# CS 170 DIS 02

### Released on 2019-01-28

## 1 Squaring vs multiplying: matrices

The square of a matrix $A$ is its product with itself, $AA$.

(a) Show that five multiplications are sufficient to compute the square of a $2 \times 2$ matrix.

(b) What is wrong with the following algorithm for computing the square of an $n \times n$ matrix?

"Use a divide-and-conquer approach as in Strassen's algorithm, except that instead of getting 7 subproblems of size $n/2$, we now get 5 subproblems of size $n/2$ thanks to part (a). Using the same analysis as in Strassen's algorithm, we can conclude that the algorithm runs in $\Theta(n^{\log_2 5})$ time."

(c) In fact, squaring matrices is no easier than multiplying them. Show that if $n \times n$ matrices can be squared in $\Theta(n^c)$ time, then any $n \times n$ matrices can be multiplied in $\Theta(n^c)$ time.

**Solution:**

a)

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}^2 = \begin{bmatrix} a^2 + bc & b(a+d) \\ c(a+d) & bc + d^2 \end{bmatrix}$$

Hence the 5 multiplications $a^2, d^2, bc, b(a+d)$ and $c(a+d)$ suffice to compute the square.

b) We have:

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix}^2 = \begin{bmatrix} A^2 + BC & AB + BD \\ CA + DC & CB + D^2 \end{bmatrix} \neq \begin{bmatrix} A^2 + BC & B(A+D) \\ C(A+D) & BC + D^2 \end{bmatrix}.$$

We end up getting 5 subproblems that are *not of the same type as the original problem*: We started with a squaring problem for a matrix of size $n \times n$ and three of the 5 subproblems now involve *multiplying* $n/2 \times n/2$ matrices. Hence the recurrence $T(n) = 5T(n/2) + O(n^2)$ does not make sense.

(Also, note that matrices don't commute! That is, in general $BC \neq CB$, so we cannot reuse that computation)

c) Given two $n \times n$ matrices $X$ and $Y$, create the $2n \times 2n$ matrix $A$:

$$A = \begin{bmatrix} 0 & X \\ Y & 0 \end{bmatrix}$$

It now suffices to compute $A^2$, as its upper left block will contain $XY$:

$$A^2 = \begin{bmatrix} XY & 0 \\ 0 & YX \end{bmatrix}$$

Hence, the product $XY$ can be calculated in time $O(S(2n))$. If $S(n) = O(n^c)$, this is also $O(n^c)$.

Note: This is an example of a reduction, and is an important concept that we will see over and over again in this course. We are saying that matrix squaring is no easier than matrix multiplication — because we can trick any program for matrix squaring to actually solve the more general problem of matrix multiplication.

## 2 Recurrence Relations

Solve the following recurrence relations and give a $\Theta$ bound for each of them.

(a) (i) $T(n) = 3T(n/4) + 4n$

(ii) $T(n) = 45T(n/3) + .1n^3$

(iii) $T(n) = T(n-1) + c^n$, where $c$ is a constant.

(b) $T(n) = 2T(\sqrt{n}) + 3$, and $T(2) = 3$. (Hint: this means the recursion tree stops when the problem size is 2)

**Solution:**

(a) (i) Since $\log_4 3 < 1$, by the Master Theorem, $T(n) = \Theta(n)$.

(ii) Since $\log_3 45 > 3$, by the Master Theorem, $T(n) = \Theta(n^{\log_3 45})$.

(iii) Expanding out the recurrence, we have $T(n) = \sum_{i=0}^{n} c^i$. From the previous problem, we know that this is $\Theta(1)$, $\Theta(n)$, or $\Theta(c^n)$, depending on if $c < 1$, $c = 1$, or $c > 1$.

(b) *Solution 1:* A priori, this problem does not immediately look like it satisifies the Master's theorem because the the recurrence relation is not a map $n \to n/b$. However, we notice that we may be able to *transform* the recurrence relation into one about another variable such that it satisfies the Master's Theorem. Let $k$ be the solution to

$$n = 2^k.$$

Then we can rewrite our recurrence relation as

$$T(2^k) = 2T(2^{k/2}) + 3.$$

Now, if we let $S(k) = T(2^k)$, then the recurrence relation becomes somethign more managable:

$$S(k) = 2S(k/2) + 3.$$

By Master's theorem, this has a solution of $\Theta(k)$. Since $n = 2^k$, then $k = \log n$ and therefore $\Theta(k) = \Theta(\log n)$.

The intuition between the transformation between $n$ and $k$ is that $n$ could be a number and $k$, the number of bits required to represent $n$. The recurrence relation is easier expressed in terms of the number of bits instead of the actual numbers.

*Solution 2:* The recursion tree is a full binary tree of height $h$, where $h$ satisfies $n^{1/2^h} = 2$. Solving this for $h$, we get that $h = \log\log n)$. The work done at every node of this recursion tree is constant, so the total work done is simply the number of nodes of the tree, which is $2^{h+1} - 1 = \Theta(\log n)$, so $T(n) = \Theta(\log n)$.

## 3 Complex numbers review

(a) Write each of the following numbers in the form $\rho(\cos\theta + i\sin\theta)$ (for real $\rho$ and $\theta$):

(i) $-\sqrt{3} + i$

(ii) The three third roots of unity

(iii) The sum of your answers to the previous item

(b) Let sqrt($x$) represent one of the complex square roots of $x$, so that $(\text{sqrt}(x))^2 = x$. What are the possible values of sqrt(sqrt($-1$))?

You can use any notation for complex numbers, e.g., rectangular, polar, or complex exponential notation.

**Solution:**

(a) (i) $-\sqrt{3} + i = 2(\cos\frac{5\pi}{6} + i\sin\frac{5\pi}{6})$

(ii) $(\cos 0 + i\sin 0), (\cos\frac{2\pi}{3} + i\sin\frac{2\pi}{3}), (\cos\frac{4\pi}{3} + i\sin\frac{4\pi}{3})$

(iii) $0$

(b) $\sqrt{-1} = \pm i$;
$\sqrt{i} = \pm\frac{\sqrt{2}}{2}(1+i)$, $\sqrt{i} = \pm\frac{\sqrt{2}}{2}(1-i)$.

Alternatively, $-1 = \cos\pi + i\sin\pi = \cos 3\pi + i\sin 3\pi$.
So, $\sqrt{\cos\pi + i\sin\pi} = \{(\cos\frac{\pi}{2} + i\sin\frac{\pi}{2}), (\cos\frac{3\pi}{2} + i\sin\frac{3\pi}{2})\}$.
Therefore:
$\sqrt{(\cos\frac{\pi}{2} + i\sin\frac{\pi}{2})} = \sqrt{(\cos\frac{5\pi}{2} + i\sin\frac{5\pi}{2})} = \{(\cos\frac{\pi}{4} + i\sin\frac{\pi}{4}), (\cos\frac{5\pi}{4} + i\sin\frac{5\pi}{4})\}$, and
$\sqrt{(\cos\frac{3\pi}{2} + i\sin\frac{3\pi}{2})} = \sqrt{(\cos\frac{7\pi}{2} + i\sin\frac{7\pi}{2})} = \{(\cos\frac{3\pi}{4} + i\sin\frac{3\pi}{4}), (\cos\frac{7\pi}{4} + i\sin\frac{7\pi}{4})\}$.

## 4 Practice with Polynomial Multiplication with FFT

(a) Suppose that you want to multiply the two polynomials $x+1$ and $x^2+1$ using the FFT. Choose an appropriate power of two, find the FFT of the two sequences, multiply the results componentwise, and compute the inverse FFT to get the final result.

(b) Repeat for the pair of polynomials $1 + x + 2x^2$ and $2 + 3x$.

**Solution:**

(a) Using $\omega = i$, the FFT of $x+1$ is $\text{FFT}(1,1,0,0) = (2, 1+i, 0, 1-i)$ and the FFT of $x^2+1$ is $\text{FFT}(1,0,1,0) = (2,0,2,0)$. Hence, the FFT of their product is $(4,0,0,0)$, corresponding to the coefficients $(1,1,1,1)$.

(b) Using $\omega = i$, the FFT of $2x^2 + x + 1$ is $\text{FFT}(1,1,2,0) = (4, -1+i, 2, -1-i)$ and the FFT of $3x+2$ is $\text{FFT}(2,3,0,0) = (5, 2+3i, -1, 2-3i)$. The FFT of their product is then $(20, -5-i, -2, -5+i)$, corresponding to the coefficients $(2,5,7,6)$.