Nov 4

```
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⌋) )
```

$T(\lfloor \frac{n}{2} \rfloor)$ — MergeSort $(a_L, \lfloor \frac{n}{2} \rfloor)$

$O(n)$ — MERGE

$T(n-\lfloor \frac{n}{2} \rfloor)$ — MergeSort $(a_R, n-\lfloor \frac{n}{2} \rfloor)$

$T(n)$ = max runtime of MergeSort on any input of size n

Overall:

$T(1) \leq O(1)$

$n > 1$

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

$$\Rightarrow T(n) \leq \begin{cases} O(1) & n = 1 \\ O(n) + T(\lfloor \frac{n}{2} \rfloor) + T(n-\lfloor \frac{n}{2} \rfloor) & \text{o/w} \end{cases}$$

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

c_1, c_2 are constants independent of n

$$\Rightarrow T(n) \leq \begin{cases} c & \text{if } n=1 \\ \leq cn + T(\lfloor \frac{n}{2} \rfloor) + T(n-\lfloor \frac{n}{2} \rfloor) & \end{cases} \quad c = \max(c_1, c_2)$$

Note: Can replace $T(\lfloor \frac{n}{2} \rfloor)$ by $T(\lfloor x \rfloor)$ by $T(x)$ does not change the asymptotics. $T(\lceil x \rceil)$ by $T(x)$

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

---

LEMMA: $T(n) \leq cn \log_2(n+1) + cn$

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

---

1) $O(n \log n)$ is the best known runtime for any general purpose sorting algo
2) Any comparison based algo takes $\Omega(n \log n)$ comparisons
3) Can do faster e.g if the values are bounded (eg. HW0 Q1)
4) Can faster runtime for "almost sorted" inputs.

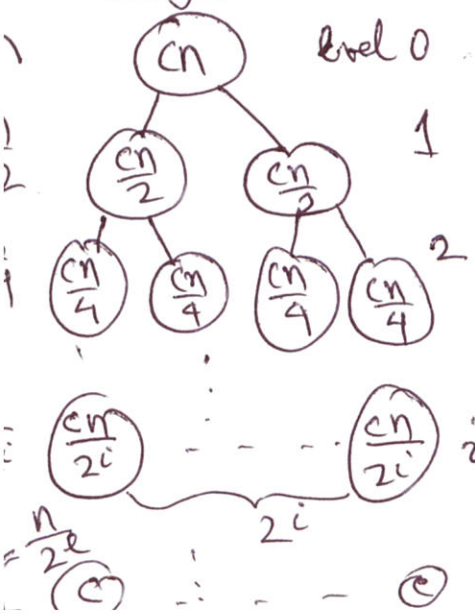
Strategies to solve recurrences

1) "Unroll" the recurrence & use the pattern
2) Guess the final answer & use induction on n ($cn \log_2(n+1)$ for LEMMA)

Strategy 1

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

'level 0' 'from $\frac{n}{2}$' 'from $\frac{n}{4}$'



contribution of level i = $\frac{cn}{2^i} \cdot 2^i = cn$

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

$= cn(\ell + 1)$

l is such $\frac{n}{2^l} = 1 \Rightarrow 2^l = n \Rightarrow l = \log_2 n$

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

$T(1) \leq c$ ~~$cn\log_2 n + 1$~~ $c \cdot 1 \cdot \log_2 1 + c \cdot 1 = c$

$\Rightarrow T(1) \leq c \cdot 1 \log_2 1 + c \cdot 1$

$\Rightarrow T(n) \leq cn \log_2 n + cn.$

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

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

Base case: $n = 1$ ✓

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

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

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

$= \frac{cn}{2}\log_2 n$

Inductive step:

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

$\leq cn + 2(\frac{cn}{2}\log_2 n)$

$= cn + cn\log_2 n$ ✓