MA3111: Mathematical Image Processing
Term project

To submit, zip your report and all *.m files into mipXXXXX.zip where XXXXX is your student ID

In the term project, you will

1. Implement several Discrete Fourier Transform (DFT) algorithms, both for 1-D and 2-D. Each algorithm take even-sized and odd-sized input.
2. Compare their run times on inputs of different sizes.
3. Report your observation and explanation.

Part 1: 1-D DFT

The functions implemented in this part takes a single input `x` of size N. **`x` could be a row or column vector.**

Functions

Implement the following functions, each defined in a separate file as discussed in class.

1. [10] `my_naive_dft(x)`

   The $\Theta(N^2)$ direct computation of DFT.

2. [10] `my_fft_time_rec(x)`

   The decimation-in-time FFT algorithm. If N is even, then break the task down to two $N/2$-point decimation-in-time FFTs. Otherwise call `my_naive_dft`. Make this function **recursive**, i.e., it calls itself when N is even.

3. [10] Bonus: `my_fft_time_iter(x)`

   Same as above, except this function is **iterative**. It can call MATLAB's function `bitrevorder`. It can also call `my_naive_dft` once on a sequence of length m where m is odd and $N = m \times 2^k, k \in \mathbb{Z}$.

4. [10] `my_fft_freq_rec(x)`

   The decimation-in-frequency FFT algorithm. If N is even, then break the task down to two $N/2$-point decimation-in-frequency FFTs. Otherwise call `my_naive_dft`. Make this function **recursive**, i.e., it calls itself when N is even.

5. [10] Bonus: `my_fft_freq_iter(x)`

   Same as above, except this function is **iterative**. It can call MATLAB's function `bitrevorder`. It can also call `my_naive_dft` once on a sequence of length m where m is odd and $N = m \times 2^k, k \in \mathbb{Z}$.

Experiments [10]

1. Modify `test_1d.m` if needed and run it. Attach the generated plots.
2. Observe the run time of various algorithms under different input sizes. Explain.

Part 2: 2-D DFT

The functions implemented in this part takes a single input `x`, which is an $M \times N$ matrix.

Functions

Implement the following functions, each defined in a separate file as discussed in class.

1. `my_super_naive_dft2(x)`
   [10] The $\Theta(M^2N^2)$ direct computation of 2-D DFT.

2. `my_naive_dft2(x)`
   [10] The $\Theta(MN(M+N))$ approach that performs `my_naive_dft` per column followed by `my_naive_dft` per row on the input matrix to obtain 2-D DFT.

3. `my_fft2(x)`
   [10] Let `X` be your favorite among `my_fft_time_rec`, `my_fft_time_iter`, `my_fft_freq_rec` and `my_fft_freq_iter`. Then, perform `X` per column followed by `X` per row on the input matrix to obtain 2-D DFT.

Experiments [10]

1. Modify `test_2d.m` if needed and run it. Attach the generated plots.
2. Observe the run time of various algorithms under different input sizes. Explain.

Questions

Answer all the following questions as detailed as possible.

1. [5] What does function `check_error` do?
2. [5] What does function `measure_runtime` do?
3. [5] What does script `test_1d` do?
4. [5] What does script `test_2d` do?

Final notes

1. It is okay if your fast algorithm is not as fast as MATLAB's functions `fft(2)`.
2. It is okay if your fast algorithm is not as fast as the naive approach on certain input. Try to explain though.
3. If it takes too long, you can choose not to report the run time of the most naive algorithm on large input.
4. **Absolutely no copying codes from anywhere including your peers, internet, or MATLAB's special built-in functions not mentioned above.**