CX2001 Algorithms **2020/21 Semester 1**

1: Introduction to Algorithms

School of Computer Science and Engineering *Nanyang Technological University*

1.1 What is an algorithm?

An algorithm is a sequence of unambiguous instructions to systematically solve a well-defined computational problem.

A good algorithm have the following properties:

- **Correctness**: The output results must be correct and consistent for every given input instance.
- **Precision**: There is no ambiguity. Each step of the algorithm must be precisely stated. Given the same inputs, the algorithm always produce the same output results.
- **Finiteness**: The algorithm terminates after a finite number of instructions have been executed.

1.2 What are the differences between an algorithm and a computer program?

A computer program is an instance, or concrete representation of an algorithm in some programming language.

- Control Structures
  - Selection Structure: if/else statement, switch-case statement
  - Repetition Structure: for-loop, while-loop
- Recursive and non-recursive function

Implementation is the task of turning an algorithm into a computer programe.

1.2.1 Example 1: Design Algorithms of an Arithmetic Series

Several algorithms can obtain the sum of an arithmetic sequence.

$$S = 1 + 2 + 3 + 4 + \ldots + n$$

Method 1 Iteratively summing up 1 to n

Algorithm 1 Summing Arithmetic Sequence

```
function Method_One(n)
begin
$sum \leftarrow 0$
for $i = 1$ to $n$ do
    $sum \leftarrow sum + i$
end
```

Method 2 Using its summation formula

$$S = \frac{n(1+n)}{2}$$

Algorithm 2 Summing Arithmetic Sequence

```
function Method_Two(n)
begin
$sum \leftarrow n(1+n)/2$
end
```

Method 3 Using recursive approach

Algorithm 3 Summing Arithmetic Sequence

```
function Method_Three(n)
begin
if n=1 then
    return 1
else
    return n+Method_Three($n-1$)
end
```

1.2.2 Example 2: Fibonacci Sequence

Let Fibonacci numbers denote as F_n. Each number is the sum of the two preceding ones starting from 0 and 1. It is defined as

$$F_0 = 0, F_1 = 1$$

and

$$F_n = F_{n-1} + F_{n-2}$$

for $n > 1$ The sequence is $0, 1, 1, 2, 3, 5, 8, 13, 21, 34, \ldots$

Algorithm 4 Fibonacci Sequence: A Simple Recursive Function

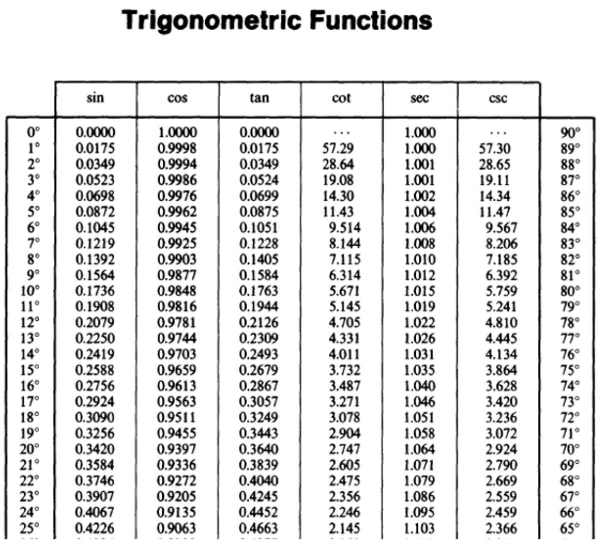
```
function Fibonacci_Recursive(n)
begin
if n<1 then
    return 0
if n==1 OR n==2 then
    return 1
return Fibonacci_Recursive($n-1$)+Fibonacci_Recursive($n-2$)
end
```

Algorithm 5 Fibonacci Sequence: A Simple Iterative Function

```
function Fibonacci_Iterative(n)
begin
if n<1 then
    return 0
if n==1 OR n==2 then
    return 1
$F_2 \leftarrow 1$
$F_1 \leftarrow 1$
for $i = 3$ to $n$ do
    begin
    $F_i \leftarrow F_{i-2} + F_{i-1}$
    $F_{i-2} \leftarrow F_{i-1}$
    $F_{i-1} \leftarrow F_i$
    end
return $F_n$
end
```

1.2.3 Example 3: Design Algorithm of the Sine Function

Method 1 Using trigonometry table



Trigonometric Functions

	sin	cos	tan	cot	sec	csc	
0°	0.0000	1.0000	0.0000	· · ·	1.000	· · ·	90°
1°	0.0175	0.9998	0.0175	57.29	1.000	57.30	89°
2°	0.0349	0.9994	0.0349	28.64	1.001	28.65	88°
3°	0.0523	0.9986	0.0524	19.08	1.001	19.11	87°
4°	0.0698	0.9976	0.0699	14.30	1.002	14.34	86°
5°	0.0872	0.9962	0.0875	11.43	1.004	11.47	85°
6°	0.1045	0.9945	0.1051	9.514	1.006	9.567	84°
7°	0.1219	0.9925	0.1228	8.144	1.008	8.206	83°
8°	0.1392	0.9903	0.1405	7.115	1.010	7.185	82°
9°	0.1564	0.9877	0.1584	6.314	1.012	6.392	81°
10°	0.1736	0.9848	0.1763	5.671	1.015	5.759	80°
11°	0.1908	0.9816	0.1944	5.145	1.019	5.241	79°
12°	0.2079	0.9781	0.2126	4.705	1.022	4.810	78°
13°	0.2250	0.9744	0.2309	4.331	1.026	4.445	77°
14°	0.2419	0.9703	0.2493	4.011	1.031	4.134	76°
15°	0.2588	0.9659	0.2679	3.732	1.035	3.864	75°
16°	0.2756	0.9613	0.2867	3.487	1.040	3.628	74°
17°	0.2924	0.9563	0.3057	3.271	1.046	3.420	73°
18°	0.3090	0.9511	0.3249	3.078	1.051	3.236	72°
19°	0.3256	0.9455	0.3443	2.904	1.058	3.072	71°
20°	0.3420	0.9397	0.3640	2.747	1.064	2.924	70°
21°	0.3584	0.9336	0.3839	2.605	1.071	2.790	69°
22°	0.3746	0.9272	0.4040	2.475	1.079	2.669	68°
23°	0.3907	0.9205	0.4245	2.356	1.086	2.559	67°
24°	0.4067	0.9135	0.4452	2.246	1.095	2.459	66°
25°	0.4226	0.9063	0.4663	2.145	1.103	2.366	65°

Reference: Abelsson. (2006). Illustration of CORDIC operation. Retrieved from https://commons.wikimedia.org/wiki/File:CORDIC-illustration.png.

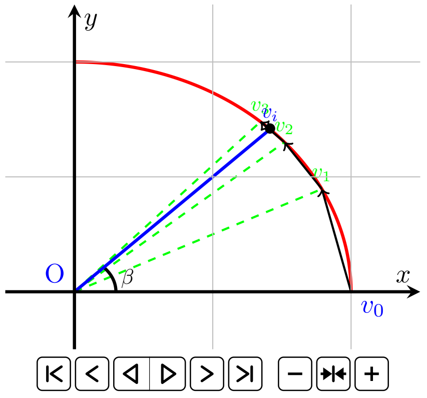
Method 2 Using Maclaurin Series

$$\sin x = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n+1)!} x^{2n+1} = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \dots$$

Maclaurin Seies is a special case of Taylor series with **a=0**.

$$f(x) = f(a) + \frac{f'(a)}{1!}(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \frac{f^{(3)}(a)}{3!}(x-a)^3 + \dots$$

Method 3 Using CORDIC (COordinate Rotation DIgital Computer)
CORDIC was conceived in 1956 by Jack E. Volder at the aeroelectronics department of Convair out of necessity to replace the analog resolver in the B-58 bomber's navigation computer with a more accurate real-time digital solution. Therefore, CORDIC is sometimes referred to as digital resolver. [1]



Let $v_0 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ and v_i can be obtained by:

$$v_i = R_i v_{i-1}$$

where R_i is the rotation matrix.

$$R_i = \begin{bmatrix} \cos(\gamma_i) & -\sin(\gamma_i) \\ \sin(\gamma_i) & \cos(\gamma_i) \end{bmatrix}$$

We can simplify the rotation matrix by using these two trigonometric identities,$\cos(\gamma_i) = \frac{1}{\sqrt{1+\tan^2(\gamma_i)}}$ and $\sin(\gamma_i) = \frac{\tan(\gamma_i)}{\sqrt{1+\tan^2(\gamma_i)}}$,

$$R_i = \frac{1}{\sqrt{1+\tan^2(\gamma_i)}} \begin{bmatrix} 1 & -\tan(\gamma_i) \\ \tan(\gamma_i) & 1 \end{bmatrix}$$

when $\gamma_i << 1$, $\tan(\gamma_i) \approx \gamma_i$. To make it easy implement in digital computer system, we select to rotate the vector, v_0 by $\pm 2^{-i}$ iteratively with increasing i. The final expression is:

$$v_i = \frac{1}{\sqrt{1+2^{-2i}}} \begin{bmatrix} 1 & -\sigma_i 2^{-i} \\ \sigma_i 2^{-i} & 1 \end{bmatrix} \begin{bmatrix} x_{i-1} \\ y_{i-1} \end{bmatrix}$$

where σ_i can be 1 (rotate counterclockwise) or -1 (clockwise) depending on the wanted angle β and $\arctan(\frac{y_{i-1}}{x_{i-1}})$.

[1]https://en.wikipedia.org/wiki/CORDIC

Algorithm 6 CORDIC Algorithm

```
function cordic($\beta$,n)
begin
$i \leftarrow 0$
$v \leftarrow \begin{bmatrix} 1 \\ 0 \end{bmatrix}$
for $i = 1$ to $n$ do
    begin
    if $\beta < 0$ then
        $\sigma \leftarrow -1$
    else
        $\sigma \leftarrow 1$
    $v \leftarrow \frac{1}{\sqrt{1+2^{-2i}}} \begin{bmatrix} 1 & -\sigma_i 2^{-i} \\ \sigma_i 2^{-i} & 1 \end{bmatrix} v$
    $\beta \leftarrow \beta - \sigma 2^{-1}$
    $i \leftarrow i + 1$
    end
end
```

1.3 What is CX2001 Algorithms?

- Design algorithms
  - A high level specification of a method/way to solve a problem
  - Outline its corresponding computer program
- Analyze algorithms' efficiency
- It is not a programming course but we assumed that you can write the programming codes.

1.3.1 Learning Journey in Computer Science/Engineering

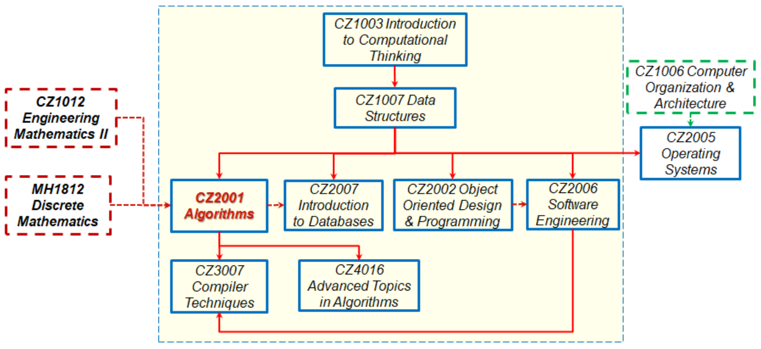

CZ1003 Introduction to Computational Thinking
CZ1007 Data Structures
CZ1006 Computer Organization & Architecture
CZ2005 Operating Systems
CZ1012 Engineering Mathematics II
MH1812 Discrete Mathematics
CZ2001 Algorithms
CZ2007 Introduction to Databases
CZ2002 Object Oriented Design & Programming
CZ2006 Software Engineering
CZ3007 Compiler Techniques
CZ4016 Advanced Topics in Algorithms

Figure 1.1: The relationship between Algorithms and other courses

Algorithms: To design an algorithm that makes efficient use of the computer's resources.

Object Oriented Design & Programming and **Software Engineering**: To design an algorithm that is easy to understand, code and debug.

1.3.2 Learning Outcomes

- Conduct complexity analysis of basic algorithm
- Design and analyse algorithms using the suitable strategies to solve a problem to solve a problem
  - incremental
  - divide and conquer
  - data-structure
  - greedy approaches
- Compare the efficiencies of different algorithms for the same problem
  1. Searching
  2. Sorting
  3. Graph Traversal
- Describe various heuristic problem-solving methods.
- Implement algorithms from pseudo code into real code.

1.3.3 Syllabus Overview

Week 1-3 Basic Algorithm Analysis

Week 4 Searching

Week 5-8 Sorting

Week 9-11 Graph

Week 12 Basic Computability Theory

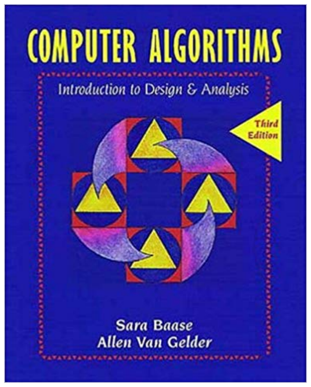
Week 13 Revision

Week 7 is e-learning week. Mid-term quiz will be conducted in tutorial class during Week 8. All materials can be found in CX2001 course site at NTULearn. Online video lectures and learning activities are available in LAMS but they are not graded.

CA components	Percentage
Assigments	40%
Quizzes	30%
Example Class Presentation	30% (2 group projects $\times$ 15%)

1.4 References

Textbook:



- Computer Algorithms: Introduction to Design and Analysis. Sara Baase and Allen Van Gelder, 2000. Third Edition. Addison Wesley.
  - ISBN-10: 0-201-61244-5
  - Library: QA76.6.B111 2000

Reference Materials:

1. Introduction to the Design and Analysis of Algorithms. Anany Levitin, 2012. Third Edition. Addison Wesley.
   - ISBN-10: 0-132-31681-1
   - Library: QA76.9.A43L666 2012 (ebook is available)
2. Introduction to Algorithms. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest and Clifford Stein, 2009. Third Edition. The MIT Press.
   - ISBN-10: 0-262-03384-4
   - Library: QA76.6.C811 2009 (ebook is available)
3. Algorithms. Richard Johnsonbaugh and Marcus Schaefer, 2004. Pearson Prentice Hall.
   - ISBN-10: 0-023-60692-4/ 0-131-22853-6(Int'l ed.)
   - Library: QA76.9.A43J65
4. Algorithm Design by J. Kleinberg and E. Tardos, 2005, Addison-Wesley.
   - ISBN-10:0-321-29535-8
   - Library: QA76.9.A43K64