ECE406 **Algorithm Design & Analysis** **Winter 2025**

Assignment #3

due Fri Feb 14 11:59pm

See Appendix for submission instructions

Exercise 1 (Fast Fourier Transform) In this question we will use the FFT to multiply the following two 4-bit binary numbers:

$$a = 1101 \quad \text{and} \quad b = 1011.$$

Download the file `A3Q1.py`. Steps (i) and (v) below are already completed in the script. You will fill in parts (ii), (iii) and (iv) and submit on LEARN (plus a screenshot on Crowdmark).

(i) The first step is to express a and b as polynomial functions $A(x)$ and $B(x)$:

$$A(x) = a_0 + a_1 x + a_2 X^2 + a_3 x^3 \quad \text{and} \quad B(x) = b_0 + b_1 x + b_2 X^2 + b_3 x^3$$

where a_0 is the least significant bit of a (i.e., $a_0 = 0$) and similarly for b_0. Define two arrays `Acoeff` $= [a_0 \cdots a_3]$ and `Bcoeff` $= [b_0 \cdots b_3]$ containing the coefficients of the polynomials $A(x)$ and $B(x)$.

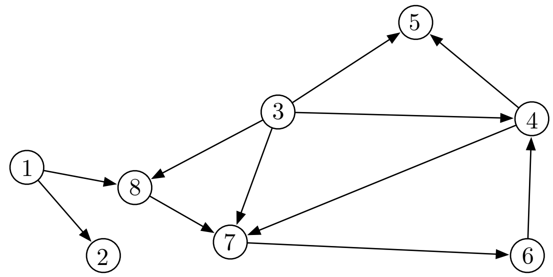
(ii) Next, take the FFT of each polynomial to obtain the value representation of A and B. We can do this using numpy's fft function `fft(coeffs,` n`)`. Since each polynomial has degree $d = 3$, evaluate at $n \geq 2d + 1$ roots of unity.

(iii) Next we find the value representation of $C(x) = A(x) \cdot B(x)$ by simply computing $C(x_k) = A(x_k)B(x_k)$ for each $k = 1, \ldots, n-1$, where $A(x_k)$ and $B(x_k)$ are from (ii).

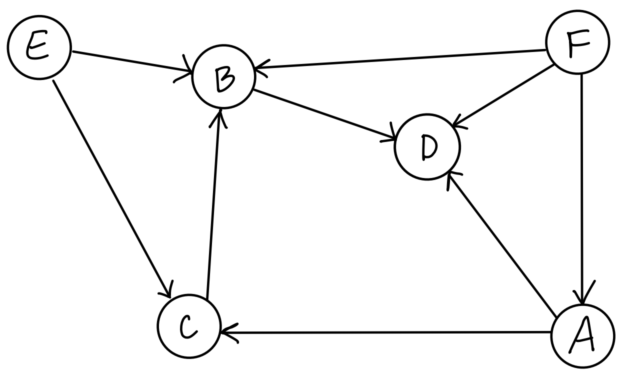
(iv) Next, take the inverse FFT of the value representation of $C(x)$ to obtain the coefficients `Ccoeff`. This can be done using the numpy function `ifft(Cvalues)`. From this, you now have the coefficients of $C(x)$. If you evaluate $C(2)$, you should get the product $a \cdot b$ in decimal (rather than binary).

(v) How would you keep the result in binary, directly from the coefficients of $C(x)$? That is, defining $c = a \cdot b$, take the the coefficients `Ccoeff` (which may or may not be binary) and compute a python array `c` such that `c[0]` is the least significant bit of the binary number c, and `c[n-1]` is the most significant bit.

Exercise 2 (DFS Edge Types) Perform a depth-first search on the following graph by hand; whenever there is a choice of vertices, pick the one with the lower number. Classify each edge as a tree edge, forward edge, back edge, or cross edge, and give the `pre` and `post` number of each vertex.



Exercise 3 (Topologically Sorting a DAG) Consider the following directed acyclic graph.



(i) Perform a depth-first search by hand, computing the `pre` and `post` times for each vertex; whenever there's a choice of vertices, use alphabetical ordering. Use the `post` numbers to generate a topological sort of the vertices. That is, assign a number to each vertex such that for every edge (u, v) in the topological sort, u has a lower number than v.

(ii) So far we have looked at generating a topological sort using the `post` numbers. Next we will explore whether or not the `pre` numbers can be used to derive a valid topological sort.

Using the result you got from running DFS on G (including `pre` and `post` numbers), find a new ordering by assigning one number to each node descending `pre` numbers. Is this a valid topological sort? In other words, for all edges (u, v) in the new sorting, is u lower than v?

Exercise 4 (Breadth-First Search) Suppose that in BFS we initialize a pointer $\texttt{prev}(v) = \texttt{nil}$ for each vertex $v \in V$ (as in Dijkstra's algorithm). Then, we add the following line of code to $\texttt{bfs}(G, s)$, immediately after the line $\texttt{inject}(Q, v)$:

$$\texttt{prev}(v) = u.$$

The `prev` values can be used to reconstruct the shortest path from s to each vertex v.

(i) Write the pseudo-code for an algorithm that takes as input a vertex $v \in V$, and the `prev` pointers, and outputs the sequence of vertices on the shortest path from s to v.

(ii) Characterize the run-time of your algorithm in big-O notation.

Appendix

How do I submit this assignment? To submit the assignment, please complete the following steps.

(i) Upload your typed or handwritten work for *each* question using the ECE406 **Crowdmark** site. This includes your answers to the programming questions, where you should submit screenshots of the Python functions you have implemented.

(ii) Upload your source code for each programming question to the **A3 Dropbox submission folder** in LEARN. The starter code for each question specifies the filename convention you should follow.