

# Algorithms: Assignment #1

Due on March 1, 2016

Zhaoyang Li (2014013432)

## 目 录

<b>Problem 1</b>	<b>3</b>
(1) . . . . .	3
(2) . . . . .	3
(3) . . . . .	3
(4) . . . . .	3
(5) . . . . .	4
(6) . . . . .	4
<b>Problem 2</b>	<b>4</b>

**Problem 1****(1)**Prove that  $2n + \Theta(n^2) = \Theta(n^2)$ .

Proof:

Let  $f(n) = \Theta(n^2)$ . By definition, there exists positive constants  $c_1, c_2$  such that

$$c_1 n^2 \leq f(n) \leq c_2 n^2$$

Let  $g(n) = f(n) + 2n$ . We have

$$c_1 n^2 + 2n \leq g(n) \leq c_2 n^2 + 2n$$

For all  $n > \frac{2}{c_2}$ ,  $2n < c_2 n^2$ .  $c_2 n^2 + 2n < 2c_2 n^2$ . Meanwhile,  $c_1 n^2 + 2n > c_1 n^2$ ,

$$c_1 n^2 \leq g(n) \leq 2c_2 n^2$$

Choosing  $c'_1 = c_1, c'_2 = 2c_2$ , by definition we have  $g(n) = \Theta(n^2)$ . QED.**(2)**Prove that  $\Theta(g(n)) \cup o(g(n)) = \emptyset$ .

Proof:

Assume that there exists an  $f$  such that  $f(n) \in \Theta(g(n)) \cup o(g(n))$ .Since  $f(n) \in \Theta(g(n))$ , there exists positive constants  $c_1, c_2, n_0$  such that for all  $n > n_0$ ,

$$c_1 g(n) \leq f(n) \leq c_2 g(n)$$

Since  $f(n) \in o(g(n))$ , for all  $c > 0$ , there exists  $n_1 > 0$ , such that for all  $n > n_1$ ,  $0 \leq f(n) < cg(n)$ . Let  $c$  be  $c_1$ , we have

$$f(n) < c_1 g(n)$$

Let  $n_2 = \max(n_0, n_1)$ . For all  $n > n_2$ , we have both  $f(n) < c_1 g(n)$  and  $f(n) \geq c_1 g(n)$  in contradiction. So  $\Theta(g(n)) \cup o(g(n)) = \emptyset$ . QED.**(3)**Prove that  $\Theta(g(n)) \cap o(g(n)) \neq O(g(n))$ .

Proof:

Let  $g(n) = n^2$ ,  $f(n) = n(1 + \sin n)$ . For all  $n > 0$ ,  $0 < f(n) \leq g(n)$ , so  $f(n) \in O(g(n))$ .However,  $\forall k \in \mathbb{N}^*, f(k\pi + \frac{3}{2}\pi) = 0$ . Thus, there does not exist such  $c > 0$  that  $\forall n > n_0, cn^2 < f(n)$ , meaning that  $f(n) \notin \Theta(g(n))$ .Meanwhile,  $\forall n = k\pi + \frac{1}{2}\pi (k \in \mathbb{N}^*), f(n) = g(n)$ . Thus,  $\forall n_0 > 0$ , there exists  $n$  s.t.  $f(n) = g(n)$ . Thus,  $f(n) \notin o(g(n))$ .

QED.

**(4)**Prove that  $\max((f(n), g(n))) = \Theta(f(n) + g(n))$ .

Proof:

Let  $h(n) = \max((f(n), g(n)))$ .

$$\frac{1}{2}(f(n) + g(n)) \leq h(n) \leq f(n) + g(n)$$

Choosing  $c_1 = \frac{1}{2}, c_2 = 1$ , by definition we have  $h(n) = \Theta(f(n) + g(n))$ . QED.

(5)

Solve the recurrence  $T(n) = 2T(\sqrt{n}) + 1$ .

Let  $m = \log n$ , then  $T(2^m) = 2T(2^{\frac{m}{2}}) + 1$ .

Let  $S(m) = T(2^m)$ , then  $S(m) = 2S(\frac{m}{2}) + 1$ .

We have  $a = 2, b = 2, F(m) = m^0$ , and we have that  $m^{\log_b a} = m^1$ .

Since  $m^1$  is polynomially larger than  $m$ , applying case 1 in the Master Theorem,

$$S(m) = \Theta(m^2)$$

Thus,

$$T(n) = \Theta(\log^2 n)$$

QED.

(6)

Solve the recurrence  $nT(n) = (n-2)T(n-1) + 2$ .

Multiply by  $(n-1)$ ,

$$n(n-1)T(n) = (n-1)(n-2)T(n-1) + 2(n-1)$$

Let  $g(n) = n(n-1)T(n)$ . Thus,

$$g(n) = g(n-1) + 2(n-1)$$

$$g(n) = \sum_{i=1}^n 2(n-1) + g(1) = n(n-1) + g(1)$$

$$T(n) = \frac{g(n)}{n(n-1)} = 1 + \frac{g(1)}{n(n-1)} = \Theta(1)$$

QED.

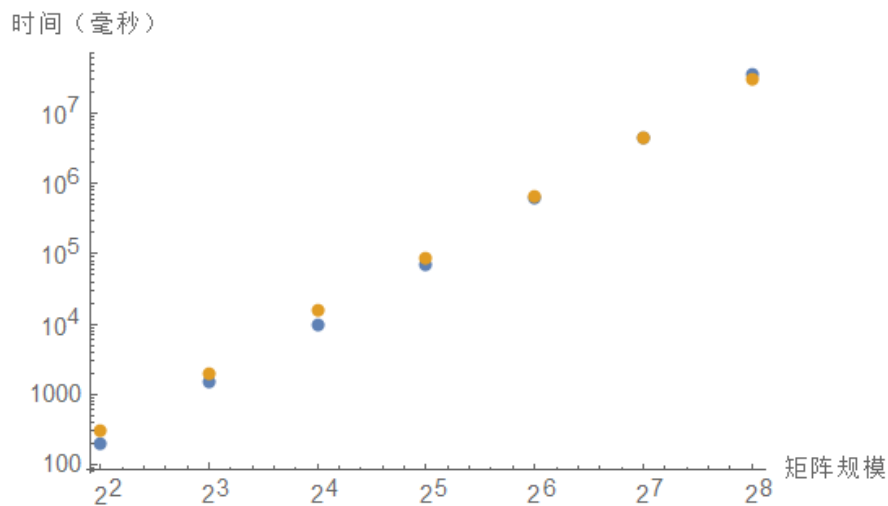
## Problem 2

Compare naive matrix multiplication and the Strassen's algorithm.

Both implemented in an recursive way in C++. To be compiled with Visual Studio 2012.  
 For code and run-time screenshots, please refer to the attachments.  
 Results<sup>a</sup> are as follows:

CPU Time Consumed (microseconds)		
Size of matrix	Naive	Strassen's
1	0	0
2	0	100
4	200	300
8	1500	2000
16	9600	16100
32	69600	88500
64	609200	643500
128	4418300	4537000
256	35934400	30109700

Seems that the cross-over point is somewhere between 128 and 256.  
 A double-logarithmic chart visualizing the data above:



Perfect linearity, verifying polynomial complexity.

<sup>a</sup>on an Intel Core i5 4120U Processor, 4GB DDR3 Memory, Windows 10 Pro