# CS 170

## 1. Study Group

None

## 2. Modular Fourier Transform

(a) proof by computing directly:
$1^4 \equiv 1 \pmod 5$
$2^4 \equiv 1 \pmod 5$
$3^4 \equiv 1 \pmod 5$
$4^4 \equiv 1 \pmod 5$
$1 + 2 + 2^2 + 2^3 = 0 \pmod 5$

(b) $M_4(2) = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 3 \\ 1 & 4 & 1 & 4 \\ 1 & 3 & 4 & 2 \end{bmatrix}$ $u = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 2 \end{bmatrix}$

$$v = M_4(2) \cdot u = \begin{bmatrix} 3 \\ 3 \\ 2 \\ 2 \end{bmatrix}$$

(c) $M_4^{-1}(2) = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 3 & 4 & 2 \\ 1 & 4 & 1 & 4 \\ 1 & 2 & 4 & 3 \end{bmatrix}$

$$M_4^{-1}(2) \cdot v \cdot 4^{-1} = \begin{bmatrix} 0 \\ 4 \\ 0 \\ 3 \end{bmatrix} \cdot 4 = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 2 \end{bmatrix} = u$$

(d) let $p_1 = \begin{bmatrix} 3 \\ 0 \\ 2 \\ 0 \end{bmatrix}, p_2 = \begin{bmatrix} 3 \\ -1 \\ 0 \\ 0 \end{bmatrix}$

$$p_1^t = M_4(2)p_1 = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \end{bmatrix}$$

$$p_2^t = M_4(2)p_2 = \begin{bmatrix} 2 \\ 1 \\ -1 \\ 0 \end{bmatrix}$$

$$p^t = p_1^t \cdot p_2^t = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

$$p = M_4^{-1}(2) \cdot 4^{-1} \cdot p^t = \begin{bmatrix} 4 \\ 2 \\ 1 \\ 3 \end{bmatrix}$$

So the result is $4 + 2x + x^2 + 3x^3$.

## 4. Triple sum

Hint1: $0 \leq A[i] \leq n$
Hint2: $i, j, k$ is not necessarily distinct
Hint3: using FFT

## 5. Searching for Viruses

(a) the pseudocode is as follows:

**function find_viruses($s_1, s_2$)**

viruses = [ ]

for $i = 0$ to $m - n - 1$

differ = 0

for $j = 0$ to $n - 1$

if $s_1[j]$ != $s_2[i + j]$:

differ += 1

if differ $\leq k$

viruses.add($s_2[i : i + n]$)

return viruses

**Runtime analysis:**

It is obvious that the runtime is $\mathcal{O}(mn)$.

(b) the pseudocode is as follows:

**function fast_find_viruses($s_1, s_2, viruses$)**

if len($s_2$) $\leq$ len($s_1$):

return

mid = $s_2.length/2$

fast_find_viruses($s_1, s_2[0 : mid], viruses$)

fast_find_viruses($s_1$, $s_2[mid : s_2.length], viruses$)

viruses.extend(find_viruses($s_1, s_2[mid - k : mid + k]$))

**Runtime analysis:**

$\mathcal{T}(m) = 2\mathcal{T}(m/2) + \mathcal{O}(k)$

since $k \leq n < m$, we know that $\mathcal{T}(m) = \mathcal{O}(mlogm)$

**Correct Answer:**

refer to the solution for help.

## 6. FFT Coding

My python implementation (only works for $n = 2^k$):

```
import numpy as np
def fft(p, n):
    if n == 1:
      return p
   p_even = p[::2]
   p_odd = p[1::2]
   even = fft(p_even, n//2)
   odd = fft(p_odd, n//2)
   ret = np.empty(n, dtype=np.complex)
   w = np.exp(2*np.pi*1j/n)
   wi = 1
   for i in range(n//2):
      ret[i] = even[i] + wi*odd[i]
      ret[i+n//2] = even[i] - wi*odd[i]
      wi *= w
   return ret

def ifft(p, n):
    def helper(p, n):
      if n == 1:
         return p
      p_even = p[::2]
      p_odd = p[1::2]
      even = helper(p_even, n//2)
      odd = helper(p_odd, n//2)
      ret = np.empty(n, dtype=np.complex)
      w = np.exp(-2*np.pi*1j/n)
      wi = 1
      for i in range(n//2):
         ret[i] = even[i] + wi*odd[i]
         ret[i+n//2] = even[i] - wi*odd[i]
         wi *= w
      return ret
    return helper(p, n) / n
```