18-799 Algorithms and Computation in Signal Processing
Midterm Exam Solutions
Total: 6 Questions. Maximum points: 100 / March 2005

1. *(24 pts, 2 each)* Which of the following statements is true? $f(n)$ is an arbitrary positive real-valued function of n.

   Answer with true or false or just leave blank. Wrong answers will be assigned negative points (such that if you answer half right and half wrong, then the total is zero).

   (a) $2n = O(n)$: True
   (b) $\Omega(f(n)) \subset \Theta(f(n))$: False. For example, if $f(n) = n$, then $n^2 = \Omega(n)$, but $n^2 \neq \Theta(n)$.
   (c) $\sin(f(n)) = O(0.5)$: True. $O(0.5) = O(1)$, and $\sin(\alpha)$ is bounded by $[-1, 1]$.
   (d) $n^{\frac{1}{1000}} = O(\log(n))$: False. $\log(n)$ grows slower than any polynomial n^ϵ, $\epsilon > 0$.
   (e) $n^2 + \log^2(n) + 2^n = O(n^2)$: False. 2^n dominates.
   (f) $1/n = \Theta(1)$: False. $1/n = O(1)$, since $1/n \leq 1$, but not $\Omega(1)$, since it approaches 0 arbitrarily close.
   (g) $n^n = \Theta(n^{n/2})$: False.
   (h) $n! = n(n-1)(n-2)\cdots 3 \cdot 2 \cdot 1 = O(n^n)$: True.
   (i) $\log_2(n) = \Theta(\log_3(n))$: True.
   (j) $2^n = \Theta(3^n)$: False.
   (k) $n^{1000} = \Omega(1.01^n)$: False. Polynomials grow slower than any exponential function a^n, $a > 1$.
   (l) If $f(n) = O(n)$ then $2^{f(n)} = O(2^n)$: False. Let $f(n) = 2n$. Then $2^{f(n)} \neq O(2^n)$.

2. *(5 pts)* Why does the best software for matrix-matrix multiplication achieve in general a higher (MFLOPS) performance than the best software for matrix-vector multiplication?

   MMM operates on $O(n^2)$ data and is computed (by the available software) with a runtime of $O(n^3)$. This implies an average data reuse of $O(n)$, which leads to high performance (if implemented properly): more computation than memory traffic (loads/stores). In contrast, MVM operates on $O(n^2)$ data and also has $O(n^2)$ runtime complexity, which implies a lower degree of data reuse and thus lower performance.

   Note that simply staing "'cache performance is better in MMM" does not provide any explanation why.

3. *(5 pts)* Give a 4×4 matrix M, which has no zero entries and is diagonalized by the DFT_4, i.e., $\mathrm{DFT}_4^{-1}\, M\, \mathrm{DFT}_4$ is diagonal.

   Any circulant matrix (of the same size) is diagonalized by the DFT_4. Example:

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 4 & 1 & 2 & 3 \\ 3 & 4 & 1 & 2 \\ 2 & 3 & 4 & 1 \end{bmatrix}.$$

   Note that a matrix with all 1's ($a_{ij} = 1$) is also a circulant matrix and is therefore diagonalized by DFT_4.

4. *(12 pts, 3 each)* Give the tightest upper bound (in O-notation) that you know for the complexity of the matrix-vector multiplication $y = Mx$, where M is an $n \times n$ matrix, and

   (a) $M = \mathrm{DFT}_n$: $O(n \log(n))$ using FFTs.

(b) M is a generic (any) matrix: $O(n^2)$. Note that this is a Matrix-Vector multiplication, not Matrix-Matrix multiplication.

(c) M is a Toeplitz matrix: $O(n\log(n))$ using FFTs.

(d) M is a circulant matrix: $O(n\log(n))$ using FFTs.

5. *(24 pts)* Solve the following recurrence using generating functions.

$$\begin{aligned} f_0 &= 1, \\ f_1 &= 4, \\ f_n &= 4f_{n-1} - 3f_{n-2}, \quad \text{for } n \geq 2. \end{aligned}$$

You can check the result with the first few values: $f_0, f_1, f_2 = 13, f_3 = 40$.

Solution:
Our generating function is:

$$\begin{aligned} F(x) &= \sum_{n\geq 0} f_n x^n \\ \sum_{n\geq 2} f_n x^n &= 4\sum_{n\geq 2} f_{n-1}x^n - 3\sum_{n\geq 2} f_{n-2}x^n \\ F(x) &= f_0 + f_1 + \sum_{n\geq 2} f_n x^n \\ F(x) &= 1 + 4x + 4\sum_{n\geq 2} f_{n-1}x^n - 3\sum_{n\geq 2} f_{n-2}x^n \\ F(x) &= 1 + 4x + 4x\sum_{n\geq 1} f_k x^k - 3x^2\sum_{n\geq 0} f_k x^k \\ F(x) &= 1 + 4x + 4x\sum_{n\geq 0} f_k x^k - 3x^2\sum_{n\geq 0} f_k x^k - 4x \\ F(x) &= 1 + 4xF(x) - 3x^2F(x) \\ F(x) &= 1/(3x^2 - 4x + 1) \\ F(x) &= A/(1-x) + B/(1-3x) \end{aligned}$$

$$A = -1/2, B = 3/2$$

Solution: $f_n = (3^{n+1} - 1)/2$

Ensure this is right by using the values given in the question.

6. *(30 pts)* The discrete cosine transform used in JPEG and MPEG coding is defined by the (real) matrix

$$\mathrm{DCT} = \left[\cos\frac{k(\ell+1/2)\pi}{n}\right]_{k,\ell=0,\ldots,n-1}.$$

Similar to the DFT, there is a fast algorithm for computing $y = \mathrm{DCT}_n\, x$, which can be represented as follows.

$$\mathrm{DCT}_1 = \mathrm{I}_1 = [1], \quad \text{(base case)}$$

and, for n even,

$$\mathrm{DCT}_n = L^n_{n/2}(\mathrm{I}_{n/2} \oplus S_{n/2})(\mathrm{I}_2 \otimes \mathrm{DCT}_{n/2})(\mathrm{I}_{n/2} \oplus D_{n/2})(\mathrm{DFT}_2 \otimes \mathrm{I}_{n/2})(\mathrm{I}_{n/2} \oplus J_{n/2}),$$

where $L^n_{n/2}$ is the stride permutation matrix, $D_{n/2}$ is a $n/2 \times n/2$ diagonal matrix in which all diagonal entries are $\neq 0, 1, -1$, $J_{n/2}$ is a permutation matrix, and $S_{n/2}$ is the $n/2 \times n/2$ matrix:

$$S_{n/2} = \begin{bmatrix} 1 & 1 & & & & \\ & 1 & 1 & & & \\ & & \cdot & \cdot & & \\ & & & \cdot & \cdot & \\ & & & & 1 & 1 \\ & & & & & 1 \end{bmatrix}.$$

Note that in the $S_{n/2}$ matrix shown above, the dots represent 1's, and the elements left blank are zeros (i.e., the matrix has $n-1$ 1's). $S_1 = [1]$.

For a 2-power $n = 2^k$, determine the arithmetic cost $C(n)$ of this algorithm, when applied recursively. $C(n) = (A(n), M(n))$, where $A(n)$ is the number of additions and $M(n)$ is the number of multiplications required. (Do not give the cost as function of k but as function of n.)

Hints: Determine the recurrence for $A(n)$ and $M(n)$ and solve it. You may use the following formula.

The recurrence

$$\begin{aligned} f_0 &= c \\ f_k &= a f_{k-1} + s_k, \quad k \geq 1, \end{aligned}$$

has the solution

$$f_k = a^k c + \sum_{i=0}^{k-1} a^i s_{k-i}.$$

(If you use this formula, you'll have to simplify this expression of course.)

As a reminder,

$$A \oplus B = \begin{bmatrix} A & \\ & B \end{bmatrix}.$$

Solution:
The costs are always presented as $C = (A(n), M(n))$ where $A(n)$ corresponds to additions, and $M(n)$, multiplications.

- $I_{n/2} \oplus J_{n/2} : (0,0)$. J is a permutation matrix, and hence has exactly one 1 in each row. I too has exactly one 1 per row. So no additions or multiplications are performed.
- $DFT_2 \otimes I_{n/2} : (n, 0)$. The DFT_2 involves 2 additions, zero multiplications. The tensor product causes this cost to be multiplied by $n/2$.
- $I_{n/2} \oplus D_{n/2} : (0, (n/2))$. I does not contribute to the cost. The diagonal matrix involved one multiplication per row, since the diagonal elements are not zeros or 1s or -1s.

- $I_2 \otimes DCT_{n/2}$: $(2T(n/2), 2T(n/2))$. This is the recursion step. Multiplication by a factor of 2 comes from the tensor product.
- $I_{n/2} \oplus S_{n/2}$: $((n/2)-1, 0)$. The S matrix has two 1's in each row, excecpt the last row. This means each row (except the last) involved one addition (no multiplications). Hence, $(n/2)-1$ additions. The I matrix here does not contribute to the cost at all.
- $L^n_{n/2}$: $(0,0)$. This is because permutations never involve additions or multiplications.

Note that in the base case DCT_1 has 0 additions and 0 multiplications.

The recursion for the additions is:

$$T(n) = n + 2T(n/2) + n/2 - 1 = 2T(n/2) + 3n/2 - 1, \quad T(1) = 0.$$

Let $n = 2^k$

$$f_k = 2f_{k-1} + 3 * 2^{k-1} - 1, \quad f_0 = 0$$

This is of the form of the equation provided in the question. Using that equation,

$$\begin{aligned} f_k &= \sum_{i=0}^{k-1} 2^i (3 * 2^{k-i-1} - 1) \\ &= 3 * 2^{k-1} k - (2^k - 1) \\ &= (3/2) k 2^k - 2^k + 1 \\ &= (3/2) n \log_2 n - n + 1 \end{aligned}$$

The recursion for the multiplication is:

$$T(n) = 2T(n/2) + n/2, \quad T(1) = 0.$$

Again, setting $n = 2^k$ and $T(n) = f_k$, and using the provided equation,

$$f_k = \sum_{i=0}^{k-1} 2^i 2^{k-i-1} = 2^k k/2 = (n/2)\log_2 n$$

Therefore, the answer is:

$$C(n) = (((3/2) n \log_2 n - n + 1), ((n/2)\log_2 n))$$