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:

\[ T(n) \leq 2 \cdot T\left(\frac{n}{2}\right) + Cn \]
\[ T(2) \leq C' \quad (= 2C) \]

The "solution" to this recurrence relation is $O(n\log n)$. 
Claim \[ T(n) \leq C n \log n \]

Proof By induction, "guess & verify."

\[
T(n) \leq 2 \, T\left(\frac{n}{2}\right) + Cn
\]
\[
\leq 2 \left( C \cdot \frac{n}{2} \log \frac{n}{2} \right) + Cn
\]
\[
\leq 2 \left( C \cdot \frac{n}{2} (\log n - 1) \right) + Cn
\]
\[
= Cn \log n - \frac{Cn}{2} + Cn
\]

Proof By "unrolling recursion."

\[
T(n) \leq 2 \, T\left(\frac{n}{2}\right) + Cn
\]
\[
\leq 2 \left( 2 \, T\left(\frac{n}{4}\right) + C \cdot \frac{n}{2} \right) + Cn
\]
\[
= 4 \, T\left(\frac{n}{4}\right) + Cn + Cn
\]
\[
= 2^i \, T\left(\frac{n}{2^i}\right) + iCn
\]
\[
= n \cdot T(1) + Cn \log n \quad \text{for } i = \log n
\]
\[
\leq 2 \, Cn \log n
\]
Proof  "By picture"

\[
\begin{array}{c}
\text{n} \\
\downarrow \\
C_n \\
\downarrow \\
C_n/2, C_n/2 \\
\downarrow \downarrow \downarrow \\
C_n/4, C_n/4, C_n/4, C_n/4 \\
\downarrow \downarrow \downarrow \downarrow \\
C_n/4, C_n/4, C_n/4, C_n/4 \\
\downarrow \downarrow \downarrow \downarrow \\
C_n/2, C_n/2, C_n/2, C_n/2 \\
\downarrow \downarrow \downarrow \downarrow \\
C_n, C_n, C_n, C_n \\
\end{array}
\]

Total time = \( Cn \) (level 1) \\
+ 2 \times C \cdot \frac{n}{2} \) (level 2) \\
+ 4 \times C \cdot \frac{n}{4} \) (level 3) \\
= C n \log n.

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

- Given n integers \( a_1, a_2, \ldots, a_n \) and an
- Select one of the \( a_i \) 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 \( L_1 \circ L_2 \).

- Worst case running time is \( \Omega(n^2) \).
- "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
  \[
  |L_1| = 1, |L_2| = \frac{n}{2}, \text{ i.e. if b is the median}.
  \]
- Can we find median in \( O(n) \) time? YES! \( T(n) \leq 2T(\frac{n}{2}) + O(n) \)

Recall: we desire...
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.

$$\left| \left\{ a_i \mid a_i \leq a_s \right\} \right| = \left\lfloor \frac{n}{2} \right\rfloor.$$  

I.e. the middle element in sorted order.

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

All these

\[
\begin{align*}
    a_1 & \leq a'_1 \leq b_1 \\
    a_2 & \leq a'_2 \leq b_2 \\
    \vdots & \\
    a_n & \leq a'_n \leq b_n
\end{align*}
\]

All these

\[
\begin{align*}
    c_1 & \leq c'_1 \\
    c_2 & \leq c'_2 \\
    \vdots & \\
    c_n & \leq c'_n
\end{align*}
\]

All these

\[
\begin{align*}
    a_{\frac{n}{5}} & \leq a'_{\frac{n}{5}} \leq b_{\frac{n}{5}} \\
    b_{\frac{n}{5}} & \leq C_{\frac{n}{5}} \leq C'_{\frac{n}{5}}
\end{align*}
\]
Algorithm

- Divide n numbers into \( \frac{n}{5} \) lists of size 5 each.
- Sort each list. Let the list be
  \[ a_i \leq a'_i \leq b_i \leq c_i \leq c'_i. \]
- Recursively find median of \( b_1, b_2, \ldots, b_{\frac{n}{5}} \).
  Call it \( b^* \) (median of medians).
- Reorder rows/indices so that
  \[ b_1, b_2, \ldots, b_{\frac{n}{10}} \leq b^* \]
  \[ b_{\frac{n}{10}+1}, \ldots, b_{\frac{n}{5}} > b^*. \]

\[ |A| = \frac{3n}{10} \]
\[ |C| = \frac{3n}{10}. \]
- 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

\[ A \quad \underline{x} \quad C \]

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 \( x \).

- How to fix the algorithm?
- First homework.

- Design algorithm that solves a more general problem: Given \( a_1, \ldots, a_n \) and \( 1 \leq k \leq n \), algo. finds \( k^{th} \) element in sorted order.

- Depending on whether \( \text{rank} (b^x) < k \), drop either A or C.

- New recurrence relation

\[
T(n) \leq T\left( \frac{n}{5} \right) + T\left( \frac{7}{10} n \right) + O(n).
\]

\[
\]

Lemma: Let \( a, b, C \) be positive constants s.t. \( a + b < 1 \). If

\[
T(n) \leq T(an) + T(bn) + Cn
\]

\( T(1) \leq C \) then \( T(n) \leq O(n) \).
Proof. We'll prove that $T(n) \leq \beta n$

where $\beta = \frac{C}{1-(a+b)}$.

By induction,

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

$\leq \beta an + \beta bn + Cn$

$= \left( \beta(a+b) + C \right) n$

$\leq \beta n$

provided that $\beta(a+b) + C \leq \beta$

if $C \leq (1-(a+b)) \beta$

if $\frac{C}{1-(a+b)} \leq \beta$.