SC1007
Data Structures and Algorithms

Introduction

Dr Liu Siyuan (syliu@ntu.edu.sg)

N4-02C-117a

Office Hour: Mon & Wed 4-5pm

School of Computer Science and Engineering

Course Schedule

Week	Lecture Topic	Tutorial	Lab	Assignment Deadline
1	Introduction to Data Structure			
2	Introduction Linked List (LL)			
3	Linked List (LL) – Linear Search		Lab 1 (LL)	
4	Stack and Queue (SQ) – Arithmetic Operations	T1(LL)	Lab 2 (SQ)	AS1: LL (**09/02/2024**)
5	Binary Trees (BT) and Binary Search Trees			
6	Binary Trees - Binary Search and AVL Trees	T2 (SQ)	Lab 3 (BT)	AS2: SQ (**23/02/2024**)
7	Analysis of Algorithm (AA)	T3 (BT & BST)	Lab 4 (BST)	AS3: BT (**01/03/2024**)
	Lab Test 1 (Recess Week: 05/03/2024 – 06/03/2024)			
8	Hash Table, Graph Representation			
9	BFS, DFS	T4 (AA)	Lab 5 (Hash Table)	
10	Backtracking, Permutation	T5 (Hash Table)	Lab 6 (Graph, BFS, DFS)	AS4: Hash Table (**29/03/2024**)
11	Dynamic Programming		Lab 7 (Backtracking)	
12	Matching	T6 (Graph)	Lab 8 (DP)	AS5: Graph (**12/04/2024**)
13	Revision			AS6: Permutation/ Matching (**19/04/2024**)
14	**Lab Test 2 + Quiz (23/04/2024 – 24/04/2024)**			

Course Schedule (Part-Time)

Week	Date	1830-1930	1930-2030	2030-2130	Venue
7	29-Feb-24	T3 (BT & BST)	Lab 4 (BST)	Lab 4 (BST)	SW2
Recess	07-Mar-24		**Lab Test 1**		SW2
8	14-Mar-24	Analysis of Algorithm (AA)		Hash Table	TR+3
9	21-Mar-24	Graph Representation, BFS, DFS	T4 (AA)	Lab 5 – Hash Table	SW2
10	28-Mar-24	Backtracking, Permutation	Lab 6 - Graph	Lab 6 - Graph	SW2
11	04-Apr-24	Dynamic Programming	Lab 7 - Backtracking	Lab 7 - Backtracking	SW2
12	11-Apr-24	T5 (Hash Table)	Lab 8 – Dynamic Programming	Lab 8 – Dynamic Programming	SW2
13	18-Apr-24	Matching Problem	Matching Problem	T6 (Graph)	TR+3
	25-Apr-24		**Lab Test 2 + Final Quiz**		SW2

Overview of SC1007

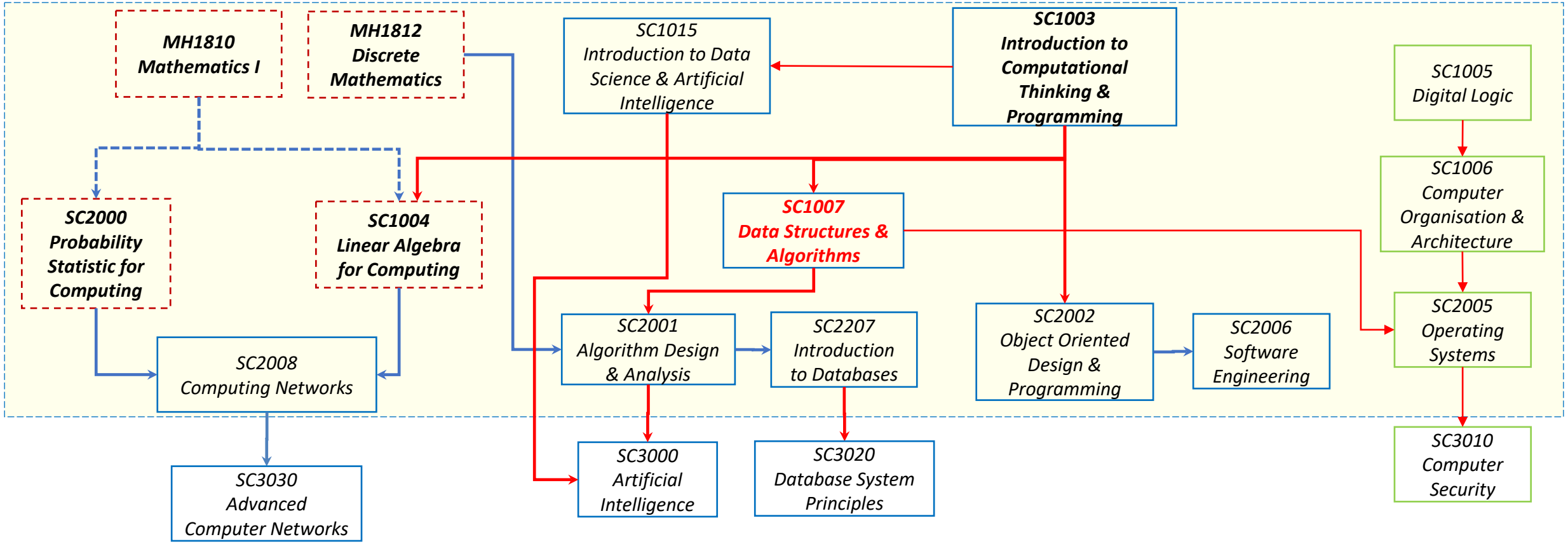
Data Structures:

- Concepts of pointers and structures (aggregates)
- Introduce some classical data structures
  - Linear: Linked list, stack, queue
  - Non-linear: tree
- Implement these data structures

Algorithms:

- Analysis of Algorithm – time complexity and space complexity
- Introduce to some typical algorithms and their applications
- Introduce to some algorithm design strategies

Why Learn Algorithms?



Why Learn Algorithms?

- Given two arrays num1 and num2. Both are sorted in ascending order. The length is m and n, respectively. Please find the median of the two arrays. The time complexity needs to be $O(\log(m+n))$.
- You are given a grid with two dimensions. The grid cell values are 1 or 0 only, where 1 represents land and 0 represents water. An island is lands surrounded by water. Each island is formed by connecting nearby lands vertically and/or horizontally. Please give the number of island in the given grid.
- Given a string s, please find the longest palindrome substring (e.g., tenet).

Why Learn Algorithms?

To Continuously Build a Way of Thinking.

What is an Algorithm?

- Appear in Webster's New World Dictionary after 1957
- It is derived from the name of a Persian Mathematician in the 9th century.
- Euclidean algorithm for finding the greatest common divisor of two numbers – Euclid's Elements (300 B.C.)

algorithm

/ˈalgərɪð(ə)m/

noun

noun: **algorithm**; plural noun: **algorithms**

a process or set of rules to be followed in calculations or other problem-solving operations, especially by a computer.
"a basic **algorithm for** division"

Origin

late 17th century (denoting the Arabic or decimal notation of numbers): variant (influenced by Greek *arithmos* 'number') of Middle English *algorism*, via Old French from medieval Latin *algorismus* . The Arabic source, *al-Ḵwārizmī* 'the man of Ḵwārizm' (now Khiva), was a name given to the 9th-century mathematician Abū Ja'far Muhammad ibn Mūsa, author of widely translated works on algebra and arithmetic.

Translate algorithm to Choose language

Use over time for: algorithm

Definitions from Oxford Languages Feedback

https://en.wikipedia.org/wiki/Algorithm
Knuth's The Art of Computer Programming

What is an Algorithm?

- An algorithm is 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.

Introduction to The Design & Analysis of Algorithms

-Anany Levitin

- 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.

Introduction to Algorithms

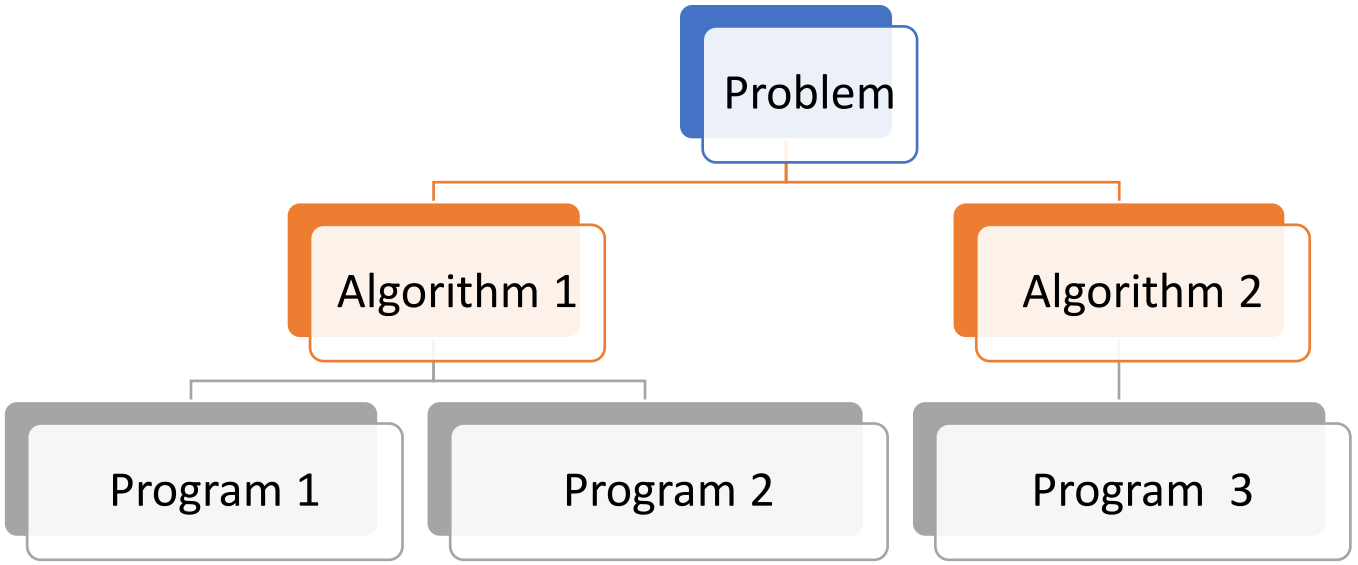
-T. H. Cormen et. al.

What is an Algorithm?

- Correctness:
  - Output results must be correct and consistent for every given input instance
- Precision:
  - A series of well-defined and systematic steps
  - The steps should not contain any ambiguous word like maybe, roughly, about etc.
- Finiteness:
  - Terminates in a finite number of instructions

Algorithm VS. Program

- A computer program is an instance, or concrete representation of an algorithm in some programming languages.
- Implementation is the task of turning an algorithm into a computer program.



Example 1: Arithmetic Series

- There are many ways (algorithms) to solve a problem
- Summing up 1 to n, i.e., $1 + 2 + \cdots + n$

Algorithm 1 Summing Arithmetic Sequence

```
1: function Method_One(n)
2: begin
3: sum ← 0
4: for i = 1 to n do
5:     sum ← sum + i
6: end
```

Algorithm 2 Summing Arithmetic

```
1: function Method_Two(n)
2: begin
3: sum ← n * (1 + n)/2
4: end
```

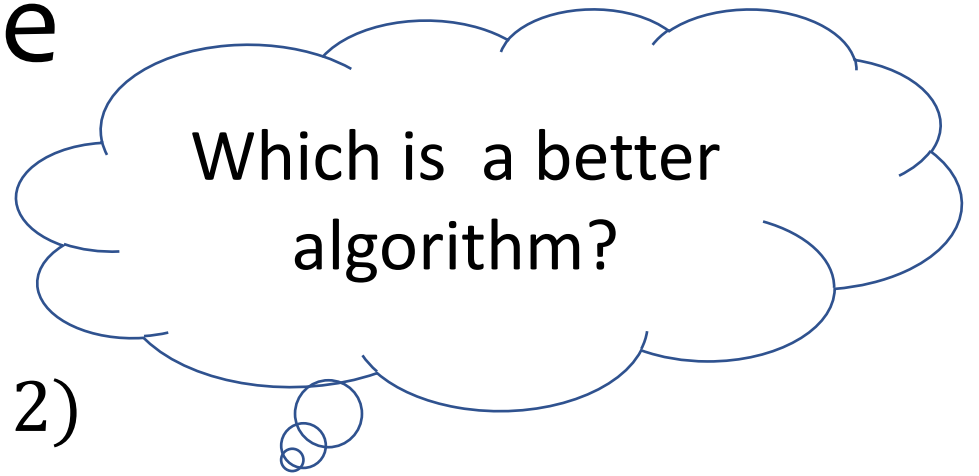
Algorithm 3 Summing Arithmetic Sequence

```
1: function Method_Three(n)
2: begin
3: if n=1 then
4:     return 1
5: else
6:     return  n+Method_Three(n − 1)
7: end
```

Example 2: Fibonacci Sequence

- 1, 1, 2, 3, 5, 8, …
- The n^{th} term is

$$f(n) = f(n-1) + f(n-2)$$



Algorithm 4 Fibonacci Sequence: A Simple Recursive Function

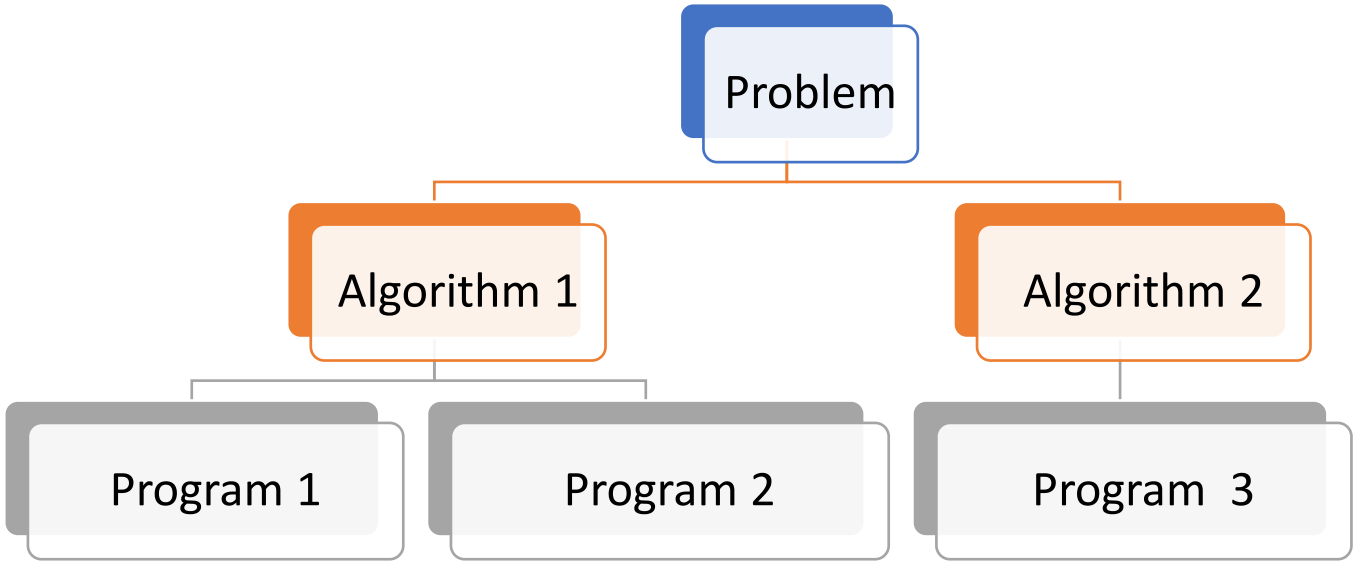
```
1: function Fibonacci_Recursive(n)
2: begin
3: if n<1 then
4:     return 0
5: if n==1 OR n==2 then
6:     return 1
7: return Fibonacci_Recursive(n − 1)+Fibonacci_Recursive(n − 2)
8: end
```

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_1 ← 1
8: F_2 ← 1
9: for i = 3 to n do
10:    begin
11:    F_i ← F_{i−2} + F_{i−1}
12:    F_{i−2} ← F_{i−1}
13:    F_{i−1} ← F_i
14:    end
15: return F_n
16: end
```

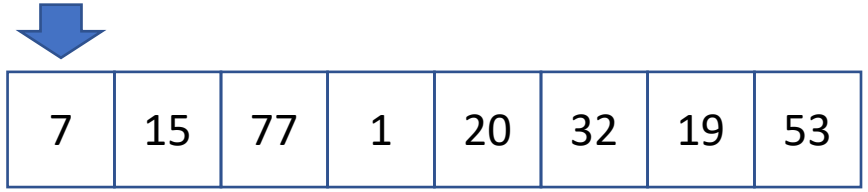
Problem Types

- Searching
- Graph Problems
- Combinatorial Problems
- Sorting (SC2001/CZ2101)
- String Processing (SC2001/CZ2101)
- Geometric Problems
- Numerical Problems



Searching: Find a Search Key in a Given Set

20

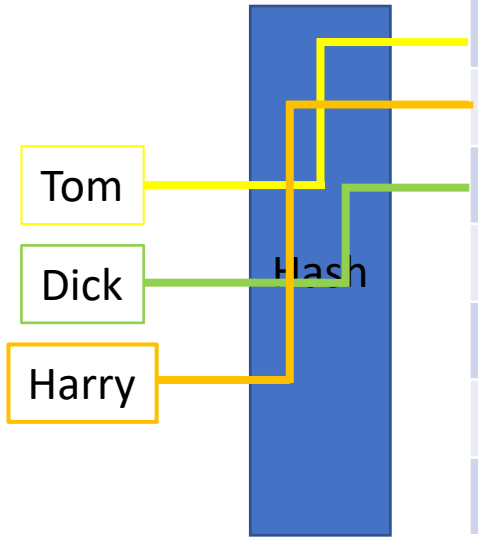


Linear Search/ Sequential Search

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

Sudoku

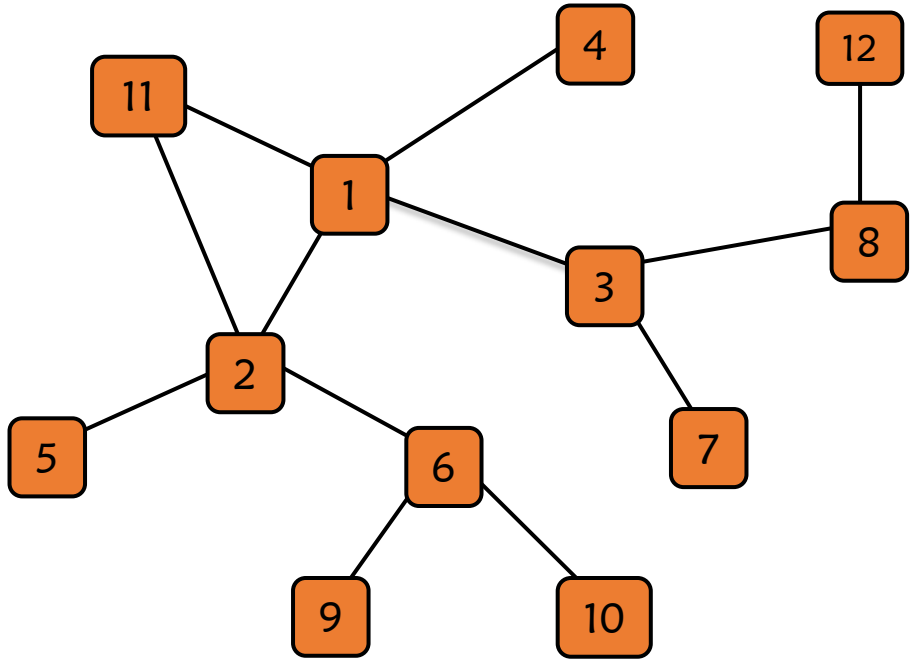
Hash Table

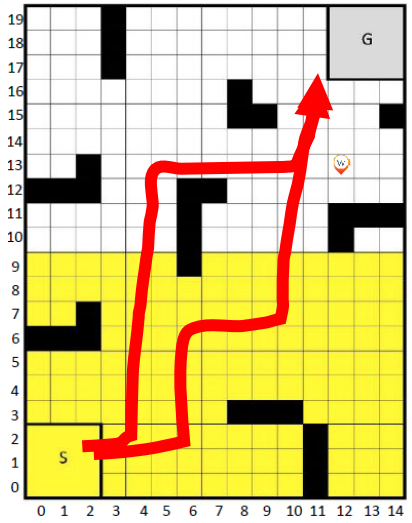


Index	Data
001	Tom, +123456
002	Harry, +369852
003	Dick, +965483
..	…
…	…
..	..
..	..

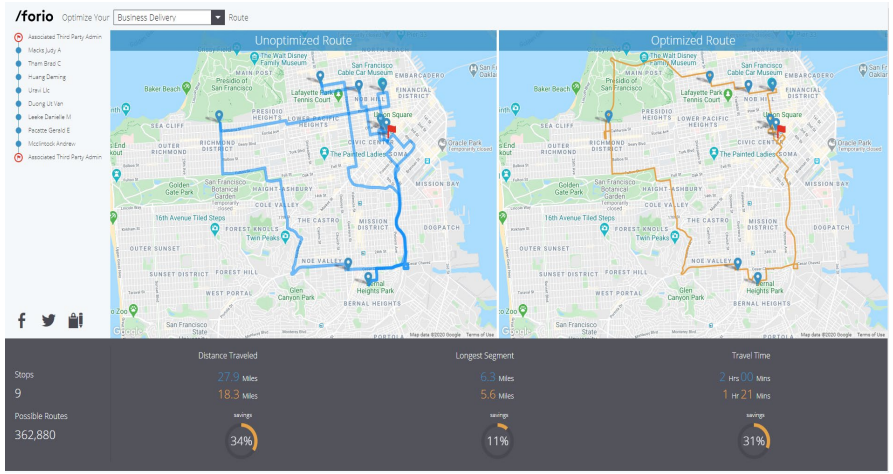
Graph Problems

- A graph is a mathematical structure consisting of a collection of vertices and edges.
- Each edge has one or two vertices associated to it.

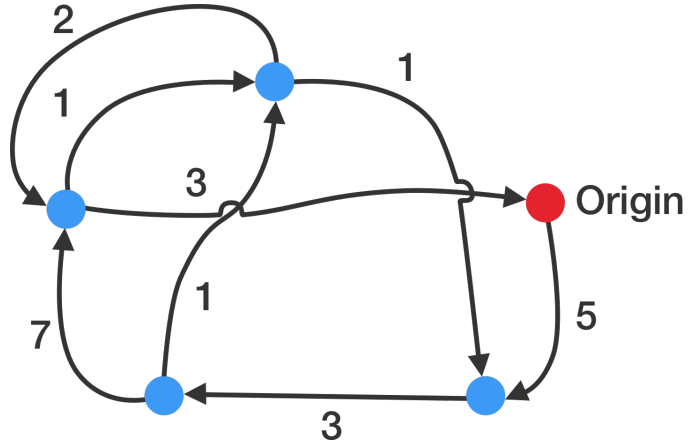




Path Finding



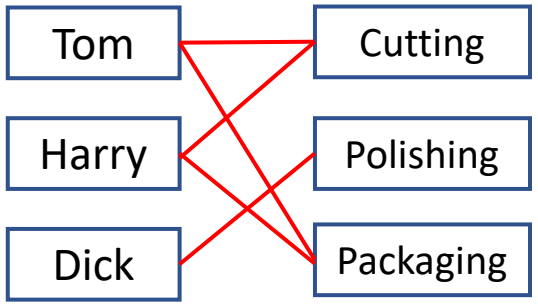
https://forio.com/app/showcase/route-optimizer/

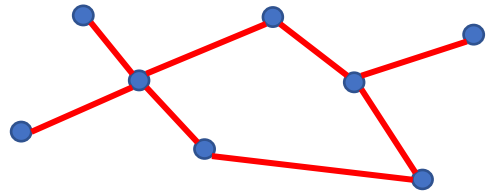


Travelling Salesman Problem

Combinatorial Problems

- The study of arrangements, patterns, designs, assignments schedules, connections and configurations.





Minimum Vertex Cover Problem

Sorting Problems

- Rearrange items of a given list in certain order

Numerical Order

Lexicographical Order

- Find the median

7	15	77	1	32	19	53

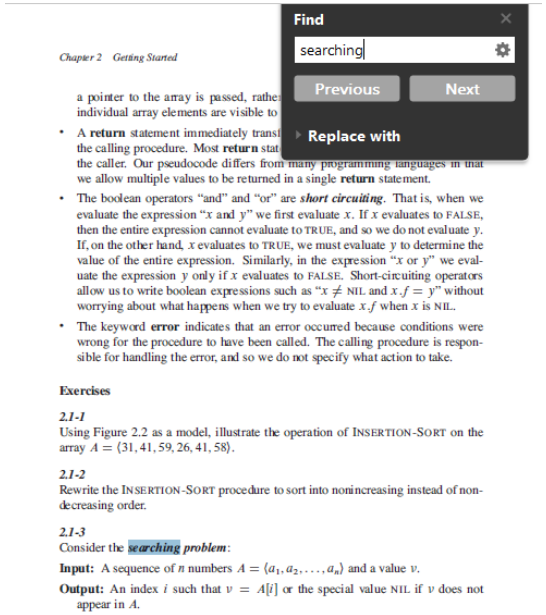
19

String Processing

String matching

PNEUMONOULTRAMICROSCOPICSILICOVOLCANOCONIOSIS

M	I	C	R	O



Text Matching

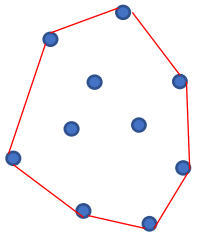
```
   1 attaaaggtt tataccttcc caggtaacaa accaaccaac tttcgatctc ttgtagatct
  61 gttctctaaa cgaactttaa aatctgtgtg gctgtcactc ggctgcatgc ttagtgcact
 121 cacgcagtat aattaataac taattactgt cgttgacagg acacgagtaa ctcgtctatc
 181 ttctgcaggc tgcttacggt ttcgtccgtg ttgcagccga tcatcagcac atctaggttt
 241 cgtccgggtg tgaccgaaag gtaagatgga gagccttgtc cctggtttca acgagaaaac
 301 acacgtccaa ctcagtttgc ctgttttaca ggttcgcgac gtgctcgtac gtggctttgg
 361 agactccgtg gaggaggtct tatcagaggc acgtcaacat cttaaagatg gcacttgtgg
 421 cttagtagaa gttgaaaaag gcgttttgcc tcaacttgaa cagccctatg tgttcatcaa
 481 acgttcggat gctcgaactg cacctcatgg tcatgttatg gttgagctgg tagcagaact
 541 cgaaggcatt cagtacggtc gtagtggtga gacacttggt gtccttgtcc ctcatgtggg
 601 cgaaatacca gtggcttacc gcaaggttct tcttcgtaag aacggtaata aaggagctgg
 661 tggccatagt tacggcgccg atctaaagtc atttgactta ggcgacgagc ttggcactga
 721 tccttatgaa gattttcaag aaaactggaa cactaaacat agcagtggtg ttacccgtga
 781 actcatgcgt gagcttaacg gaggggcata cactcgctat gtcgataaca acttctgtgg
 841 ccctgatggc taccctcttg agtgcattaa agaccttcta gcacgtgctg gtaaagcttc
 901 atgcactttg tccgaacaac tggactttat tgacactaag aggggtgtat actgctgccg
 961 tgaacatgag catgaaattg cttggtacac ggaacgttct gaaaagagct atgaattgca
1021 gacacctttt gaaattaaat tggcaaagaa atttgacacc ttcaatgggg aatgtccaaa
1081 ttttgtattt cccttaaatt ccataatcaa gactattcaa ccaagggttg aaaagaaaaa
1141 gcttgatggc tttatgggta gaattcgatc tgtctatcca gttgcgtcac caaatgaatg
1201 caaccaaatg tgcctttcaa ctctcatgaa gtgtgatcat tgtggtgaaa cttcatggca
1261 gacgggcgat tttgttaaag ccacttgcga attttgtggc actgagaatt tgactaaaga
1321 aggtgccact acttgtggtt acttacccca aaatgctgtt gttaaaattt attgtccagc
1381 atgtcacaat tcagaagtag gacctgagca tagtcttgcc gaataccata atgaatctgg
1441 cttgaaaacc attcttcgta agggtggtcg cactattgcc tttggaggct gtgtgttctc
1501 ttatgttggt tgccataaca agtgtgccta ttgggttcca cgtgctagcg ctaacatagg
1561 ttgtaaccat acaggtgttg ttggagaagg ttccgaaggt cttaatgaca accttcttga
1621 aatactccaa aaagagaaag tcaacatcaa tattgttggt gactttaaac ttaatgaaga
```

SARS-CoV-2/human/USA/UNC_200265_2020/2020, complete genome

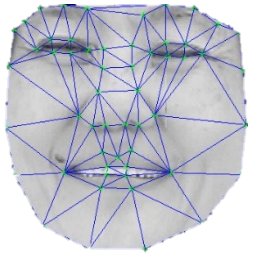
```
   1 aacaaaccaa ccaactttcg atctcttgta gatctgttct ctaaacgaac tttaaaatct
  61 gtgtggctgt cactcggctg catgcttagt gcactcacgc agtataatta ataactaatt
 121 actgtcgttg acaggacacg agtaactcgt ctatcttctg caggctgctt acggtttcgt
 181 ccgtgtttgca gccgatcatc agcacatcta ggttttgtcc gggtgtgacc gaaaggtaag
 241 atggagagcc ttgtccctgg tttcaacgag aaaacacacg tccaactcag tttgcctgtt
 301 ttacaggttc gcgacgtgct cgtacgtggc tttggagact ccgtggagga ggtcttatca
 361 gaggcacgtc aacatcttaa agatggcact tgtggcttag tagaagttga aaaaggcgtt
 421 ttgcctcaac ttgaacagcc ctatgtgttc atcaaacgtt cggatgctcg aactgcacct
 481 catggtcatg ttatggttga gctggtagca gaactcgaag gcattcagta cggtcgtagt
 541 ggtgagacac ttggtgtcct tgtccctcat gtgggcgaaa taccagtggc ttaccgcaag
 601 gttcttcttc gtaagaacgg taataaagga gctggtggcc atagttacgg cgccgatcta
 661 aagtcatttg acttaggcga cgagcttggc actgatcctt atgaagattt tcaagaaaac
 721 tggaacacta aacatagcag tggtgttacc cgtgaactca tgcgtgagct taacggaggg
 781 gcatacactc gctatgtcga taacaacttc tgtggccctg atggctaccc tcttgagtgc
 841 attaaagacc ttctagcacg tgctggtaaa gcttcatgca ctttgtccga acaactggac
 901 tttattgaca ctaagagggg tgtatactgc tgccgtgaac atgagcatga aattgcttgg
 961 tacacggaac gttctgaaaa gagctatgaa ttgcagacac cttttgaaat taaattggca
1021 aagaaatttg acatcttcaa tggggaatgt ccaaattttg tatttccctt aaattccata
1081 atcaagacta ttcaaccaag ggttgaaaag aaaaagcttg atggctttat gggtagaatt
1141 cgatctgtct atccagttgc gtcaccaaat gaatgcaacc aaatgtgcct ttcaactctc
1201 atgaagtgtg atcattgtgg tgaaacttca tggcagacgg gcgattttgt taaagccact
1261 tgcgaatttt gtggcactga gaatttgact aaagaaggtg ccactacttg tggttactta
1321 ccccaaaatg ctgttgttaa aatttattgt ccagcatgtc acaattcaga agtaggacct
1381 gagcatagtc ttgccgaata ccataatgaa tctggcttga aaaccattct tcgtaagggt
1441 ggtcgcacta ttgcctttgg aggctgtgtg ttctcttatg ttggttgcca taacaagtgt
1501 gcctattggg ttccacgtgc tagcgctaac ataggttgta accatacagg tgttgttgga
1561 gaaggttccg aaggtcttaa tgacaacctt cttgaaatac tccaaaaaga gaaagtcaac
1621 atcaatattg ttggtgactt taaacttaat gaagagatcg ccattatttt ggcatctttt
```

Severe acute respiratory syndrome coronavirus 2 isolate Wuhan-Hu-1, complete genome.

Computational Geometric Problem



Convex Hull



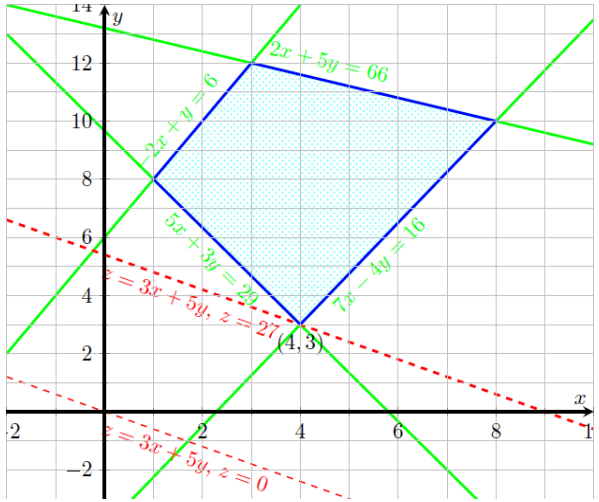
Delaunay Triangulation

- Finding the convex hull of a set of points
- Finding the closest pair of points in a set of points
- Finding the intersection of two line segments or two circles
- Testing whether a point is inside or outside a polygon
- Finding the Voronoi diagram of a set of points
- Finding the shortest path between two points in a planar graph with obstacles
- Constructing a Delaunay triangulation Computing the area of a polygon or the volume of a polyhedron
- Detecting and resolving collisions between objects in a 2D or 3D space

Numerical Problem and Optimization Problem

- Use numerical approximation for the mathematical analysis
- Widely used for solving problems of engineering and mathematical models
  - Newton's method
  - Gaussian elimination
- Linear programming is an optimization technique for a system of linear constraints and a linear objective function

$$\min 3x + 5y$$
$$\text{subject to } 5x + 3y \geq 29$$
$$-2x + y \leq 6$$
$$2x + 5y \leq 66$$
$$7x - 4y \leq 16$$



How do we solve these problems?

How do We Solve these Problems?

- Select appropriate data structures
  - Arrays
  - Linked Lists
    - Singly linked list, doubly linked list, circular linked list etc.
  - Stack and Queue
  - Trees
  - Graphs

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
- …etc.

- Decrease-and-Conquer
- Transform-and-Conquer
- Iterative Improvement

Call for Participations

- Study 1:
  - A VR game to facilitate the learning of graph search algorithm.
  - Time consumed: 30 minutes (game playing and questionnaire)
  - Time and venue: March 16 and 17, 2024 in school
  - Reward: $5
- Study 2:
  - A web game to facilitate the learning of the second part of this module
  - Time consumed: 30 minutes × 3 sessions (each session consists of game playing and questionnaire)
  - Time and venue: Anytime before attending tutorials. The game can be played online.
  - Reward: $15
- Registration:
  - Scan the code, or
  - Drop me email: syliu@ntu.edu.sg