

**Assignment : Fast Polynomial Multiplication with DFT/FFT implementation, RSA Encryption , Image compression (20 Marks)****Guidelines:**

- Code has to be developed in **PYTHON (Open Source compiler IDEs)**
- Assignment will have to be carried out in teams of size TWO. Form your team and register in the Google form ( will be shared in the Group)**
- Submission ( Code, Readme files, Test Data , Summary Report etc , Snapshot of results) will have to be completed , **on or before deadline**, dropped in the Google Drive shared folder
- Approx **4 - 5 Weeks** of time will be available before submission. Actual **Demo** dates will be broadcast. Hence look out !! ( Most likely to be a Google Meet)
- Follow fair code of ethics and , **develop your version** of code. You can discuss/consult with anyone, but write your version of the code. Plagiarism will get you zero marks !!
- You will be called upon to Demo the assignment, to match with submission data you have provided in the Google Drive.

**Problem Definition, Data Sets, Testing and Logging Stats****Problem:**

- Implement 1-D & 2-D Fourier Transform & RSA Encryption on a  $M \times N$  matrix to achieve Fast Polynomial Multiplication, Secure transmission and Lossy Image compression.
- Step (a) to be done in the following sequence:
  - Implement 1-D DFT ,on coefficient vectors of two polynomials  $A(x)$ ,  $B(x)$  by multiplication of Vandermonde matrix . (  $O(n^2)$  - Complexity)
  - Implement 1-D FFT on the same vectors, of  $A(x)$  and  $B(x)$ . Ensure above two steps produce same results. (  $O(n \log n)$  – Complexity)
  - Pointwise multiply results of Step (ii) to produce  $C(x)$  in P-V form
  - RSA encrypt (128-bit , 256-bit and 512-bit ) , with public key , the  $C(x)$  in PV form, for transmission security and decrypt with a private key and verify .
  - Implement 1-D Inverse FFT (I-FFT) on  $C(x)$ , in PV form (Interpolation) to get  $C(x)$  in Coefficient form (CR) Polynomial.
  - Verify correctness of  $C(x)$  , by comparing with the coefficients generated by a Elementary “Convolution For Loop” on the Coefficients of  $A(x)$  and  $B(x)$
  - Implement a 2-D FFT and 2-D I-FFT module using your 1-D version (This just means , applying FFT on the Rows First and Columns Next on  $M \times N$  matrix of numbers !!)
  - Verify your of Step (vii) correctness on a Grayscale matrix ( which has random integer values in the range 0-255; 255  $\rightarrow$  White & 0  $\rightarrow$  Black))
  - Apply your 2D-FFT on TIFF/JPG (lossless) Grayscale image and drop Fourier coefficients below some specified magnitude and save the 2D- image to a new file.  
( relates to % compression – permanent Lossy compression)  
( by sorting and retaining only coefficients greater than some(quantization) value. Rest are made 0.)
  - Apply 2D I-FFT, on the Quantized Grayscale image and render it to observe Image Quality.

**Data Set Generation and Preparatory Reading Links :**

- a) For 1-D, DFT, FFT and Inverse DFT, Inverse FFT use randomly generated polynomial coefficient vectors for  $A(x)$  and  $B(x)$  of varying degree-bound sizes namely,  $n = 4, 8, 16, 32, 64, \dots 1024$  and  $2048$
- b) For 2-D Grayscale image, use randomly generated pixel values in the range 0-255 for testing 2-D FFT and 2-D Inverse FFT.
- c) For testing on actual image, use a Grayscale TIFF or lossless JPEG image and use the Python OpenCV image manipulation API e.g `imread()`, `imshape()` etc., for accessing the raw pixel data of the 2-D image matrix
- d) For comparing the Efficiency/Asymptotic complexity of your DFT, FFT, Inverse DFT and Inverse FFT, for increasing values of  $n$ , compare with Python Numpy built-in FFT/IFFT functions .
- e) Below are set of links for your review, before getting into the Assignment :

- L1) [Fast Fourier Transform. How to implement the Fast Fourier... | by Cory Maklin | Towards Data Science](#)
- L2) [Understanding the FFT Algorithm | Pythonic Perambulations](#)
- L3) [Image Transforms - Fourier Transform \( 2-D FFT\)](#)
- L4) [\(2\) Image Compression and the FFT - YouTube \( Steve Brunton \)](#)
- L5) [\(2\) Image Compression and the FFT \(Examples in Python\) - YouTube \( Steve Brunton\)](#)
- L6) [OpenCV: Basic Operations on Imag](#)
- L7) [Unraveling The JPEG](#)
- L8) [The RSA Homonym](#)
- L9) [RSA Encryption/Decryption Example - YouTube](#)

**Demo:**

- All of the team members should be present ( Most likely to be a Google Meet)
- Should produce pdf of the report with Performance Metrics / Plots and Screen snap shots
- Should be able to demo the 1-D & 2-D FFT, Inverse FFT, Encryption, Decryption for variable size of Polynomial degree and RSA key size

**Report:**

- About 10-15 pages in size
- Should contain your design and implementation details, snap shots results, critical code section developed by your team
- Timings for various degree polynomial multiplication and Timings for Encryption and Decryption, and FFT/IFFT with comparative numbers from Python Numpy built-in functions

**Last para of your report should contain your observations on the Learning Outcomes of this project.**