Fast Fourier Transform

$w = e^{2\pi i/N}$ "primitive root of unity"

will evaluate polynomial at points $1, w, w^2, \ldots, w^{N-1}$

$$\begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & w & w^2 & \cdots & w^{n-1} \\ 1 & w^2 & w^4 & \cdots & w^{2n-2} \\ & & \vdots & & \end{bmatrix} \begin{bmatrix} c_0 \\ \vdots \\ c_{n-1} \end{bmatrix} = \begin{bmatrix} y_0 \\ \vdots \\ y_{n-1} \end{bmatrix}$$

↑ DFT (a vandermonde matrix)

*given c, can quickly compute Fc

** Note FFT does not actually compute matrix-vector multiplication

FFT: algorithm for quickly computing $P(1), P(w), \ldots P(w^{n-1})$ for some degree $< N$ polynomial P (evaluate degree n-1 polynomial at n points)
(applying DFT to vector of P's coeff $P_0 - P_{n-1}$)

$$P(x) = p_0 + p_1 x + p_2 x^2 + \ldots + p_{n-1} x^{n-1}$$
$$= (p_0 + p_2 x^2 + \ldots + p_{n-2} x^{n-2}) + x(p_1 + p_3 x^2 + \ldots + p_{n-1} x^{n-2})$$
$$= P_{even}(x^2) + x P_{odd}(x^2)$$

# flops to evaluate deg < N polynomial on N roots of unity : $T(n) = 2T\left(\frac{n}{2}\right) + \theta(N) = \theta(N \log N)$

Polynomial multiplication algorithm:
1. Use FFT to compute $\hat{a} = Fa$
2. Use FFT to compute $\hat{b} = Fb$
3. for $i=0$ to $n-1$: $\hat{c}_i \leftarrow \hat{a}_i \cdot \hat{b}_i$ ($\hat{c}$ is evaluation of C on $1, w, \ldots, w^{N-1}$)
4. $c \leftarrow F^{-1}\hat{c}$
5. return c (coeff vector of $C = A * B$)

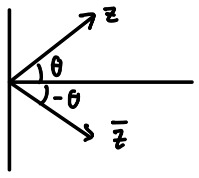
$F^{-1} = \frac{1}{N}\bar{F}$ $\bar{F}$: entrywise complex conjugate

$F^{-1}\hat{c} = \frac{1}{N}\overline{F\bar{\hat{c}}}$

↑ another application of FFT

$z = a + ib = re^{i\theta}$

$\bar{z} = a - ib = re^{-i\theta}$



Polynomial Multiplication with FFT:

$$\begin{tikzcd}
\text{Coefficient representation} \\ a_0, a_1, \dots a_d \arrow[r, shift left, "\text{Evaluation}"] & \text{Value representation} \\ F(v_0) \dots F(v_d) \arrow[l, shift left, "\text{Interpolation}"]
\end{tikzcd}$$

* d degree polynomial represented by d+1 distinct points

To multiply $p(x) \cdot g(x)$ of degree d:

$$\begin{tikzcd}
\boxed{\text{1. Given two polynomials } p(x), g(x)} \arrow[r, "\text{Evaluation}", "\text{FFT}"'] & \boxed{\text{2. } 2d+1 \text{ points that represent } p(x); \ 2d+1 \text{ points that represent } g(x)} \arrow[d, "\text{multiply points pairwise}"] \\
\boxed{\text{4. coefficient form of } p(x) \cdot g(x)} & \boxed{\text{3. } 2d+1 \text{ points that represent } p(x) \cdot g(x)} \arrow[l, "\text{Interpolation}"', "\text{FFT}^{-1}"]
\end{tikzcd}$$

Note: If the number of coefficients of a polynomial is not a power of 2, zero pad the polynomial.

$$x^2 + 2x + 7 \longrightarrow 0x^3 + x^2 + 2x + 7$$

Note: Your TA probably will not cover all the problems. This is totally fine, the discussion worksheets are not designed to be finished in an hour. They are deliberately made long so they can serve as a resource you can use to practice, reinforce, and build upon concepts discussed in lecture, readings, and the homework.

1 FFT Intro

We will use ω_n to denote the first n-th root of unity $\omega_n = e^{2\pi i/n}$. The most important fact about roots of unity for our purposes is that the squares of the $2n$-th roots of unity *are* the n-th roots of unity.

Fast Fourier Transform! The *Fast Fourier Transform* $\text{FFT}(p, n)$ takes arguments n, some power of 2, and p is some vector $[p_0, p_1, \dots, p_{n-1}]$.

Here, we describe how we can view FFT as a way to perform a specific matrix multiplication involving the DFT matrix. Note, however, that the FFT algorithm will not explicitly compute this matrix. We have written out the matrix below for convenience.

Treating p as a polynomial $P(x) = p_0 + p_1x + \ldots + p_{n-1}x^{n-1}$, the FFT computes the value of $P(x)$ for all x that are n-th roots of unity by computing the result of the following matrix multiplication in $\mathcal{O}(n \log n)$ time:

$$\begin{bmatrix} P(1) \\ P(\omega_n) \\ P(\omega_n^2) \\ \vdots \\ P(\omega_n^{n-1}) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \omega_n^1 & \omega_n^2 & \dots & \omega_n^{(n-1)} \\ 1 & \omega_n^2 & \omega_n^4 & \dots & \omega_n^{2(n-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega_n^{(n-1)} & \omega_n^{2(n-1)} & \dots & \omega_n^{(n-1)(n-1)} \end{bmatrix} \cdot \begin{bmatrix} p_0 \\ p_1 \\ p_2 \\ \vdots \\ p_{n-1} \end{bmatrix}$$

If we let $E(x) = p_0 + p_2x + \ldots p_{n-2}x^{n/2-1}$ and $O(x) = p_1 + p_3x + \ldots p_{n-1}x^{n/2-1}$, then $P(x) = E(x^2) + xO(x^2)$, and then $FFT(p, n)$ can be expressed as a divide-and-conquer algorithm:

1. Compute $E' = \text{FFT}(E, n/2)$ and $O' = \text{FFT}(O, n/2)$.
2. For $i = 0 \ldots n-1$, assign $P(\omega_n^i) \leftarrow E((\omega_n^i)^2) + \omega_n^i O((\omega_n^i)^2)$

Also observe that:

$$\frac{1}{n}\begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \omega_n^{-1} & \omega_n^{-2} & \dots & \omega_n^{-(n-1)} \\ 1 & \omega_n^{-2} & \omega_n^{-4} & \dots & \omega_n^{-2(n-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega_n^{-(n-1)} & \omega_n^{-2(n-1)} & \dots & \omega_n^{-(n-1)(n-1)} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \omega_n^1 & \omega_n^2 & \dots & \omega_n^{(n-1)} \\ 1 & \omega_n^2 & \omega_n^4 & \dots & \omega_n^{2(n-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega_n^{(n-1)} & \omega_n^{2(n-1)} & \dots & \omega_n^{(n-1)(n-1)} \end{bmatrix}^{-1}$$

(You should verify this on your own!) And so given the values $P(1), P(\omega_n), P(\omega_n^2)\ldots$, we can compute P by finding the result of the following matrix multiplication in $O(n \log n)$ time:

$$\begin{bmatrix} p_0 \\ p_1 \\ p_2 \\ \vdots \\ p_{n-1} \end{bmatrix} = \frac{1}{n}\begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \omega_n^{-1} & \omega_n^{-2} & \dots & \omega_n^{-(n-1)} \\ 1 & \omega_n^{-2} & \omega_n^{-4} & \dots & \omega_n^{-2(n-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega_n^{-(n-1)} & \omega_n^{-2(n-1)} & \dots & \omega_n^{-(n-1)(n-1)} \end{bmatrix} \cdot \begin{bmatrix} P(1) \\ P(\omega_n) \\ P(\omega_n^2) \\ \vdots \\ P(\omega_n^{n-1}) \end{bmatrix}$$

This can be done in $O(n \log n)$ time using a similar divide and conquer algorithm.

(a) Let $p = [p_0]$. What is FFT$(p, 1)$?

DFT: [1] $FFT(p,1) = p_0$

polynomial is a constant

(b) Use the FFT algorithm to compute FFT([1, 4], 2) and FFT([3, 2], 2).

FFT (p,n) = polynomial p evaluated at the nth roots of unity

$FFT([1,4],2) = E(x^2) + x\,O(x^2)$ using 2nd roots of unity 1, -1

even $FFT([1],1) = 1$ odd $FFT([4],1) = 4$

$P(1) = 1 + 1 \cdot 4 = 5$

$P(-1) = 1 - 1 \cdot 4 = -3$

$\boxed{FFT([1,4],2) = [5,-3]}$

$FFT([1],1) = E(x^2)$, $FFT([4],1) = O(x^2)$ } evaluated at $w_1 = w_2^2$

1st roots of unity   2nd roots of unity

$FFT([3,2],2)$

$FFT([3],1) = 3$ $FFT([2],1) = 2$

$P(1) = 3 + 1 \cdot 2 = 5$

$P(-1) = 3 - 1 \cdot 2 = 1$

$\boxed{FFT([3,2],2) = [5,1]}$

(c) Use your answers to the previous parts to compute FFT([1, 3, 4, 2], 4).

$FFT[[1,3,4,2],4]$

$FFT([1,4],2) = [5,-3]$ $FFT([3,2],2) = [5,1]$

E(1) E(-1)   O(1) O(-1)

4th roots of unity: $1, i, -1, -i$

$P(1) = E(1) + 1 \cdot O(1) = 5 + 1 \cdot 5 = 10$

$P(i) = E(i^2) + i\,O(i^2) = E(-1) + i\,O(-1) = -3 + i \cdot 1 = -3 + i$

$P(-1) = E((-1)^2) - 1 \cdot O((-1)^2) = E(1) - O(1) = 5 - 5 = 0$

$P(-i) = E((-i)^2) - i\,O((-i)^2) = E(-1) - i\,O(-1) = -3 - i$

$\boxed{FFT([1,3,4,2],4) = [10, -3+i, 0, -3-i]}$

(d) Describe how to multiply two polynomials $p(x), q(x)$ in coefficient form of degree at most d.

1. zero pad p and q so that their number of terms is a power of 2 (they have degree $2^k - 1$, where $k > 2d$ and k is power of 2)
2. Evaluate p and q at 2^k points using FFT
3. Compute dot product between these points
4. Interpolate to get pq using FFT^{-1}

2 (Challenge Problem) Cartesian Sum

Let A and B be two sets of integers in the range 0 to $10n$. The *Cartesian sum* of A and B is defined as

$$A + B = \{a + b \mid a \in A, b \in B\}$$

i.e. all sums of an element from A and an element with B. For example, $\{1, 3\} + \{2, 4\} = \{3, 5, 7\}$.

Note that the values of $A + B$ are integers in the range 0 to $20n$. Design an algorithm that finds the elements of $A + B$ in $\mathcal{O}(n \log n)$ time, which additionally tells you for each $c \in A + B$, *how many* pairs $a \in A, b \in B$ there are such that $a + b = c$.

Hint: Notice that $(x^1 + x^3) \cdot (x^2 + x^4) = x^3 + 2x^5 + x^7$

Define two polynomials:

$$P = x^{a_1} + x^{a_2} + \ldots + x^{|A|} \qquad Q = x^{b_1} + x^{b_2} + \ldots + x^{|B|}$$

Multiply them:

$$R(x) = P(x)Q(x) = r_0 + r_1 x + \ldots + r_{20n} x^{a_{10n} b_{10n}}$$

$$r_k = \sum_{j=0}^{k} p_j q_{k-j}$$

$$p_j q_{k-j} = \begin{cases} 1 & \text{if } j \in A \text{ and } k-j \in B \\ 0 & \text{otherwise} \end{cases}$$

$\hookleftarrow$ serves as counter

Cartesian sum $A+B$ found in exponents of R.
Counts found in corresponding coefficient.

4 Connectivity vs Strong Connectivity

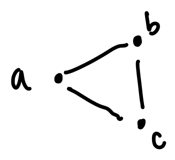
(a) Prove that in any connected undirected graph $G = (V, E)$ there is a vertex $v \in V$ such that removing v from G gives another connected graph.

T= DFS tree of G

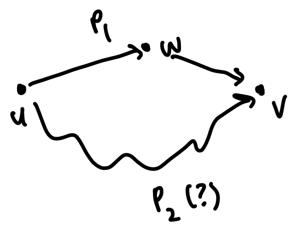
v = a leaf of T

Removing v does not disconnect the graph because any path $u \to w$ does not have to pass through v

(b) Give an example of a strongly connected directed graph $G = (V, E)$ such that, for *every* $v \in V$, removing v from G gives a directed graph that is not strongly connected.



(c) Let $G = (V, E)$ be a connected undirected graph such that G remains connected after removing any vertex. Show that for every pair of vertices u, v where $(u, v) \notin E$ there exist two different u-v paths.



G-w still connected

removing w destroys P_1

Therefore there must exist a P_2

5 Short Answer

For each of the following, either prove the statement is true or give a counterexample to show it is false.

(a) If (u, v) is an edge in an undirected graph and during DFS, $\text{post}(v) < \text{post}(u)$, then u is an ancestor of v in the DFS tree.

True Case 1: $\text{pre}(u) < \text{pre}(v) < \text{post}(v) < \text{post}(u)$ u is ancestor $u \to v$

Case 2: $\text{pre}(v) < \text{post}(v) < \text{pre}(u) < \text{post}(u)$ X

not possible to visit and return from v first bc edge $u - v$

(b) In a directed graph, if there is a path from u to v and $\text{pre}(u) < \text{pre}(v)$ then u is an ancestor of v in the DFS tree.

False

$$\begin{tikzcd} u \arrow[r, bend right] & w \arrow[l, bend right] \arrow[r] & v \end{tikzcd}$$

Start DFS from w

Just because you visit one vertex before another doesn't mean that one is an ancestor of the other.

(c) In any connected undirected graph G there is a vertex whose removal leaves G connected.

True

remove a leaf in DFS

Discussion 2 Attendance/Feedback

bit.ly/disc02-feedback