CSCI 5454: Algorithms: Exam 1 makeup

Ashutosh Gandhi

April 6, 2023

Problem 1

Given the sequence 1,0,0,1,0,0,1,0,0 ... n times
For any j^{th} FFT coefficient would be:

$$\begin{aligned}
A_j &= \sum_{k=0}^{n-1} a_k(\omega_n^j)^k \\
&= 1 * (\omega_n^j)^0 + 1 * (\omega_n^j)^3 + 1 * (\omega_n^j)^6 + ... + 1 * (\omega_n^j)^{n-3} \quad \# \text{ All other coefficients are 0} \\
&= (\omega_n^{3j})^0 + (\omega_n^{3j})^1 + 1 * (\omega_n^{3j})^2 + ... + (\omega_n^{3j})^{n/3-1} \\
&= \frac{(\omega_n^{3j})^n - 1}{\omega_n^{3j} - 1} \quad \# \text{ Using the GP formula with } r = \omega_n^{3j} \\
&= \frac{(\omega_n^n)^{3j} - 1}{\omega_n^{3j} - 1} \\
&= \frac{1 - 1}{\omega_n^{3j} - 1} \quad \# \quad \omega_n^n = 1 \\
A_j &= 0 \quad \text{when} \quad \omega_n^{3j} \neq 1
\end{aligned}$$

ω_n^{3j} would be 1 for j=0, j=n/3, j=2n/3 For those 3 values of j, we have

$$\begin{aligned}
A_j &= \sum_{k=0}^{n/3-1} (\omega_n^{cn/3})^{3k} \quad \# \text{ where c is 0,1,2} \\
&= \sum_{k=0}^{n/3-1} (\omega_n^n)^{ck} \\
&= \sum_{k=0}^{n/3-1} 1 \quad \# \quad \omega_n^n = 1 \\
&= 0 + (n/3 - 1) + 1 \\
A_j &= n/3
\end{aligned}$$

Thus $A_0, A_{n/3}, A_{2n/3} = n/3$ and all other FFT coefficients are 0

Problem 2

The completed code is:

```
def partition(A, p):
    i = -1 # initial value of i
    j = -1 # initial value of j
    while (j < n-1): # while some unprocessed element remains...
        if(A[j+1] < p): # element A[j+1] must be belong to "first" partition
            swap(A, i+1, j+1) # Swap A[i+1] and A[j+1]
            i += 1 # First partition now goes up to i + 1
            j += 1 # Second partition now goes up to j + 1
        else: # element A[j+1] must belong to the "second" partition
            j += 1 # Second partition now goes up to j + 1
    return i + 1 # return the pivot index
```

Initially, everything is unprocessed and the 0 to i and i to j partitions are empty so we initialize i and j to -1. Next, we iterate over all unprocessed elements and compared them with the pivot element, if the element is lesser then we swap i+1, j+1 and increment the 0 to i and i to j partitions. Otherwise, we just increment the i to j partition. Finally, i+1 is returned as the new pivot position as all elements from 0 to i would be less than pivot element.

Since we are iterating over each element of the array once the running time of the code is $O(n)$.