Homework #1

(Due: Oct 4)

Task 1. [**20 Points**] **Recurrence(Recurrence(Recurrence(Recurrence(Recurrence(...)))))**

Use the *Master Theorem* to solve the following recurrences.

(*a*) [**5 Points**] For $a > 1$ and $b > 0$,

$$T(n) = \begin{cases} \Theta(1) & \text{if } n \leq b, \\ aT(n-b) + n & \text{otherwise.} \end{cases}$$

(*b*) [**5 Points**] For $a \geq 1$, $b > 1$ and $n = 2^k$ for some integer $k \geq 0$,

$$T(n) = \begin{cases} \Theta(1) & \text{if } n \leq 2, \\ aT(n^{\frac{1}{b}}) + 1 & \text{otherwise.} \end{cases}$$

(*c*) [**5 Points**] The following recurrences arise during the analysis of the span (i.e., critical pathlength) of a multithreaded implementation of Floyd-Warshall's APSP (All-Pairs Shortest Paths) algorithm. Solve for $T_A(n)$.

$$T_A(n) = \begin{cases} \Theta(1) & \text{if } n \leq 1, \\ 2\left(T_A(\frac{n}{2}) + \max\{T_B(\frac{n}{2}), T_C(\frac{n}{2})\} + T_D(\frac{n}{2})\right) + \Theta(1) & \text{otherwise.} \end{cases}$$

$$T_B(n) = \begin{cases} \Theta(1) & \text{if } n \leq 1, \\ 2\left(T_B(\frac{n}{2}) + T_D(\frac{n}{2})\right) + \Theta(1) & \text{otherwise.} \end{cases}$$

$$T_C(n) = \begin{cases} \Theta(1) & \text{if } n \leq 1, \\ 2\left(T_C(\frac{n}{2}) + T_D(\frac{n}{2})\right) + \Theta(1) & \text{otherwise.} \end{cases}$$

$$T_D(n) = \begin{cases} \Theta(1) & \text{if } n \leq 1, \\ 2T_D(\frac{n}{2}) + \Theta(1) & \text{otherwise.} \end{cases}$$

(*d*) [**5 Points**] For $a \geq 1$, $b > 1$, $n = b^k$ for some integer $k \geq 0$, and a nonnegative function $f(n)$ defined on exact powers of b,

$$T(n) = \begin{cases} \Theta(1) & \text{if } n \leq 1, \\ f(n) + \sum_{i=1}^{k} \left(\frac{a}{2}\right)^i T\left(\frac{n}{b^i}\right) & \text{otherwise.} \end{cases}$$

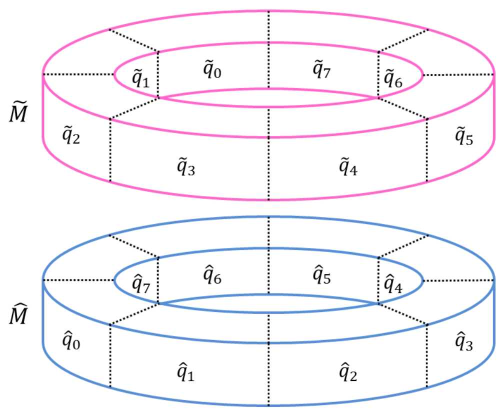

Figure 1: Concentric magnetic rings.

Task 2. [25 Points] Futile Attraction

Consider two magnetic rings $\widehat{M}$ and $\widetilde{M}$ of the same size, each divided into $n \geq 1$ segments with each segment subtending exactly $\frac{2\pi}{n}$ radians at the center of the ring (see Figure 1). In each ring the segments are numbered from 0 to $n-1$ in such a way that segment $k \in [0, n)$ is always adjacent to segments $((k+1) \bmod n)$ and $((n+k-1) \bmod n)$. For each $i \in [0, n)$, the center of the top surface of the i-th segment of $\widehat{M}$ contains a point magnetic charge of magnitude $\hat{q}_i$ amp-meter. Similarly, the center of the bottom surface of segment $j \in [0, n)$ of $\widetilde{M}$ contains a point magnetic charge of magnitude $\tilde{q}_j$ amp-meter.

We know that if two point magnetic charges of magnitude $\hat{q}_i$ and $\tilde{q}_j$ are placed at a distance r (in meters) in a medium of permeability μ (in newton/amp^2), then the magnetic force (in newtons) between them is given by:

$$f_{i,j} = \mu \frac{\hat{q}_i \tilde{q}_j}{4\pi r^2}.$$

A positive value of $f_{i,j}$ indicates repulsion, and a negative value means attraction.

When $\widetilde{M}$ is placed directly above $\widehat{M}$ at a very small distance $r > 0$ with each segment $i \in [0, n)$ of $\widehat{M}$ perfectly aligned with segment $((i+k) \bmod n)$ of $\widetilde{M}$, the total force acting between the two rings is approximated as:

$$F_k = \sum_{i=0}^{n-1} f_{i,(i+k) \bmod n}.$$

Give an efficient algorithm to determine a value of k that results in the maximum attraction between the two rings, i.e., $F_k = \min_{l=0}^{n-1} \{F_l\}$.

Task 3. [25 Points] More than this - there is nothing... (Bryan Ferry / Norah Jones)

Consider an $n \times n$ ($n \geq 1$) matrix $\mathcal{M}_{n,\vec{\alpha},\vec{\beta}}$, where for $\vec{\alpha} = \langle \alpha_0, \alpha_1, \ldots, \alpha_{n-1} \rangle$, $\vec{\beta} = \langle \beta_0, \beta_1, \ldots, \beta_{n-1} \rangle$ and $i, j \in [0, n)$, the entry in the i-th row and the j-th column of the matrix is given by

$$\mathcal{M}_{n,\alpha,\beta}[i,j] = \alpha_{(i+j) \bmod n} + \beta_{(n-1+i-j) \bmod n}.$$

For example, for $n = 4$, we have:

$$\mathcal{M}_{4,\vec{\alpha},\vec{\beta}} = \begin{bmatrix} \alpha_0+\beta_3 & \alpha_1+\beta_2 & \alpha_2+\beta_1 & \alpha_3+\beta_0 \\ \alpha_1+\beta_0 & \alpha_2+\beta_3 & \alpha_3+\beta_2 & \alpha_0+\beta_1 \\ \alpha_2+\beta_1 & \alpha_3+\beta_0 & \alpha_0+\beta_3 & \alpha_1+\beta_2 \\ \alpha_3+\beta_2 & \alpha_0+\beta_1 & \alpha_1+\beta_0 & \alpha_2+\beta_3 \end{bmatrix}$$

Show that though an $n \times n$ matrix has n^2 entries, the product of two matrices as described above can be computed in $o\left(n^2\right)$ time. More specifically, prove that the number of distinct numbers you need in order to completely represent the n^2 entries of $\mathcal{M}_{n,\vec{a},\vec{b}} \times \mathcal{M}_{n,\vec{c},\vec{d}}$ is not more than $4n$ (each entry of the product must be computable in constant time from those numbers), and the complexity of computing those numbers is not more than $\mathcal{O}(n \log n)$.

Task 4. [10 Points] The Fat Fourier Transform (FFT)

Recall that the *discrete Fourier transform* (DFT) of a vector X of n complex numbers is given by another complex vector Y of the same length, where $Y[i] = \sum_{0 \leq j < n} X[j] \cdot \omega_n^{-ij}$ for $0 \leq i < n$, and $\omega_n = e^{2\pi\sqrt{-1}/n}$.

Figure 2 shows one implementation of the DFT computation above. If $n = \mathcal{O}(1)$, we compute DFT using direct formula. Otherwise, for any factorization $n = n_1 n_2$, we observe that

$$Y[i_1 + i_2 n_1] = \sum_{j_2=0}^{n_2-1} \left[\left(\sum_{j_1=0}^{n_1-1} X[j_1 n_2 + j_2] \omega_{n_1}^{-i_1 j_1} \right) \omega_n^{-i_1 j_2} \right] \omega_{n_2}^{-i_2 j_2}.$$

Observe that both the inner and outer summations in the equation above are DFT's. The FFT routine in Figure 2 implements this equation by first computing n_2 transforms of size n_1 each (the inner sum), multiplying the results by *twiddle factors* (i.e., $\omega_n^{-i_1 j_2}$), and finally computing n_1 transforms of size n_2 each (the outer sum).

Analyze the running time of the FFT implementation given in Figure 2 on an input of size $n = 2^k$ for some integer $k \geq 0$.

FFT(X, n)

(Input is a vector of length $n = 2^k$ for some integer $k \geq 0$. Output is the in-place FFT of X.)

1. **Base Case:** If n is a small constant then compute FFT using the direct formula and return.
2. **Divide-and-Conquer:**
   (a) **Divide:** Let $n_1 = 2^{\lceil \frac{k}{2} \rceil}$ and $n_2 = 2^{\lfloor \frac{k}{2} \rfloor}$. Observe that $n_2 \in \{n_1, 2n_1\}$.
   (b) **Transpose:** Treat X as a row-major $n_1 \times n_2$ matrix. Transpose X in-place.
   (c) **Conquer:** ***for*** $i \leftarrow 0$ ***to*** $n_2 - 1$ ***do*** FFT($X[\ i \times n_1,\ i \times n_1 + n_1 - 1\ ]$, n_1)
   (d) **Multiply:** Multiply each entry of X by the appropriate twiddle factor.
   (e) **Transpose:** Treat X as a row-major $n_2 \times n_1$ matrix. Transpose X in-place.
   (f) **Conquer:** ***for*** $i \leftarrow 0$ ***to*** $n_1 - 1$ ***do*** FFT($X[\ i \times n_2,\ i \times n_2 + n_2 - 1\ ]$, n_2)
   (g) **Transpose:** Treat X as a row-major $n_1 \times n_2$ matrix. Transpose X in-place.
   (h) ***return*** X

Figure 2: A divide-and-conquer algorithm for computing FFT.