CS526 hw1 P1

(a) $T(n) = n \lg n + 2T(n/2)$. Suppose $n = 2^k$.

$$\rightarrow T(n) = n \cdot k + 2\left[\tfrac{n}{2}(k-1) + 2\left[\tfrac{n}{4}(k-2) + \cdots + 2 \cdot 1 + 2 \cdot T(1) \cdots\right]\right]$$
$$= nk + n(k-1) + n(k-2) + \cdots + n \cdot 1 + n \cdot T(1)$$
$$= n(k + (k-1) + (k-2) + \cdots + 1) + O(n) = O(nk^2) = O(n \lg^2 n)$$

Could also use tree analysis

(b) $T(n) = \frac{n}{\lg n} + 2 \cdot T(n/2)$. Suppose $n = 2^k$.

$$T(n) = \frac{n}{k} + 2 \cdot \left[\frac{n/2}{k-1} + 2\left[\frac{n/4}{k-2} + \cdots + \frac{2}{1} + 2 \cdot T(1) \cdots\right]\right]$$
$$= n\left(\frac{1}{k} + \frac{1}{k-1} + \frac{1}{k-2} + \cdots + \frac{1}{1}\right) + n \cdot T(1) = n \cdot H_k + O(n).$$

Note the "Harmonic number" $H_k \sim \ln k = \Theta(\lg \lg n)$, so $T(n) = O(n \lg \lg n)$.

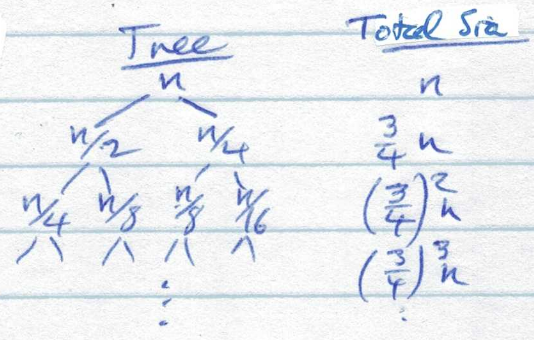
(c) $T(n) = n + \sqrt{n} \cdot T(\sqrt{n})$. Suppose $n = 2^k = 2^{2^l}$

$$T(n) = n + \sqrt{n}\left[\sqrt{n} + \sqrt[4]{n} \cdot \left[\sqrt[4]{n} + \sqrt[8]{n} \cdot \left[\cdots + 2 \cdot T(2) \cdots\right]\right]\right]$$
$$= \underbrace{n + n + n + \cdots}_{l \text{ copies of } n} + \frac{n}{2}T(2) = n \cdot l + O(n)$$
$$= O(n \lg \lg n).$$

Remark: Depth of recurrence is $\lg \lg n$, not $\lg n$.

(d) $T(n) = T(n/2) + T(n/4) + n$

Tree analysis: problems on each level have total size = $\frac{3}{4}$ the previous total size.



So summing the non-recursive "n" term over all problems, we get: $T(n) \le n + \left(\frac{3}{4}\right)n + \left(\frac{3}{4}\right)^2 n + \left(\frac{3}{4}\right)^3 n + \cdots$ [∞ series]

$$= \frac{1}{(1 - 3/4)} n = 4n = O(n).$$

CS526 hw1 P2

(a) $M_n(\omega)$ has two identical rows: row 0 and row k are both all 1's. so it cannot have full rank, and is not invertible.

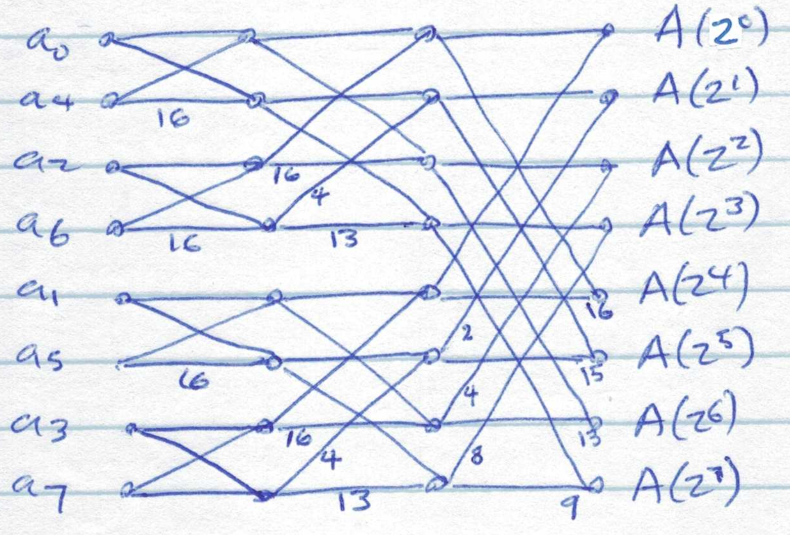
(b)
$$M_4(i) = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & i \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -i \end{bmatrix} \begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & -1 \end{bmatrix}$$

Correspond to two layers of FFT network, other solutions possible.

(c) Idea: replace each integer label k by (2^k mod 17):

k	0	1	2	3	4	5	6	7
(2^k mod 17)	1	2	4	8	16	15	13	9

(we omit the 0/1 labels)



CS526 hw1 P3

Suppose $\vec{r} = (r_1, r_2, \ldots, r_n)$ is our input array of n numbers. For $1 \le a \le b \le n$, the routine PolyProduct $(\vec{r}, a, b)$ should return the product $\prod_{i=a}^{b} (X - r_i)$

as a polynomial of degree $b-a+1$.

```
PolyProduct(r, a, b):          <- subproblem size b-a+1, a subarray
    if a==b:
        return X - r_a
    else:
        mid = ⌊(a+b)/2⌋
polynomials,  -> A = PolyProduct(r, a, mid)
about half each -> B = PolyProduct(r, mid+1, b)
product of polys -> C = A * B
        return C    <- degree b-a+1
```

At the top level, we call PolyProduct$(\vec{r}, 1, n)$.

This is a divide and conquer. Since we can compute the product of two degree m polynomials in $O(m \lg m)$ time, we get:

$$T(n) = 2 \cdot T(n/2) + O(n \lg n)$$
$$= O(n \lg^2 n)$$

by problem 1(a).

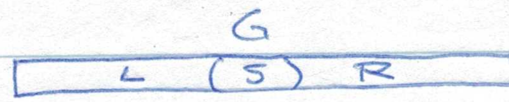
CS526 hw1 P4

Charles M. Fiduccia,
"Polynomial Evaluation via the Division Algorithm: the Fast Fourier Transform Revisited".
In Proceedings of the Fourth Annual ACM Symposium on Theory of Computing (STOC). Denver, CO, ACM Press, pages 88-93, 1972.

Many other (more recent) answers possible!

CS526 hw1 P5

This is a D&C approach, similar to what we saw for planar graphs. Say a graph is "nice" if the vertices are ordered $v_1, v_2, v_3, \ldots$, and there is no edge $v_i - v_j$ with $|i-j| \geq 10$.



NICEMIS(G):
- n = # vertices of G
- if $n \leq 20$:
  - return MIS found exhaustively
- else:
  - S = middle 9 vertices // a separator
  - L = vertices to left of S // at most $n/2$
  - R = vertices to right of S // at most $n/2$
  - for every independent $I_S \subseteq S$:
    - N = neighbors of I_S // cannot use
    - I_L = NICEMIS(G[L-N]) // a nice graph
    - I_R = NICEMIS(G[L-N]) // a nice graph
    - $I = I_S \cup I_L \cup I_R$
  - return I^* = largest I seen

Note G[L-N] means the subgraph of G on vertices in L-N.
Idea: We don't know I^*, but we can try all possible choices of $I^* \cap S$, and then recurse left and right. Note there are no edges between L and R.

Analysis: $T(n) \leq 2^9 \cdot 2 \cdot T(n/2) + O(n) \Rightarrow T(n) = O(n^{10})$.
Remark: DP approach has $n \cdot 2^9$ subproblems, $O(n)$ time.