

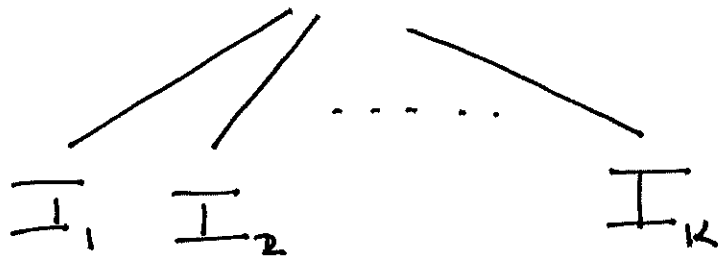
ese 102 4-30-24

11

# Divide & Conquer

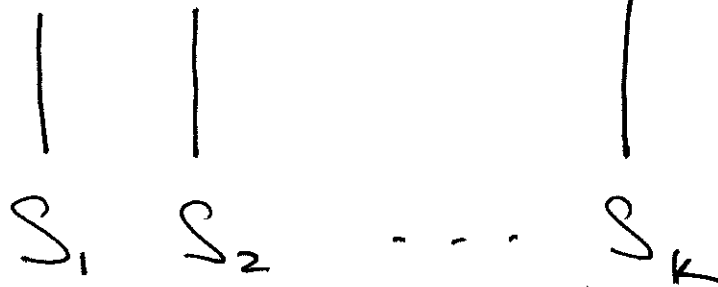
Problem instance:  $I$

divide



subinstances

solve



solutions

combine

Problem Solution:  $S$

Notation: array  $A$

subarray  $A[i \dots j]$  if  $i \leq j$   
 $\hookrightarrow (A_i, A_{i+1}, \dots, A_j)$

Merge Sort

$A[p \dots r]$  unsorted

split

$A[p \dots q]$

$A[q+1 \dots r]$  unsorted

recur

$A[p \dots q]$

$A[q+1 \dots r]$  sorted

Merge

$A[p \dots r]$  sorted

split point  $q = \lfloor \frac{p+r}{2} \rfloor$

MergeSort(A, p, r) Pre:  $p \leq r$

#comp

1. if  $p < r$

0

2.  $q = \lfloor \frac{p+r}{2} \rfloor$

0

3. MergeSort(A, p, q)

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

4. MergeSort(A, q+1, r)

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

5. Merge(A, p, q, r)

$n-1$

$$n = \text{length}(A[p \dots r]) = r - p + 1$$

Thm

MergeSort(A, p, r) sorts  $A[p \dots r]$

in increasing order.

I. if  $m=1$ , then  $r-p+1=1$ , so  $r=p$ .

In this case the array is already sorted, since its length 1.

The test on line 1 is false, so we do nothing.

II. let  $m>1$  and assume Mergesort correctly sorts any subarray of length less than  $m$ . We must show Mergesort correctly sorts  $A[p \dots r]$

observe  $m>1$  implies  $r>p$  so condition on line 1 is true, so lines 2-5 are executed.

Also

$$p < r \Rightarrow p < \frac{p+r}{2} < r \Rightarrow p \leq \left\lfloor \frac{p+r}{2} \right\rfloor < r$$

$$\Rightarrow \boxed{p \leq q < r}$$

so

$$\text{len}(A[p..q]) = q - p + 1 < r - p + 1 = m \quad \checkmark$$

and

$$\text{len}(A[q+1..r]) = r - (q+1) + 1 = r - q < r - q + 1$$

$$\leq r - p + 1 = m \quad \checkmark$$

By the induction hypothesis, lines 3 and 4 correctly sort subarrays  $A[p..q]$  and  $A[q+1..r]$ . Since line 5 correctly merges these subarrays,  $A[p..r]$  is sorted when Merge Sort halts.

# Runtime

In sorting algorithms, the runtime is (typically) the # of comparison operations performed.

Let  $T(n) = \# \text{ comparisons } (A_i < A_j)$

performed by  
MergeSort(A, 1, n)

Three measures of runtime:

- Best case
- average case  $\leftarrow$  maybe later
- worst case  $\leftarrow T(n)$

Note:

The worst case # comparisons  
by Merge( $A, l, q, r$ ) is

$$\text{len}(A[l \dots r]) - 1 = (r - l + 1 - 1) = r - l$$

At top level, this is  $n - 1$

Note

show that if  $l = 1, r = n$  then

$$q = \left\lfloor \frac{l+r}{2} \right\rfloor = \left\lfloor \frac{n+1}{2} \right\rfloor$$

and

$$\text{len}(A[l \dots q]) = \left\lceil \frac{n}{2} \right\rceil$$

and

$$\text{len}(A[q+1 \dots r]) = \left\lfloor \frac{n}{2} \right\rfloor$$

Thus  $T(n)$  satisfies

$$T(n) = \begin{cases} 0 & \text{if } n=1 \\ T(\lfloor \frac{n}{2} \rfloor) + T(\lceil \frac{n}{2} \rceil) + (n-1) & \text{if } n \geq 2 \end{cases}$$

By Master Thm.

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

comp.  $n$  to  $n^{\log_2 2} = n$

Case 2:  $T(n) = \Theta(n \log n)$



Apply iteration method: first simplify by assuming  $n$  is an exact power of 2.

Then  $\lfloor \frac{n}{2} \rfloor = \lceil \frac{n}{2} \rceil = \frac{n}{2}$  at all levels of recursion.

$$T(n) = \begin{cases} 0 & n=1 \\ 2T(\frac{n}{2}) + (n-1) & n \geq 2 \\ & (n \text{ power of } 2) \end{cases}$$

$$T(n) = (n-1) + 2T\left(\frac{n}{2}\right)$$

$$= (n-1) + 2\left(\left(\frac{n}{2}-1\right) + 2T\left(\frac{n}{2^2}\right)\right)$$

$$= (n-1) + (n-2) + 2^2 T\left(\frac{n}{2^2}\right)$$

$$= (n-1) + (n-2) + 2^2 \left(\left(\frac{n}{2^2}-1\right) + 2T\left(\frac{n}{2^3}\right)\right)$$

$$= (n-1) + (n-2) + (n-2^2) + 2^3 T\left(\frac{n}{2^3}\right)$$

$$\vdots$$

$$= \sum_{i=0}^{k-1} (n-2^i) + 2^k T\left(\frac{n}{2^k}\right)$$

$$= kn - \frac{2^k - 1}{2 - 1} + 2^k T\left(\frac{n}{2^k}\right)$$

recursion terminates when  $\frac{n}{2^k} = 1$

$\therefore$   $\boxed{k = \lg(n)}$

$$\therefore \boxed{T(n) = n \lg n - n + 1}$$

$$\uparrow$$

exact solution when

$n$  is a power of 2

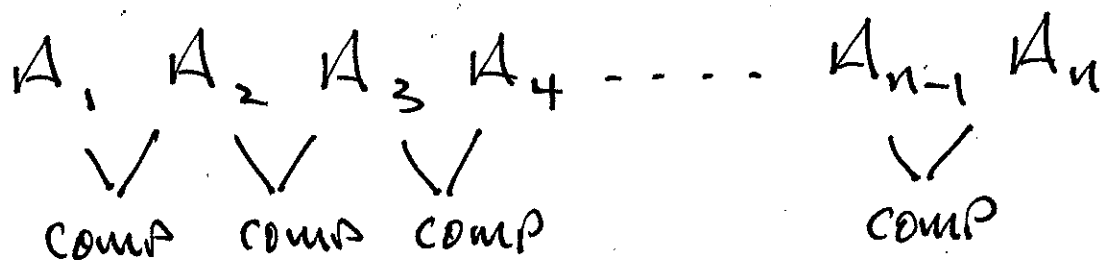
## Exercise

write an <sup>recursive</sup> algorithm (based on merge sort) that just checks that an array is sorted.

- Prove its correctness
- analyze <sup>worst case</sup> runtime (# comparisons)

answer =  $\Theta(n)$

note: iterative version:



$$\# \text{comp} = n - 1$$

Exercise

Given  $A = (A_1, A_2, \dots, A_n)$ , a

Pair of indices  $(i, j)$  is called

an inversion iff:  $i < j$  and  $A_i > A_j$

Write an algorithm (based on MergeSort) that counts # inversions.

note # of Pairs of indices  $= \binom{n}{2} = \frac{n(n-1)}{2}$

- write algorithm
- Prove correctness
- analyze runtime (# comparisons)

Two ways

Sort as you go:  $\Theta(n \log n)$

Don't sort:  $\Theta(n^2)$