Basic Algorithmic Techniques

- Divide & conquer
- Greedy
- Dynamic Programming
- Amortized Analysis

Divide & Conquer

- Divide the problem of size n into two (sub-) problems of size $\frac{n}{2}$.
- Solve each (sub-) problem recursively.
- Combine two solutions to get a solution to the original problem.

Merge Sort

Recurrence relation :

$$T(n) \leq 2\,T\left(\frac{n}{2}\right) + Cn$$

$$T(2) \leq C' \ (= 2C).$$

The "solution" to this recurrence relation is $O(n \log n)$.

<u>Claim</u> $T(n) \leq c\, n \log n$

<u>Proof</u> By induction, "guess & verify".

$$
\begin{aligned}
T(n) &\leq 2\, T\left(\frac{n}{2}\right) + cn \\
&\leq 2\left(c \cdot \frac{n}{2} \log \frac{n}{2}\right) + cn \\
&\leq 2\left(c \cdot \frac{n}{2} (\log n - 1)\right) + cn \\
&= c\, n \log n - \cancel{cn} + \cancel{cn}
\end{aligned}
$$

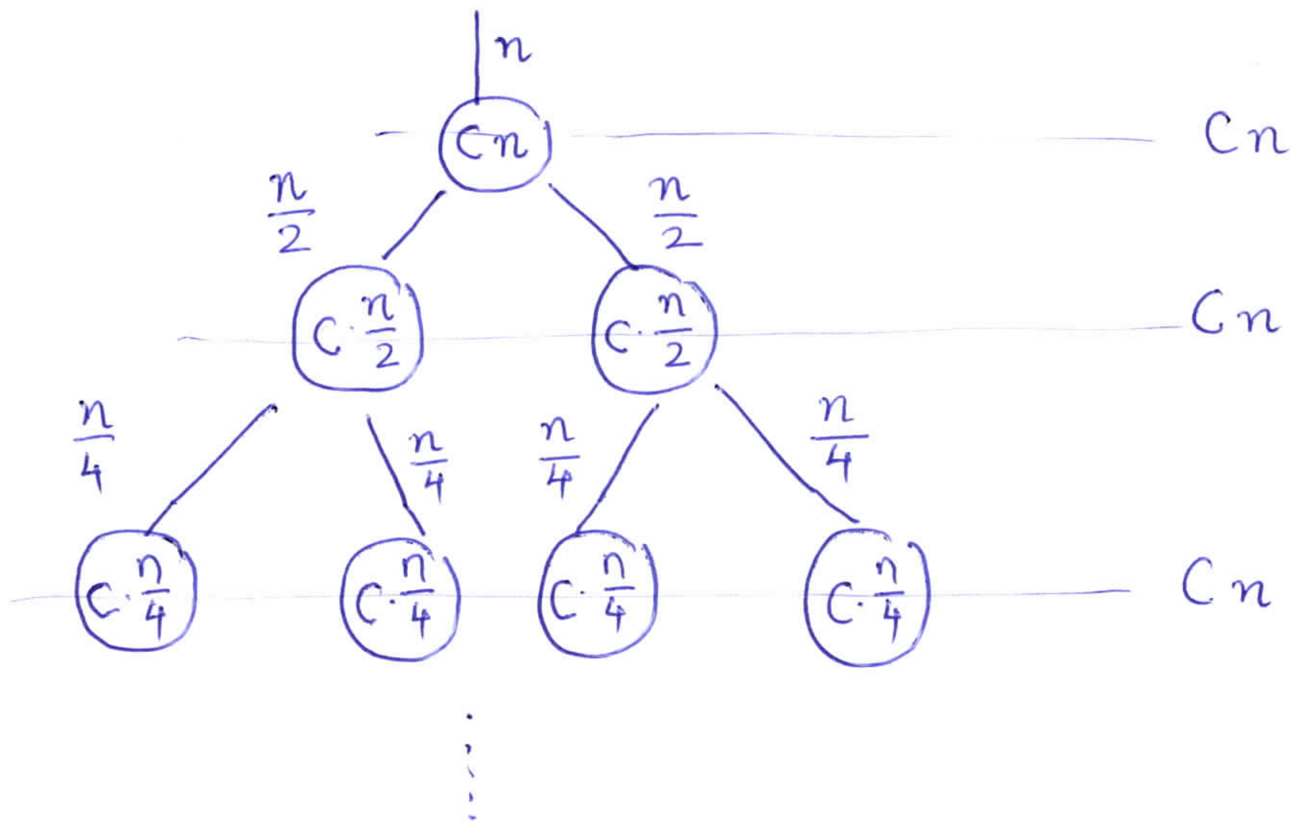
$\blacksquare$

<u>Proof</u> By "unrolling recursion".

$$
\begin{aligned}
T(n) &\leq 2\, T\left(\frac{n}{2}\right) + cn \\
&\leq 2\left(2\, T\left(\frac{n}{4}\right) + c \cdot \frac{n}{2}\right) + cn \\
&= 4\, T\left(\frac{n}{4}\right) + \underline{cn + cn} \\
&\ \ \vdots \\
&= 2^i\, T\left(\frac{n}{2^i}\right) + i\, c\, n \\
&= n \cdot T(1) + c\, n \log n \qquad i = \log n \\
&\leq 2\, c\, n \log n
\end{aligned}
$$

$\blacksquare$

Proof "By picture"



$$
\begin{aligned}
\text{Total time} &= Cn && \text{(level 1)} \\
&+ 2 \cdot c \cdot \frac{n}{2} && \text{(level 2)} \\
&+ 4 \cdot c \cdot \frac{n}{4} && \text{(level 3)} \\
&\;\vdots \\
&= Cn \log n.
\end{aligned}
$$

$\blacksquare$

Note We'll see at least one more recurrence relation; however the relation $T(n) \le 2\,T(\frac{n}{2}) + O(n)$ is the most important. Its solution is $\underline{O(n \log n)}$.

Quick Sort

$$\overbrace{a_1, a_2, \dots, a_n}^{L}$$

- Given n integers $a_1, a_2, \dots, a_n$
- Select one of the a_{i_0} as "pivot" b.
- Partition L into

$$L_1 = \{ a_i \mid a_i \leq b \}.$$

$$L_2 = \{ a_i \mid a_i > b \}.$$

- Recursively sort L_1, sort L_2.
- Output $L_1 \circ L_2$.

—— x ——

- Worst case running time is $\Omega(n^2)$,
- $\because$ if "pivot" is badly chosen, it could be the case that always $|L_1| = 1$, $|L_2| = n-1$.
- This runs in time $O(n \log n)$ if $|L_1| = |L_2| = \frac{n}{2}$, i.e. if b is the median.
- Can we find median in $O(n)$ time? YES!

Recall: we desire $T(n) \leq 2T(\frac{n}{2}) + O(n)$

Finding Median in $O(n)$ time.

Def Given integers $a_1, a_2, \ldots, a_n$, the median is the number a_s s.t.

$$|\{a_i \mid a_i \leq a_s\}| = \lfloor \tfrac{n}{2} \rfloor.$$

I.e. the middle element in sorted order.

Theorem There is $O(n)$-time algorithm to find median.

All these $\leq b^*$

$$a_1 \leq a_1' \leq b_1 \leq c_1 \leq c_1'$$
$$a_2 \leq a_2' \leq b_2 \leq c_2 \leq c_2'$$
$$\vdots$$

A

b^*

C

All these $> b^*$

$$a_{\frac{n}{5}} \leq a'_{\frac{n}{5}} \leq b_{\frac{n}{5}} \leq c_{\frac{n}{5}} \leq c'_{\frac{n}{5}}$$

Algorithm

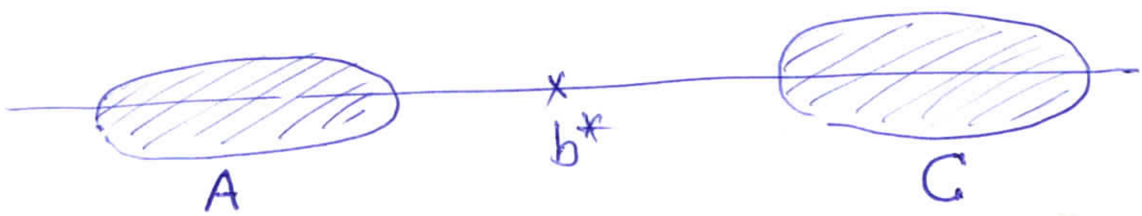
- Divide n numbers into $\frac{n}{5}$ lists of size 5 each.
- Sort each list. Let i^{th} list be

$$a_i \le a_i' \le \underline{b_i} \le c_i \le c_i'.$$

- Recursively find median of $b_1, b_2, \ldots, b_{\frac{n}{5}}$. Call it b^* (median of medians)
- Reorder rows/indices so that

$$b_1, b_2, \ldots, b_{\frac{n}{10}} \le b^*$$

$$b_{\frac{n}{10}+1}, \ldots, b_{\frac{n}{5}} > b^*.$$



$|A| = \frac{3n}{10}$ $\qquad$ $|C| = \frac{3n}{10}$.

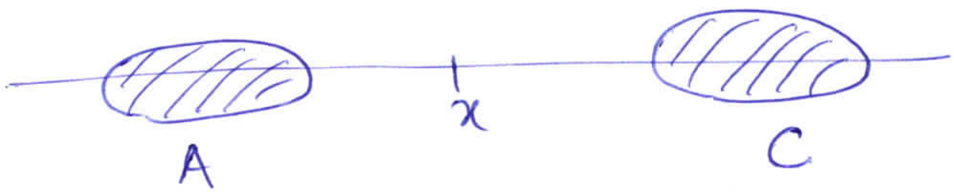
- Drop A, C.
- Find median of remaining $n - \frac{3n}{10} - \frac{3n}{10}$

$= \frac{2}{5}n$ elements. (Recursively).

Recurrence Relation

$$T(n) \leq T\left(\frac{n}{5}\right) + T\left(\frac{2}{5}n\right) + O(n).$$

- There is a flaw in the analysis, proof of correctness however.
- Let x be the true (hypothetical) median. If it were the case that



then dropping A, C would preserve x as the median of remaining elements.
- However this need not be the case.
- In fact, it could be that $x \in A \cup C$ and if one drops A, C, one loses the true median :(.
- How to fix the algorithm?

- First homework.
- Design algorithm that solves a more general problem: Given $a_1, \dots, a_n$ and $1 \le k \le n$, algo. finds $\underbrace{k^{th} \text{ element}}_{\text{rank } k}$ in sorted order.
- Depending on whether rank (b^*) $<k$, $=k$, $>k$ drop either A or C.
- New recurrence relation

$$T(n) \le T\left(\frac{n}{5}\right) + T\left(\frac{7}{10}n\right) + O(n).$$

$\blacksquare$

<u>Lemma</u> Let a, b, C be positive constants s.t. $a+b<1$. If

$$T(n) \le T(an) + T(bn) + Cn$$

$T(1) \le C$ then $T(n) \le O(n)$.

<u>Proof</u> We'll prove that $T(n) \le \beta n$

where $\beta = \dfrac{c}{1-(a+b)}$.

By induction,

$$
\begin{aligned}
T(n) &\le T(an) + T(bn) + cn \\
&\le \beta \cdot an + \beta \cdot bn + cn \\
&= (\beta(a+b) + c)\, n \\
&\le \beta n
\end{aligned}
$$

provided that $\beta(a+b) + c \le \beta$

if $c \le (1-(a+b))\beta$

if $\dfrac{c}{1-(a+b)} \le \beta$.

$\blacksquare$