CSE 102 10-21-21

- Exam: Midterm 1 today

Ex. $T(n) = 2T\left(\frac{n}{2}\right) + \frac{n}{\log n}$

comp. $\frac{n}{\log n}$ to $n^{\log_2(2)} = n^1$ ← winner

$$\frac{\frac{n}{\log n}}{n^{1-\varepsilon}} = \frac{n/\log n}{n/n^{\varepsilon}} = \frac{n^{\varepsilon}}{\log n} \longrightarrow \infty \quad \left(\begin{matrix}\text{for any} \\ \varepsilon > 0\end{matrix}\right)$$

$\therefore \frac{n}{\log n} = \omega(n^{1-\varepsilon})$ for all $\varepsilon > 0$.

Recall $\omega(n^{1-\varepsilon}) \cap O(n^{1-\varepsilon}) = \phi$

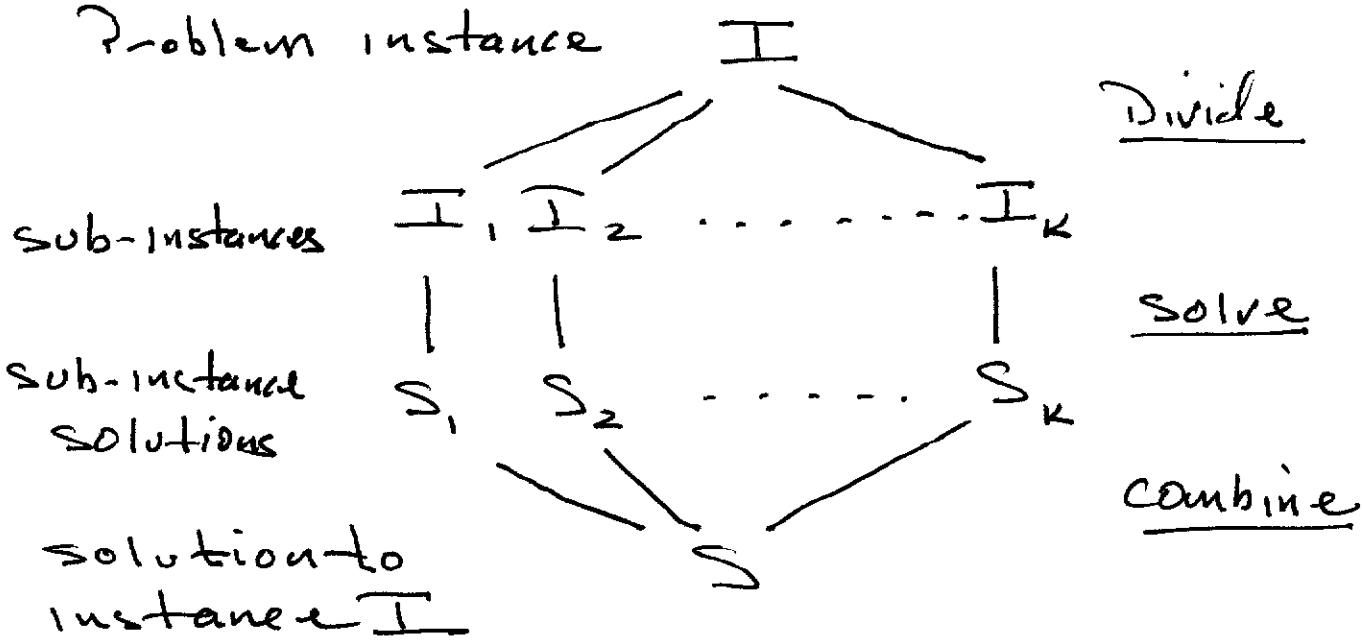
$\therefore \frac{n}{\log n} \neq O(n^{1-\varepsilon})$ for any $\varepsilon > 0$

$\therefore$ not in case 1 $\therefore$ master thm does not apply.

Exercise:

find an example where $f(n) = \Omega(n^{\log_b(a)+\varepsilon})$ for some $\varepsilon > 0$ (so case 3 is indicated), but the regularity cond. fails. i.e. there does not exist c in $0 < c < 1$ s.t. $af(\frac{n}{b}) \leq cf(n)$ for all suff. large n.

Divide and Conquer (recursion)



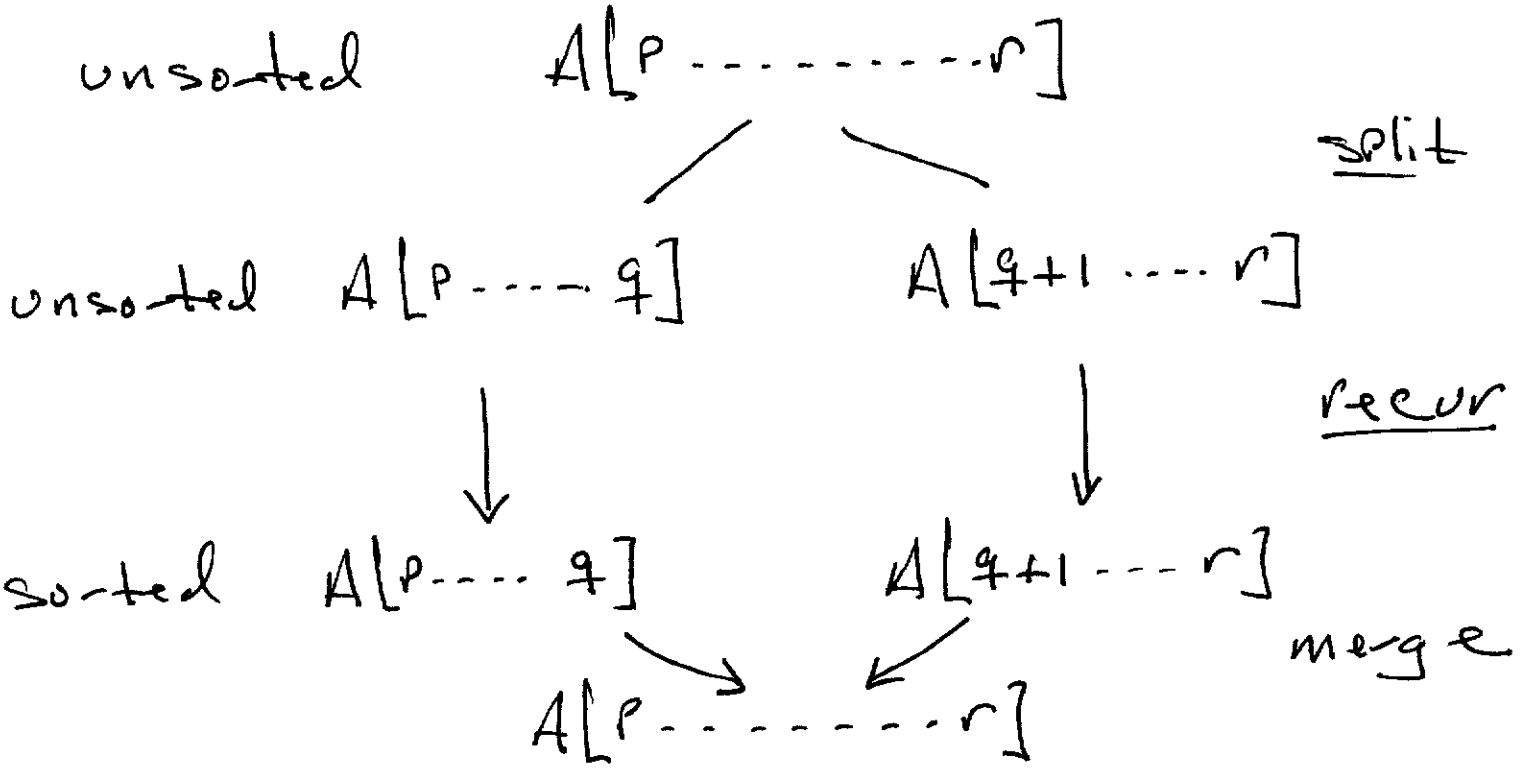
Ex. MergeSort

Let A be an array of length $n = \text{length}[A]$.

notation: $A[p \ldots r] = (A[p], A[p+1], \ldots, A[r])$

a subarray of length:

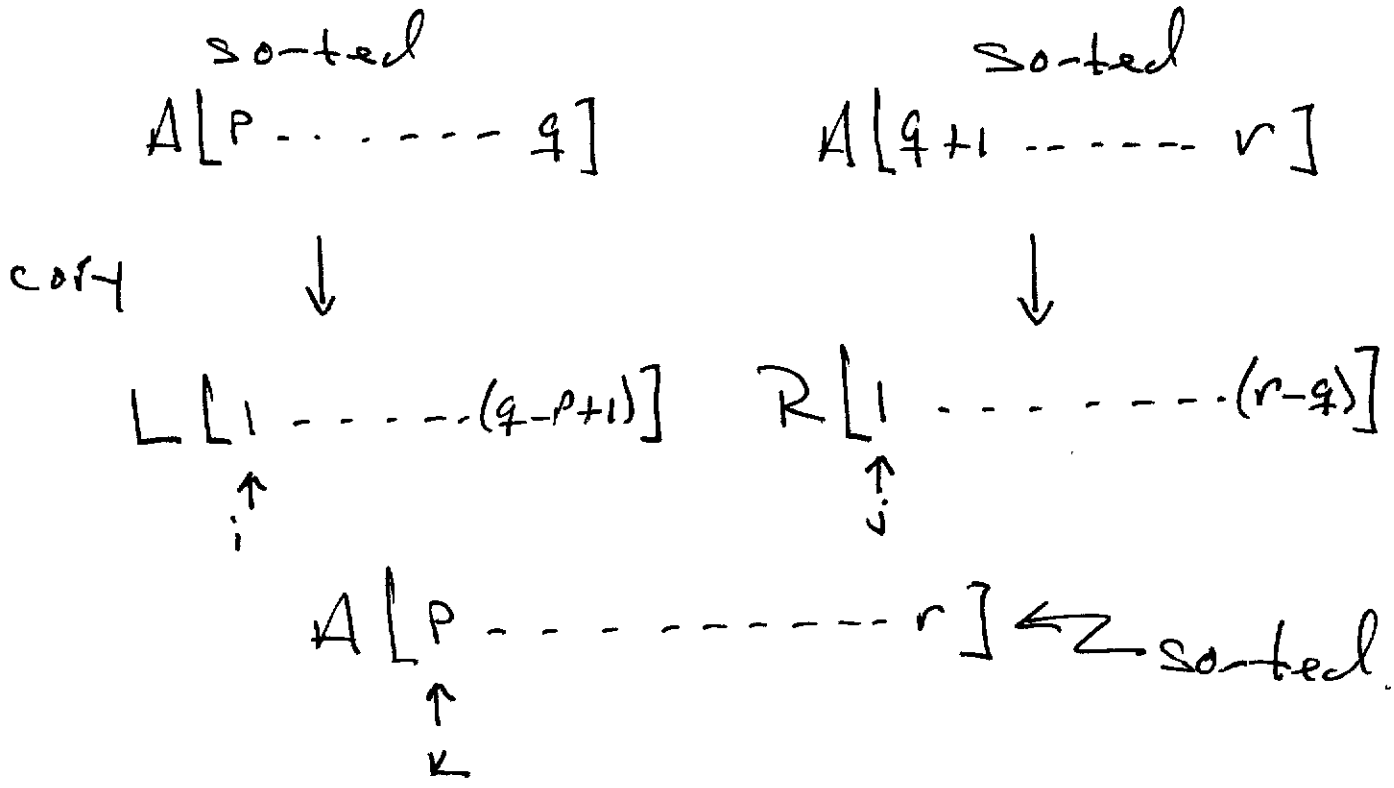
$$\text{length}[A[p \ldots r]] = r - p + 1$$



Pseudo-code:

	Cost if $p=1$, $r=n$
MergeSort(A, p, r) Pre: $p \le r$.	
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)	$n-1$

how does Merge() work



Exercise:

write pseudo code for -

Merge(A, p, q, r).

or see sec. 2.3 at CLRS

To sort A[1-----n], call

MergeSort(A, 1, n)

Prove correctness of MergeSort(A, p, r)

by induction on

$$m = \text{length}(A[p \ldots r]) = r - p + 1.$$

note: we Assume Merge(A, p, q, r)

is correct.

Proof.

I. If $n=1$, then $r=p$, so cond. on line 1 is false, we do nothing. But an array of len. 1 is already sorted.

II. Let $n>1$. Assume MergeSort() correctly sorts any sub array of length less than n. We must show that MergeSort(A, p, r) causes $A[p \ldots r]$ to be sorted where $n = r-p+1$.

$n>1 \Rightarrow r-p+1>1 \Rightarrow r>p$

so cond. on line 1 is true, so lines 2–5 are executed.

also

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

$$\Rightarrow p \le q < r$$

so that

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

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

$$\hookrightarrow \le r - p + 1 = m$$

By the ind. hypothesis, the recursive calls on lines 3 and 4 cause both $A[p \ldots q]$ and $A[q+1 \ldots r]$ to be sorted.

By our assumption on Merge(), line 5 causes $A[p \ldots r]$ to be sorted.

■

Runtime of MergeSort()

In sorting algorithms we (almost) always pick basic operation to be: comparison of array elements.

we count # of comparisons on input arrays of length n in

- best case
- worst case ← mostly this.
- average case ← occassionally.

we do worst case analysis on mergeSort().

Let $T(n)$ denote the worst case # of array comparisons performed by MergeSort$(A, 1, n)$

Exercise: Show that if $p=1$, $r=n$ and $q=\lfloor \frac{p+r}{2} \rfloor = \lfloor \frac{n+1}{2} \rfloor$, then

$$\text{len}(A[p \cdots q]) = \lceil \frac{n}{2} \rceil$$

and

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

Remark:

The worst case # of comparisons by Merge(A, p, q, r) is

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

so $T(n)$ satisfies:

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

By Master Thm:

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

compare n to $n^{\log_2 2} = n^1$

case 2 gives: $\boxed{T(n) = \Theta(n \log n)}$