CMPS 101: Winter 2016: Programming HW 2 (Update 3)

Due: 5th February 2016

- The assignment is to be attempted in groups of two. If you choose to not work with a partner, one point will be automatically deducted from your score.
- Each group needs to submit only one set of solutions.
- The names of the group members, and their UCSC ID (@ucsc.edu email address) should prominently be written in the README file.
- The solutions need to be submitted via git as outlined in this piazza post `https://piazza.com/class/iizg7fr5ykq6ot?cid=30`. However, if there is any conflict between the Piazza guide and the instructions in this homework, the instructions in this homework have higher priority and must be adhered to.
- Download the test script from `https://users.soe.ucsc.edu/~vishy/winter2016/hw/hw2test.txt`. Rename it to `hw2test.py`, and follow the instructions at the top of the file to check that your assignment passes all the tests before submission. Otherwise you are liable to lose points.
- The submission must be tagged and submitted before 3:30 pm on the due date.
- You are required to read the homework and start working on it soon after it is assigned.
- Although no points are given for coding style, unreadable or messy code can be penalized at the grader's option.
- Clearly acknowledge sources, and mention if you discussed the problems with other students or groups. In all cases, the course policy on collaboration applies, and you should refrain from getting direct answers from anybody or any source. If in doubt, please ask the instructors or TAs.

In the class we discussed how we can design divide and conquer algorithms for various tasks. Here we will see how divide and conquer can be used to solve a linear algebra problem. You will need to read and understand the following to solve the homework:

Hadamard Matrix A Hadamard matrix of size $n = 2^k$ for some $k > 0$ is defined as follows:

$$H_n = \begin{bmatrix} H_{n/2} & H_{n/2} \\ H_{n/2} & -H_{n/2} \end{bmatrix}$$

with

$$H_2 := \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}.$$

Matrix Vector Multiplication Given a $n \times n$ matrix A and a n-dimensional vector x, the i-th element of the matrix vector product

$$A \cdot x = b$$

can be calculated as

$$b_i = \sum_{j=1}^{n} A_{ij} \cdot x_j, \tag{1}$$

were A_{ij} denotes the (i, j)-th entry of A, while x_i and b_i refer to the i-th entries of the vectors x and b respectively.

Divide and Conquer Suppose we split a vector x of size n into two blocks x_1 and x_2 of size $n/2$ as follows:

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

Then

$$b = H_n \cdot x$$

can be written as

$$\begin{aligned} \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} &= \begin{bmatrix} H_{n/2} & H_{n/2} \\ H_{n/2} & -H_{n/2} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \\ &= \begin{bmatrix} H_{n/2} \cdot x_1 + H_{n/2} \cdot x_2 \\ H_{n/2} \cdot x_1 - H_{n/2} \cdot x_2 \end{bmatrix} \end{aligned} \tag{2}$$

Question 1 (1 point): Implement a function `matmult(A, x)` which takes as input a two dimensional Numpy array of size $n \times n$, and an one dimensional Numpy array of size n and computes and returns $A \cdot x = b$, using equation (1).

- Do not use the built in matrix vector multiplication routines of Numpy.
- Use only elementary operations (e.g., swapping, loops etc).

What is the time complexity of your algorithm?

Question 2 (1 point): Implement a function `hadmat(k)` which takes an input an integer k and produces a Hadamard matrix of size 2^k.

- Do not use the built in Numpy routines to create a Hadamard matrix.
- However, you are permitted the use of the `concatenate` function from Numpy.

Question 3 (2 points): Describe an efficient algorithm which uses Equation (2) in order to multiply a $2^k \times 2^k$ Hadamard matrix with an arbitrary 2^k dimensional vector. Analyze the time complexity of your algorithm and prove that it is faster than brute-force matrix multiplication.

Question 4 (1 point) Implement `hadmatmult(H, x)` which takes as input a $2^k \times 2^k$ Hadamard matrix H and an one dimensional Numpy array x of size 2^k and computes and returns $H \cdot x = b$. Your function must implement the efficient algorithm that you developed above. Verify the correctness of your algorithm by calling `matmult` and checking that its output matches that of `hadmatmult`.

Question 5 (1 point) For $k \in \{1, 2, \cdot, 12\}$ create Hadamard matrices of size H_{2^k} and corresponding random vectors x of size 2^k. Plot the time it takes to compute $H_{2^k} \cdot x$ for different values of $n = 2^k$ with $k = \{1, 2, \ldots 12\}$, using `matmult` and `hadmatmult`. Comment on the trends that you observe

Bonus problem (not graded) Write a function `efficienthadmatmult(x)` which takes as input a vector x of size 2^k and multiplies it with H_{2^k}, without explicitly forming H_{2^k}.

What You Need to Submit Only one submission is required per group.

- A README file which contains the names of the team members and their UCSC ID.
- Submit **one** python file which is named `assign2.py`, which contains the solutions to problems 1, 2, 4, and 5. Clearly delineate the solutions for ease of grading.
- Submit a PDF file named `matmulttime.pdf`, which contains the plot for problem 5. The x axis should be labeled as n and the y axis should be labeled as `time`. The legend of the plots should clearly indicate which curve is for for `matmult` and which one is for `hadmatmult`.

- Submit **one** text file which is named `analysis.txt`, and contains the following:
  - Time complexity analysis for problem 1
  - The derivation of the algorithm, and its time complexity analysis for problem 3.
  - Your comments on the trends that you observe for problem 5.

  Clearly delineate the three sections. If you want, you may submit a LaTeX file instead of a text file.

Learning Outcomes After this homework you should have developed a deeper understanding of

- How matrix-vector products work for dense matrices
- The definition of a Hadamard matrix
- How to use divide and conquer to solve problems
- How to analyze recurrence relations and infer time complexity of recursive algorithms

You will also be able to observe a difference in the time that it takes to multiply a Hadamard matrix with a vector using the brute force algorithm, and the recursive algorithm.

Changelog

- Update 1: Fixed the definition of the Hadamard matrix by adding a negative sign. Propagated the change to the other parts of the document.
- Update 2: Added instructions to run the test script before submission.
- Update 3: Made it clear that I am looking for time complexity as a function of n and not k for problem 5.