HW2 – Intro to Algorithms
John Carroll

Exercises 3.2-2 and 3.2-3, Problem 3-1 parts a-c, Exercises 4.2-2 and 4.2-4, Problem 4-1 a-f

**1) Exercise 3.2-2**

**Prove equation (3.16).**

**Ans:** Let's have $\log_a b^c = c\log_a b$. (Equation *)

**Claim 1:** Let $\log_a b = x$. Therefore, $b = a^x$. Then,

$$\log_a b^c = \log_a (a^x)^c = \log_a a^{xc} = xc = \log_a b \cdot c.$$

□

**Proof:** Let $\log_b a = x$. Therefore, $a= b^x$. Then, using Equation *,

$$a^{\log_b c} = (b^x)^{\log_b c} = b^{x \log_b c} = b^{\log_b c^x} = c^x = c^{\log_b a}.$$

□

**2) Exercise 3.2-3**

**Prove equation (3.19). Also prove that $n! = \omega(2^n)$ and $n! = o(n^n)$.**

**Ans:**

$$\begin{aligned} n! &= o(n^n)\ , \\ n! &= \omega(2^n)\ , \\ \lg(n!) &= \Theta(n \lg n)\ , \end{aligned}$$

Obviously $n! \leq n^n$, so we know that log n! is O(n log n). A lower bound for the factorial function would be found using the following:

$$\begin{aligned} n! &= n \times (n-1) \times \ldots \times \frac{n}{2} \times + \left(\frac{n}{2} - 1\right) \times \ldots \times 2 \times 1 \\ &\geq \frac{n}{2} \times \frac{n}{2} \times \ldots \times \frac{n}{2} \times 1 \times \ldots \times 1 \times 1 \\ &= \left(\frac{n}{2}\right)^{n/2} \end{aligned}$$

Therefore

$$\log n! \geq \log \left(\frac{n}{2}\right)^{\frac{n}{2}} = \left(\frac{n}{2}\right) \log \left(\frac{n}{2}\right).$$

In other words, log n! is in $\Omega$(n log n). Thus, log n! = $\Theta$(n log n).

The equation can be proved by using Stirling's approximation:

The equation holds for all $n \geq 1$:

$$n! = \sqrt{2\pi n}\left(\frac{n}{e}\right)^n .e^{\alpha_n}$$

$$= \sqrt{2\pi}.n^{1/2}.\frac{n^n}{e^n}.e^{\alpha_n}$$

$$= \sqrt{2\pi}.n^{1/2}.n^n.\frac{1}{e^n}.e^{\alpha_n} \qquad \left[Since,\ x^a + x^b = x^{a+b}\right]$$

$$= \sqrt{2\pi}.n^{1/2+n}.e^{\alpha_n - n} \qquad \left[Since,\ x^a + x^b = x^{a+b}, \frac{x^a}{x^b} = x^{a-b}\right]$$

$$n! = \sqrt{2\pi}\ n^{n+1/2}\ e^{-n} \qquad \left[Since,\ e^{\alpha_n} = 0\ \ \text{constant}\right]$$

Apply log both sides then

$$\lg n! = \lg(\sqrt{2\pi}) + \lg n^{n+1/2} + \lg e^{\alpha_n - n}$$

$$= \lg(\sqrt{2\pi}) + (n + 1/2)\log n + (\alpha_n - n)\lg e$$

$$= \lg(\sqrt{2\pi}) + n\lg n + 1/2\lg n - n \qquad \therefore \log_e^{\ e} = 1$$

$$\approx n\lg n - n$$

$$= \Theta(n\lg n)$$

$n! = o(n^n)$

Apply limit theorem then we get

$$\lim_{n\to\infty}\frac{n!}{n^n} = \lim_{n\to\infty}\frac{\sqrt{2\pi n}\left(\frac{n}{e}\right)^n}{n^n}$$

$$= \lim_{n\to\infty}\sqrt{2\pi n}\,\frac{n^n}{n^n e^n}$$

$$= \lim_{n\to\infty}\sqrt{2\pi n}\left(\frac{n}{ne}\right)^n$$

$$= \lim_{n\to\infty}\sqrt{2\pi n}\left(\frac{1}{e}\right)^n$$

$$= \lim_{n\to\infty}\sqrt{2\pi n}\left(\frac{1}{e^n}\right)$$

$$= \sqrt{2\pi n}\lim_{n\to\infty}\left(\frac{1}{e^n}\right)$$

$$= \sqrt{2\pi n}\lim_{n\to\infty}\left(\frac{1}{e^\infty}\right)$$

$$= \sqrt{2\pi n}.\left(\frac{1}{\infty}\right)$$

$$= 0$$

Since the limit is equal to zero, $\log_2 n$ has a smaller order of growth than $\sqrt{n}$ . So, $\lim_{n\to\infty}\frac{n!}{n^n}=0$ ,we can use the little-oh notation.

Thus, $n!\in o(n^n)$

The given equation:

$$n!=\omega\left(2^n\right)$$

Apply Limit theorem then we get

$$\begin{aligned}\lim_{n\to\alpha}\frac{n!}{2^n}&=\lim_{n\to\alpha}\frac{\sqrt{2\pi n}\left(\frac{n}{e}\right)^n}{2^n}\\&=\lim_{n\to\infty}\sqrt{2\pi n}\,\frac{n^n}{2^n e^n}\\&=\lim_{n\to\infty}\sqrt{2\pi n}\left(\frac{n}{2e}\right)^n\\&=\infty\end{aligned}$$

Thus $2^n$ grows very fast, $n!$ grows still faster. We can write symbolically that $n!\in\omega(2^n)$

3) **<u>Problem 3-1 a-c</u>**

### *3-1 Asymptotic behavior of polynomials*

Let

$$p(n) = \sum_{i=0}^{d} a_i n^i ,$$

where $a_d > 0$, be a degree-$d$ polynomial in $n$, and let $k$ be a constant. Use the definitions of the asymptotic notations to prove the following properties.

a) **If $k \geq d$, then p(n) = O($n^k$).**

**Ans:**

For a given function m $g(n)$, we denote $O(g(n))$ is the set of functions given by

$$O(g(n)) = \{f(n) : \text{there exist positive constants } c \text{ and } n_0 \text{ such that } 0 \leq f(n) \leq cg(n) \text{ for all } n \geq n_0\}$$

From the given statement $f(n) = p(n)$ and $g(n) = n^k$, where $k$ is a constant.

Given $k \geq d$

$$\Rightarrow n^d \leq n^k$$

$$\Rightarrow c_2 n^d \leq c_2 n^k$$

From (1) and (2) we have

$$0 \leq p(n) \leq c_2 n^d \leq c_2 n^k \text{ for all } n \geq n_0$$

Hence from the definition, $\boxed{p(n) = O(n^k)}$.

b) **If $k \leq d$, then p(n) = Ω($n^k$).**

**Ans:**

For a given function $g(n)$, we denote $\Omega(g(n))$ is the set of functions given by

$$\Omega(g(n)) = \{f(n) : \text{there exist positive constants } c \text{ and } n_0 \text{ such that } 0 \leq cg(n) \leq f(n) \text{ for all } n \geq n_0\}$$

From the given statement $f(n) = p(n)$ and $g(n) = n^k$, where $k$ is a constant.

Given $k \leq d$

$$\Rightarrow n^k \leq n^d$$

$$\Rightarrow c_1 n^k \leq c_1 n^d \qquad \text{......(3)}$$

From (1) and (3) we have

$$0 \leq c_1 n^k \leq c_1 n^d \leq p(n) \text{ for all } n \geq n_0$$

Hence from the definition, $\boxed{p(n) = \Omega(n^k)}$.

c) **If $k = d$, then p(n) = Θ($n^k$).**

**Ans:**

For a given function $g(n)$, we denote $\Theta(g(n))$ is the set of functions given by

$$\Theta(g(n)) = \{f(n): \text{there exist positive constants } c_1, c_2 \text{ and } n_0 \text{ such that}$$
$$0 \le c_1 g(n) \le f(n) \le c_2 g(n) \text{ for all } n \ge n_0\}$$

From the given statement $f(n) = p(n)$ and $g(n) = n^k$, where $k$ is a constant.
From (1) we have

$$0 \le c_1 n^d \le p(n) \le c_2 n^d \text{ for all } n \ge n_0$$

Hence from the definition, $\boxed{p(n) = \Theta(n^k)}$.

4) **Exercise 4.2-2**

**Write pseudocode for Strassen's algorithm.**

**Ans:**

```
def Matrix(a,b):
   result = []

   for i in range(0,len(a)):
      new_array = []
      result.extend(new_array)
      for j in range(0,len(b[0])):
                 ssum = 0
                 for k in range(0,len(a[0])):
                 ssum += a[i][k] * b[k][j]
        result[i][j] = ssum
    return result
```

5) **Exercise 4.2-4**

What is the largest $k$ such that if you can multiply $3 \times 3$ matrices using $k$ multiplications (not assuming commutativity of multiplication), then you can multiply $n \times n$ matrices in time $o(n^{\lg 7})$? What would the running time of this algorithm be?

**Ans:**

Strassen's algorithm takes the approach of a recursive multiply with a base condition of 2x2 matrices. We are asked to apply an algorithm using a base condition of a 3x3 matrix and told it will take k multiplications.

Consider the comparative recursions:

Strassen: $T(n) = 7T(n/2) + \Theta(n^2)$
3x3: $T(n) = kT(n/3) + \Theta(n^2)$

As the hint points out, case 1 of the Master Theorem applies and the recursive term dominates. Concentrating on the 3x3 recursion, we want to solve for k such that the number of multiplies will be less than $n^{\lg 7}$. We do so as follows:

$\Theta(n^{\lg 7}) \geq \Theta(n^{\log[3]k})$, so

$n^{\lg 7} \geq n^{\log[3]k}$
$\lg 7 \geq \log_3 k$

Utilizing maple as suggested to solve for $k$ we find that $21.8499 \geq k$.
Therefore the largest $k$ possible, while still doing better than $o(n^{\lg 7})$ using this 3x3 method is 21.

Plugging 21 back into the recurrence and solving using case 1 of the Master Theorem provides us with a running time of:

$$\Theta(n^{\log[3](21)}) \approx \Theta(n^{2.7712})$$

6) <u>**Problem 4-1 a-f**</u>

Give asymptotic upper and lower bounds for $T(n)$ in each of the following recurrences. Assume that $T(n)$ is constant for $n \le 2$. Make your bounds as tight as possible, and justify your answers.

**Case 3 of the master theorem is applied for parts a, b and d.**

**a)** $T(n) = 2T(n/2) + n^4$.

**Ans:**

Consider the recurrence relation, $T(n) = 2T(n/2) + n^4$.

Here *a*= 2, *b*=2 and $f(n) = n^4$.

$$n^{\log_b a} = n^{\log_2 2}$$
$$= n^{\log 1}$$
$$= n \qquad (\text{since } \log 1 = 1\ )$$

Apply case 3 of master theorem (refer theorem 4.1).

$$f(n) = n^4$$
$$= \Omega\left(n^{\log_b a + \epsilon}\right)$$
$$= \Omega\left(n^{\log_2 2 + 2}\right)$$

Substitute the values of *a*, *b*, and *k* to find the value of $\frac{a}{b^k}$

$$\frac{a}{b^k} = \frac{2}{2^4}$$
$$= \frac{2}{16}$$
$$= \frac{1}{8} < 1$$

Thus running time is $\boxed{\Theta(n^4)}$.

**b)** $T(n) = T(7n/10) + n.$

**Ans:**

Consider the recurrence relation, $T(n) = T(7n/10) + n.$

Here, *a* = 1; *b* = 10/7 = 1.42 and $f(n) = n.$

$$\begin{aligned} n^{\log_b a} &= n^{\log_{10/7} 1} \\ &= n^0 \\ &= 1 \end{aligned}$$

Apply case 3 of master theorem (refer theorem 4.1).

$$f(n) = n = \Omega(n^{\log_{10/7} 1+1})$$

Substitute the values of *a*, *b*, and *k* to find the value of $\frac{a}{b^k}$ .

$$\begin{aligned} \frac{a}{b^k} &= \frac{1}{\left(\frac{10}{7}\right)^1} \\ &= \frac{7}{10} < 1 \end{aligned}$$

Thus, the running time is $T(n) = \boxed{\Theta(n)}$.

**c)** $T(n) = 16T(n/4) + n^2.$

**Ans:**

Consider the recurrence relation, $T(n) = 16T(n/4) + n^2$.

Here, $a$ = 16; $b$ = 4 and $f(n) = n^2$.

$$
\begin{aligned}
n^{\log_b a} &= n^{\log_4 16} \\
&= n^{\log_4 4^2} \\
&= n^{2\log_4 4} \qquad \left(\text{since} \log_b a = 1\right) \\
&= n^2
\end{aligned}
$$

Apply case 2 of master theorem (refer theorem 4.1).

$$
\begin{aligned}
f(n) &= \Theta\left(n^{\log_b^a}\right) \\
&= \Theta\left(n^{\log_4^{16}}\right) \\
&= \Theta\left(n^{\log_4 4^2}\right) \\
&= \Theta\left(n^{2\log_4 4}\right) \\
&= \Theta\left(n^2\right)
\end{aligned}
$$

Thus, the running time $T(n) = \Theta\left(n^2 \log n\right)$.

**d)** $T(n) = 7T(n/3) + n^2.$

**Ans:**

Consider the recurrence relation, $T(n) = 7T(n/3) + n^2.$

Here, $a$ = 7, $b$ = 3 and $f(n) = n^2.$

$n^{\log_b a} = n^{\log_3 7}$

The value of $n^{\log_3 7}$ is between 1 and 2.

$1 < n^{\log_3 7} < 2$

$$\begin{aligned} f(n) &= n^2 \\ &= \Omega\left(n^{\log_b a+\epsilon}\right) \\ &= \Omega\left(n^{\log_3 7+\epsilon}\right) \text{ for some constant } \epsilon > 0 \end{aligned}$$

Also, $\frac{a}{b^k} = \frac{7}{3^2}$

$= \frac{7}{9} < 1$

According to case 3 of master theorem (refer theorem 4.1), the running time $T(n) = \Theta\left(n^2\right).$

**e)** $T(n) = 7T(n/2) + n^2.$

**Ans:**

Consider the recurrence relation, $T(n) = 7T(n/2) + n^2.$

Here, $a$ = 7, $b$ = 2 a nd $f(n) = n^2 = n^{\log_2 7}.$

The value of $n^{\log_2 7}$ in between 1 and 2

$2 < n^{\log_2 7} < 3$

So, we have $f(n) = n^2.$

$$\begin{aligned} &= O\left(n^{\log_b a-\epsilon}\right) \\ &= O\left(n^{\log_2 7-\epsilon}\right) \text{ for some constant } \epsilon > 0 \end{aligned}$$

According to case 1 of master theorem (refer theorem 4.1), the running time $T(n) = \Theta\left(n^{\log_2 7}\right).$

**f)** $T(n) = T(n-2) + n^2.$

**Ans:**

Consider the recurrence relation, $T(n) = 2T(n/4) + \sqrt{n}.$

Here, $a$ = 2, $b$ = 4 and $f\ (n) = \sqrt{n}.$

$$\log_b a = \log_4 2 = \sqrt{n}$$

Apply case 2 of master theorem (refer theorem 4.1).

$$\sqrt{n} = \Theta\left(n^{\log_b^a}\right) = \Theta\left(n^{\log_4 2}\right)$$

$$\begin{aligned} T(n) &= \Theta\left(n^{\log_b^a} \lg n\right) \\ &= \Theta\left(n^{\log_4 2} \lg n\right) \\ &= \Theta\left(\sqrt{n} \lg n\right) \end{aligned}$$

Thus, the running time $T(n) = \Theta\left(\sqrt{n} \lg n\right).$