Lecture 3

Let us recall our algorithm for multiplying two polynomials $A(x)$ and $B(x)$:

Step 1: Convert $A(x)$ and $B(x)$ from coefficient form to point-value representation.

Step 2: Obtain their product $P(x)$ in point-value representation.

Step 3: Convert $P(x)$ into coefficient representation.

We have seen how to implement Step 1 in $O(n\log n)$ time, where $A(x)$ and $B(x)$ are degree $n-1$ polynomials.

Easy to see that Step 2 takes $O(n)$ time.

What about Step 3?

$$P(x) = p_0 + p_1 x + p_2 x^2 + \dots + p_{N-1} x^{N-1}$$

where $p_0, p_1, \dots, p_{N-1}$ are unknown to us.
Recall that N is the smallest power of 2 that is $\geq 2n-1$.

What we know are $P(1), P(\omega), P(\omega^2), \dots, P(\omega^{N-1})$.
That is,

$$\begin{bmatrix} P(1) \\ P(\omega) \\ \vdots \\ P(\omega^{N-1}) \end{bmatrix} = \begin{bmatrix} 1 & 1 & \dots & 1 \\ 1 & \omega & \dots & \omega^{N-1} \\ & & \vdots & \\ 1 & \omega^{N-1} & \dots & \omega^{(N-1)^2} \end{bmatrix} \begin{bmatrix} p_0 \\ p_1 \\ \vdots \\ p_{N-1} \end{bmatrix}$$

So $\begin{bmatrix} p_0 \\ p_1 \\ \vdots \\ p_{N-1} \end{bmatrix} = \begin{bmatrix} 1 & 1 & \dots & 1 \\ 1 & \omega & \dots & \omega^{N-1} \\ & & \vdots & \\ 1 & \omega^{N-1} & \dots & \omega^{(N-1)^2} \end{bmatrix}^{-1} \begin{bmatrix} P(1) \\ \vdots \\ P(\omega^{N-1}) \end{bmatrix}$

Let X denote the above $N \times N$ matrix, i.e.

$$X = \begin{bmatrix} 1 & 1 & \dots & 1 \\ 1 & \omega & \dots & \omega^{N-1} \\ & & \vdots & \\ 1 & \omega^{N-1} & \dots & \omega^{(N-1)^2} \end{bmatrix}$$

What is the inverse of X? Any guesses?

A reasonable guess is $\begin{bmatrix} 1 & 1 & \dots & 1 \\ 1 & 1/\omega & \dots & 1/\omega^{N-1} \\ & & & \\ 1 & 1/\omega^{N-1} & \dots & 1/\omega^{(N-1)^2} \end{bmatrix}$

Call this matrix Y.
What is XY?

<u>Claim</u>. $XY = \begin{bmatrix} N & 0 & \dots & 0 \\ 0 & N & \dots & 0 \\ 0 & 0 & \dots & N \end{bmatrix}$

<u>Proof</u>. Consider the (j,k)-th element of XY.
This is the dot product of

$$\left(1 \;\; \omega^j \;\; \omega^{2j} \; \dots \; \omega^{(N-1)j}\right)\left(1 \;\; \frac{1}{\omega^k} \; \dots \; \frac{1}{\omega^{(N-1)k}}\right)$$

$$= 1 + \omega^{j-k} + \omega^{2(j-k)} + \dots + \omega^{(N-1)(j-k)}$$

When $j = k$, the above sum is N.
When $j \neq k$, this is $1 + \omega^t + \dots + \omega^{t(N-1)}$
where $t = j-k$.

The sum $1 + \omega^t + \dots + \omega^{t(N-1)}$

$$= \frac{\omega^{tN} - 1}{\omega^t - 1} = 0 \quad \text{since } \omega^N = 1. \quad \square$$

So the inverse of X is $\frac{1}{N}\begin{bmatrix} 1 & 1 & \dots & 1 \\ 1 & \omega^{N-1} & \dots & \omega^{(N-1)^2} \\ 1 & \omega^{N-2} & \dots & \omega^{(N-1)(N-2)} \\ & & \vdots & \\ 1 & \omega & \dots & \omega^{N-1} \end{bmatrix}$

(note that $\frac{1}{\omega} = \omega^{N-1}$)

Our problem now is to compute the following:

$$\frac{1}{N}\begin{bmatrix} 1 & 1 & \dots & 1 \\ 1 & \omega^{N-1} & \dots & \omega^{(N-1)^2} \\ & & \vdots & \\ 1 & \omega & \dots & \omega^{N-1} \end{bmatrix}\begin{bmatrix} q_0 \\ q_1 \\ \vdots \\ q_{N-1} \end{bmatrix}$$

where $q_0 = P(1)$, $q_1 = P(\omega)$, $\dots$, $q_{N-1} = P(\omega^{N-1})$.

The above matrix multiplication is called Inverse Fourier Transform.

Let us create the polynomial

$$Q(x) = q_0 + q_1 x + \dots + q_{N-1} x^{N-1}$$

We need to evaluate $Q(x)$ at $x = 1, \omega^{N-1}, \omega^{N-2}, \dots, \omega$ which are the N-th roots of 1.

Or in other words, evaluate $Q(x)$ by FFT at $x = 1, \omega, \dots, \omega^{N-1}$ to obtain the vector

$\begin{bmatrix} Q(1) \\ Q(\omega) \\ \vdots \\ Q(\omega^{N-1}) \end{bmatrix}$ and permute this vector and scale it down by N $\rightsquigarrow \frac{1}{N}\begin{bmatrix} Q(1) \\ Q(\omega^{N-1}) \\ \vdots \\ Q(\omega) \end{bmatrix}$ These are the coefficients of P.

We will now look at a problem in graphs.

- we will see a randomized algorithm for this problem.

The algorithm will make random choices and it may sometimes return the wrong answer. However its error probability will be $\leq 1/4$.

- such an algorithm is called a Monte Carlo algorithm.

The problem we will consider is the global min-cut problem.

the set of vertices

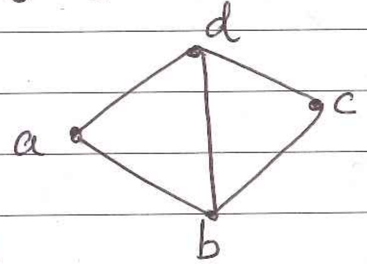
the set of edges

Input: a connected undirected graph $G = (V, E)$

A cut C is a subset of E such that the graph $(V, E-C)$ is disconnected.

That is, a cut C is a set of edges whose removal leaves the resulting graph disconnected.

For example, consider the graph:



Here $V = \{a, b, c, d\}$ and $E = \{(a,b), (b,c), (c,d), (a,d), (b,d)\}$.

$C = \{(a,b), (a,d)\}$ is a cut. Similarly, $C' = \{(a,b), (c,d), (b,d)\}$ is a cut and so on.

A cut corresponds to a partition of V into A and $V-A$, where neither $V-A$ nor A is empty.

We are interested in a min-cut, i.e., a minimum-size set of edges whose removal leaves the graph disconnected.

- such a cut is also called a global min-cut

(we will later look at s-t min-cut problem where we are given a pair of vertices s and t and we want a min-cut that leaves s and t in different connected components)

Let us fix a (global) min-cut C.
Suppose C has k edges, i.e., $|C| = k$.

Exercise. Show that G has at least $\frac{nk}{2}$ edges, where $|V| = n$.

What is the probability that an edge picked uniformly at random from G belongs to C?

- the answer is $\frac{|C|}{|E|} = \frac{k}{|E|} \leq \frac{k}{nk/2}$

since $|E| \geq nk/2$.

So the answer is $\leq 2/n$.

Let e be the edge picked uniformly at random.
We showed that $Pr[e \in C] \leq 2/n$.

This means with high probability, an edge picked uniformly at random is not in C.

Now we want to get rid of e so that we have a smaller graph where C is still a min-cut.

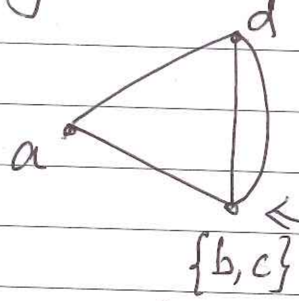
Going back to our earlier example, suppose $e = (b, c)$ is the edge picked uniformly at random.

What should we do with e so as to get a new graph (in fact, a smaller graph) without e such that our min-cut $C = \{(a, b), (a, d)\}$ is still a min-cut in this new graph?

Idea: Contract this edge.

What does this mean? Merge the endpoints of e into a "super-vertex". Self-loops are removed but all parallel edges are preserved.

Example: Contracting (b, c) in that graph results in the graph



this is the vertex formed by unifying b and c.

Let G_0 be the given graph G.

$$G_0 \xrightarrow[\text{random edge}]{\text{contract a}} G_1$$

What do we do next? We have G_1 on $n-1$ vertices. Let us repeat the same step.

$$G_1 \xrightarrow[\text{random edge}]{\text{contract a}} G_2$$

Again repeat this step on G_2.

So we get $G_0 \longrightarrow G_1 \longrightarrow G_2 \longrightarrow G_3 \longrightarrow \cdots \longrightarrow G_{n-2}$