Divide and Conquer

– Divide the problem into parts.
– Solve each part.
– Combine the Solutions.
– Complexity is usually of form $T(n) = aT(n/b) + f(n)$.
– Chapter 5 of the book

Merge Sort

1. Divide the array into two parts.
2. Sort each of the parts recursively.
3. Merge the two sorted parts. Since the two parts are sorted, merging is easier than sorting the whole array.

Input: $A[i], \ldots, A[j]$, where $i \leq j$.
Output: Sorted $A[i], \ldots, A[j]$.

MergeSort(A, i, j)

(* Assumption: $i \leq j$. *)

1. If $i = j$, then return Endif
2. Let $k = \lfloor \frac{i+j-1}{2} \rfloor$.
3. MergeSort(A, i, k)
4. MergeSort$(A, k+1, j)$
5. Merge(A, i, k, j)

End

Merge(A, i, k, j)

(* Assumption: $i \leq k < j$; $A[i : k]$ and $A[k + 1 : j]$ are sorted. *)

1. $p1 = i$; $p2 = k + 1$; $p3 = i$
2. While $p1 \leq k$ and $p2 \leq j$ {
   2.1 If $A[p1] \leq A[p2]$, then $B[p3] = A[p1]$, $p1 = p1 + 1$
   Else $B[p3] = A[p2]$, $p2 = p2 + 1$; Endif
   2.2 $p3 = p3 + 1$;
   }
3. (* Copy the remaining elements into B *)
   While $p1 \leq k$ { $B[p3] = A[p1]$; $p1 = p1 + 1$; $p3 = p3 + 1$; }
   While $p2 \leq j$ { $B[p3] = A[p2]$; $p2 = p2 + 1$; $p3 = p3 + 1$; }
4. For $r = i$ to j { $A[r] = B[r]$ }

End

Complexity of Merge: $O(j - i + 1)$, that is the size of the two parts to be merged.

Complexity of MergeSort:
$T(n) \leq T(\lceil \frac{n}{2} \rceil) + T(\lfloor \frac{n}{2} \rfloor) + cn.$
Gives $T(n) = O(n \log n)$.
– For n being power of 2:
$T(n) = 2T(n/2) + cn.$
$T(n) = 4T(n/4) + 2(cn/2) + cn = 4T(n/4) + 2cn.$
$T(n) = 8T(n/8) + 4(cn/4) + 2cn = 8T(n/8) + 3cn.$
...
$T(n) = O(n \log n)$
For n not a power of two, it follows using monotonicity of the complexity formula.

Tiling Problem

– Input: A n by n square board, with one of the 1 by 1 square missing, where $n = 2^k$ for some $k \geq 1$.

– Output: A tiling of the board using a tromino, a three square tile obtained by deleting the upper right 1 by 1 corner from a 2 by 2 square.

– You are allowed to rotate the tromino, for tiling the board.

Tiling Problem

Base Case: A 2 by 2 square can be tiled.
Induction:
– Divide the square into 4, $n/2$ by $n/2$ squares.
– Place the tromino at the "center", where the tromino does not overlap the $n/2$ by $n/2$ square which was earlier missing out 1 by 1 square.
– Solve each of the four $n/2$ by $n/2$ boards inductively.

Complexity: $T(n) = 4T(n/2) + c$.
Gives, $T(n) = \Theta(n^2)$ using the master recurrence theorem.

Another way: There are $(n^2 - 1)/3$ tiles to be placed, and placing each tile takes $O(1)$ time!

Finding closest pair of points on a plane.

Input: Given n points on a plane (via coordinates (a, b), where a, b are non-negative rational numbers.)
Output: A pair of points which are closest among all pairs.
Note: There could be several closest pairs. We only choose one such pair.

Closest Pair of Points

1. Find x_c, the median of the x-cordinates of the points.
2. Divide the points into two groups of equal size based on them having x-coordinate $\leq x_c$ or $\geq x_c$ (note that several points may have same x-coordinate x_c).
3. Find closest pair among each of the two groups inductively.
4. Let the closest pair (among the two groups) have distance δ.

5. Consider all points which have x-coordinate between $x_c - \delta$ and $x_c + \delta$ and sort them according to y-coordinate
6. For each point find the distance between it and the next 7 points in the list as formed in step 5.
7. Report the shortest distance among all the distance found above (along with δ).

End

Correctness:
Note that the two points with shortest distance may be:
(a) In same group as in step 2
(b) In different groups

(a) Done in the individual group, inductively.
(b) Consider any pair of points (x, y) and (x', y') which were in different groups, but have distance $< \delta$.
Then $|x - x'| < \delta$ and $|y - y'| < \delta$.
(i) In particular, x, x' lie in the open interval $(x_c - \delta, x_c + \delta)$.

$(x_c - \delta, y + \delta)$	$(x_c - \delta/2, y + \delta)$	$(x_c, y + \delta)$	$(x_c + \delta/2, y + \delta)$	$(x_c + \delta, y + \delta)$
$(x_c - \delta, y + \delta/2)$	$(x_c - \delta/2, y + \delta/2)$	$(x_c, y + \delta/2)$	$(x_c + \delta/2, y + \delta/2)$	$(x_c + \delta, y + \delta/2)$
$(x_c - \delta, y)$	$(x_c - \delta/2, y)$	(x_c, y)	$(x_c + \delta/2, y)$	$(x_c + \delta, y)$

Without loss of generality, assume that in the sorting (as done in step 5) (x', y') appears after (x, y).
(ii) consider the eight squares given by the end points of the form:
$(x_c + k(\delta/2), y + k'(\delta/2))$,
where $-2 \leq k \leq 2$ and $0 \leq k' \leq 2$.
Each of these squares can have at most one point (otherwise, they are both in same group (of step 2) and their distance is $< \delta$).
(Here, for the points with x-coordinate exactly x_c, we place it in the square to the left/right based on which group we placed them in step 2).
Thus, considering next seven points as done in step 6 is enough.

Complexity:
$T(n) \leq 2T(\lceil \frac{n}{2} \rceil) + cn \log n$.
Gives $T(n) = O(n(\log n)^2)$.

Smarter way: Don't need to sort every time, but only once (before the start of the algorithm). This will give
$T(n) \leq 2T(\lceil \frac{n}{2} \rceil) + cn$.

Matrix Multiplication

$C = A \times B$, for $n \times n$ matrices.

$C(i,j) = \Sigma_{k=1}^{n} A(i,k) * B(k,j)$

$O(n^3)$ operations.

Strassen's Matrix Multiplication algorithm

$$A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}$$

$$B = \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix}$$

$$AB = \begin{pmatrix} a_{11}b_{11} + a_{12}b_{21} & a_{11}b_{12} + a_{12}b_{22} \\ a_{21}b_{11} + a_{22}b_{21} & a_{21}b_{12} + a_{22}b_{22} \end{pmatrix}$$

where $a_{11}b_{11}$ etc are smaller matrix multiplications.

This however doesn't help in general.

$$
\begin{aligned}
q_1 &= (a_{11} + a_{22}) * (b_{11} + b_{22}) \\
q_2 &= (a_{21} + a_{22}) * b_{11} \\
q_3 &= a_{11} * (b_{12} - b_{22}) \\
q_4 &= a_{22} * (b_{21} - b_{11}) \\
q_5 &= (a_{11} + a_{12}) * b_{22} \\
q_6 &= (a_{21} - a_{11}) * (b_{11} + b_{12}) \\
q_7 &= (a_{12} - a_{22}) * (b_{21} + b_{22})
\end{aligned}
$$

$$
AB = \begin{pmatrix} q_1 + q_4 - q_5 + q_7 & q_3 + q_5 \\ q_2 + q_4 & q_1 + q_3 - q_2 + q_6 \end{pmatrix}
$$

Complexity: $T(n) = 7T(n/2) + O(n^2)$
Gives, $T(n) = \Theta(n^{lg7})$, using the Master Theorem.
lg7 = approximately 2.807

Best known algorithm: $O(n^{2.376})$.
Best known lower bound: $\Omega(n^2)$: need to look at all the entries of the matrices.