Assignment 1. Solution

Chapter 2.1

Problem 9.

(a) $n(n+1)$ can be approximated to, $n(n+1) \approx n^2$

The higher order in the function, $n(n+1)$ overrides the lower power. meaning $n^2 + n$ can be approximated to n^2.

$\underbrace{n^2}_{\text{is higher power.}}$

When compared with $2000n^2$, it has the same order of growth given n^2+n and $2000n^2$ has the highest power same which is n^2.

You can bolster your argument by taking limit.

$$\lim_{n\to\infty} \frac{n(n+1)}{2000n^2} = \lim_{n\to\infty} \frac{n^2+n}{2000n^2} = \lim_{n\to\infty} \frac{1}{2000} + \frac{1}{2000n}$$

$$= \frac{1}{2000}$$

~~Therefore~~, By taking limit and the value being constant, we can conclude $n(n+1)$ and $2000n^2$ have same order of growth.

(b) $100n^2$ and $0.01n^3$

Taking limits,

$$\lim_{n\to\infty} \frac{100n^2}{0.01n^3} = \lim_{n\to\infty} \frac{10000}{n} = 0$$

Therefore, $0.01n^3$ has a higher order of growth in comparison to $100n^2$.

(c) $\log_2 n$ and $\ln n$.

First step is to convert the logarithm functions into same base using

$$\log_a n = \log_a b \log_b n.$$

Or you can take take the limits directly.

$$\lim_{n\to\infty} \frac{\log_2 n}{\ln \ldots} = \frac{\log_2 e \log_e n}{\ldots} = \log_2 e \text{ (a constant)}$$

Therefore same rate of

(d) $\log_2^2 n$ and $\log_2 n^2$

Using log simplication.

$\log_2^2 n = \log_2 n \cdot \log_2 n$

$\log_2 n^2 = 2 \log_2 n$

Taking limit;

$$\lim_{n\to\infty} \frac{\log_2 n \cdot \cancel{\log_2 n}}{2\,\cancel{\log_2 n}} = \lim_{n\to\infty} \frac{\log_2 n}{2} \Rightarrow \neq 0$$

Even in the second step it is clear. $\log_2^2 n$ has a higher order of growth.

(e) 2^{n-1} and 2^n

$$2^{n-1} = \frac{2^n}{2}$$

2^{n-1} and 2^n has same order of growth as 2^n is within a constant multiple.

(f) $(n-1)!$ and $n!$

~~$n! = (n-1)!\, n!$~~

$n! = (n-1)!\; n$

$$\lim_{n\to\infty} \frac{(n-1)!}{n!} = \lim_{n\to\infty} \frac{\cancel{(n-1)!}}{\cancel{(n-1)!}\, n} = \lim_{n\to\infty} \frac{1}{n} = 0$$

Therefore $n!$ has higher order of growth.

Problem 6.

(a) Every polynomial of degree k, $p(n) = a_k n^k + a_{k-1} n^{k-1} + \cdots a_0$ with $a_k > 0$, belongs to $\Theta(n^k)$

Proof: By taking the limit for the function, $p(n)$ and n^k, we want to show if $p(n)$ belongs ~~Lim~~ to $\Theta(n^k)$ or otherwise.

$$\underset{n\to\infty}{Lt} \frac{p(n)}{n^k}. \qquad \left(\underset{n\to\infty}{Lt} = \lim_{n\to\infty} \right)$$

$$= \underset{n\to\infty}{Lt} \frac{a_k n^k + a_{k-1} n^{k-1} + \cdots + a_0}{n^k}.$$

$$= \underset{n\to\infty}{Lt} \left(a_k + \frac{a_{k-1}}{n} + \cdots + \frac{a_0}{n^k} \right).$$

$a_k > 0$

$$= \underset{n\to\infty}{Lt} (a_k) > 0.$$

Therefore, $p(n) \in \Theta(n^k)$.

(b) Exponential function a^n have different orders of growth for different values of base $a > 0$.

Proof: Assume there are 2 functions a_1^n and a_2^n, ~~where~~ Taking limit.

$$\cancel{\underset{n\to\infty}{Lt}} \ \cancel{a} \ \lim_{n\to\infty} \frac{a_1^n}{a_2^n} = \begin{cases} 0 & \text{if } a_1 < a_2 \Leftrightarrow a_1^n \in o(a_2^n) \\ 1 & \text{if } a_1 = a_2 \Leftrightarrow a_1^n \in \Theta(a_2^n) \\ \infty & \text{if } a_1 > a_2 \Leftrightarrow a_2^n \in o(a_1^n) \end{cases}$$

Chapter 2.3.

Problem 1

(b) $2 + 4 + 8 + 16 + \cdots + 1{,}024$

or $2 + 2^2 + 2^3 + 2^4 + \cdots + 2^{10}$

or $\displaystyle\sum_{i=1}^{10} 2^i = 2 \cdot \frac{2^{10}-1}{2-1} = 2{,}046$ Use Geometric series sum.

(c) $\displaystyle\sum_{i=3}^{n+1} 1 = (n+1) - 3 + 1 = n-1$ (why? the summation 1 simply mean it is going to add 1 from $i=3$ to $n+1$)

(d) $\displaystyle\sum_{i=3}^{n+1} i = \underbrace{\sum_{i=0}^{n+1} i}_{\text{leads to a series in Arithmetic Progression.}} - \underbrace{\sum_{i=0}^{2} i}_{\text{First 3 terms.}} = \frac{n(n+1)(n+2)}{2} - 3$

$\displaystyle = \frac{n^2+3n+4}{2}$

(e) $\displaystyle\sum_{i=0}^{n-1} i(i+1) = \sum_{i=0}^{n-1} (i^2+i) = \sum_{i=0}^{n-1} i^2 + \sum_{i=0}^{n-1} i = \frac{(n-1)\,n\,(2n-1)}{6} + \frac{(n-1)\,n}{2}$

Sum of ~~Geometric series or~~ Squares

$\displaystyle = \frac{(n^2-1)\cdot n}{3}$

(g) $\displaystyle\sum_{i=1}^{n}\sum_{j=1}^{n} ij = \sum_{i=1}^{n} i \sum_{j=1}^{n} j = \sum_{i=1}^{n} i\, \frac{n(n+1)}{2} = \frac{n(n+1)}{2}\sum_{i=1}^{n} i = \frac{n(n+1)}{2}\,\frac{n(n+1)}{2}$

Solving inner summation

$\displaystyle = \frac{n^2(n+1)^2}{4}$

(h) $\displaystyle\sum_{i=1}^{n} 1/i(i+1) = \sum_{i=1}^{n} \left(\frac{1}{i} - \frac{1}{i+1}\right)$ [We break $\frac{1}{i(i+1)} = \frac{1}{i} - \frac{1}{i+1}$

$\displaystyle = \left[\left(\frac{1}{1} - \frac{1}{2}\right) + \left(\frac{1}{2} - \frac{1}{3}\right) + \cdots + \left(\frac{1}{n-1} - \frac{1}{n}\right) + \left(\frac{1}{n} - \frac{1}{n+1}\right)\right]$

$= 1 - \frac{1}{\ldots} = n/\ldots$

Problem 5

(a) Computes the difference between the array's largest and smallest elements.

(b) An comparison of elements.

(c) $C(n) = \sum_{i=1}^{n-1} 2 = 2(n-1)$ [How?] [Look at 1(c)]

(d) $\Theta(n)$

(e) One way to improve is to ~~use one~~ replace the two if statements by

if $A[i] <$ minval minval $\leftarrow A[i]$

else if $A[i] >$ maxval

maxval $\leftarrow A[i]$.

* There are other ways in which it can be improved.

Problem 6

(a) Algorithm returns true if its input matrix is symmetric and false if it is not.

(b) Comparison of 2 matrix elements.

(c) $C_{worst}(n) = \sum_{i=0}^{n-2} \sum_{j=i+1}^{n-1} 1 = \sum_{i=0}^{n-2} [(n-1)-(i+1)+1]$

How? Look at 1(c)!

$$= \sum_{i=0}^{n-2} (n-1-i) = (n-1) + (n-2) + \cdots + 1 = \frac{(n-1) \cdot n}{2}$$

(d) Quadratic: $C_{worst}(n) \in \Theta(n^2)$ How? By definition and...

(e) The algorithm is optimal. Why?
In worst case, it is going to compare $(n-1)n/2$ elements for the upper triangular part of the matrix. with their symmetric counterparts in the lower-triangular part, which is ... this algorithm computes.

Chapter 2.4
Problem 1(c)

$$x(n) = x(n-1) + n \quad \text{for } n>0, \; x(0)=0$$

$$
\begin{aligned}
x(n) &= x(n-1) + n \\
&= [x(n-2) + (n-1)] + n = x(n-2) + (n-1) + n. \\
&= [x(n-3) + (n-2)] + (n-1) + n = x(n-3) + (n-2) + (n-1) + n \\
&= \cdots \\
&= x(n-i) + (n-i+1) + (n-i+2) + \cdots + n. \\
&= x(0) + 1 + 2 + \cdots + n = \frac{n(n+1)}{2}
\end{aligned}
$$

1(e) $x(n) = x(n/3) + 1$ for $n>1$, $x(1)=1$

$$
\begin{aligned}
x(3^k) &= x(3^{k-1}) + 1 \\
&= [x(3^{k-2}) + 1] + 1 = x(3^{k-2}) + 2 \\
&= [x(3^{k-3}) + 1] + 2 = x(3^{k-2}) + 3 \\
&= \cdots \\
&= x(3^{k-i}) + i \\
&= \cdots \\
&= x(3^{k-k}) + k = x(1) + k = 1 + \log_3 n.
\end{aligned}
$$

3(a) Let $T(n)$ be the # of basic operations (multiplications)
The recurrence relation is

$\cancel{M(n)}$ $T(n) = T(n-1) + 2$, $M(1) = 0$

Solving the above expression

$$
\begin{aligned}
T(n) &= T(n-1) + 2 \\
&= [T(n-2) + 2] + 2 = T(n-2) + 2 + 2 \\
&= [T(n-3) + 2] + 2 + 2 = T(n-3) + 2 + 2 + 2 \\
&= M(n-i) + 2i = M(1) + 2(n-1) = 2(n-1)
\end{aligned}
$$

Problem 3(b)

Non-recursive version. Pseudocode is

Algorithm NonRecursiveS(n)

// Input = A positive number n

// Output: Sum of first n cubes.

$S \leftarrow 1$

for $i \leftarrow 2$ to n do

$S \leftarrow S + i * i * i$

return S.

# of basic operators are

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

Note: Nonrecursive variant does not have overhead of space due to stack.

Problem 9.

(a) Computes the value of smallest element in a given array

(b) Recurrence relation

$C(n) = C(n-1) + 1$ for $n > 1$, $C(1) = 0$.

If you solve it, you should get

$C(n) = n-1$.

Problem 10

Recurrence relation for computation of # of basic operations.

Let $T_w(n)$ be the # of times the adjacent matrix is checked in the worst case. (Here, the graph is complete). The recurrence relation for $T_w(n)$ is

$$T_w(n) = T_w(n-1) + (n-1) \quad \text{for } n>1,\ c(1)=0$$

How?

Solving

$$
\begin{aligned}
T_w(n) &= T_w(n-1) + n-1 \\
&= [T_w(n-2) + n-2] + n-1 \\
&= [T_w(n-3) + n-3] + n-2 + n-1 \\
&\dots \\
&= T_w(n-i) + (n-i) + (n-i+1) + \dots + (n-1) \\
&\dots \\
&= T_w(1) + 1 + 2 + 3 + \dots (n-1) \\
&= 0 + (n-1) \cdot n/2 = (n-1)n/2.
\end{aligned}
$$

Note: The worst case of the ~~ad~~ algorithm checks every element below the main diagonal of the adjacency matrix of a given graph.