NANYANG TECHNOLOGICAL UNIVERSITY

# CE1007/CZ1007 DATA STRUCTURES

## Lecture 00: **Course Introduction**

Mr Tan Kheng Leong

**College of Engineering**

School of Computer Science and Engineering

# INSTRUCTOR INFORMATION

- **Mr. TAN** Kheng Leong
- Email: **khengleong@ntu.edu.sg**
- Office: **NTU, N4-02b-63**
- Office hours:
  - Review Lecture day
  - Other times by appointment (Email)

# OVERVIEW

- Introduction to many of the basic data structures used in computer software
- Practice design and analysis of data structures
- Practice using these data structures by writing programs
- Make the transformation from programmer to computer scientist

# GOALS

- You will understand
  - What the tools are for storing and processing common data types
  - Which tools are appropriate for which need
- So that you can
  - Make good design choices as a developer, project manager, or system customer
- You will be able to
  - **Justify** your design decisions
  - **Communicate** ideas about programs clearly and precisely

# GOALS

"I will, in fact, claim that the difference between a bad programmer and a good one is whether he considers his code or his data structures more important. Bad programmers worry about the code. Good programmers worry about data structures and their relationships."

Linus Torvalds, 2006

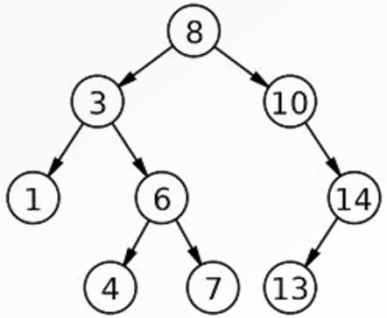
(Creator of the Linux kernel)

# DATA STRUCTURE

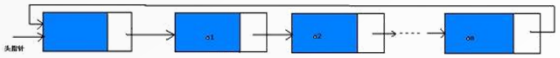
- What is Data Structure?
  - Data structure is a particular way of organizing data in a computer so that it can be used efficiently

**Stack**

**Queue**




**Tree**



**Linked List**

  - About how you store the data when you are coding

# WHAT IS A DATA STRUCTURE?

- Very simply, a particular arrangement of data
- Why do we care?
  - Each arrangement allows you to do some things efficiently…and other things less efficiently
- Solving a given problem efficiently requires
  - The right data structure(s) + right algorithm(s)
- You'll learn more about this in Algorithms

# DATA STRUCTURE

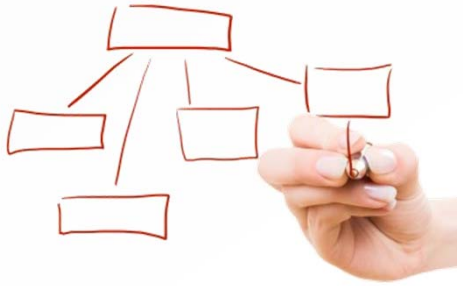
- What is Data Structure?
- Why we study it?
  - Most basic course of Computer Science
  - Program = Data Structure + Algorithm
  - If you code, you have to use data structure

# PICKING THE BEST DATA STRUCTURE FOR THE JOB

- The data structure you pick needs to support the operations you need
- Ideally it supports the operations you will use most often in an efficient manner
- Examples of operations:
  - A **List** with operations **insert** and **delete**
  - A **Stack** with operations **push** and **pop**

# DATA STRUCTURE

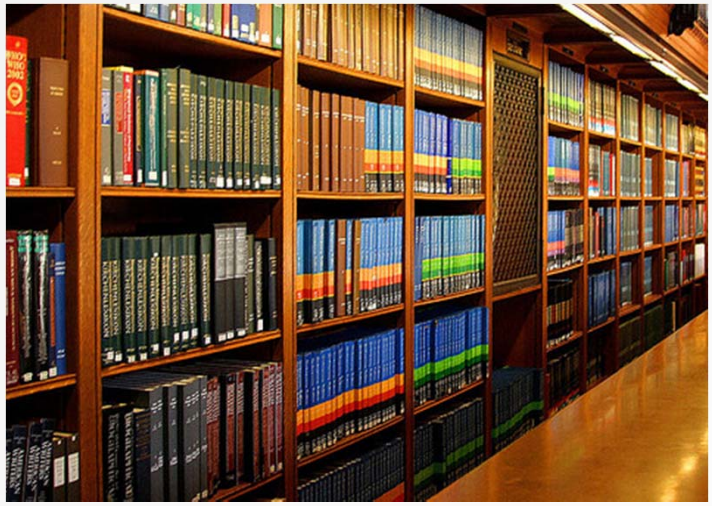
- What is Data Structure?
- Why we study it?
- How we study it?
  - Listen to my lectures
  - Have a clear picture in mind for every concept
  - Practice, practice and practice! (Coding, Debug)



# SOME EXAMPLES

# EXAMPLE 1: LIBRARY SEARCH

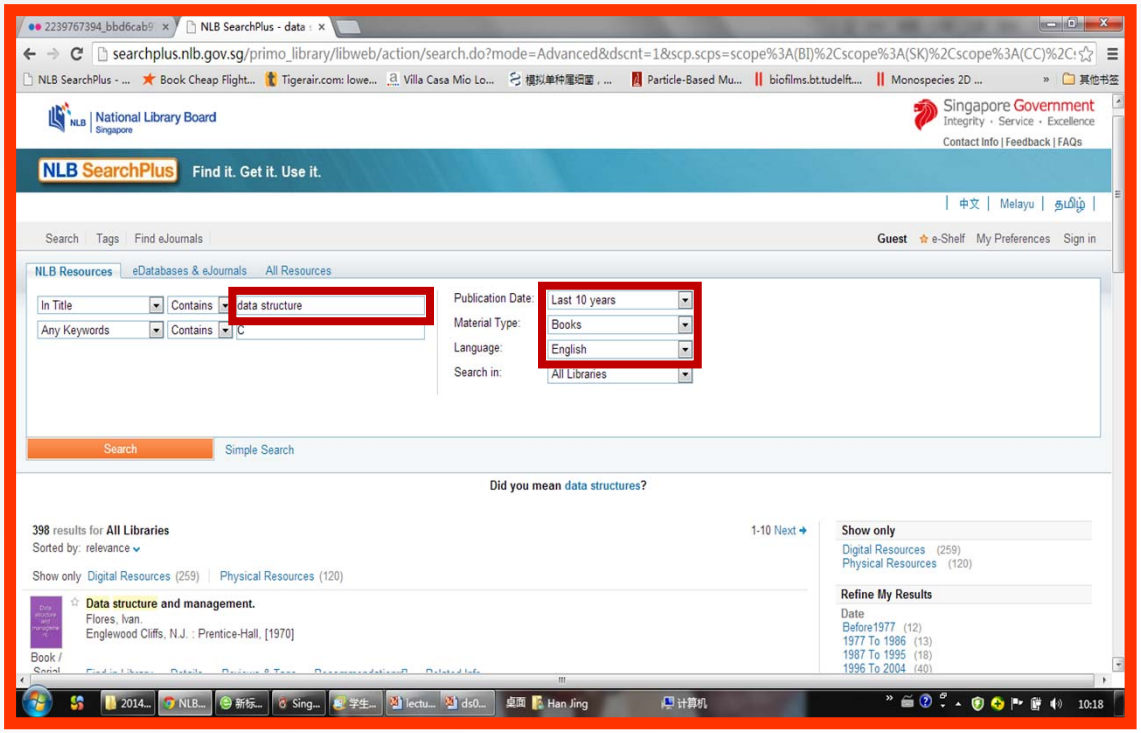
- If I want to find a book that relates to data structure




20 years ago

# EXAMPLE 1: LIBRARY SEARCH

- If I want to find a book that relates to data structure




Now

# DATA STRUCTURE

**Array**

| | Call No. | Title | Author | Year | Publisher |
|---|---|---|---|---|---|
| 001 | C001 | Data Structure | M.Weiss | 2003 | Addison Wesley |
| 002 | C002 | C Programming | X. Man | 2012 | Springer |
| 003 | E001 | English writing | B. Gao | 2000 | Wells |
| 004 | P001 | Physics | B. Jim | 1999 | Basingstoke |
| … | … | … | | | |

Simplest way to store the data

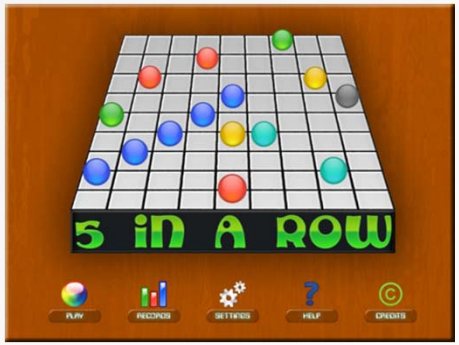
Array of a BOOK structure:

**struct BOOK libraryBook[1000];**

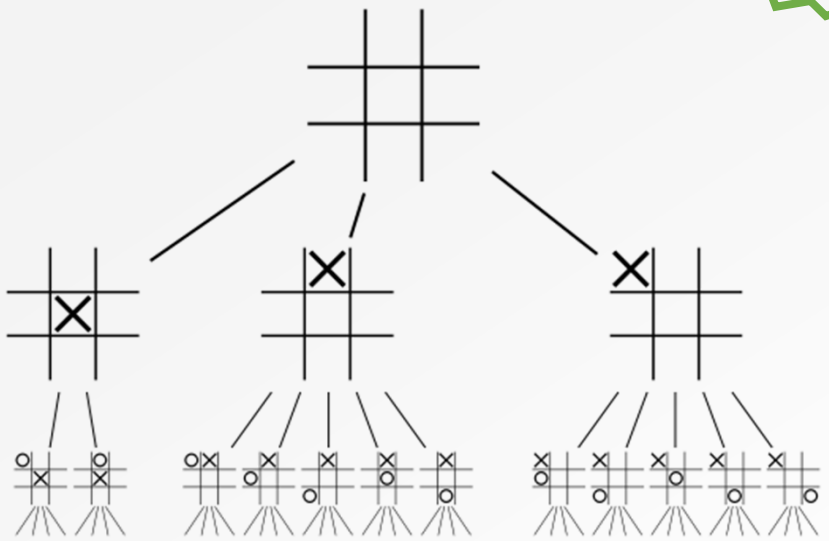
```
Struct BOOK {

char callnum[20];
char title[50];
char author[30];
int  year;
char publisher[20];}
```

# EXAMPLE 2: COMBINATORIAL GAMES

- Many combinatorial game programs (tic-tac-toe, five-in-a-row, chess, etc.) use tree structure for searching the move



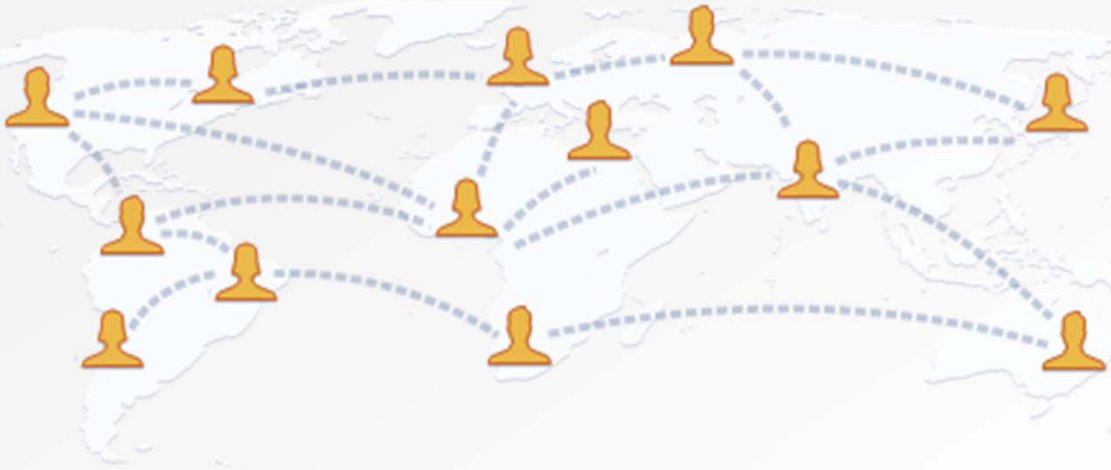

# DATA STRUCTURE



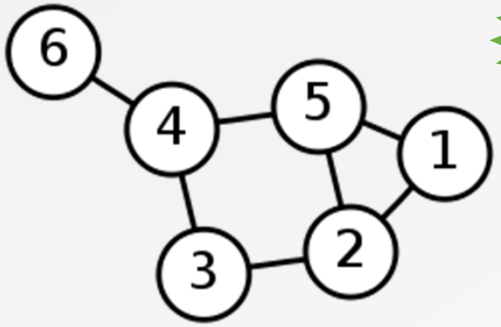
Tree

```
Struct treeNode {
   struct boardConfig bc;
   struct treeNode
                *firstChild;
   struct treeNode
                *brother;
};
```

# EXAMPLE 3: SOCIAL NETWORK



# DATA STRUCTURE




Graph

Dense

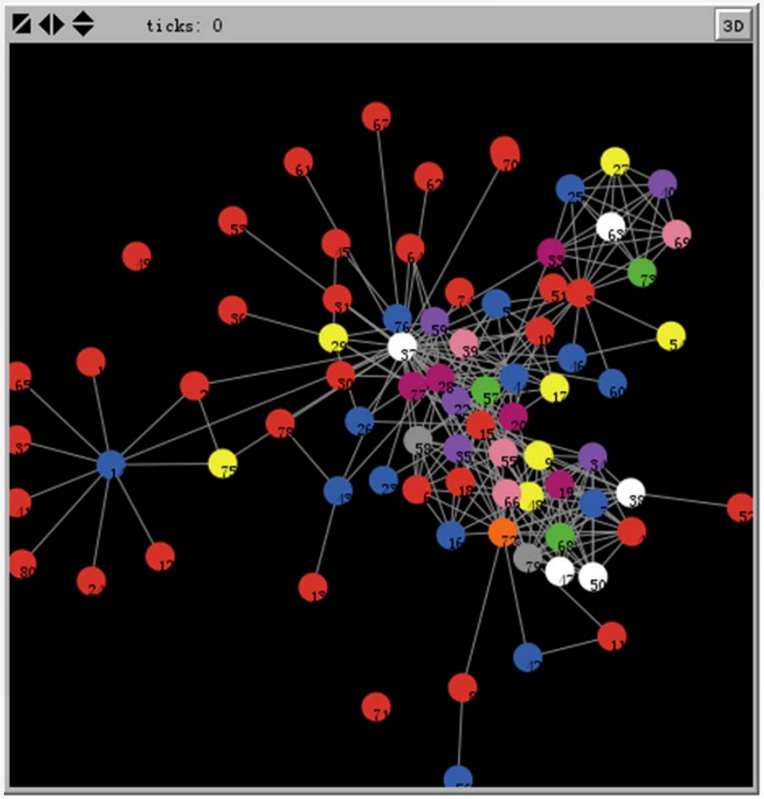
```
Struct node {
Char name[20];
…};

Boolean adjacentMatrix[6][6];
/* if node i and node j are connected,
adjacentMatrix[i][j] = True */
```

Sparse

```
Struct Node {
Char name[20];
…
struct nextNode *first;
};


Struct nextNode{
Int id;
Struct nextNode *next;
}
```

Linked List

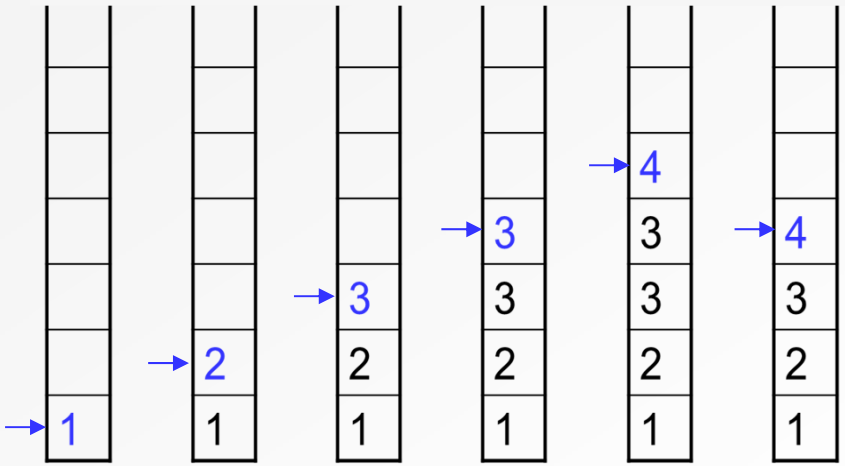
# EXAMPLE 4: GRAPH COLOURING




- Use 10 colours to colour 80 nodes
- The rule is:
  - If two nodes are linked, they should be coloured differently
- Algorithm:
  - Backtracking

- Use **graph structure** to store the graph
- For backtracking algorithm: **stack**

**Graph and Stack**




```
Struct Stack{
 int top;
 int color[80];
};
```

# SO, IF YOU WANT TO WRITE PROGRAMS

- to solve problems (big data, e-commerce, artificial intelligence, scheduling problems, …)

You need to know **DATA STRUCTURE**

# NEXT 6 WEEKS

- What will we be working with?
  - Structures
  - Pointers
  - Structures inside structures
  - Pointers to structures
  - Pointers inside structures
- Make sure you know
  - What pointers/structures are
  - How to declare and use pointers/structures

# NEXT 6 WEEKS KEYPOINTS

1. Linked List
2. Queue
3. Stack
4. Tree

# THINGS YOU SHOULD DO

- Draw lots of pictures
  - Visualizing how objects are laid out in memory helps with understanding
- Concept before code
  - Following pointers can be tricky if you don't have a mental model of the data structure
  - With the right model as a reference, you can implement the structure in any language
- Use the debugger
  - Once you start writing code, you'll do silly things with pointers and you need to be able to track down your mistakes

# ROADMAP (REVISION LECTURES)

| Week | Wed (TCT-LT) |
|---|---|
| 8 | Introduction to Dynamic Data Structures<br>Linked Lists |
| 9 | Linked Lists |
| 10 | Stacks and Queues |
| 11 | Binary Trees |
| 12 | Binary trees<br>Binary Search Trees |
| 13 | Revision |

# ROADMAP (LABS, TUTORIALS)

| Week | Tutorial | Lab |
|---|---|---|
| 8 | No Tutorial | No Labs |
| 9 | Dynamic Data Structure & Linked Lists | Dynamic Data Structure & Linked Lists |
| 10 | Linked Lists | Linked Lists |
| 11 | Stack and Queues | Stack and Queues |
| 12 | Binary Trees | Binary Trees |
| 13 | Binary Search Trees | Binary Search Trees |

# ROADMAP (ASSIGNMENTS AND LAB TESTS)

| Week | Topic | Deadline |
|---|---|---|
| 10 | Linked Lists | Oct 25th, 2019 |
| 11 | Stacks and Queues | Nov 1st , 2019 |
| 12 | Binary Trees | Nov 8th, 2019 |
| 13 | Binary Search Trees | Nov 15th, 2019 |
| 14 | **Lab Test 2** | **Nov 18th, 2019 (Monday)** |