# CS157 Lecture #5 : DIVIDE & CONQUER

* like dynamic programming, it involves breaking the problem into subproblems

## PRODUCT OF TWO MATRICES

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \quad B = \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}$$

$$\Rightarrow AB = C = \left(\begin{array}{c|c} A_{11}B_{11} + A_{12}B_{21} & A_{11}B_{12} + A_{12}B_{22} \\ \hline A_{21}B_{11} + A_{22}B_{21} & A_{21}B_{12} + A_{22}B_{22} \end{array}\right)$$

↑ recur on these 8 (smaller) subproblems

$$\Rightarrow T(n) = 8\left(\frac{n}{2}\right) + n^2$$

↑ recur on 8 subproblems half the size of this problem; $n^2$ ← cost of additions at each step

* but this is the same as standard multiplication, just in a different order
  - ↳ except this version might fit in the cache - assuming B doesn't fit in the cache, it will never fit during all of standard mult. $\Rightarrow O(n^3)$ memory tranfer.
  - here, once the size of the subproblem reaches $O(k^2)$, where $k^3$ = cache size, we have $O(k)$ arithmetic memory

"Cache oblivious"

Hz = $\frac{\text{things}}{s}$

## FOURIER TRANSFORMS

↑ fourier transform

intensity of audio $f(t) \longrightarrow \hat{f}(y) = \sum_t e^{i \cdot \frac{y}{n} \cdot 2\pi \cdot t} f(t)$

$e^{i\theta} = \cos\theta + i\sin\theta$

* what's happening at 1000Hz?
  - ↳ if you discretize $f(t)$, it's just a long vector. If you take the dot product of that with $\sin(1000 \cdot 2\pi \cdot t)$, it will give you the times where the audio is at 1000Hz.
  - ↳ what about $\cos(1000 \cdot 2\pi \cdot t)$? If you multiply it with sin, you get 0. to get both components, take $\sum_t e^{i \cdot 1000 \cdot 2\pi \cdot t} f(t)$

* suppose $t \in \{0, \ldots, n-1\}$, then $y \in \{0, \ldots, n-1\}$. In what sense is this "enough"?
  - ↳ Claim that $f(t) \to \hat{f}(y)$ invertible.

* let $w$ = nth root of unity = $e^{i(\frac{1}{n})2\pi}$

$$W = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & \cdots \\ 1 & w & w^2 & w^3 & & \\ 1 & w^2 & w^4 & w^6 & \cdots & \\ 1 & w^3 & w^6 & w^9 & \cdots & \end{pmatrix} \longrightarrow Wf(t) = \hat{f}(y) \Rightarrow W : f(t) \to \hat{f}(y)$$

★ **CONVOLUTION**: A, B matrices, then $C = A * B$ is the convolution of A, B

defined as: $C \equiv 0$

for all i, j

$C(i+j) += A(i) \cdot B(j)$

↳ $[1\ 1\ 1] * [1\ 0\ 1] = [1\ 1\ 2\ 1\ 1]$

1 copy of A

1 copy of A shifted by 2

* unfair coin, $P(\text{tails}) = .3$, $P(\text{heads}) = .7$ $\Rightarrow P = [.3\ .7]$ 2nd unfair coin $[.4\ .6]$

↳ how can you compute the distribution of flipping each coin once?

$[.3\ .7] * [.4\ .6] = [.12\ .46\ .42]$

↑ no heads ↑ 1 head ↑ 2 heads

➡ $f * g = \hat{f} \cdot \hat{g}$ = fourier transform of the f.t. of f times the f.t. of g

★ how do you compute the fourier transform? (i.e. how do you compute W)

Claim: take every other column of W up to the halfway point (1st half of rows) and these are exactly the columns of the "W" of half the size.

let $n = 8$. then $\omega^8 = 1$, $\omega^{10} = \omega^2$, ... ⇒ bottom half of the matrix exactly the same as the top half

↳ but what do you do with the odd columns?

↳ use the recursive answer from each even column, multiplying the answer in each row i by $\omega^i$ to get the entry of that row and column+1.