CS 170 Dis 02

Released on 2017-09-10

1 Cubed Fourier

(a) Cubing the 9^{th} roots of unity gives the 3^{rd} roots of unity. Next to each of the third roots below, write down the corresponding 9^{th} roots which cube to it. The first has been filled for you. *We will use ω_9 to represent the primitive 9^{th} root of unity, and ω_3 to represent the primitive 3^{rd} root.*

$\omega_3^0 : \omega_9^0,$,

$\omega_3^1 :$, ,

$\omega_3^2 :$, ,

(b) You want to run FFT on a degree-8 polynomial, but you don't like having to pad it with 0s to make the (degree+1) a power of 2. Instead, you realize that 9 is a power of 3, and you decide to work directly with 9th roots of unity and use the fact proven in part (a). Say that your polynomial looks like $P(x) = a_0 + a_1x + a_2x^2 + \ldots + a_8x^8$. **How do you split $P(x)$ to use the fact proven in part (a) to your advantage?** Provide either the polynomial, or explain how the vector can be divided to recurse on. *Recall that for the FFT algorithm shown in the book, we split a given polynomial $Q(x) = A_e(x^2) + xA_o(x^2)$, and we define what $A_e(x^2)$ and $A_o(x^2)$ are. Correspondingly, in lecture you saw the $\vec{a}$ split into $\vec{a}_{even}$ and $\vec{a}_{odd}$.*

2 Vandermonde Matrices

Recall that a square Vandermonde matrix is of the following form:

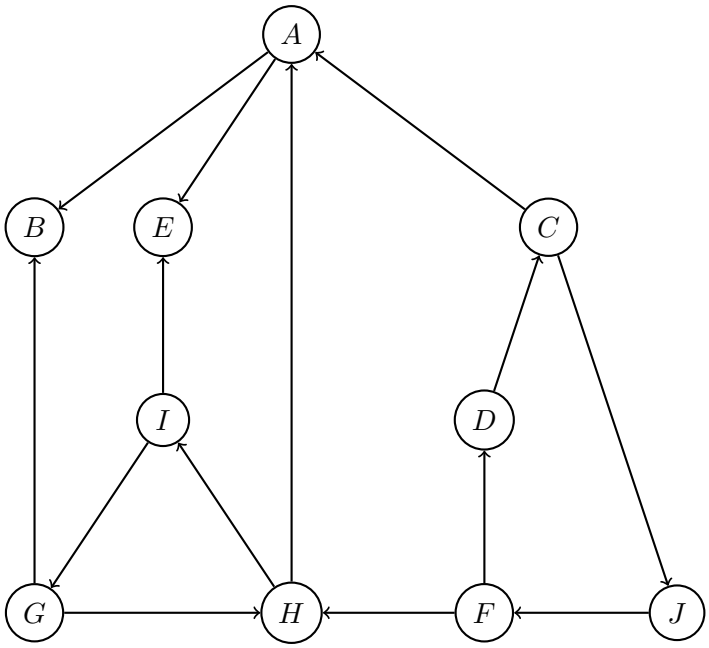
$$\begin{bmatrix} 1 & \alpha_1 & \alpha_1^2 & \ldots & \alpha_1^{n-1} \\ 1 & \alpha_2 & \alpha_2^2 & \ldots & \alpha_2^{n-1} \\ & & \ldots & & \\ 1 & \alpha_n & \alpha_n^2 & \ldots & \alpha_n^{n-1} \end{bmatrix}$$

Some real matrices have the nice property that $M^{-1} = cM^T$ for some constant c, or even that $M^{-1} = M$. Show that if M is a *real* n-by-n Vandermonde matrix and $n > 2$, then M^{-1} is not equal to cM^T for some constant c. (Hint: It suffices to show that MM^T is not a diagonal matrix, i.e. at least one of its off-diagonal entries is non-zero).

(Why are we asking you to show this? As seen in the textbook, both evaluating a polynomial at n points and going from the value of a polynomial at n points to its coefficients were equivalent to solving for either x or b in the equality $Mx = b$, where M is a Vandermonde matrix, x is a vector of the polynomial's coefficients, and b is the value of the polynomial at the distinct points $\alpha_1, \alpha_2 \ldots \alpha_n$, using the same α_i that define the Vandermonde matrix. In particular, for FFT the matrix M used has the nice property that its conjugate M^* is

proportional to its inverse M^{-1}. This exercise shows that if we want to use a matrix of only real values in FFT, we won't be able to achieve this nice property, which is what allows inverse-FFT to look so similar to FFT.)

3 Graph Traversal



(a) For the directed graph above, perform DFS starting from vertex A, breaking ties alphabetically. As you go, label each node with its pre- and post-number, and mark each edge as **T**ree, **B**ack, **F**orward or **C**ross.

(b) What are the strongly connected components of the above graph?

(c) Draw the DAG of the strongly connected components of the graph.

4 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.

(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.

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

5 True Source

Design an efficient algorithm that given a directed graph G determines whether there is a vertex v from which every other vertex can be reached. (Hint: first solve this for directed acyclic graphs. Note that running DFS from every single vertex is not efficient.)

6 Path Problems on DAGs

Let G be a directed, acyclic graph.

(a) Give an efficient algorithm to compute the number of edges in the longest path in G.

(b) Give an efficient algorithm that takes G and two vertices s, t in its input and computes the number of paths from s to t.