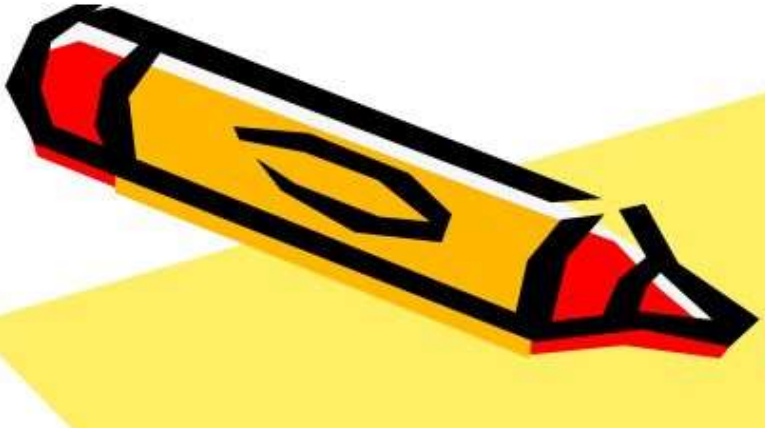


Welcome to My Class!

Nice to see you !

Who am I?

❖ Here in CSE,VU……

- **Hired Faculty/ Adjunct Faculty Since Spring 2022**

❖ My Origin……

- ***Associate Professor***
  - *Dept. of Information and Communication Engineering.*

❖ My Education…..

- **B.Sc. in ICE, RU (2008)**
- **M.Sc. in ICE, RU (2009)**
- **Ph.D. in ICS, Saitama Uni, JP (2016)**

@ RU since 2003

DR. MD. GOLAM RASHED

More details about me @ http://www.ru.ac.bd/ice

Contact me through

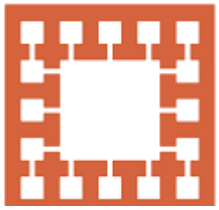
- Mobile: **+88-01717-515008**
- Email: golamrashed@ru.ac.bd
  golamrashed.ru@gmail.com

DR. MD. GOLAM RASHED

CSE 2103
(Data Structures)

Lecture on
Chapter-1: Introduction

By
Dr. M. Golam Rashed
(golamrashed@ru.ac.bd)



Department of Computer Science and Engineering (CSE)
Pundra University of Science and Technology, Bogra-Bangladesh

Text Books:

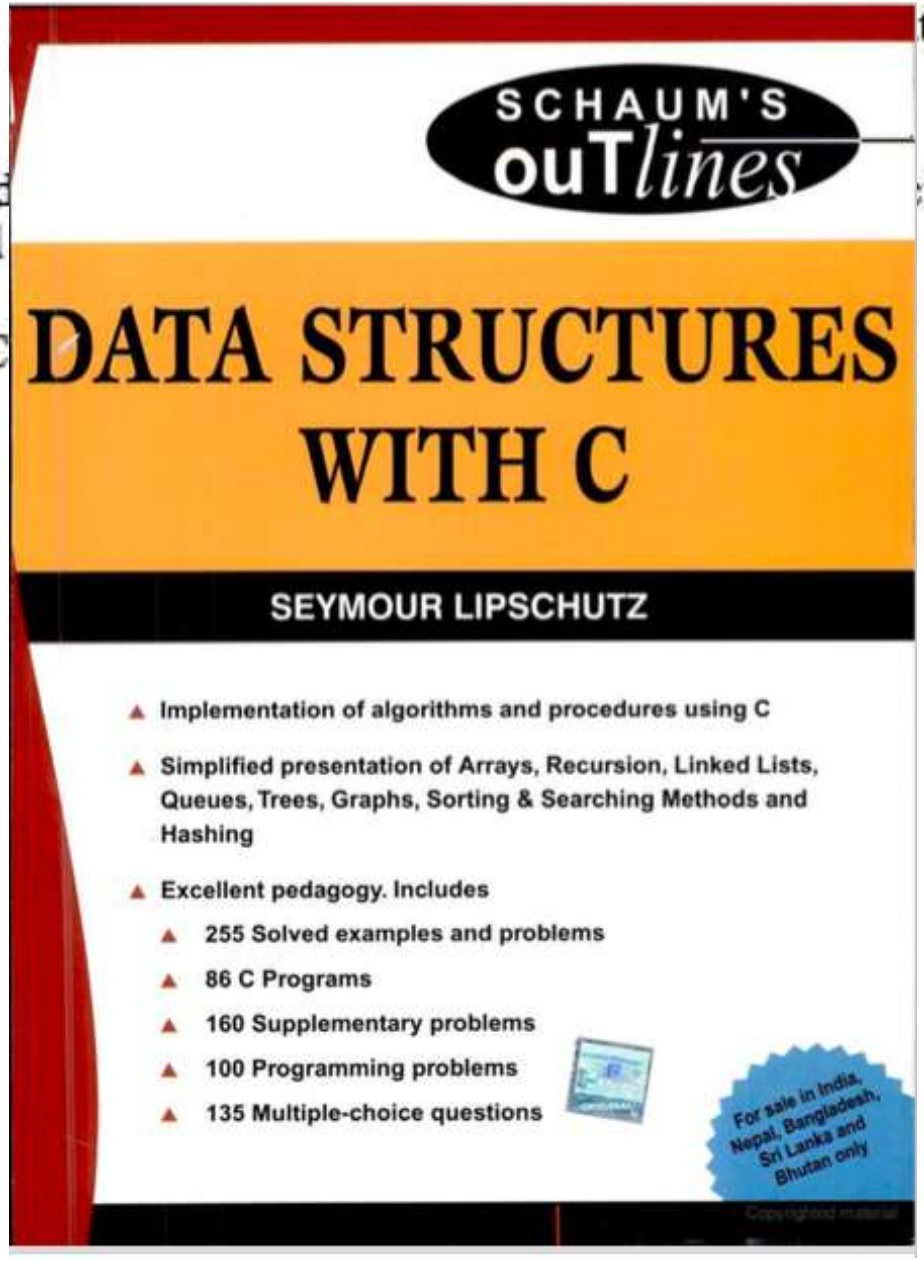
1. S Lipschutz

Reference Books:

2. E. Horowitz and
3. E. Horowitz and
4. Reingold
5. T. H. Cormen, C

ta Structures

tures



120

75

55

Seymour Lipschutz

- ✓ Data are simple values or set of values.
- ✓ Data item refers to a single unit of values
- ✓ Data items that are divided into sub-items are called group items.

For example:

An employee's name may be divided into three sub item…..

- ❖ First name
- ❖ Middle name
- ❖ Last Name

But, NID number would be normally be treated as a single item

120 km in distance

75 kgs in weight

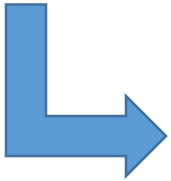
55 cm in height

Writer: Seymour Lipschutz

- Collection of data are frequently organized into a
  - Hierarchy of fields
  - Records
  - Files

Entity?

An entity is something that has certain ATTRIBUTE or PROPERTIES which may be assigned **VALUES.**



may be numeric or non-numeric

ATTRIBUTE	NAME	Age	Sex	Height	NID
Values	Jhon	30	Male	65 cm	27642847

Entity Set?

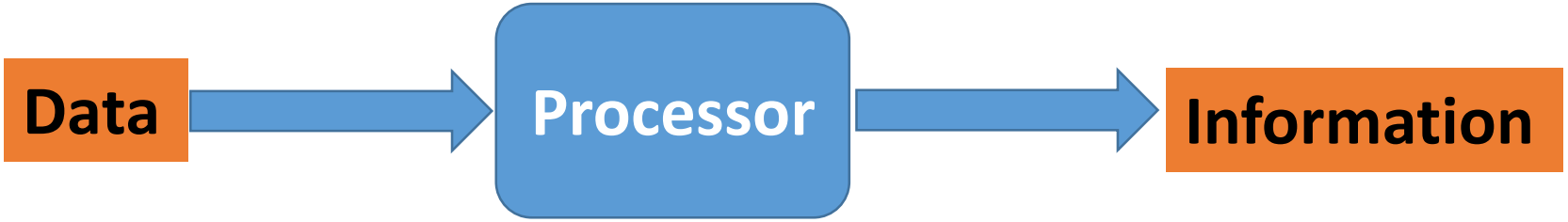
Entities with similar ATTRIBUTES form entity set.

Example:

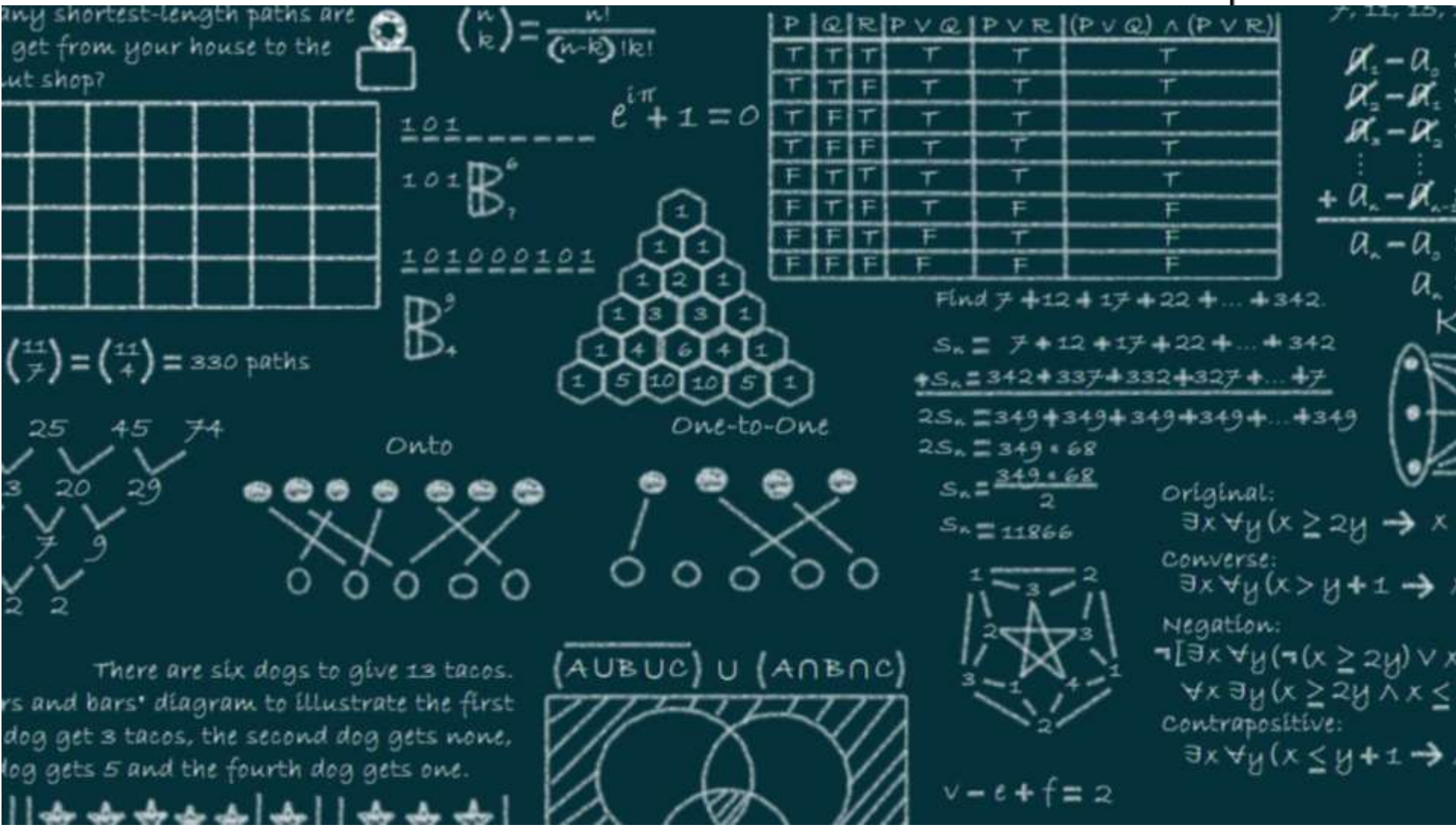
- All the employee in an organization
- All the students of any department.

✓**Each attribute of an entity set has a range of values**

Information?



Processed dare are usually called information


any shortest-length paths are
get from your house to the
ut shop?
$\binom{n}{k} = \frac{n!}{(n-k)!k!}$
$e^{i\pi} + 1 = 0$
101
101
1 0 1 0 0 0 1 0 1
$\binom{11}{7} = \binom{11}{4} = 330$ paths
P Q R P∨Q P∨R (P∨Q)∧(P∨R)
T T T T T T
T T F T T T
T F T T T T
T F F T T T
F T T T T T
F T F T F F
F F T F T F
F F F F F F
Find 7 + 12 + 17 + 22 + ... + 342.
$S_n = 7 + 12 + 17 + 22 + \ldots + 342$
$+S_n = 342 + 337 + 332 + 327 + \ldots + 7$
$2S_n = 349 + 349 + 349 + 349 + \ldots + 349$
$2S_n = 349 \cdot 68$
$S_n = \frac{349 \cdot 68}{2}$
$S_n = 11866$
25 45 74
3 20 29
7 9
2 2
Onto
One-to-One
There are six dogs to give 13 tacos.
rs and bars' diagram to illustrate the first
dog get 3 tacos, the second dog gets none,
log gets 5 and the fourth dog gets one.
(AUBUC) U (A∩B∩C)
$V - e + f = 2$
Original:
$\exists x \forall y (x \geq 2y \rightarrow x$
Converse:
$\exists x \forall y (x > y + 1 \rightarrow$
Negation:
$\neg[\exists x \forall y (\neg(x \geq 2y) \vee x$
$\forall x \exists y (x \geq 2y \wedge x \leq$
Contrapositive:
$\exists x \forall y (x \leq y + 1 \rightarrow$

Data Structure?

Data may be organized in many different ways.

The logical or mathematical model of a particular organization of data is called a data structure

✓ **Particular data model depends on TWO consideration**:

1. It must be rich enough in structure to mirror the actual relationships of the data in real world.
2. The structure should be simple enough that can be efficiently process the data when necessary.

Some Data Structure

CSE 2103

	STUDENT
1	John Brown
2	Sandra Gold
3	Tom Jones
4	June Kelly
5	Mary Reed
6	Alan Smith

One D array

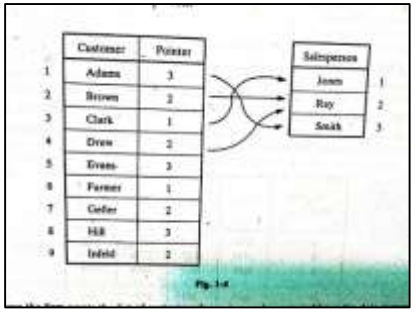
Store \ Dept.	1	2	3	4
1	2872	805	3211	1560
2	2196	1223	2525	1744
3	3257	1017	3686	1951
...	...	...	...	...
28	2618	931	2333	982

Fig. 1-2

Two D array

	Customer	Salesperson
1	Adams	Smith
2	Brown	Ray
3	Clark	Jones
4	Drew	Ray
5	Evans	Smith
6	Farmer	Jones
7	Geller	Ray
8	Hill	Smith
9	Infeld	Ray

Fig. 1-3



Linked List

	Salesperson	Pointer
1	Jones	3, 6
2	Ray	2, 4, 7, 9
3	Smith	1, 5, 8

Fig. 1-5

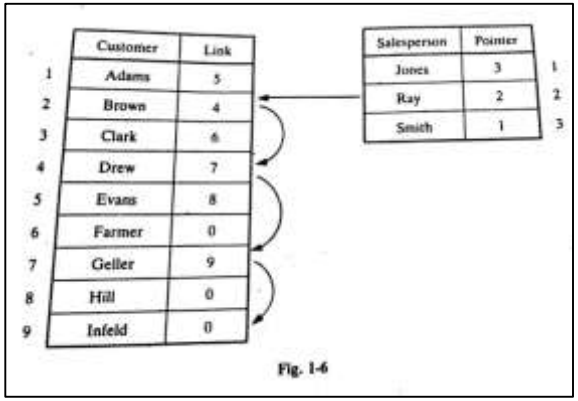


Fig. 1-6

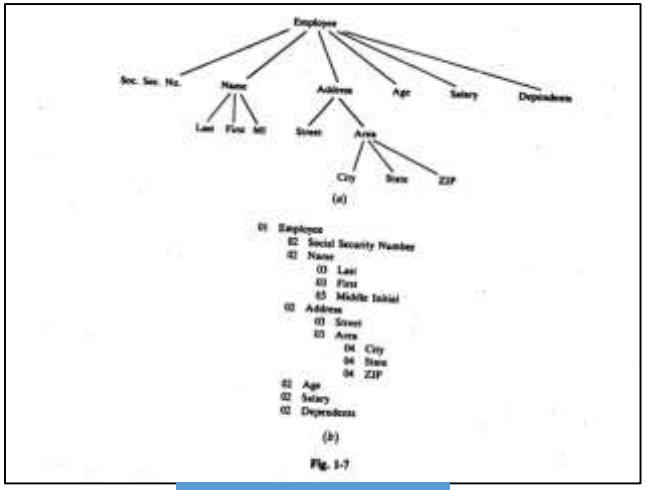


Fig. 1-7

Hierarchical

$(2x+y)(a-7b)^3$

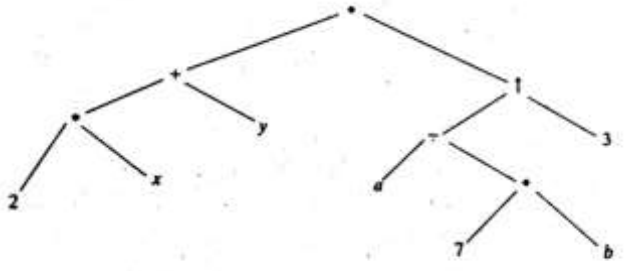


Fig. 1-8

Data Structure: Array

Simplest types of data structure

- One dimensional array /Linear

	STUDENT
1	John Brown
2	Sandra Gold
3	Tom Jones
4	June Kelly
5	Mary Reed
6	Alan Smith

Fig. 1-1

- Two dimensional Array

Store \ Dept.	1	2	3	4
1	2872	805	3211	1560
2	2196	1223	2525	1744
3	3257	1017	3686	1951
. . .	. . .	. . .	. . .	. . .
28	2618	931	2333	982

Fig. 1-2

Data Structure: Link List

	Customer	Salesperson
1	Adams	Smith
2	Brown	Ray
3	Clark	Jones
4	Drew	Ray
5	Evans	Smith
6	Farmer	Jones
7	Geller	Ray
8	Hill	Smith
9	Infeld	Ray

Fig. 1-3

	Customer	Pointer
1	Adams	3
2	Brown	2
3	Clark	1
4	Drew	2
5	Evans	3
6	Farmer	1
7	Geller	2
8	Hill	3
9	Infeld	2

	Salesperson
1	Jones
2	Ray
3	Smith

Fig. 1-4

	Salesperson	Pointer
1	Jones	3, 6
2	Ray	2, 4, 7, 9
3	Smith	1, 5, 8

Fig. 1-5

	Customer	Link
1	Adams	5
2	Brown	4
3	Clark	6
4	Drew	7
5	Evans	8
6	Farmer	0
7	Geller	9
8	Hill	0
9	Infeld	0

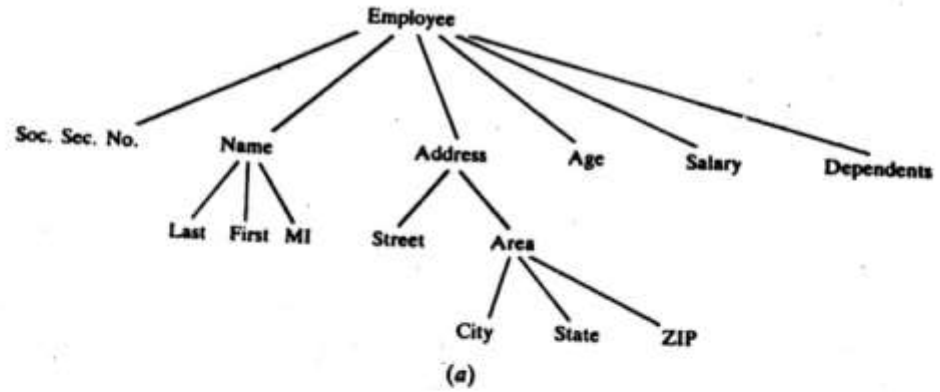
	Salesperson	Pointer
1	Jones	3
2	Ray	2
3	Smith	1

Fig. 1-6

Advantages: An integer used as a pointer requires less space than a name. Hence this representation saves spaces, if there are hundreds of customers for each salesman

Data Structure: Tree

Data frequently contain a hierarchical relationship between various elements. The data structure reflects this relationship is called a rooted tree graph or simply a tree



(a)

```
01 Employee
   02 Social Security Number
   02 Name
      03 Last
      03 First
      03 Middle Initial
   02 Address
      03 Street
      03 Area
         04 City
         04 State
         04 ZIP
   02 Age
   02 Salary
   02 Dependents
```

(b)

Fig. 1-7

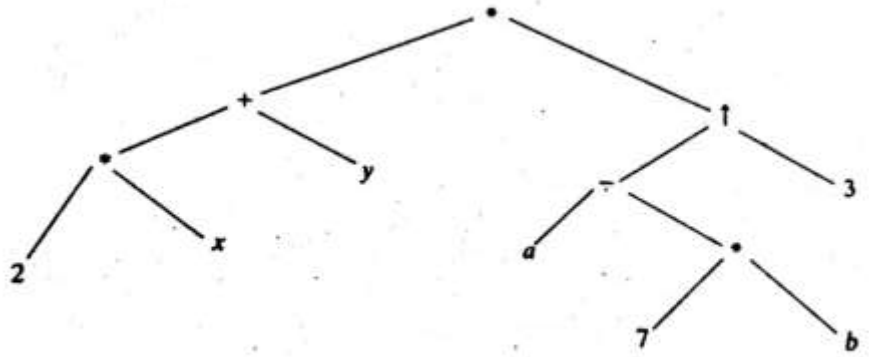
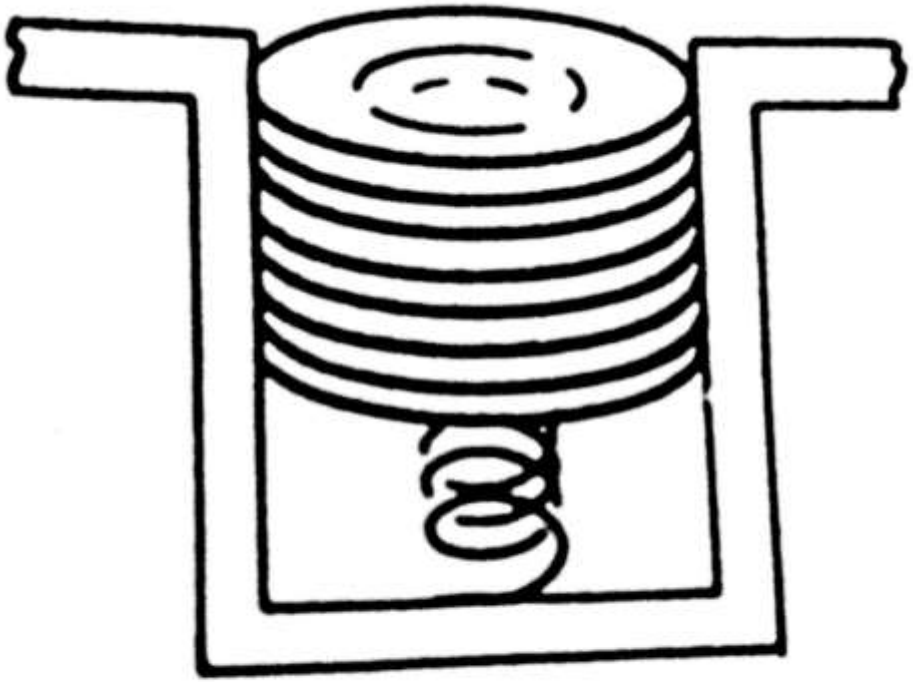


Fig. 1-8

Data Structure: Stack



(a) Stack of dishes.

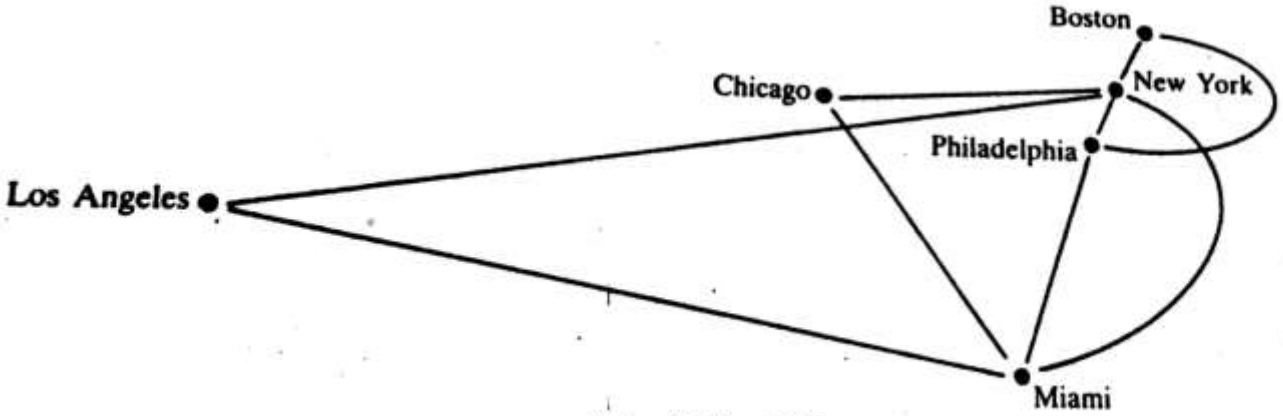
LIFO

Data Structure: Queue

(*b*) Queue waiting for a bus.

FIFO

Data Structure: Graph



(c) Airline flights.

Data Structure Operations

The data appearing in our data structure are processed by mean of certain operations.

The most frequently used of these operation are:

1. Traversing
2. Searching
3. Inserting
4. Deleting
5. Update
6. Sorting
7. Merging

Data Structure Operations

Accessing each record once so that certain items in the record may be processed (Visit).

Example:

An organization contains a membership file in which each record contains data for a given member:

Name **Address** **Tel. Number** **Age** **Sex**

(a) Suppose the organization wants to announce through a mailing.

(b) Suppose one wants to find the name of all members in a certain area.

Operation: Traversing

Data Structure Operations

Finding the location of the record with a given key value, or finding the locations of all records which satisfy one or more condition.

Example:

An organization contains a membership file in which each record contains data for a given member:

Name Address Tel. Number Age Sex

(a) Suppose one wants to obtain address for a given name.

Operation: Searching

Data Structure Operations

Adding a new record to the structure

Example:
An organization contains a membership file in which each record contains data for a given member:

Name Address Tel. Number Age Sex

(a) Suppose a new person joins the organization.

Operation: Inserting

Data Structure Operations

Removing a record from the structure

Example:

An organization contains a membership file in which each record contains data for a given member:

Name Address Tel. Number Age Sex

(a) Suppose a Member dies.

Operation: Deleting

Data Structure Operations

Changing items in the record with the new data

Example:
An organization contains a membership file in which each record contains data for a given member:

Name Address Tel. Number Age Sex

(a) Suppose a member has moved and has a new address and telephone number.

Operation: Updating

Data Structure Operations

Arranging the record in some logical order (e.g. alphabetically according to some NAME key)

Example:

An organization contains a membership file in which each record contains data for a given member:

NAME **Address** **Tel. Number** **Age** **Sex**

(a) Suppose One wants to obtain all the members list according to alphabetical order of their family name.

Operation: Sorting

Data Structure Operations

Combining the records in two different sorted files into a single sorted file.

Example: Exam Answer Script

Operation: Merging

Application Domains of Data Structures

- Data structures are like the filing cabinets and organizers of the computer world.
- They provide efficient ways to store, organize, and access data.
- Their applications are vast and span across various domains:
  - **Software Development:**
  - **Artificial Intelligence and Machine Learning:**
  - **Networking and Routing:**
  - **Other Applications:**

Application Domains of Data Structures

Software Development:

- **Arrays:** The foundation for storing collections of similar data items, used in functions, lists, and tables.
- **Linked Lists:** Useful for dynamic data (frequently changing size) like managing browser history or implementing undo/redo functionalities.
- **Stacks:** LIFO order, perfect for function calls, backtracking algorithms, and implementing the "undo" functionality.

Application Domains of Data Structures

Software Development:

- **Queues**: FIFO order, ideal for managing printer jobs, task scheduling, and handling requests in servers.
- **Trees:** Hierarchical organization for efficient searching and sorting, used in file systems, contact management, and decision trees in AI.
- **Graphs:** Model relationships between data points, used in social networks, navigation apps (finding shortest paths), and recommendation systems

Application Domains of Data Structures

Artificial Intelligence and Machine Learning:

Trees: Decision trees for classification tasks, and kd-trees for efficient nearest neighbor search in high-dimensional data.

Graphs: Knowledge graphs to represent entities and relationships, used in natural language processing and recommender systems.

Networking and Routing:

Trees: Spanning trees to find optimal paths in networks, and prefix trees for efficient IP address routing.

Graphs: Modeling network topologies, routing algorithms, and network traffic analysis.

Application Domains of Data Structures

Other Applications:

Geographic Information Systems (GIS):

Spatial data structures like quadtrees and R-trees for efficient storage and retrieval of geographic data.

Financial and Trading Systems: Priority queues for order processing and maintaining order books in stock exchanges.

Application Domains of Data Structures

- Their applications are vast and span across various domains:
  - **Software Development:**
  - **Artificial Intelligence and Machine Learning:**
  - **Networking and Routing:**
  - **Other Applications:**

Real life Applications of Data Structures

- Data structures are the fundamental building blocks for organizing and storing data efficiently in computer programs.
- They play a crucial role in various aspects of our daily lives, often working behind the scenes to ensure things run smoothly.
- Here are some real-world applications of common data structures:
- **Arrays:** Imagine the leaderboard in a video game. An array can efficiently store the scores of all players, allowing for easy ranking and retrieval. Similarly, phone contact lists and online shopping carts can be implemented using arrays.

Real life Applications of Data Structures

- **Queues:** Queues follow a first-in-first-out (FIFO) principle, mimicking real-life waiting lines. They are used in managing tasks like printing jobs, handling website traffic, and processing music playlists in media players.
- **Stacks:** Stacks function in a last-in-first-out (LIFO) manner. A classic example is the "undo" function in many software programs, which utilizes a stack to track the last performed actions and facilitate easy reversal.

Real life Applications of Data Structures

- **Trees:** Think of your family tree - a hierarchical structure that represents relationships. Similarly, computer trees organize data in a parent-child fashion. They are used for efficient searching and indexing in databases, like the directory structure on your computer or the way DNS (Domain Name System) translates website addresses.
- **Graphs:** Social media platforms like Facebook can be visualized as graphs, where users are nodes and connections between them are edges. Graphs are also used in route planning applications (like GPS) to model road networks and find optimal paths.

Algorithms ?

An algorithm is a well-defined list of step for solving problem.

The efficiency of an algorithm is obtained by measuring the TIME and SPACE it uses.

Algorithms Notation

(Largest Element in Array) A nonempty array **DATA** with **N** numerical values is given. This algorithm finds the location **LOC** and the value **MAX** of the largest element of **DATA**. The variable **K** is used as counter.

- **Step 1. [Initialize] Set K:=1, LOC:=1 and MAX := DATA[1].**
- **Step 2. [Increment counter.] Set K:=K+1.**
- **Step 3. [Test counter.] If K>N, then:**

  Write: LOC, MAX, and Exit.
- **Step 4. [Compare and update.] If MAX<DATA[K], then:**

  Set LOC:=K and MAX := DATA[K].
- **Step 5. [Repeat loop.] Go to Step 2.**

Algorithms Notation

(Largest Element in Array) A nonempty array **DATA** with **N** numerical values is given. This algorithm finds the location **LOC** and the value **MAX** of the largest element of **DATA**. The variable **K** is used as counter.

- Step 1. [Initialize] Set K:=1, LOC:=1 and MAX := DATA[1].
- Step 2. [Increment counter.] Set K:=K+1.
- Step 3. [Test counter.] If K>N, then:
  Write: LOC, MAX, and Exit.
- Step 4. [Compare and update.] If MAX<DATA[K], then:
  Set LOC:=K and MAX := DATA[K].
- Step5. [Repeat loop.] Go to Step 2.

✓ The Steps of the algorithm are executed one after the other, beginning with **Step 1**

✓ Control may be transferred to **Step n** of the algorithm by the statement "***Go to Step n***"

✓ If several statements appear in the same step, e.g.,

Set K:=1, LOC:=1 and MAX := DATA[1].

then they are executed from **LEFT TO RIGHT**

✓ The algorithm is completed when the statement

Exit. Is encountered.

Algorithms Notation

(Largest Element in Array) A nonempty array **DATA** with **N** numerical values is given. This algorithm finds the location **LOC** and the value **MAX** of the largest element of **DATA**. The variable **K** is used as counter.

- Step 1. [Initialize] Set K:=1, LOC:=1 and MAX := DATA[1].
- Step 2. [Increment counter.] Set K:=K+1.
- Step 3. [Test counter.] If K>N, then:
  Write: LOC, MAX, and Exit.
- Step 4. [Compare and update.] If MAX<DATA[K], then:
  Set LOC:=K and MAX := DATA[K].
- Step5. [Repeat loop.] Go to Step 2.

✓ The [comment] will usually appear at the beginning or the end of the step.

✓ Variable names will use capital letters as in MAX and DATA.

➢ Single-letter names of variables used as counters or subscripts will also be capitalized in the algorithms (K and N, for example).

Algorithms Notation

(Largest Element in Array) A nonempty array **DATA** with **N** numerical values is given. This algorithm finds the location **LOC** and the value **MAX** of the largest element of **DATA**. The variable **K** is used as counter.

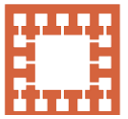
- Step 1. [Initialize] Set K:=1, LOC:=1 and MAX := DATA[1].
- Step 2. [Increment counter.] Set K:=K+1.
- Step 3. [Test counter.] If K>N, then:
  Write: LOC, MAX, and Exit.
- Step 4. [Compare and update.] If MAX<DATA[K], then:
  Set LOC:=K and MAX := DATA[K].
- Step5. [Repeat loop.] Go to Step 2.

✓ Assignment statements will use the dots-equal notation (:=).
For example, MAX := DATA[1]. (Some time ← or = is used for this operation

✓ Data may be input and assigned to variables by means of a Read statement
For example, Read: Variable names

✓ Similarly, data in variable may be output by mean of a **Write** or **Print** statement
For example, Write: Message and / or variable names.

Complexity of Algorithms



CSE 2103

The complexity of an algorithm is the function which gives the running time and/or space in terms of the input size.

In order to compare algorithms, we must have some criteria to measure the efficiency of a algorithm.

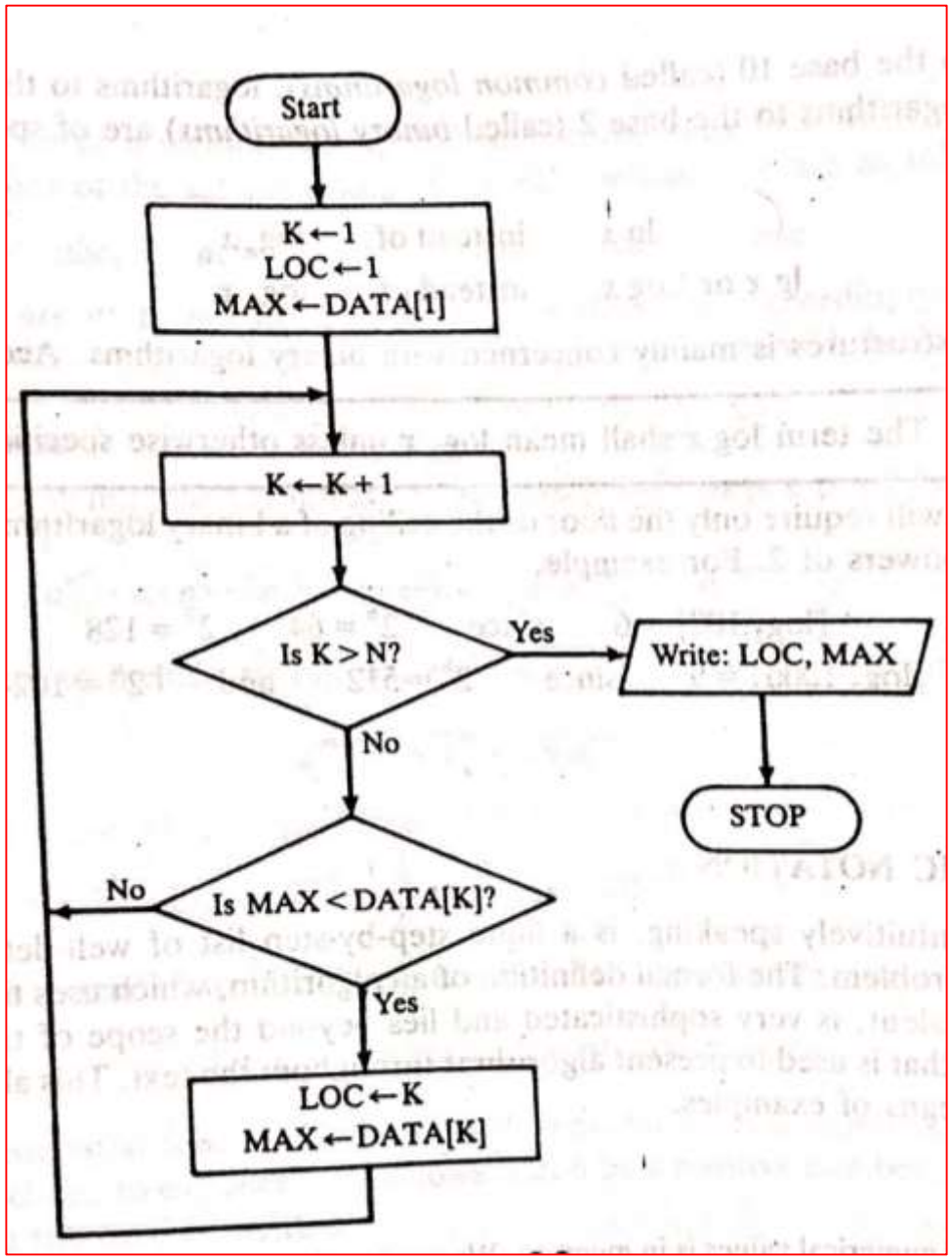
Suppose ***M*** is an algorithm, and suppose ***n*** is the size of the input data.

The ***TIME*** and ***SPACE*** used by the algorithm M are the two main measures for the efficiency of *M.*

The **TIME** is measured by counting the number of key operation-in sorting and searching algorithms. (the # of comparison)

The **SPACE** is measured by counting the maximum of memory needed by the algorithm

Flowcharts


Start
K←1
LOC←1
MAX←DATA[1]
K←K+1
Is K>N?
Yes
Write: LOC, MAX
No
STOP
No
Is MAX<DATA[K]?
Yes
LOC←K
MAX←DATA[K]

Understanding Time Complexity

“IN Computer Science, time complexity describes the amount of time that takes to run an algorithm or Code.”

How to calculate Time Complexity ?

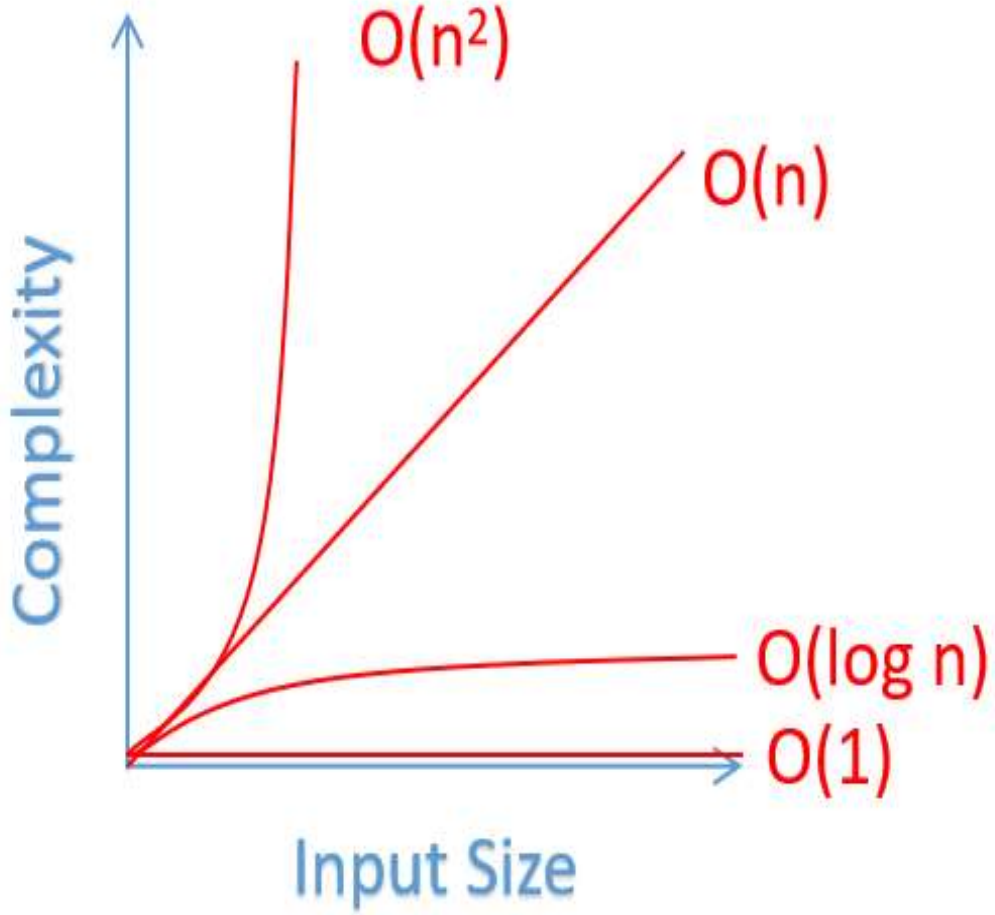
"Time complexity is calculated by counting the Number of Operation , which takes a fixed amount of time to perform."

Big O Notation

"Big O notation is a mathematical notation that describes the limiting behavior of a function when the argument tends towards a particular value or infinity."

Common Time Complexity

- O(1) Constant Time complexity
- O(N) Linear Time Complexity
- O(N^2) Quadratic Time Complexity
- O(N log N) Logarithmic Time Complexity



Constant Time Complexity-O(1)

- **Input (in single operation)**
- **Output (in single operation)**

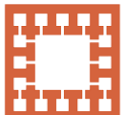
```
#include <iostream>
using namespace std;


int main() {

int a,b;
cin >> a >> b;
 //O(1+1)=O(1) constant operation.
int sum = a+b;
 cout<< sum <<endl;
 return 0;
}
```

Linear Time Complexity - O(N)

- While doing operation with ***Single*** for or while loop.

- Total Complexity(O(N))
- In 1 Sec 1e7 Or maximum 1e8 Iteration can be done.

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
    int t;
    cin >> t;
           int n;
       cin >> n;
       int arr[n];
       for(int i=0;i<n;i++){
        cin >> arr[i];
       }            // O(N) *here N is Number of
element
       sort(arr,arr+n);
       for(int i=0;i<n;i++){
        cout << i << " ";
       }     // O(N) *here N is Number of element
       cout << endl;
}
```




```
#include<bits/stdc++.h>
using namespace std;
int main()
{
   int n;
   cin >> n;

   int arr[n];
   for(int i=0;i<n;i++) cin >> arr[i];
   //Bubble sort
   for( int i = 0; i < n; i++ )
  {
      for(int j = 0; j < (n-i-1); j++) {
         if(arr[j] > arr[j+1]){
            swap( arr[j] , arr[j+1] );
         }
      }
   }
   for(auto u : arr){
      cout << u << " ";
   }
   cout << endl;

}
```

Logarithmic Time Complexity – O(log N)

Per Operation mid divides into half Value. That means

n/2 + n/4 + n/8 + n/16 + ……..

= n (1/2 + 1/4 + 1/8 + ……..)

= log(n)) // $log_2(y) = x,\ 2\hat{}x = y;$

So, Time Complexity of Binary search is O(log(n))

```
int main()
{  // Binary Search
   int n , v[n]; cin >> n;
   for(int i=0; i<n; i++){
   cin >> v[i];
   }
   int lo=0,hi=n-1,mid,fnd;
   cin >> fnd;
   while(hi>=lo){
      mid=(hi+lo)/2;
      if(v[mid] == fnd) {
         cout << mid << endl;
         break;
      }
      if(v[mid] <= fnd){
      lo=mid+1;
      }
      else
      {
         hi = mid-1;
      }
   }

}
```