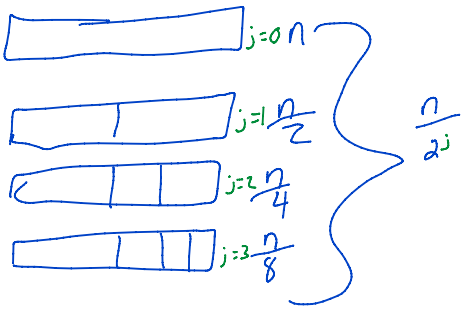
Thursday, February 24, 2022 7:29 PM

$O(\log n)$

j = # iterations



$\frac{n}{2^j} = 1$ log

$n = 2^j$

$\log n = j \log 2$

$\log n = j$

(array)

$A = [\square, \square, \square, \square]$

size = n ≥ 1

↳ goal → sum values in array

↳ for loop

↳ add each element → total variable

↳ Recursion

```
def total(A, n ): → inputs: array & size
  if n==1:
    return A[n];
  else
    A[n] + total(A, n-1);
```

Base case: 1 element A = {12} (Recursion →)

A = {12, 3}

Reverse array

['r', 'a', 'c', 'e', 'c', 'a', 'r']

a[0] a[1] a[2] a[n-3] a[n-2] a[n-1]

i+1 j-1

Swap(a[0], a[n-1])
Swap(a[1], a[n-2])
Swap(a[2], a[n-3])

Swap(a[i+1], a[j-1])

Base case

↳ a = ['A'] → array has 1 element

↳ first & last are the same

if base case
return array

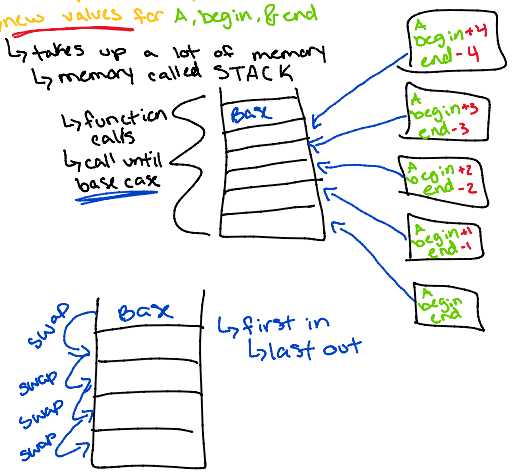
```
def array_reverse(A, begin, end):
    if A[begin] == A[end]:
        return A
    else:
        A[begin], A[end] = A[end], A[begin]
        array_reverse(A, begin+1, end-1)
```

Recursion →

change boundaries →

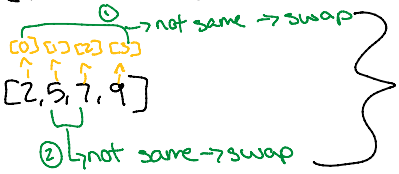
don't need return statement

- every time array-reverse is called
  - opens up new scope/environment
    - new values for A, begin, & end
- takes up a lot of memory
  - memory called STACK



- first in
- last out

Example: Reverse order



- have to change condition
  - was: if A[begin] == A[end]
  - changed to:
    - if A[begin] >= A[end]
      - begin >= end

- don't need return statement
- don't need else
- swapping → don't need to return