SC1007 Data Structures and Algorithms **2022/23 Semester 2**

1: Introduction to Algorithms

School of Computer Science and Engineering *Nanyang Technological University*

1.1 What is SC1007 Data Structures and Algorithms?

- Part 1:
  - Select appropriate data structures such as arrays, linked lists, stacks, queues and trees
  - Conduct complexity analysis of algorithms
- Part 2:
  - Implement algorithms to solver real world problems using C programming
  - Introduce algorithms
    - Search algorithms eg. sequential search (covered with linked lists), binary search (covered with binary trees), hash tables
    - Graph traversal algorithms eg. Depth First Search (DFS) and Breath First Search (BFS)
- It is not a programming course but we assumed that you can write the C programming codes.
- Prerequisite Course: SC1003 Introduction to Computational Thinking
- Some fundamental mathematic concepts are required for complexity analysis

1.1.1 Learning Journey in Computer Science/Engineering

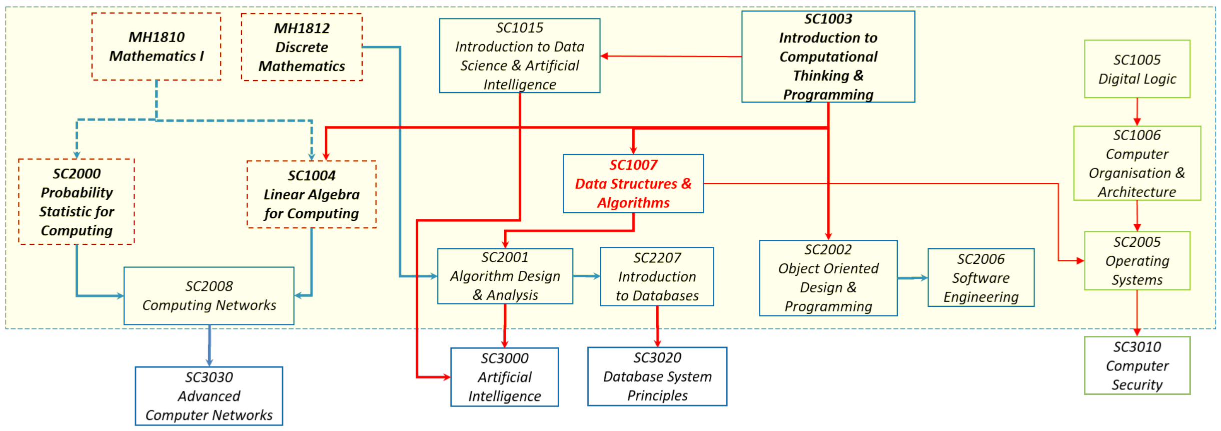

MH1810 Mathematics I
MH1812 Discrete Mathematics
SC1015 Introduction to Data Science & Artificial Intelligence
SC1003 Introduction to Computational Thinking & Programming
SC1005 Digital Logic
SC1006 Computer Organisation & Architecture
SC2000 Probability Statistic for Computing
SC1004 Linear Algebra for Computing
SC1007 Data Structures & Algorithms
SC2008 Computing Networks
SC2001 Algorithm Design & Analysis
SC2207 Introduction to Databases
SC2002 Object Oriented Design & Programming
SC2006 Software Engineering
SC2005 Operating Systems
SC3030 Advanced Computer Networks
SC3000 Artificial Intelligence
SC3020 Database System Principles
SC3010 Computer Security

Figure 1.1: The relationship between SC1007 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.

1.1.2 Syllabus Overview

Week 1 Introduction to Data Structures and Algorithm

Week 2 Linked List - Linear Search

Week 3 Stack and Queue - Arithmetic Expression

Week 4-5 Tree Traversal - Binary Search

Week 6 AVL Tree

Week 7 Analysis of Algorithm

Week 8 Hash Table + Graph Representation

Week 9 Breath First Search and Depth First Search

Week 10 Backtracking techniques and Permutation problems

Week 11-12 Dynamic Programming and Matching Problems

Week 13 Revision

All materials can be found in SC1007 course site at NTULearn. For Part 2, lectures will be conducted in YouTube: Dr. Loke's Channel. The weight of six programming assignments is 40%. The programming lab test 1 and lab test 2 will contribute another 40%. The final quiz will cover all concepts of this module (part 1 and part 2) and contribute 20%.

CA components	Percentage
6 Assignments	40%
Lab Test 1 & 2	40%
Final Quiz	20%

1.2 What is an algorithm?

In Anany Levitin's *Introduction to The Design & Analysis of Algorithms*, an algorithm is defined as a sequence of unambiguous instructions for solving a problem, i.e., for obtaining a required output for any legitimate input in a finite amount of time.

In Cormen's *Introduction to Algorithms*, an algorithm is any well-defined computational procedure that takes some value, or set of values, as input and produces some value, or set of values, as output.

As you can see in these definitions, firstly, algorithm should be a set of well-defined instructions. Given any input, it will return the corresponding output through these instructions. There is no any ambiguity. That means that today you give input x to execute the program implemented the algorithm. The program will return output y. Tomorrow, you use the same input x to rerun the same program. You should obtain the same output y. There is no randomness issue.

Of course, in practice, some programs consist of randomness elements. Here we do not consider them. Moreover, the random function in finite state machine is not a real random generator.

In Levitin's definition, it mentioned that a finite amount of time. Algorithm we discussed here must be completed in finite time. So, there is no infinite loop.

Generally, algorithms that we are going to discuss in this course needs to fulfil the following three points:

- **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.

1.3 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 eg. C Java or Python.

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

Algorithm is a description of the problem solution. Program is a realization of the algorithm by a programming language. Thus, an algorithm can be implemented in different language with different performance. Even they are implemented in the same language and these programs are correctly implemented the algorithm, their efficiency may not be the same but they usually does not have too big difference.

1.3.1 Example 1: Design Algorithms of an Arithmetic Series

For example, we would like to design an algorithm for calculating arithmetic series. There are many ways to solve the problem. The most intuitive algorithm is the first one. We just run a loop and sum up from 1 to n.

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

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 If we have learn about arithmetic algorithm, we know that the summation formula is the number of term divide by two and multiply the sum of the first term and the last term. 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 We also can use recursive approach to sum up from 1 to n. It is similar to Algorithm 1.

Algorithm 3 Summing Arithmetic Sequence

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

When we study algorithm, we not only learn how to design an algorithm to solve the problem but also study the complexity or efficiency of algorithms. Thus, algorithm is not just coding but also about mathematic analysis.

1.3.2 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

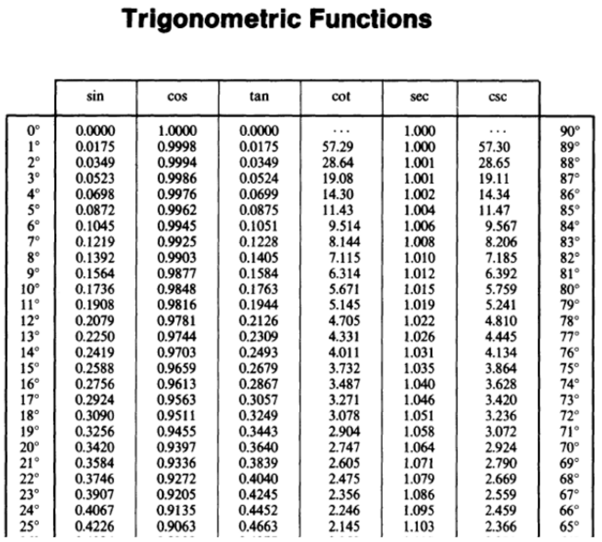
```
function Fibonacci_Recursive(n)
begin
if n<1 then
    return 0
if n==1 OR n==2 then
    return 1
return Fibonacci_Recursive($n-1$)+Fibonacci_Recursive($n-2$)
end
```

Algorithm 5 Fibonacci Sequence: A Simple Iterative Function

```
1: function Fibonacci_Iterative(n)
2: begin
3: if n<1 then
4:     return 0
5: if n==1 OR n==2 then
6:     return 1
7: $F_{-2} \leftarrow 1$
8: $F_{-1} \leftarrow 1$
9: for $i = 3$ to $n$ do
10:     begin
11:     $F_i \leftarrow F_{-2} + F_{-1}$
12:     $F_{-2} \leftarrow F_{-1}$
13:     $F_{-1} \leftarrow F_i$
14:     end
15: return $F_n$
16: end
```

1.3.3 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.

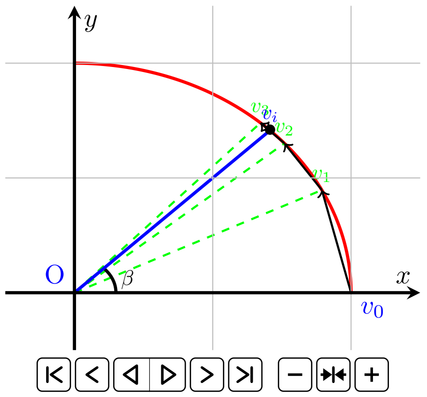
Method 2 Using Maclaurin Series

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

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 and performant 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
```

1.4 Problem Types

1. Searching
2. Graph Problems
3. Combinatorial Problems
4. Sorting
5. String Processing
6. Geometric Problems
7. Numerical Problems

1.4.1 Searching

Finding a particular piece or pieces of information from stored data is a fundamental computing problem. It is known as **searching**. There is no single searching algorithm that can resolve all situations.

If data are not sorted in order, we have to use **linear search/ sequential search**. If data is stored with a special arrangement like binary search tree or hash table, then more efficient algorithms can be used for searching.

One of real-world problems that is required searching algorithm is Sudoku. We need to search a digit from 1 to 9 that has not been appeared in the same row, column and the 3×3 subgrid.

1.4.2 Graph Problems

A graph is one of the important mathematical concepts that can be applied in many real world problems not only in computer science. You can view a graph as a mathematical structure consisting of a collection of vertices and edges. Each edge has one or two vertices associated to it. Path finding is a typical example graph problems.

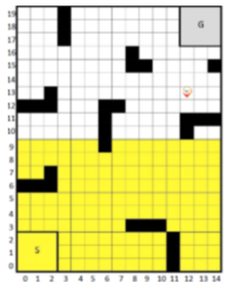


Figure 1.2: Path Finding: Finding the shortest path from a source to a goal

1.4.3 Combinatorial Problems

The study of arrangements, patterns, designs, assignments schedules, connections and configurations. In computer science, patterns of digits are used to encode complicated statements. We have applied them in cryptography. Job assignments are typical problems of matching. A supervisor needs to assign workers to tools or to work areas. University scheduling officers assign teaching duties to faculty members and tutorial groups to students. These problems are matching problems. An electrical engineer considers alternative configurations for a circuit. An industrial engineer consider alternative production schedules and workplace configurations to maximize efficient production.

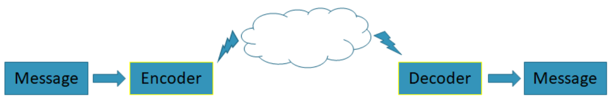


Figure 1.3: Information Transmission: It is a combinatorial problem. We need to ensure that the encoding and decoding are reliable and secure.

Three basic problems of combinatorics.

1. The existence problem: Is there at least one arrangement of a particular kind?
2. The counting problem: How many arrangements are there?
3. The optimization problem: Which arrangement(s) is the best solution?

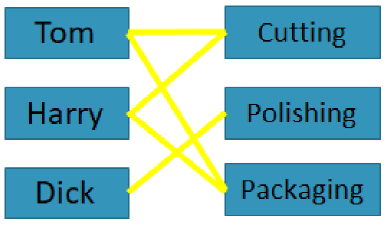


Figure 1.4: A Typical Matching Problem

1.4.4 Sorting

Given a set of records, we rearrange them in a certain order eg. numerical order (ascending or descending) or lexicographical order. This process is known as sorting. Sorting can help us to improve the searching process. For example, if books in a library are not sorted, it won't be easy to find a book you want. A second example is that, because words are well-ordered alphabetically in a dictionary, you can find a word quickly.

If we need to find the top 5% of students in a class. If the results are not sorted, the searching process will become very tedious or nearly impossible to complete the searching tasks.

In computer science, we also study about the stability of sorting algorithm. Stable sorting algorithms sort repeated elements in the same order that they appear in the input. It is important in database sorting when you need to sort the data with multiple columns.

1.4.5 String Processing

A string is a sequence of characters which can comprise letters, numbers, special characters etc. For example, a nucleic acid sequence is a series of a set of five letters that indicate the order of nucleotides forming alleles within a DNA (G, A, C, T) or RNA (G, A, C, U) molecule. The range of sequences in size is from a few nucleotides to billions of base pairs. Searching for a query sequence in a nucleic acid sequence is one of fundamental problems in bioinformatics research.

A segment of a coronavirus genome sequence is as follow:

```
>NC_ 045512.2 Severe acute respiratory syndrome coronavirus 2 isolate Wuhan-Hu-1, complete genome
ATTAAAGGTTTATACCTTCCCAGGTAACAAACCAACCAACTTTCGATCTCTTGTAGATCTGTT
CTCTAAACGAACTTTAAAATCTGTGTGGCTGTCACTCGGCTGCATGCTTAGTGCACTCACGCA
GTATAATTAATAACTAATTACTGTCGTTGACAGGACACGAGTAACTCGTCTATCTTCTGCAGG
CTGCTTACGGTTTCGTCCGTGTTGCAGCCGATCATCAGCACATCTAGGTTTCGTCCGGGTGTG
ACCGAAAGGTAAGATGGAGAGCCTTGTCCCTGGTTTCAACGAGAAAACACACGTCCAACTCAG
TTTGCCTGTTTTACAGGTTCGCGACGTGCTCGTACGTGGCTTTGGAGACTCCGTGGAGGAGGT
CTTATCAGAGGCACGTCAACATCTTAAAGATGGCACTTGTGGCTTAGTAGAAGTTGAAAAAGG
CGTTTTGCCTCAACTTGAACAGCCCTATGTGTTCATCAAACGTTCGGATGCTCGAACTGCACC
TCATGGTCATGTTATGGTTGAGCTGGTAGCAGAACTCGAAGGCATTCAGTACGGTCGTAGTGG
TGAGACACTTGGTGTCCTTGTCCCTCATGTGGGCGAAATACCAGTGGCTTACCGCAAGGTTCT
TCTTCGTAAGAACGGTAATAAAGGAGCTGGTGGCCATAGTTACGGCGCCGATCTAAAGTCATT
TGACTTAGGCGACGAGCTTGGCACTGATCCTTATGAAGATTTTCAAGAAAACTGGAACACTAA
ACATAGCAGTGGTGTTACCCGTGAACTCATGCGTGAGCTTAACGGAGGGGCATACACTCGCTA
....
```

String searching, matching and alignment are typical string processing problems. These processing can help us to have better understanding about the information.

1.4.6 Geometric Problems

Geometric problems deal with geometric objects such as points, lines and polygons. The geometric algorithms are typically applied in computer graphics, robotics and tomography. For example, **convex hull problem** is finding the smallest convex polyhedron/polygon containing all the points. **Delaunay triangulation** is a triangulation algorithm for a given set of discrete points in a plane.

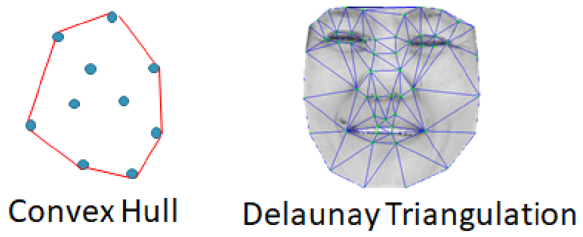


Figure 1.5: Examples of convex hull and Delaunay triangulation

1.4.7 Numerical Problems

Solving equations and systems of queations, computing definite integrals, evaluating functions etc. are numerical problems. It is widely used for solving problems of engineering and mathematical models eg. Newton's methods, Gaussian elimination and linear programming.

$$\begin{array}{lll} \text{minimize} & 3x + 5y & \\ \text{subject to} & 5x + 3y & \geq 29 \\ & -2x + y & \leq 6 \\ & 2x + 5y & \leq 66 \\ & 7x - 4y & \leq 16 \end{array}$$

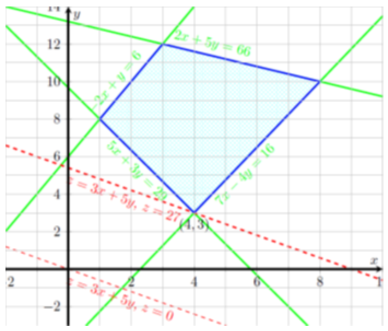


Figure 1.6: An example of a system of linear contraints and a linear objective function

1.5 Algorithm Design Strategies

A general approach to solving problems algorithmically that is applicable to a variety of problems from different areas of computing

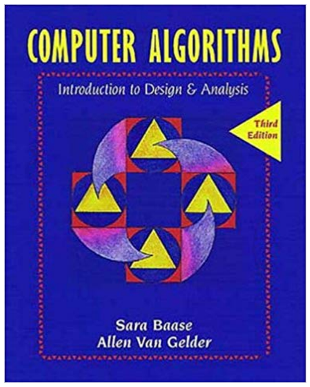
- Brute Force and Exhaustive Search
- Divide-and-Conquer
- Greedy Strategy
- Decrease-and-Conquer

- Transform-and-Conquer
- Iterative Improvement

1.6 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