Divide and Conquer Algorithms

As the name suggests, we divide the problem into smaller parts recursively and solve each individually. A prominent example for the same is Merge Sort.
We discuss the problem of Integer multiplication below.

* Master's Theorem :- $T(n) = aT(n/b) + \Theta(n^k)$

$$\Rightarrow T(n) \in \Theta(n^k) \text{ if } a < b^k$$

$$T(n) \in \Theta(n^k \log n) \text{ if } a = b^k$$

$$T(n) \in \Theta(n^{\log_b a}) \text{ if } a > b^k$$

* Integer multiplication :-

① Naive Algo

- Let x, y be two n-digit numbers. We assume that addition/multiplication of single bits is $O(1)$. We also assume bit shifting to be $O(1)$.
- Can be clearly seen that this is $O(n^2)$.

② Karatsuba's Approach

- Break up x, y into halves. That is,

$$x = \overset{a}{12} \,\vdots\, \overset{b}{34} \qquad \Rightarrow \quad xy = (10^2 a + b)(10^2 c + d)$$
$$y = \underset{c}{79} \,\vdots\, \underset{d}{53}$$

$$= 10^4 ac + 10^2 (bc + ad) + bd$$

$\Rightarrow$ Compute $\alpha = ac$, $\beta = bd$, $\gamma = (a+c)(b+d) \Rightarrow (bc + ad) = \gamma - \alpha - \beta$

Running time of this algo. would be:-

$$T(n) = 3T(n/2) + O(n)$$

From the master's theorem, we get $T(n) \in O(n^{\log_2 3}) \approx O(n^{1.584})$

This is a very active field with the upper bound being proven as $O(n\log n)$ in 2019!

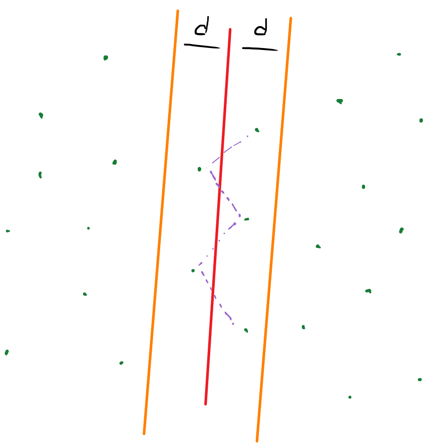
* Closest points in a plane:-

- The naive algorithm would go through all possible pairs to find the one with shortest distance, making it $O(n^2)$. We shall make it better using divide and conquer.

① Divide and Conquer:-

Divide the plane into two parts by a line. Let d_{left} be the minimum distance on the left side, and similarly d_{right} for right half.

$$d = \min(d_{left}, d_{right})$$



We then look at the points which are at a distance of 'd' from the mid-line, as a few highlighted pairs might've been missed.

Let the set of points to be checked be rep by the sequence S_y, arranged in decreasing order of y-coordinate. We prove the following lemma :-

Lemma: If the distance between P_i and P_j is less than d, then $j-i \le 15$

Proof: Divide region into squares of width $d/2$; see that a square can hold a single point.

Time of this algo :- $T(n) = \underbrace{O(n)}_{\text{Identify left and right halves}} + 2T(n/2) + \underbrace{O(n)}_{\text{Go through 'd'-region}} = O(n \log n)$

* Univariate Polynomial Multiplication :-

Let the polynomials $A(x)$, $B(x)$ be of n-order. It is quite clearly visible that the naive algorithm is $O(n^2)$ as each term in $A(x)$ needs to be multiplied with every term in $B(x)$.

Let $A(x) = A_0 + A_1 x + \ldots + A_n x^n$. We divide it into two parts as :-

$$A(x) = \underbrace{A_0(x^2)}_{\text{Even}} + x A_1(x^2) \rightarrow \text{Odd} \qquad \Rightarrow \text{Degree of } A_1, A_0 = n/2$$

i.e, $A_1(x) = A_1 + A_3 x + \ldots$

- Finding $A(x)$ from A_1, A_0 would be $O(n)$ at a given $x = a_1$.

* We need to compute $A(x)$, $B(x)$ at $2n$ points to uniquely determine the product. Instead of computing for random points, we look at the 2n-roots of unity for better computations.

- Recall :- $\omega_{j,2n} = e^{i\frac{2\pi j}{2n}} \Rightarrow \omega_{j,2n}^2 = \omega_{j,n}$

Computing $A(\omega_{j,2n}) = A_0(\omega_{j,2n}^2) + \omega_{j,2n} A_1(\omega_{j,2n}^2)$

$$= \underline{A_0(\omega_{j,n})} + \omega_{j,2n} \underline{A_1(\omega_{j,n})}$$

↳ Same problem of size $n/2$!

* Time analysis :- $T(n) = 2T(n/2) + 2n$

$$\Rightarrow T(n) = 2T(n/2) + \Theta(n) \longrightarrow O(n \log n) \text{ !}$$

$T(n)$ is time taken to compute all $(2n)^{th}$ roots of unity for a polynomial of degree 'n'.

- Reconstrunction using 2n values :-

Lemma :- Define $D(x) = \sum_{s=0}^{2n-1} C(\omega_{s,2n}) x^s$ then $c_s = \frac{1}{2n} D(\omega_{2n-s,2n})$.

The above lemma makes the reconstrunction of the polynomial into computing its value at all $\omega_{j,2n} \Rightarrow A(\omega_{j,2n}) B(j,2n) = 2T(n\log n) + O(n)$

$= T(n\log n)$