CS 170 Dis 03

Released on 2019-02-04

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.*

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

$\omega_3^1 :$, ,

$\omega_3^2 :$, ,

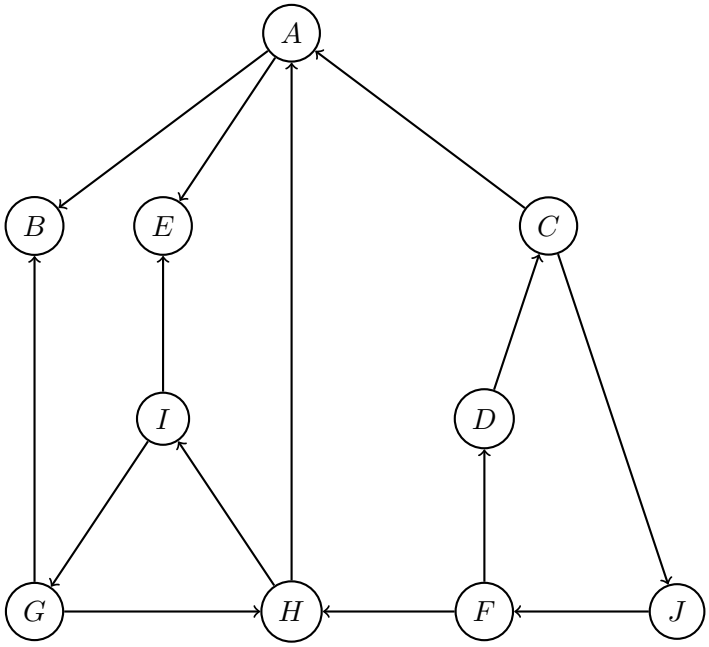
(b) You want to run FFT on a degree-8 polynomial, but you don't like having to pad it with 0s to make the (degree+1) a power of 2. Instead, you realize that 9 is a power of 3, and you decide to work directly with 9th roots of unity and use the fact proven in part (a). Say that your polynomial looks like $P(x) = a_0 + a_1x + a_2x^2 + \ldots + a_8x^8$. **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 $\vec{a}_{odd}$.*

Solution:

(a) $\omega_3^0 : \omega_9^0, \omega_9^3, \omega_9^6$
$\omega_3^1 : \omega_9^1, \omega_9^4, \omega_9^7$
$\omega_3^2 : \omega_9^2, \omega_9^5, \omega_9^8$

(b) Let $P(x) = P_1(x^3) + xP_2(x^3) + x^2P_3(x^3)$
where $P_1(x^3) = a_0 + a_3x^3 + a_6x^6$.
and $P_2(x^3) = a_1 + a_4x^3 + a_7x^6$.
and $P_3(x^3) = a_2 + a_5x^3 + a_8x^6$.

2 Graph Traversal

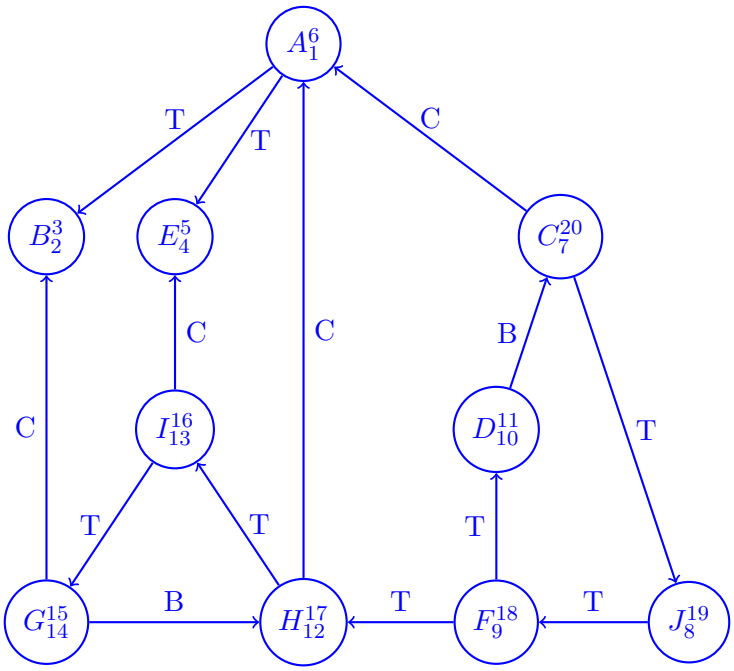


(a) For the directed graph above, perform DFS starting from vertex A, breaking ties alphabetically. As you go, label each node with its pre- and post-number, and mark each edge as **T**ree, **B**ack, **F**orward or **C**ross.

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

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

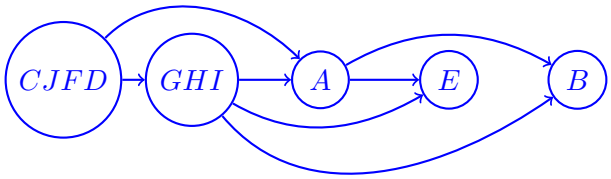
Solution:



(a)

(b)

$$\{A\}, \{B\}, \{E\}, \{G, H, I\}, \{C, J, F, D\}$$



(c)

3 Finding Clusters

We are given a directed graph $G = (V, E)$, where $V = \{1, \ldots, n\}$, i.e. the vertices are integers in the range 1 to n. For every vertex i we would like to compute the value $m(i)$ defined as follows: $m(i)$ is the smallest j such that vertex j is reachable from vertex i. (As a convention, we assume that i is reachable from i.) Show that the values $m(1), \ldots, m(n)$ can be computed in $O(|V| + |E|)$ time.

Solution: Let G^R be the graph G with its edge directions reversed. The algorithm is as follows.

procedure DFS-CLUSTERS(G)

while there are unvisited nodes in G **do**
  Run DFS on G^R starting from the numerically-first unvisited node i
  for j visited by this DFS **do** $m(j) := i$

To see that this algorithm is correct, note that if a vertex i is assigned a value then that value is the smallest of the nodes that can reach it in G^R, and every node is assigned a value because the loop does not terminate until this happens. Now observe that the set of vertices reachable by i in G^R is the set of vertices which can reach i in G.

The running time is $O(|V|+|E|)$ since computing G^R can be done in linear time (or faster if we use an adjacency matrix!), and we process every vertex and edge exactly once in the DFS.