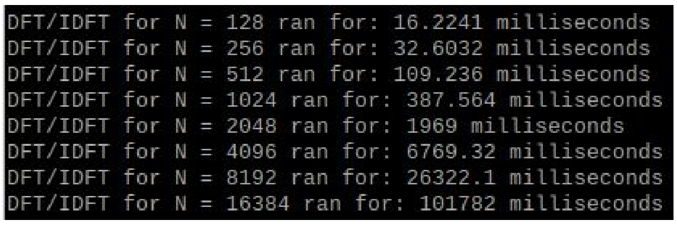
Minhaj Shah / 400119266
Jack Wawrychuk / 400145293

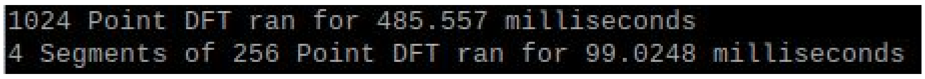
3DY4 Group 14 Lab 2

1. Shown below, is a screenshot of the respective runtimes for different values of N as they increase by a power of 2 from 2 to the power of 7, all the way until 14. As seen below, a clear relationship between the size of N and the run-time of the program is evident. The Time Complexity is exponential, resulting in a big O(N^N), where N is the number of samples. The reason behind this runtime analysis is due to the fact that for DFT and IDFT, both carry a nested loop. The inner bound of the nested loop, runs for the amount of samples we have, which is N. The outer bound runs for the corresponding output of each function which takes the size of the input, which is also N. Thus N^N is the resulting time complexity for both IDFT and DFT respectivley. With further analysis, it can be seen that, the total runtime before simplification, can be determined to be O(8* (N^N + N^N). This boils down to a complexity of N^N.



```
DFT/IDFT for N = 128 ran for: 16.2241 milliseconds
DFT/IDFT for N = 256 ran for: 32.6032 milliseconds
DFT/IDFT for N = 512 ran for: 109.236 milliseconds
DFT/IDFT for N = 1024 ran for: 387.564 milliseconds
DFT/IDFT for N = 2048 ran for: 1969 milliseconds
DFT/IDFT for N = 4096 ran for: 6769.32 milliseconds
DFT/IDFT for N = 8192 ran for: 26322.1 milliseconds
DFT/IDFT for N = 16384 ran for: 101782 milliseconds
```

2. Shown below are the measured runtimes of the 1024 Point DFT along with the 4 Segment 256 Point DFT. It can be seen that the 4 segment DFT is approximatley, 4 times faster. This behaviour is expected due to the algorithmic approach to conducting the DFT computation in segments. It takes resembles into the theory of "Divide and Conquer". By splitting up the larger problem, into smaller sub sections, ultimately, a faster run time can be achieved. After obtaining the results from the computations of each segment, the results were then averaged and a frequency spectrum was successfully plotted. From examining the spectrum plots, it was seen that the 256 Point DFT would show up at approximately 25% of the Frequency Bin Value from the 1024 Point DFT.



```
1024 Point DFT ran for 485.557 milliseconds
4 Segments of 256 Point DFT ran for 99.0248 milliseconds
```

3. For this task, it was essentially utilizing the same algorithmic approach as the previous Lab in Python. The key difference here was that many helper functions had to be created for things such as slicing the vectors to extract sub-vectors and another for convolution. Also, some work was required with passing values by pointers to be able to pass the address of a certain element in a vector, so data could be written in the correct sub-vectors of the final filtered vector. By keeping track of a sliding window style block, incrementally, the final filtered data vector was populated.