4 Sept 2019

Divide & Conquer

1. Divide into smaller problems.
2. Conquer the smaller problems
3. Combine problems together

$T(n)$ MERGE SORT (A)

$\Theta(1)$ { if $|A| \leq 1$
| return A
endif

$\Theta(1)$ $m \leftarrow \lfloor |A|/2 \rfloor$ ← = "\gets" in LaTeX

the floor

$T(n/2)$ $B \leftarrow$ MERGESORT $(A[1 \ldots m])$

$T(n/2)$ $C \leftarrow$ MERGESORT $(A[m+1 \ldots |A|])$

$\Theta(n)$ return MERGE (B, C)

given: two sorted arrays B, C
return $B \sqcup C$, sorted
↑ disjoint union

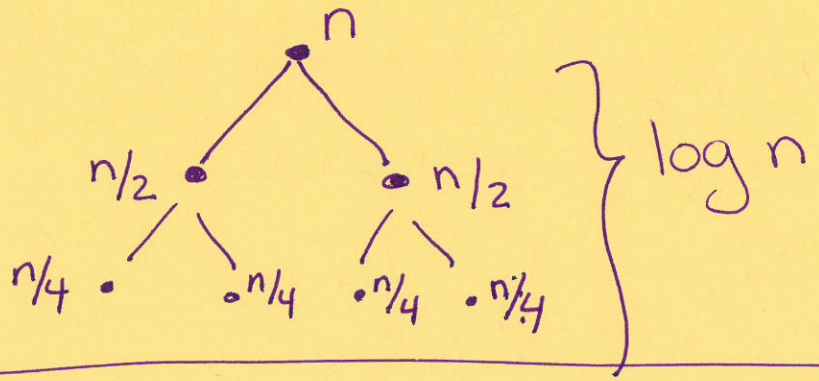
Proof of Termination:

① For a for loop, we must define a fcn

$$D: \{\text{state space}\} \longrightarrow S,$$

where:
- S is a well-~~defined~~ ordered set
  - ↳ ie., any subset has a minimum value.
- consider $A := \text{im}(D)$
  $= \{D(x) \mid x = \text{possible state}\}$
- loop terminates when $D(x)$ reaches $\min A$.
- ~~the~~ D decreases each time through the loop.

② For recursion, we need:
- base case(s) when $D(x)$ reaches $\min A$
- $D(x)$ gets smaller down recursive tree.



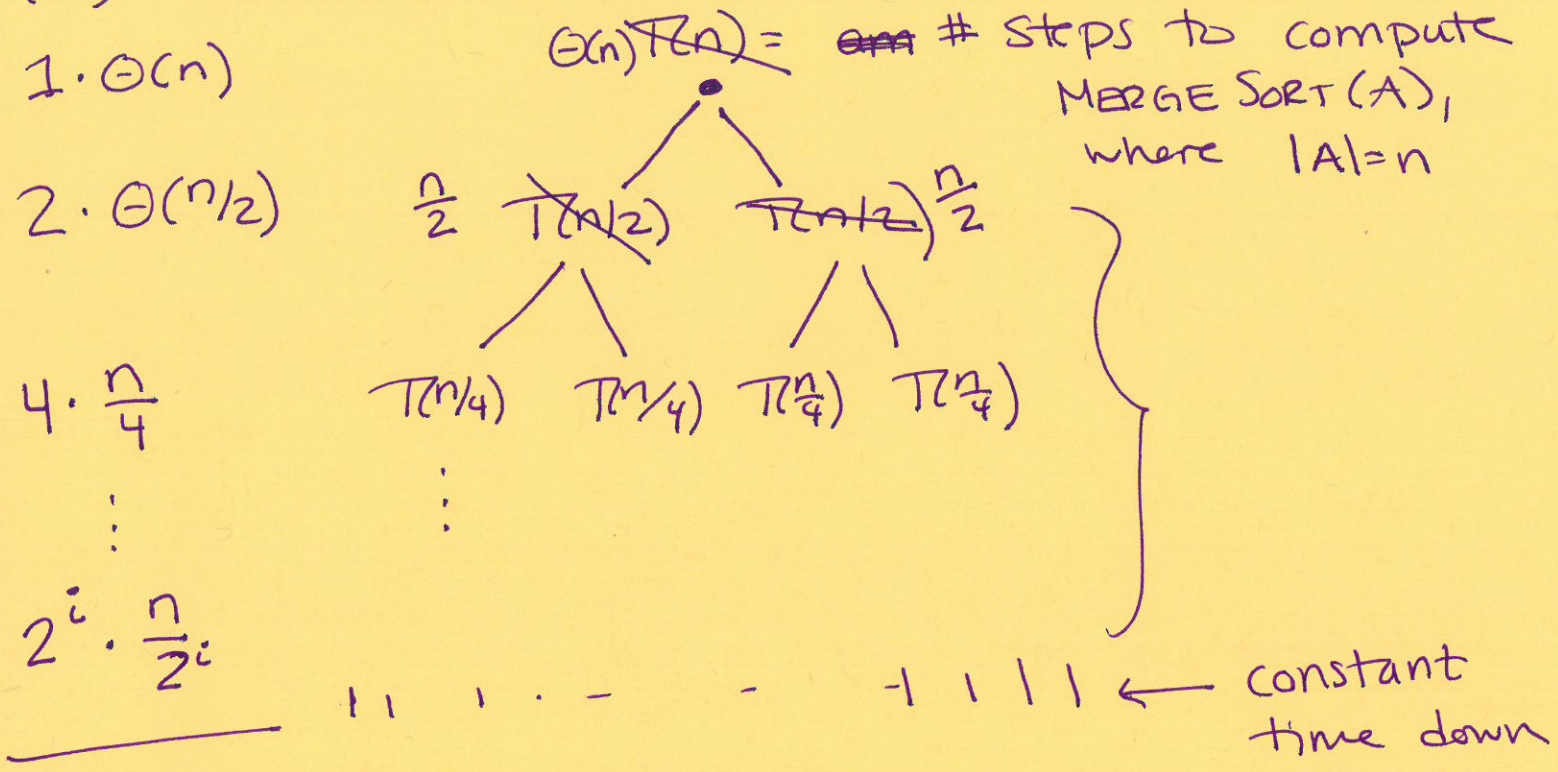
merge sort: $D = |A|$

$D: SS \longrightarrow \mathbb{N} \cup \{0\}$

∴ Since $\mathbb{N} \cup \{0\}$ is well-ordered, must reach 0 or 1.

How to solve recursions?

(a) Recursion Tree



$$\hookrightarrow \sum_{i=0}^{\log n} 2^i \frac{n}{2^i} = \sum_{i=0}^{\log n} n = n \log n$$

closed form of the recurrence.

Goal: We remember this is $\Theta(n \log n)$

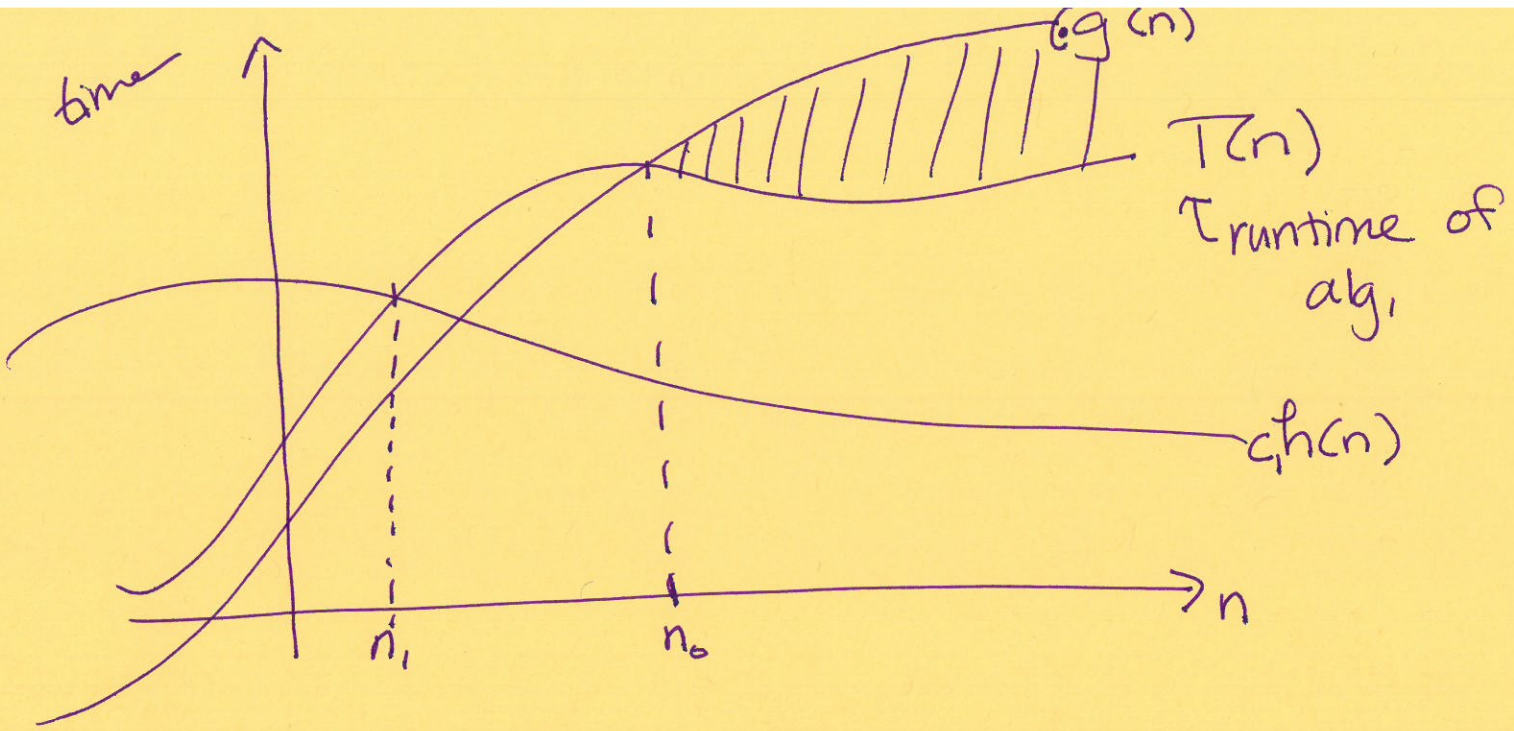
$$T(n) = \Theta(1) + \Theta(1) + T(n/2) + T(n/2) + \Theta(n)$$

$$\boxed{T(n) = 2T(n/2) + \Theta(n)}$$ merge sort recurrence relation

Other ways to solve Recurrence relations?

(b) Master's Theorem
- formula for common-style recurrence relations

(c) Substitution / Guess + Check
- ↑ induction, using the def. of big-O.



- we say $T(n)$ is $O(g(n))$ if

  $\exists\ n_0, c > 0, n_0, c \in \mathbb{R}$ such that

  $\forall\ n \geq n_0,\ T(n) \leq c \cdot g(n)$ $\boxed{\leq}$

- we say $T(n)$ is $\Omega(g(n))$ if

  $\exists\ n_1, c_1 \in \mathbb{R}, n_1 > 0, c_1 > 0$ such that

  $\forall\ n \geq n_1\ \ T(n) \geq c_1 \cdot h(n)$ $\boxed{\geq}$

  AH: if $T(n)$ is $O(g(n))$, then $g(n)$ is $\Omega(T(n))$.

- $T(n)$ is $\Theta(g(n))$ iff $\boxed{=}$

  $T(n)$ is $O(g(n))$ and $T(n)$ is $\Omega(g(n))$

- little o: $T(n)$ is $o(g(n))$ iff $\boxed{<}$

  $\lim_{n \to \infty} \frac{T(n)}{g(n)} = 0$