Evan Louie
Austin Yen
CMPS 101
5/6/16

Homework 4 - ejlouie, ayen

1. Time complexity of matmult

There are 2 for loops for matmult. The inner for loop runs for exactly the length of the vector times. The outer for loop runs the length of the vector times as well. The resr of the operations in matmult run in $O(1)$ time. The algorithm thus runs in $n \cdot n = n^2$ time.

$0 \leq n^2 + 2 \leq c \cdot n^2$ for all $n \geq n_0$

$0 \leq 1 + \frac{2}{n^2} \leq c$

$T(n) : O(n^2)$

$0 \leq \lim_{n \to \infty} \frac{2}{n^2} \leq c$

$0 \leq 0 \leq c$

Since we are multiplying an $n \times n$ matrix by a one-dimensional array, the result will also be a one dimensional array of size n, so only 2 for loops are needed.

3. The divide and conquer algorithm for Hadamard matrix multiplication involves splitting the array to be multiplied into halves until you have 4 matrices, each of size 2. It stops here because it is the base case. The matrices are then merged in the way described by equation 2 on the homework. The array on the left side, B, is added to the array on the right side, C, to create an array D. Then, we subtract array C from array B to create an array E. We then concatenate D and E in that order. This continues until the array is fully merged, yielding the correct product.

Time complexity for hadmatmult

There are 2 recurrences that are splitting the vector into 2 halves. Running hadmatmerge takes only $O(1)$ time because there are no loops are only simple arithmetic. The rest of the operations in hadmatmult, like splitting the vector only takes $O(1)$ as well, so the cost is at most just c. The recurrence relation is

$T(n) \leq 2T(n/2) + c \qquad T(1) \leq c$

Master theorem

$a = 2 \quad b = 2 \quad f(n) = n^0 = 1$

$n^{\log_2 2} = n^1 = n$

for $f(n) = 1$ to equal $n^{\log_2 2} = n^1$ must subtract constant $\epsilon = 1$

$f(n) = O(n^{\log_2 2 - 1})$ so $T(n) = \theta(n^{\log_2 2}) = n^1 =$

$T(n) = O(n)$

In question 1, we proved that the time complexity is $O(n^2)$. By showing the time complexity for hadmatmult is $O(n)$, this proves that hadmatmult is the more efficient algorithm.

5. The graph shows that hadmatmult is faster than matmult, but not by much. The reason for this, is that hadmat is ran more times within hadmatmult than within matmult. Without running hadmat, hadmatmult is much faster than matmult. You can see this in our graph of efficient hadmatmult.