1 Syllabus

1. Asymptotic notation, recurrence.
2. Divide and Conquer.
3. Dynamic Programming.
4. Greedy Algorithm.
5. Graph Algorithm.
6. NPC

2 Basic

2.1 What is an algorithm?

Unambiguous, mechanically executable sequence of elementary operations.

There are certain types of algorithm:

Traditional (This courses main focus.)	Modern algorithm research
Deterministic	Randomized
Exact	Approximate
Off-line	On-line
Sequential	Parallel

2.2 Input & Output

View algorithm as a function with well defined inputs mapping to specific outputs. For example:

Input: $A[1...n]$ // Positive real number, distinct.

Output: $MAX A[i], 1 <= i <= n$.

2.2.1 Algorithm 1

Stupid way.

```
procedure FindMax
    for i = 1 to n do
        count = 0
        for j = 1 to n do
            if A[i] > A[j] then
                count = count + 1
            end if
        end for
        if count = n then
            return A[i]
        end if
    end for
end procedure
```

Algorithm 1: Stupid Find Max Algorithm

Analysis: Worst Case, n^2 comparison.

2.2.2 Algorithm 2

Sort & Find.

```
procedure FindMax
    Ā = sort(A)
    return Ā[n]
end procedure
```

Algorithm 2: Sort & Find Max Algorithm

Analysis: Worst Case, sorting takes $c\ n \log n$ time.

2.2.3 Algorithm 3

Dynamically store the biggest one.

```
procedure FindMax
    current = 1
    for i = 2 to n do
        if A[i] > A[current] then
            current = i
        end if
    end for
    return A[current]
end procedure
```

Algorithm 3: Search & Find Max Algorithm

2.3 Can we do better?

It depends on the operations allowed. For example the dropping the curtain and find the first appearing one.

3 Asymptotic Notation – big "O" notation

3.1 Growth of Functions

The growth of function in Table 1 increase downwards.

Table 1: Function List

$\log_{10} n$	binary search
n	input
n^2	pairs
$10^{10}n^{10}$	
$1.000.1^n$	
2^n	Binary string of length n
$n!$	Permutation

Let $f(n)$, $g(n)$ be function.

3.2 big "O" notation

Definition 3.2.1. *$f(n) = \mathcal{O}(g(n))$, if $\exists n_0 \in \mathbb{N}$, $c \in \mathbb{R}^+$, s.t. $\forall n \geq n_0$, $f(n) \leq c * g(n)$, and $\lim_{n\to\infty} \frac{f(n)}{g(n)} \neq \infty$, i.e. it is $\lim_{n\to\infty} \frac{f(n)}{g(n)} < k$, for some constant k.*

Table 2 shows the basic definition of all the asymptotic notations.

Table 2: Definition for all Asymptotic Notation

$f(n)$	$\lim_{n\to\infty} \frac{f(n)}{g(n)}$	relation
$\mathcal{O}(g(n))$	$\neq \infty$	$\leq$
$\Omega(g(n))$	$\neq \infty$	$\leq$
$\Theta(g(n))$	$= k > 0$	$=$
$o(g(n))$	$= 0$	$<$
$\omega(g(n))$	$= \infty$	$>$

3.3 Asymptotic Relation's feature

Theorem 3.3.1. *Multiplying by positive constant does NOT change asymptotic relations. i.e. if $f(n) = \mathcal{O}(g(n))$, then $100 * f(n) = \mathcal{O}(g(n))$.*

Proof. $f(n) = \mathcal{O}(g(n)) \Rightarrow \exists n_0 \exists c, \forall n \geq n_0, f(n) \leq c * g(n)$,

then, $\exists n_0 \exists c'$, s.t. $\forall n \geq n_0$, $100 * f(n) \leq c' * g(n) = 100c * g(n)$. □

Example:

$$C * 2^n = \Theta(2^n) \tag{1}$$
$$(C * 2)^n \neq \Theta(2^n) \tag{2}$$

Claim 3.3.2. *Show:* $2n\log(n) - 10n = \Theta(n\log(n))$

Proof. First show: $2n\log(n) - 10n = \mathcal{O}(n\log(n))$

For $n_0 = 1$, $c = 2$

$$2n\log(n) - 10n \leq 2n\log(n)$$

Now show: $2n\log(n) - 10n = \Omega(n\log(n))$

For $n_0 = 2^10$, $c = 1$,

$$\begin{aligned} 2n\log(n) - 10n &\geq n\log(n) + n\log(2^{10}) - 10n \\ &= n\log(n) + 10n - 10n \\ &= n\log(n) \end{aligned}$$

$n_0 = 1$ ($n_0 = 2^{10}$) means n is at least 1 (or 2^10). □

Corollary 3.3.3. $\mathcal{O}(1)$ *means* ***Any Constant****.*

Attention: *Asymptotic notation has limit. It is not applicable for all scenarios.*

3.4 Properties of $\log(n)$

Definition 3.4.1. $n = C^{\log_c n}$*,* $c > 1$*,* $\lg n = \log_2 n$*,* $\ln n = \log_e n$*.*

Theorem 3.4.2. $\forall a, b > 1$

$$\log_b(n) = \frac{\log_a(n)}{\log_a(b)}$$
$$\log_b(n) = \Theta(\log_a(n))$$

Theorem 3.4.3. $\forall a, b \in \mathbb{R}$

$$\log(a^n) = n * \log(a)$$
$$\log(a * b) = \log(a) + \log(b))$$
$$a^{\log(b)} = b^{\log(a)}$$

Theorem 3.4.4. $\lg(n)$ *is to* n *as* n *is to* 2^n*.*

3.5 Something More

Theorem 3.5.1. *Let* $f(n)$ *be a polynomial function, then* $\log(f(n)) = \Theta(\log(n))$*.*

Proof. The asymptotic result of n^2 and n^10 are the same.

Definition 3.5.2. $\log^*(n) = o(\log\log\log\log\log(n)) = \alpha$*.*

Example: $\lg^*(2^{2^{2^{2^2}}}) = 5$.

4 Series

4.1 Some Definition

Definition 4.1.1. *Harmonic Series:*

$$\sum_{i=1}^{n} \frac{1}{i} = \Theta(\log(n))$$

Definition 4.1.2. *Geometric Series:*

$$\sum_{i=0}^{n-1} x^i = \frac{x^n - 1}{x - 1} = \begin{cases} \Theta(x^n) & \text{if } \forall x > 1, \\ \Theta(1) & \text{if } \forall x < 1, \\ \Theta(n) & \text{if } \forall x = 1. \end{cases}$$

Definition 4.1.3. *Arithmetic Series:*

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2} = \Theta(n^2)$$

4.2 Some Theorem

Suppose I want to know if $f(n) = o(g(n))$.

Theorem 4.2.1. *If* $\log(f(n)) = o(\log(g(n))$, *then* $f(n) = o(g(n))$.

Example: Let $f(n) = n^3$, $g(n) = 2^n$. Then $\log(f(n)) = \log(n^3) = 3\log(n)$, $\log(g(n)) = \log(2^n) = n$.

$$\text{i.e. } \log(f(n)) < \log(g(n) \Rightarrow f(n) < g(n)$$

Note that this theorem stands for 'o', NOT TRUE for '$\mathcal{O}$'.

Example: $\log(n^3) = \mathcal{O}(\log(n^2))$, but $n^3 \neq \mathcal{O}(n^2)$.

5 Induction

5.1 When to use?

Prove statement for all $n \in \mathbb{N}$, s.t. $n \geq n_0$.

5.2 Definition

Basically, induction has two parts:

1. Base case(s) – Sometimes there are more than one base cases.

   Prove statement for some n. – Often $n_0 = 0$ or 1.

2. Induction Hypothesis

   Assume statement hold true for all $m \leq n$.

   Prove the hypothesis implies that it hold true for $n + 1$.

Note that the process may be different from previous, which just hypothesize $n-1$ is true and prove for n.