CS 170 DISCUSSION 3

FAST FOURIER TRANSFORM AND GRAPHS

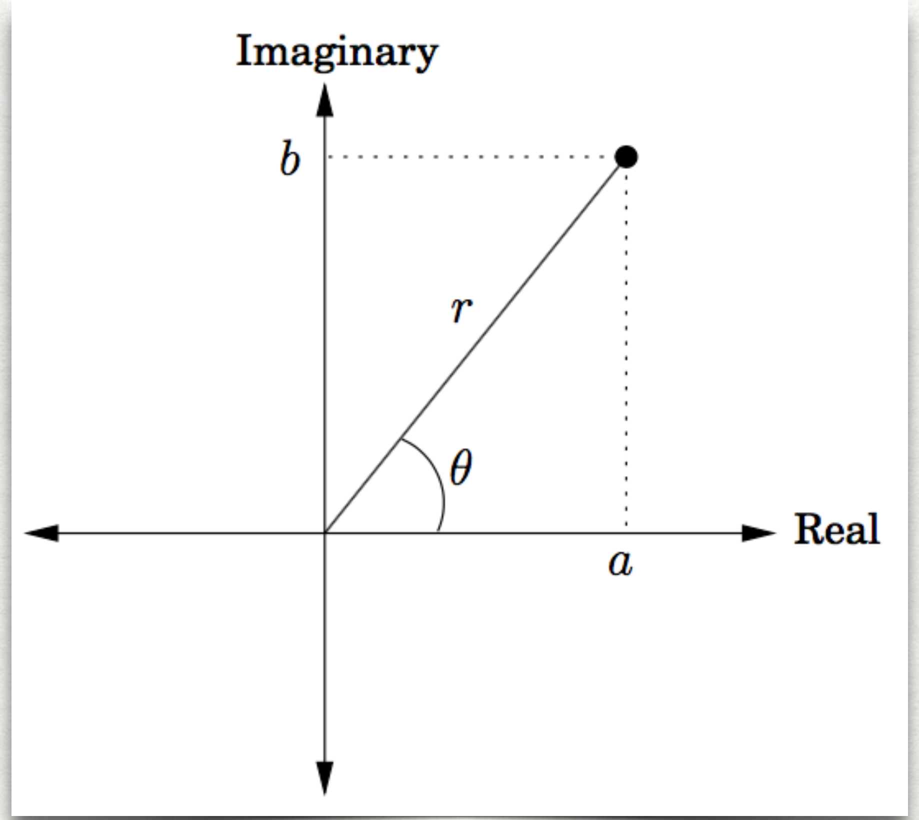
Raymond Chan
UC Berkeley Fall 17

COUPLE THINGS

- Slides and notes will be posted here
  - http://raychan3.github.io/cs170/fa17.html
- Fill this out (name and section) if plan to attend any of my sections:
  - https://goo.gl/forms/459vf15Q4ad8pFgm2
- Anonymous feedback form:
  - https://goo.gl/forms/mM8JnAvlDAcEb2sI2
- Both will be on website.

COMPLEX NUMBERS

- Complex numbers can be represented as *a + bi* or in polar coordinates with *r* and θ



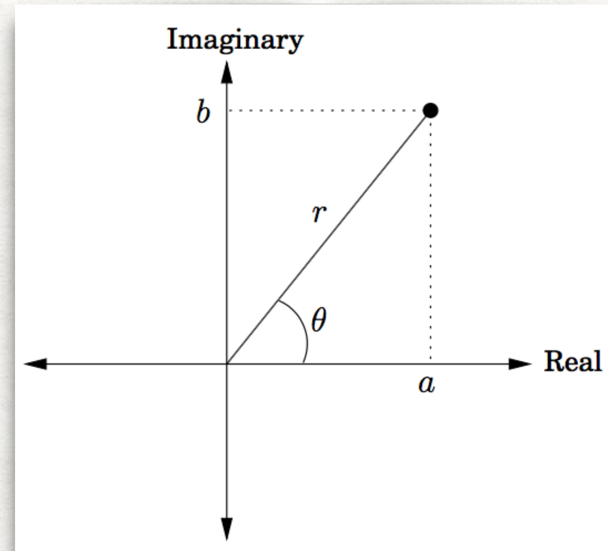
$z = a + bi$

$r = \sqrt{(a^2 + b^2)}$

$\theta = \tan^{-1}(b/a)$

COMPLEX NUMBERS

- $z = a + bi$
- $= r(\cos(\theta) + i\sin(\theta)) = re^{\theta i}$
- $\cos(\theta) + i\sin(\theta) = e^{\theta i}$ *Euler's Formula*



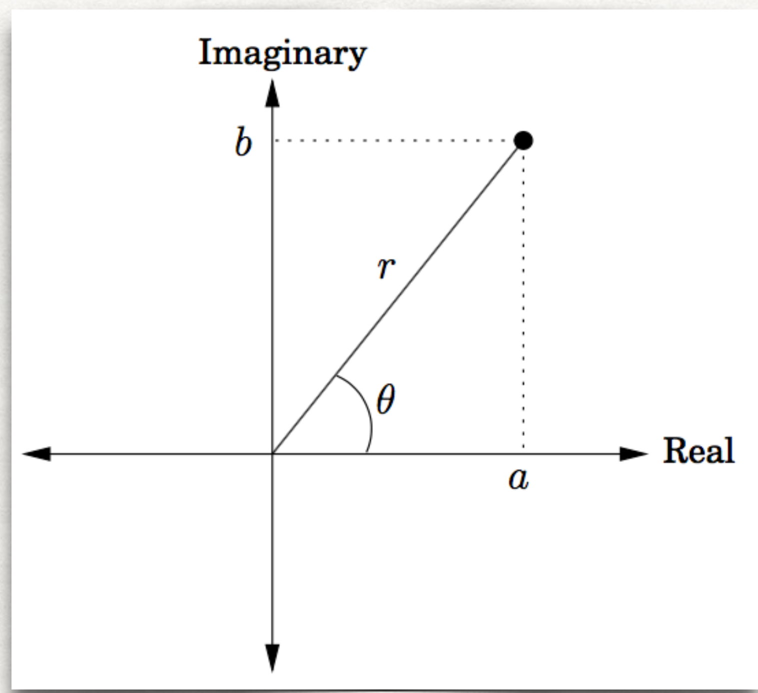
$z = a + bi$

$r = \sqrt{(a^2 + b^2)}$

$\theta = \tan^{-1}(b/a)$

COMPLEX NUMBERS

- *$z = a + bi$*
- *$= r(\cos(\theta) + i\sin(\theta)) = re^{\theta i}$*
- *$\cos(\theta) + i\sin(\theta) = e^{\theta i}$ Euler's Formula*



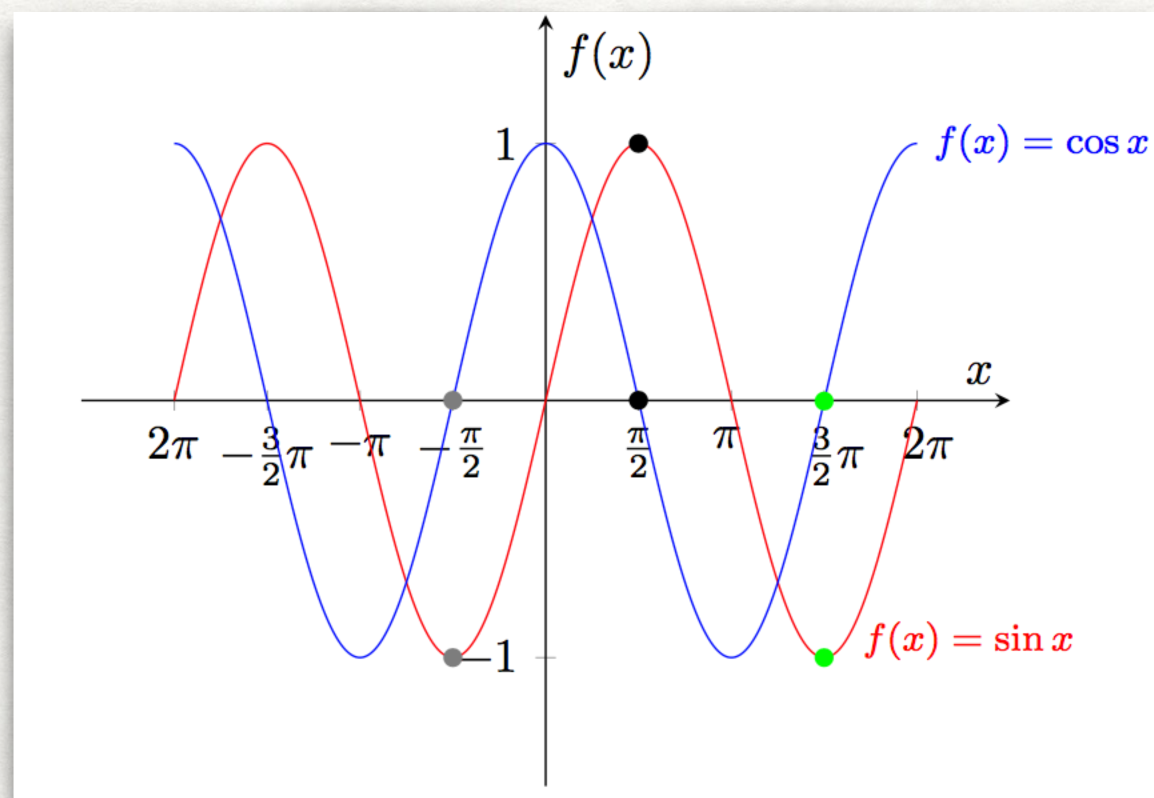
$z = a + bi$

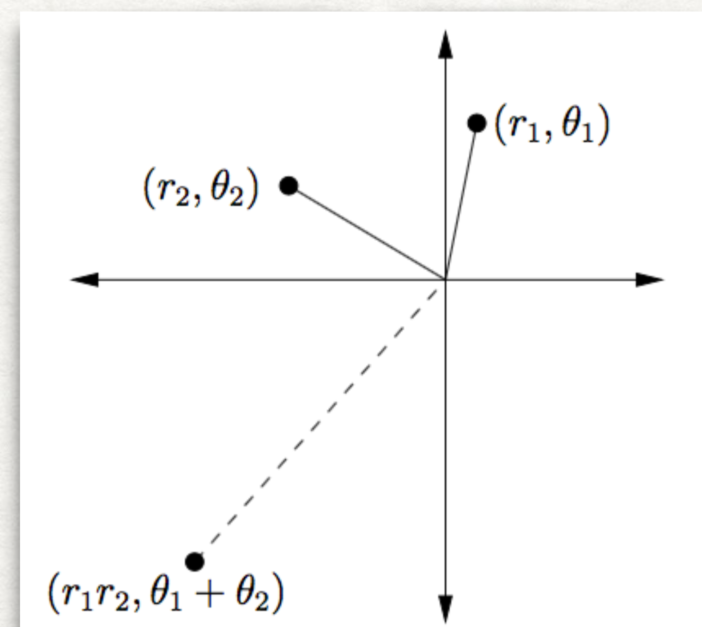
$r = \sqrt{(a^2 + b^2)}$

$\theta = \tan^{-1}(b/a)$

COMPLEX NUMBERS

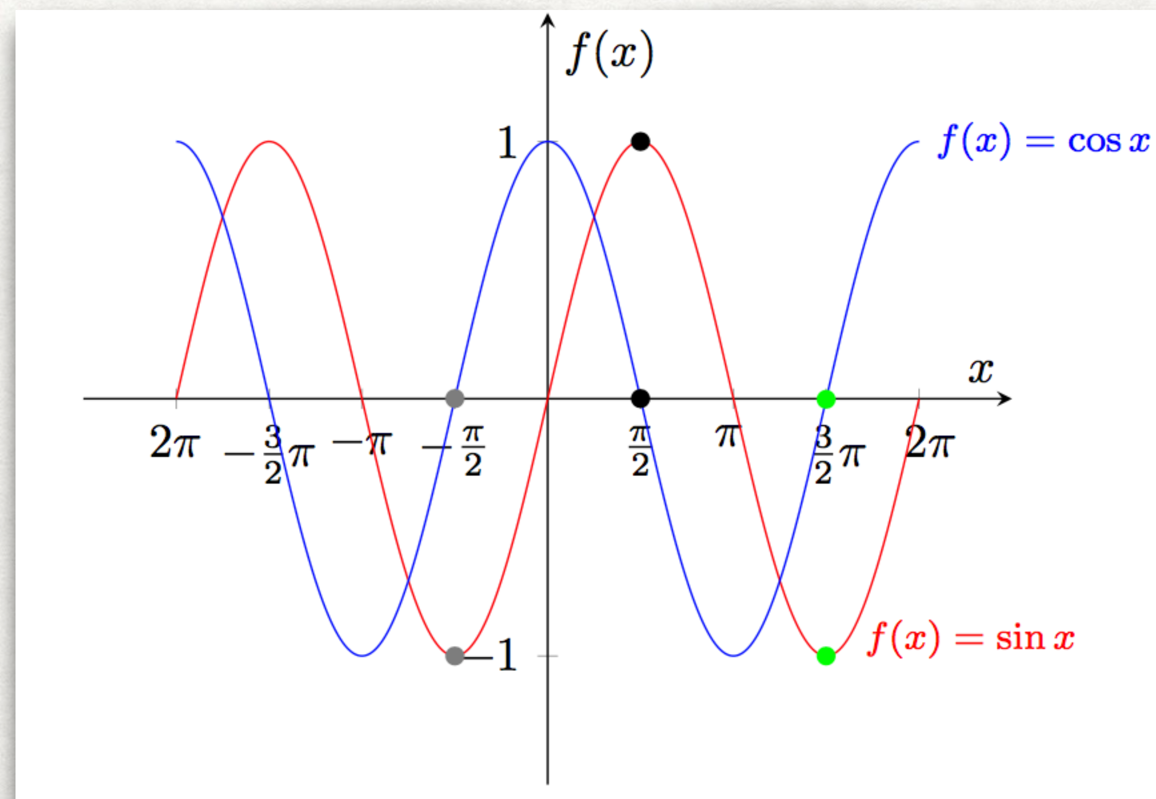
- $(r_1, \theta_1) \cdot (r_2, \theta_2) = (r_1 r_2, \theta_1 + \theta_2)$
- Multiplying by (1, π) and (1, xπ) (whiteboard)





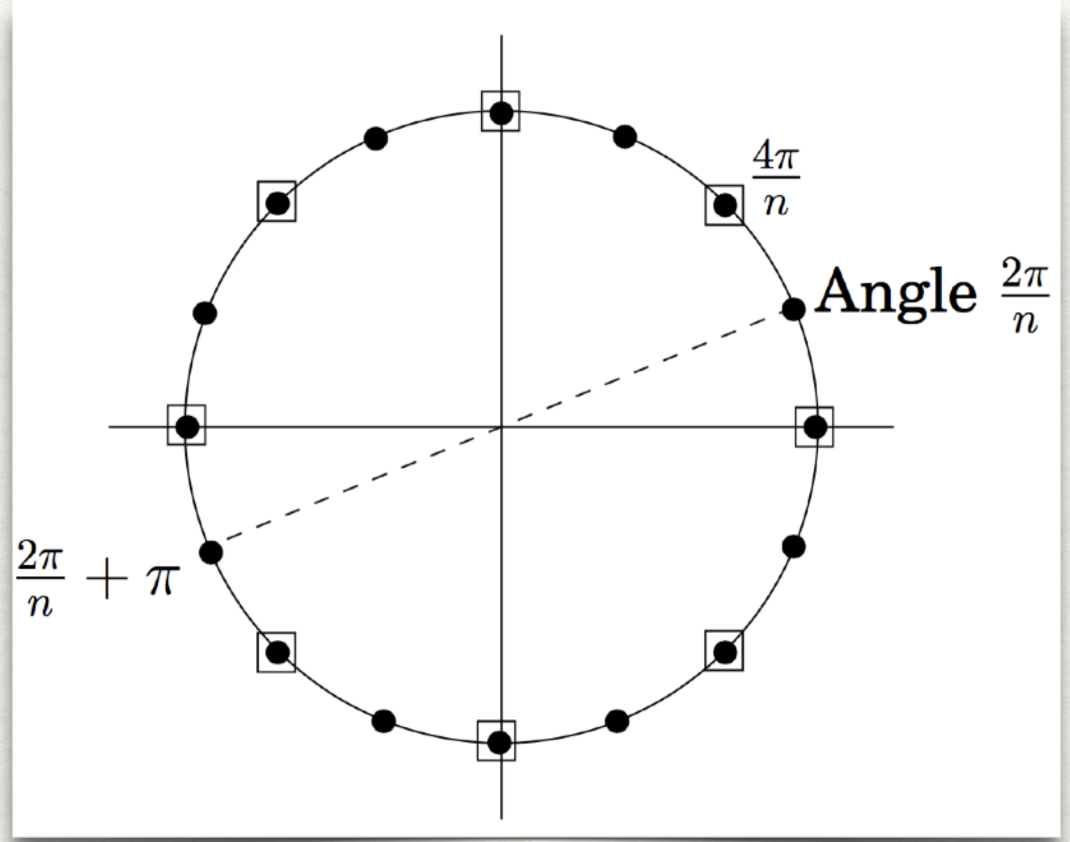
COMPLEX NUMBERS

- Adding π to θ for cos θ + isin θ till negate the value.
- We want the ***n*th complex roots of unity** (whiteboard)
  - All solutions to $z^n = 1$



COMPLEX NUMBERS

- Visualizing the roots of unity points on the unit circle.



POLYNOMIAL MULTIPLICATION

- Want to multiply two polynomials
  - $A(x)=a_0+ax^1+a_2x^2+\ldots a_dx^d$ and $B(x)=b_0+bx^1+b_2x^2+\ldots b_dx^d$
- $C(x) = A(x) \cdot B(x) = c_0+c_1x+c_2x2+\cdots+c_{2d}x^{2d}$
- where $c_k = a_0b_k + a_1b_{k-1} \ldots a_kb_0 = \Sigma_{i=0}\, k a_i b_{k-i}$
- Slow due to pairwise multiplication —> $O(d^2)$.
- Since any polynomial of degree d can be determined by $d + 1$ points, we can do better.

POLYNOMIAL MULTIPLICATION 2

- Selection ($O(n)$)
  - Pick points $x_0, x_1, \ldots, x_{n-1}$ such that $n \geq 2d + 1$
- Evaluation ($O(?)$)
  - Compute $A(x_0), A(x_1), \ldots, A(x_{n-1})$ and $B(x_0), B(x_1), \ldots, B(x_{n-1})$
- Multiplication ($O(n)$)
  - Compute $C(x_k) = A(x_k) \cdot B(x_k)$, $k = 0, 1, \ldots, n - 1$
- Interpolation ($O(?)$)
  - Recover $C(x)=c_0+c_1x+c_2x^2+\ldots c_{2d}x^{2d}$ from $C(x_k)$, $k=0, 1, \ldots, n-1$

EVALUATION VIA DIVIDE AND CONQUER

- Interpolation is inverse of evaluation.
- Evaluation needs to be sub $O(n^2)$ time.
- What points should we pick for x_i?
- Recursively use the nth root of unity.

EVALUATION VIA DIVIDE AND CONQUER

- Let $\boldsymbol{\omega} = \ (1,\ \ 2\pi/n) = e^{i2\pi/n}$
- Let $\boldsymbol{\omega}^k = \ (1,\ \ 2\pi/n) = e^{i(2k\pi/n)}$
- The *n*th roots of unity are $\boldsymbol{\omega}^0, \boldsymbol{\omega}^1, \boldsymbol{\omega}^2, \ldots, \boldsymbol{\omega}^{n-1}$
- $\boldsymbol{\omega}^i = -\boldsymbol{\omega}^{i+n/2}$
- When we square both values, we have $(\boldsymbol{\omega}^i)^2 = (\boldsymbol{\omega}^{i+n/2})^2 = \boldsymbol{\omega}^{2i}$
- $\boldsymbol{\omega}^{2i}$ is by n/2 -th roots of unity.
- Divide and conquer step is to square our values.

EVALUATION VIA DIVIDE AND CONQUER

- $A(x) = A_e(x^2) + x\, A_o(x^2)$
- Even degree polynomial: $A_e(x) = a_0 + a_2 x + a_4 x^2 + \dots$
- Odd degree polynomial: $A_o(x) = a_1 + a_3 x + a_5 x^2 + \dots$
- Recursively compute A_e and A_o on n/2 -th roots of unity and combine
- $A(\omega^i) = A_e(\omega^{2i}) + \omega^i A_o(\omega^{2i})$
- $A(\omega^{i + n/2}) = A_e(\omega^{2i}) - \omega^i A_o(\omega^{2i})$
- Pick n as the next power of 2 ≥ degree + 1

EVALUATION VIA DIVIDE AND CONQUER

- Reusing $A_e(\boldsymbol{\omega}^{2i})$ and $A_o(\boldsymbol{\omega}^{2i})$
- Runtime: $T(n) = 2T(n/2) + O(n) = O(n \log n)$

DEPTH FIRST SEARCH

```
DFS(G, v)
  label v as visited and set previsit number
  for all of v's neighbors u:
    if vertex u has not been visited:
      DFS(G, u)
      set postvisit number
```

DEPTH FIRST SEARCH

```
DFS(G, v)
  Use S as stack with v as first value
  while S is not empty
    pop first value of S as v
    if v not visited:
      label v as visited
      for all of v's neighbors u:
        push u onto stack
```

GRAPH EDGES

- Types of edges
  - **Tree edges** are edges that Depth First Search uses.
  - **Forward edges** connect nodes to a non child descendant.
  - **Back edges** lead to an ancestor in the DFS tree.
  - **Cross edges** lead to neither descendant nor ancestor. Leads to already explored nodes (with postvisit number).