1 Midterm Prep: Divide and Conquer

Given a set of points \( P = \{(x_1, y_1), (x_2, y_2), \ldots (x_n, y_n)\} \), a point \((x_i, y_i) \in P\) is Pareto-optimal if there does not exist any \( j \neq i \) such that such that \( x_j > x_i \) and \( y_j > y_i \). In other words, there is no point in \( P \) above and to the right of \((x_i, y_i)\). Design a \( O(n \log n) \)-time divide-and-conquer algorithm that given \( P \), outputs all Pareto-optimal points in \( P \).

(Hint: Split the array by \( x \)-coordinate. Show that all points returned by one of the two recursive calls is Pareto-optimal, and that you can get rid of all non-Pareto-optimal points in the other recursive call in linear time).

2 Midterm Prep: FFT

(a) Cubing the 9\textsuperscript{th} roots of unity gives the 3\textsuperscript{rd} roots of unity. Next to each of the third roots below, write down the corresponding 9\textsuperscript{th} roots which cube to it. The first has been filled for you. We will use \( \omega_9 \) to represent the primitive 9\textsuperscript{th} root of unity, and \( \omega_3 \) to represent the primitive 3\textsuperscript{rd} root.

\[
\begin{align*}
\omega_0^0 & : \omega_9^0, \\
\omega_1^1 & : , \\
\omega_2^2 & : , 
\end{align*}
\]

(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 \( P(x) = a_0 + a_1x + a_2x^2 + \ldots + a_8x^8 \). Describe a way to split \( P(x) \) into three pieces (instead of two) so that you can make an FFT-like divide-and-conquer algorithm.

(c) What is the runtime of FFT when we divide the polynomial into three pieces instead of two?
3 Midterm Prep: DFS

Suppose we just ran DFS on a directed (not necessarily strongly connected) graph $G$ starting from vertex $r$, and have the pre-visit and post-visit numbers $\text{pre}(v), \text{post}(v)$ for every vertex. We now delete vertex $r$ and all edges adjacent to it to get a new graph $G'$. Given just the arrays $\text{pre}(v), \text{post}(v)$, describe how to modify them to arrive at new arrays $\text{pre}'(v), \text{post}'(v)$ such that $\text{pre}'(v), \text{post}'(v)$ are a valid pre-visit and post-visit ordering for some DFS of $G'$.

4 Midterm Prep: Shortest Paths

You are given a strongly connected directed graph $G = (V, E)$ with positive edge weights, and there is a special node $v_0 \in V$. Give an efficient algorithm that computes the length of the shortest path from $s$ to $t$ that passes through $v_0$ for all pairs $s, t$. Your algorithm should take $O(|V|^2 + |E| \log |V|)$ time.