Nov 7

```
MergeSort(a, n)
O(1) { If n = 1 return a_1
O(n) { a_L = a_1, ..., a_⌊n/2⌋
       a_R = a_⌊n/2⌋+1, ..., a_n
     return MERGE( MergeSort(a_L, ⌊n/2⌋),
                   MergeSort(a_R, n-⌊n/2⌋))
            O(n)
```

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

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

$T(n)$ def = max runtime of MergeSort over all inputs of size n

$$T(n) \leq O(1) + O(n) + T(\lfloor \tfrac{n}{2} \rfloor) + T(n - \lfloor \tfrac{n}{2} \rfloor) + O(n)$$

If $n = 1$, $T(1) = O(1)$

$n > 1$, $T(n) \leq O(n) + T(\lfloor \tfrac{n}{2} \rfloor) + T(n - \lfloor \tfrac{n}{2} \rfloor)$.

Rewrite

$$T(n) \leq \begin{cases} O(1) & \text{if } n = 1 \\ O(n) + T(\lfloor \tfrac{n}{2} \rfloor) + T(n - \lfloor \tfrac{n}{2} \rfloor) & \text{otherwise} \end{cases}$$

(By definition of Big-Oh)

$$T(n) \leq \begin{cases} c_1 & \text{if } n = 1 \\ c_2 n + T(\lfloor \tfrac{n}{2} \rfloor) + T(n - \lfloor \tfrac{n}{2} \rfloor) & \text{o.w.} \end{cases}$$

($c = \max(c_1, c_2)$)

$$T(n) \leq \begin{cases} c & \text{if } n = 1 \\ cn + T(\lfloor \tfrac{n}{2} \rfloor) + T(n - \lfloor \tfrac{n}{2} \rfloor) & \text{o.w.} \end{cases}$$

$= \lceil \frac{n}{2} \rceil$

Rule of thumb: for asymptotics of $T(n)$:

$T(\lfloor x \rfloor) \mapsto T(x)$, $T(\lceil x \rceil) = T(x)$

$$\Rightarrow T(n) \leq \begin{cases} c & \text{if } n = 1 \\ cn + 2T(\frac{n}{2}) & \text{o/w} \end{cases}$$

Lemma: $T(n) \leq cn\log_2 n + cn$

$\Rightarrow$ MergeSort runs in $O(n\log n)$ time

Some remarks

1. $O(n\log n)$ is best known upper bound for general sorting problem.
2. Can do faster if domain of a_i's have $O(n)$ size
   (Q1 on HW0 $a_i \in \{A, B\} \Rightarrow O(n)$ runtime)
3. Can have faster algos for "almost" sorted inputs
4. Any comparison based algo for sorting has to use $\Omega(n\log n)$ comparisons
   ↑ Mergesort, Quicksort, heapsort ...

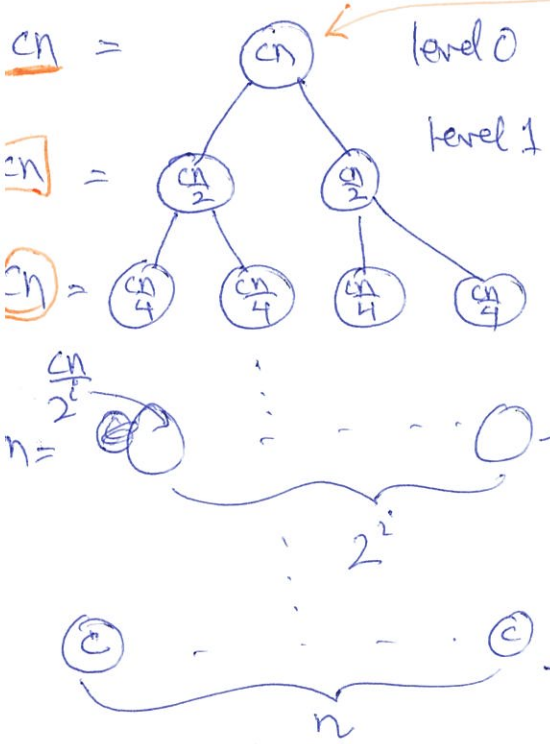
Strategies for solving recurrences:

1. "Unroll" the recurrence and use the pattern
2. Guess the answer and verify using induction on n.

Pf of Lemma 1:

Assume: n is a power of 2

Strategy 1: $T(n) \leq cn + 2T\left(\frac{n}{2}\right)$



$\leq cn + 2\left(\frac{cn}{2} + 2T\left(\frac{n}{4}\right)\right)$

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

$= cn + cn + 4\left(\frac{cn}{4} + 2T\left(\frac{n}{8}\right)\right)$

$= cn + cn + cn + 8T\left(\frac{n}{8}\right)$

contribution from level i:

$= 2^i \cdot \frac{cn}{2^i} = cn$

$\Rightarrow T(n) \leq cn$ (# levels)

$= cn(\ell + 1)$

Note: $\frac{n}{2^\ell} = 1 \Rightarrow 2^\ell = n$

$\Rightarrow \ell = \log_2 n$

$\Rightarrow T(n) \leq cn(\log_2 n + 1) = cn\log_2 n + cn$ ∎

Strategy 2: Guess: $T(n) \leq cn\log_2 n + cn$

Base case: $n=1$; $T(1) \; c$ $\quad c\cdot 1\cdot\log_2 1 + c\cdot 1 = c$

$\Rightarrow T(1) \leq c$ ✓

Inductive hypothesis: $T(\frac{n}{2}) \leq c\frac{n}{2}\log_2\frac{n}{2} + \frac{cn}{2}$

I. S.:

By recursion:

$T(n) \leq cn + 2T(\frac{n}{2})$

$\overset{\text{by}(*)}{\leq} cn + 2\left(\frac{cn}{2}\log_2 n\right)$

$= cn + cn\log_2 n$ ✱

$= \frac{cn}{2}\left(\log_2\frac{n}{2} + 1\right)$

$= \frac{cn}{2}\left(\log_2 n - \log_2 2 + 1\right)$

$= \frac{cn}{2}\log_2 n$ —(✱)