CMPS 102 — Quarter Spring 2017 – Homework 2

VLADOI MARIAN

April 28, 2017

I have read and agree to the collaboration policy.Vladoi Marian
Name of students I worked with: Victor Shahbazian and Mitchell Etzel

Solution to Problem 1: Divide and Conquer

Proof. 1A. We should consider the Walsh Hadamard matrix of order n, named H(n).

$$\begin{bmatrix} h_{00} & h_{01} & h_{02} \text{ ... } h_{0n} \\ h_{10} & h_{11} & h_{12} \text{ ... } h_{1n} \\ \text{.........} & & \\ h_{n0} & h_{n1} & h_{n2} \text{ ... } h_{nn} \end{bmatrix}$$

Were $h_{i,j} \forall\ i, j\ \in\ 0,\ n-1$ are elements of the matrix. These elements can be found as follow :
For $i, j \geq 0$, we write the binary numbers of i,j as :
$\sum_{x=0}^{n-1} i_x 2^x\ and\ \sum_{x=0}^{n-1} j_x 2^x$; Where $i_x \in \{0, 1\} \forall \{0, 1, ...n-1\}$; And $j_x \in \{0, 1\} \forall i \{0, 1, ...n-1\}$.
This implies that $hi, j = (-1)^{\sum_{x=0}^{n-1} \oplus i_x, j_x}$ where $\oplus$ denotes adition modulo -2.
The idea is that we use the base 2 of the indeces i,j to create the dot product.
The smallest Walsh Hadamard is H(0). Using the previous formula for h(0,0) we get the matrix [1]. Every time we increase a Walsh hadamard matrix H(n) to H(n+1), the final matrix wil be a square matrix of size 2^{n+1}.H(n) will always be the top left matrix in H(n+1). We want to show that any position h(i,j) in this H(n) matrix = $h(i, j + 2^{n-1})$ in H(n+1) matrix = $h(i + 2^{n-1}, j)$ in H(n+1) matrix = $-h(i + 2^{n-1}, j + 2^{n-1})$ in H(n+1) matrix. Acording to our dot product of indices (in binary) we will always have this case. □

Proof. 1.B Proof by induction:
Base case: for n = 1 we have :

$$H_1 = \frac{1}{\sqrt{2^1}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} = \frac{1}{\sqrt{2^1}} \sqrt{\sum_{i=1}^{2^1} i^2} = 1$$

Assume for any $k > 1$ that :

$$H_1 = \frac{1}{\sqrt{2^k}} \begin{bmatrix} H_{k-1} & H_{k-1} \\ H_{k-1} & H_{k-1} \end{bmatrix} = \frac{1}{\sqrt{2^k}} \sqrt{\sum_{i=1}^{2^k} i^2} = \frac{1}{\sqrt{2^k}} \sqrt{2^k} = 1$$

We must show that the statement is true for k+1:

$$H_1 = \frac{1}{\sqrt{2^k}} \begin{bmatrix} H_k & H_k \\ H_k & -H_k \end{bmatrix} = \frac{1}{\sqrt{2^{k+1}}} \sqrt{\sum_{i=1}^{2^{k+1}} i^2} = \frac{1}{\sqrt{2^k}} \sqrt{\sum_{i=1}^{2^k} i^2 (By\ Induction\ Hypothesis = 1)} \frac{1}{\sqrt{2^{k+1}}} \sqrt{2^{k+1}} = 1$$

The Eucledian norm of every column and every row = 1 $\forall n$ □

Proof. 1.C We proved on Claim 2 that The Eucledian norm of every column and every row of H_n is 1. Because we are dealing with Walsh-Hadamard matrix, the same aproached can be used to prove that every column of H_n has Euclidean norm of 1.
We will prove next, that the dot product of any two columns of H_n equals to 0.
Our H_n matrix, whose columns and rows are indexed by vectors x, y $\in \{0,1\}^n$, has entries:

$$H_n(y,x) = (-1)^{\sum_i^{\infty} x_i y_i}$$

Suppose that x and z are two different columns, $x_j \neq z_j$. The inner product between the two columns is :

$$\sum_{y=1}^{\infty} H(y,x)H(y,z) = \sum_{y=1}^{\infty} (-1)^{\sum_{i=1}^{\infty}} y_i(x_i + z_i)$$

Next we decompose y in y_j, y_{-j}, wher $y_{-j} \in \{0,1\}^{n-1}$ The inner product becomes :

$$\sum_{y=1}^{\infty} H(y,x)H(y,z) = \sum_{y_j, y_{-j}} (-1)^{y_j} (-1)^{\sum_{i \neq j} y_i(x_i+z_i)} = \sum_{y_{-j}} (-1)^{\sum_{i \neq j} y_i(x_i+z_i)} \sum_{y_j=0}^{1} (-1)^{y_j} = 0$$

□

Proof. 1.D We will use Divide and Conquer aproach.
Supose we have a H_n $matrix$. This is a 2^n by 2^n matrix. We know that this H_n can be computed in terms of H_{n-1}. Then the column vector of this matrix, called v has size n. We want to express this vector in terms of two vectors of size $= \frac{n}{2} = 2^{n-1}$. Lets name the upper part of this vector v_x and the lower part of this vector v_y . So each iteration, the size of our problem decreases.We will use the recursion until we hit the base case. Then we will start to construct our solution of the algorithm.

$$H_n v = \begin{bmatrix} H_{n-1} & H_{n-1} \\ H_{n-1} & -H_{n-1} \end{bmatrix} \begin{bmatrix} v_x \\ v_y \end{bmatrix} = \begin{bmatrix} H_{n-1}v_x & H_{n-1}v_y \\ H_{n-1}v_x & -H_{n-1}v_y \end{bmatrix}$$

$$H_{n-1} v = \begin{bmatrix} H_{n-2} & H_{n-2} \\ H_{n-2} & -H_{n-2} \end{bmatrix} \begin{bmatrix} v_{x_x} \\ v_{y_y} \end{bmatrix} = \begin{bmatrix} H_{n-2}v_{x_x} & H_{n-1}v_{y_y} \\ H_{n-1}v_{x_x} & -H_{n-1}v_{y_y} \end{bmatrix}$$

The algorithm for this problem.

MatrixVector(Hn, v)
1. If H_0 return v.
2. vector v_x = the upper part of the vector v, having size 2^{n-1}
3. vector v_y = the lower part of the vector v, having the same size
4. vector t_1 = MatrixVector($H_{n-1}v_x$)
5. vector t_2 = MatrixVector ($H_{n-1}v_y$)
6. vector t = $[t_1 + t_2, t_1 + (-t_2)]$ we are constructing the solution vector from the base case
7. Return t

Assume that T(n) is the cost of solving $H_n v$. Then the cost of solving $H_{n-1}v_x,\ and\ H_{n-1}v_y\ is\ T(\frac{n}{2})$. The cost of solving the base case, making the aditions , is O(n). The recurrence relation is $T(n) = 2T(\frac{n}{2}) + O(n)$. Solving this with Master Theorem , case number 2, we get the running time O(n log(n)). □