UNIT I

Fundamentals

* Computer - Electronic Machine (Hardware & Software) → does the human work & minimizes the workload of humans.
  1. I/P - Users (I/P device - keyboard, Mouse)
  2. Processing - CPU (Control Processing Unit) (Brain of computer).
  3. O/P - Monitor, printer.

Programming Language, Machine language.

Computer Hardware.

→ physical components.

→ Fuction efficiently & produce useful output only when H/W & S/W work together.

→ Internal Computer Hardware.

process / Store the instructions delivered by the program / OS.

1. Mother board
2. CPU
3. RAM
4. Hard drive
5. Solid State Drive
6. Optical Drive
7. GPU
8. NIC
9. Heat Sink

① Mother Board → Central hub.
holds the CPU & functions are carried out.

② CPU – brain of Computer. Clock speed – measures the Computer performance & Efficiency.

③ RAM – Temporary Memory. Volatile Memory (Stored data is cleared when Computer is Switched off).
Information is immediately accessible to programs.

④ Hard drive – Store Temporary & permanent data.

⑤ SSD – Non-Volatile, Safe to store. Data remains permanent even when the Computer is Switched off.

⑥ GPU – Extension to CPU. Graphical Data.

⑦ NIC – Connect to Network / LAN / N/W Adapter.

External Hardware Components.

* peripheral Components, Controls either I/P or O/P functions

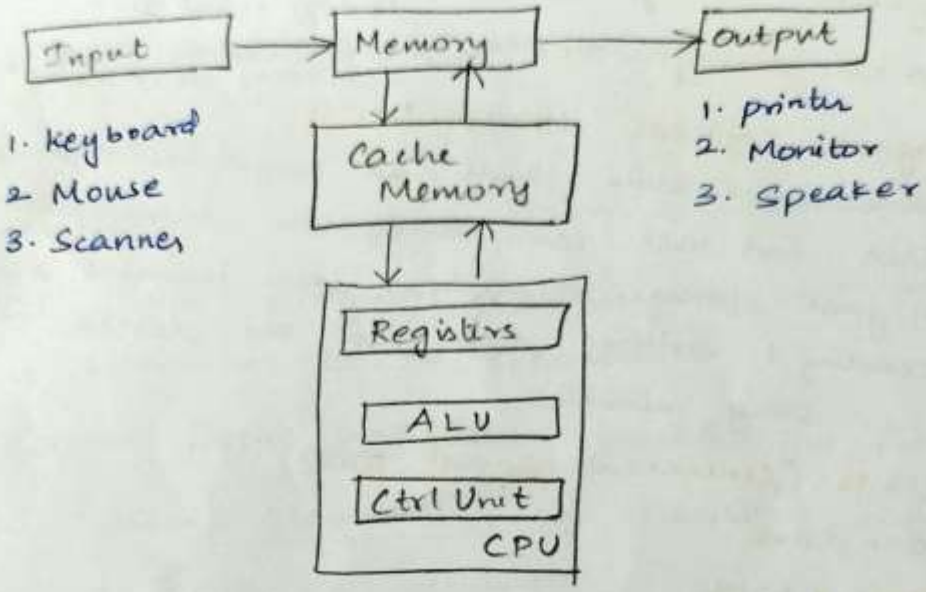
1. Mouse.
2. Keyboard
3. Microphone
4. Camera
5. Memory Card.

Hardware – Tangible Components – run the instruction provided by the Software.

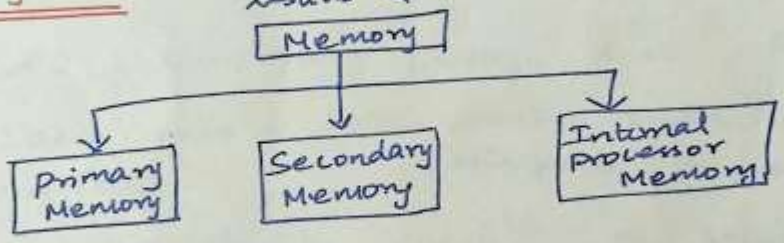
Software – Intangible Components – User – H/W interact & Command – Specific task.

① Application Software
② System Software.

Basic Computer Organization.



(2) Memory Unit → Store data & instruction & intermediate processing results & Final processed information.



Primary Memory.

* Built-in Unit of the Computer
* Data is stored in Machine Understandable binary form

① ROM (Read-Only Memory) - (Non-Volatile)

- permanent memory Contents Cannot be changed by end user.
- BIOS information - which performs POST (Power-on Self Test).

② RAM - Volatile

- Temporary memory. power off Contents get erased.
- Data is of no usage, Contents are erased.

③ Cache Memory

- Store data & related application that was last processed by CPU
- CPU & main memory - intermediate cache is placed

Secondary Memory.

* External Storage device connected to computer.
* Connected externally. Non-Volatile.

① Magnetic storage Device → read, erased & rewritten.
Floppy, Hard disk

② Optical storage Device → laser beam.
CD-ROM, CD-RW, DVD

③ Magneto-optical Storage Device.
Store information such as programs, Files & backup data. End user can modify the information.
Higher storage capacity → laser beams & magnets for reading & writing data to the device.
Ex: Sony minidisc

④ USB (Universal Serial Bus)
pen drive.
Compactable.
Storage Capacity is large.

③ CPU

* Brain of Computer
* processing in s/y.
* Controls the Components of s/y.

Main operations

1. Fetching instructions from memory
2. Decode the inst to decide what operation to be done
3. Execute the instruction
4. Store the result in memory.

Main Components:

1. ALU
2. CU
3. Registers.

1. ALU (Arithmetic Logic Unit)
   - ↳ Arithmetic operations (add, sub, mul, div)
   - ↳ Logical operations (AND, OR, NOT)

2. Control Unit (CU)
   - Controls the flow of data & information
   - CU uses program counter reg for fetching the next instruction to be execution.
   - fetches instn and gives to ALU for processing.
   - CU uses Status register handling conditions such as overflow of data.

3. Registers.
   - CPU - temporary storage unit - Registers.
   - Data, instn & intermediate results are stored in registers.

   (i) program counter (PC)
   (ii) Instruction Register (IR) - instn to be decoded by CU
   (iii) Accumulator - results produced by CPU
   (iv) Mem. Add. Register (MAR) - Address of next. location in Mem. to be accessed
   (v) Mem. Buffer Register (MBR) - storing data sent/ Received to CPU.
   (vi) Mem. Data Register (MDR) - Data & operands.

Algorithm

* Set of instruction when executed in a sequence will get a solution.

Representation of Algorithm

1. Normal English
2. Flowchart
3. Pseudocode

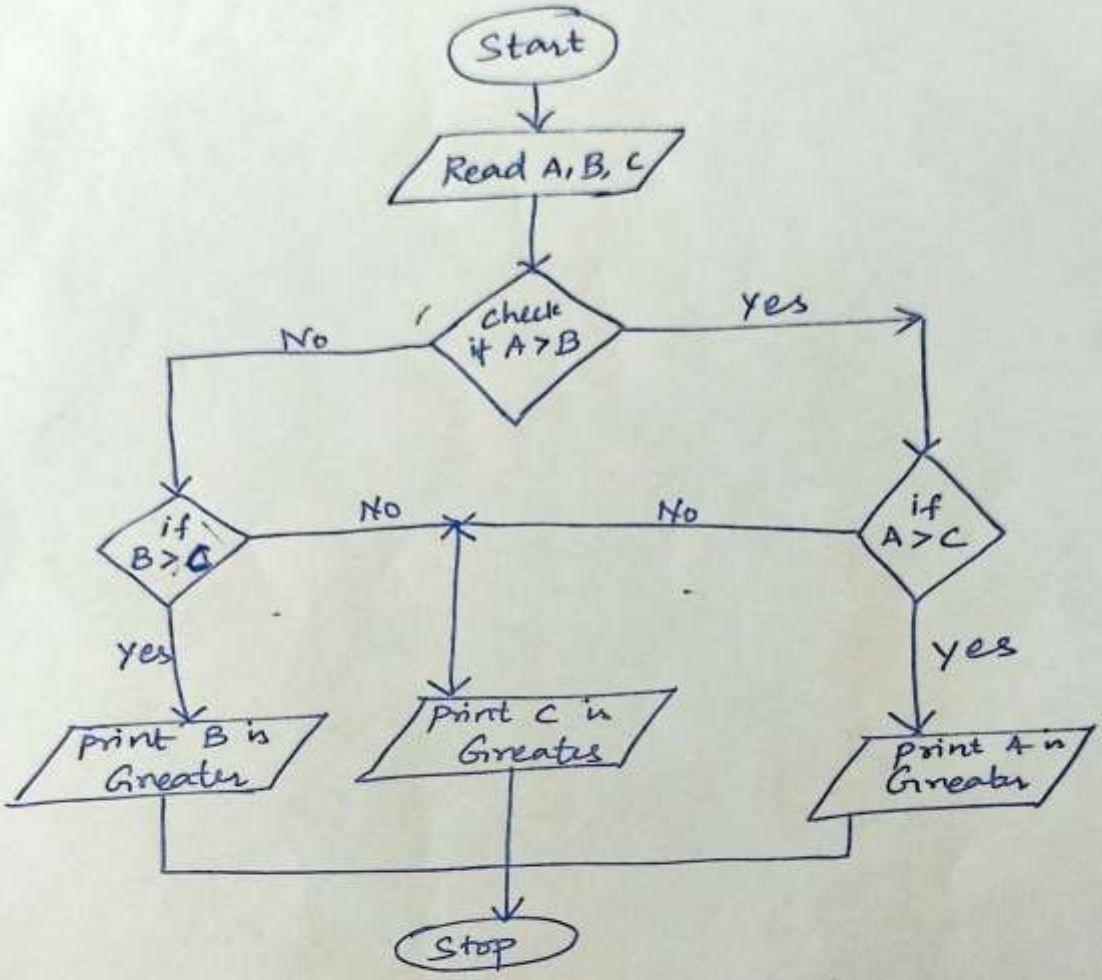
Flowchart.

① Flow lines
② Terminal Symbols
③ Input/ output Symbol
④ Process Symbol
⑤ Decision Symbol
⑥ Connectors.

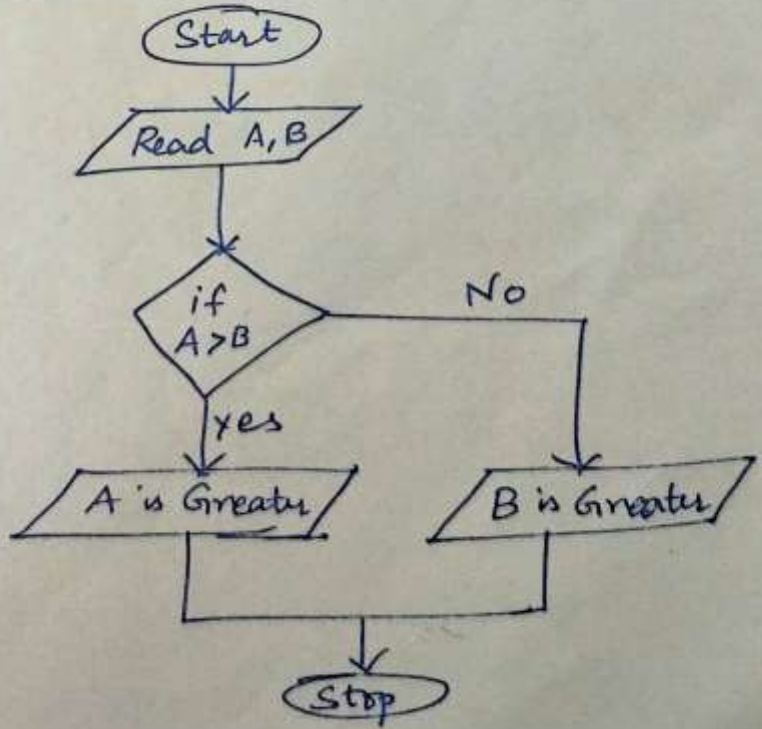
Example:

1. Addition of two numbers
2. Greatest of two numbers, even/odd, Neg/positive.
3. Calculate the SI, Area of circle
4. Calculate Total & Average of Students in a class
5. Sum of 5 numbers.

Greatest of Three Numbers.



Greatest of two numbers.



* Algorithm. After plan for developing pgm by logic - correct seq of Procedure of instn required to carryout the task. (1)

* Algorithm - logic of the program. It is the basic tool used to develop problem solving.

* " a Unambiguous - (clear) sequence of instructions designed in such a way that if the instn are executed in specified sequence, the desired results will be obtained".

Characteristics:

* Algorithm - precise & Unambiguous
  - instn should not be repeated again.
  - instn should be in sequence.
  - Result produced after algorithm terminates.

Representation of Algorithms.

* Normal English
* Flowchart
* Pseudocode
* Decision table
* program.

Flowchart

* pictorial representation of an algorithm. Sequential steps are represented in flowchart using standard symbols.
* layout & visual representations of the plan.
* Symbol - oriented design - identifies the type of stmt from the symbols used.
* Arrows ← Connecting b/w 2 symbols.
* Process of drawing flowchart for an algorithm is called Flowcharting.
* Flowcharts : Easy to Understand.
  - helps in reviewing & debugging of a pgm.
  - easy to analyze & compare various methods.

Symbols used:

① Flowlines → L ← R ↑ Top ↓ Bottom — Exact sequence in which instructions are to be executed. Drawn with arrow head.

② Terminal Symbol — Start / Stop [oval shape] → begin + end
→ start (no entering ↓)
→ stop (no ↓ exit flow line.)

③ Input/output Symbol. [parallelogram]
- Input (Read) & output (write)
- denotes fn of I/O devices in the pgm.

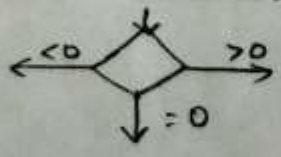
④ Process Symbol [Rectangle]
- used for calculations & initializations of memory locations.
- Arithmetic operations, data movements.

⑤ Decision Symbol — Y / N [Diamond shaped symbol]
- Indicate at a point decision is to be made b/w two alternatives.
- Yes (True), No (False).

⑥ Connectors — (n) (n) [circle & a digit/letter placed in link to specify the link].

Rules: Example: Greatest of 2 nos, odd/even, +ve/-ve number.

* Standard symbols should be used.
* only one should enter decision symbol, 2/3 flow lines can leave decision according to the possible answer.



* Annotation Symbol - describe data/ computational steps more clearly ---- [This is top secret data

- Flow lines should not cross eachother.

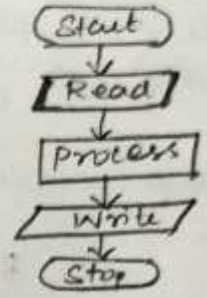
words in Flowchart - common stmts & easy to understand.

Design structures in Flowcharts.

Design flowchart for any program & later we can combine to get the solution for complex problems, 3 designs:

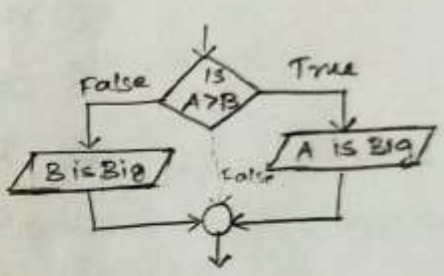
① Sequence structure.

- simplest, series of stmts
- sequence (same direction)
- can include any number of instn.



② Selection structure.

- Decision (make questions).
- operations done on decision made.
- 2 exists (two branches
  ↓
  rejoined to single flow to exit the structure)



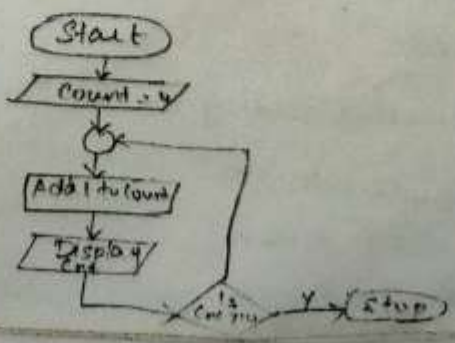
- Selection (True → one-sided selection)
  + null-sided should end with Exit.

③ Loop structure.

Execute sequence of steps in no. of times until particular Condition is met

(i) Top tested loop → if Count < 10 → T → goes into loop
F → goes to stmt after loop.

(ii) Bottom tested loop → Trailing decision loop
decision - last stmt of loop.



Pseudocode.

* Flowcharting symbols were established bf structured program*
* So flowchart will not be able to handle some concepts.
* pseudocode - formal design tool with structured design
  - Visual, narrative, used for planning prog. logic

Pseudocode (Also called PDL) {
- pseudo (initiation / false).
- code (set of stmts / instn in pgm. lang).

Program Design language
- written in English, very clear.
- Rules:
  1. Write one stmt per line.
  2. Capitalize initial keywords.
  3. Indent to show hierarchy
  4. End multiline structure.
  5. keep stmt lang independent.

Example. To calculate student total and average.

```
READ    name, class, M1, M2, M3
        Total = M1+M2+M3
        Average = Total/3
IF      average is greater than 75
        Rank = Distinction
ENDIF
WRITE   name, Total, Average, Rank
```

Advantages.
- Word processor
- Easily Modified
- read & understood easily
- Converting pseudocode to pgm. lang is easy when compared with flowchart.

Disadvantages.
- Not Visual (pictorial represent)
- No standardised style/ Format

Computer Software

* Set of instructions grouped into programs that make the computer to function in the desired way.
* Collection of pgm to perform a task.
* instructs H/w what to do.
* pgm - Set of instn written in High level lang Understandable by Computer stored in memory

CPU fetches an instn, decodes it & the executes required operation.

After execution - pgm counter - next instn fetched & executed

Types of Software.

* Application Software
* System Software.

① System Software.

* Set of programs that goverens the operation of computer system and make H/w to work.
* Controls the internal operations of computer Sly.
* Types
  - OS - Controls all components (CPU, Mem, I/O devices) of computer. Ex: DOS, WINDOWS, UNIX, LINUX.
  - Language processors. Conversion of high level language to Machine language.
    - Assembler
    - Interpreter
    - Compiler
  - Utility pgms - Virus Scanner, Disk defragementer.

② Application Software

* Set of pgms to carryout operations of specific Applin.
* It is written by programmers to enable computer to perform specific task.
* Types
  - Customised Applin S/W - based on customer requirements.
  - General Applin S/W - General requirements for a specific task. Customers can use it simultaneously.
* Example: Result preparation, Railway reservation, billing

Notes:

int a[10] char - 1, int - 2

float b[20] float - 4

Scanf("%f", ...

* OS - control all Hlw parts of computer system
  - Manage resources & overall operations of computer
* Function of os - Memory Mgt
  - File Mgt
  - Security Mgt
  - Process Mgt
  - User Interface
* Compiler - Convert high-level lang to machine lang
  - Source code to binary code
* Interpreter - Line-by-line execute the source code.
* Assembler - Convert Assembly lang to machine lang

Examples:

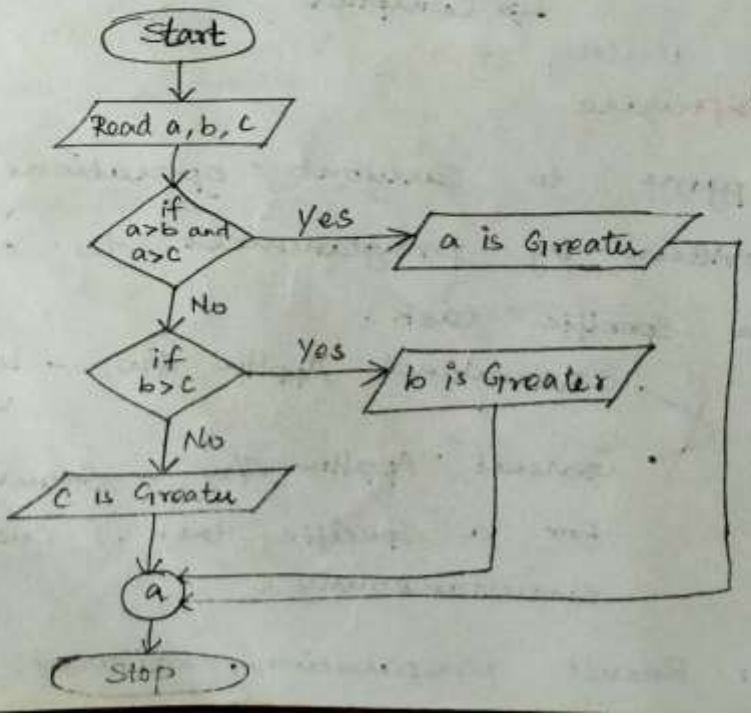
① Area of Circle.

② Convert Celcius to Fahrenheit. F = (1.8 * C) + 32.

③ Greatest of 3 numbers.

Internet Terminologies

1. Web page
2. Home page
3. Browser
4. URL
5. ISP
6. Download & Upload
7. online & offline
8. Hypertext
9. Web server
10. Web site.

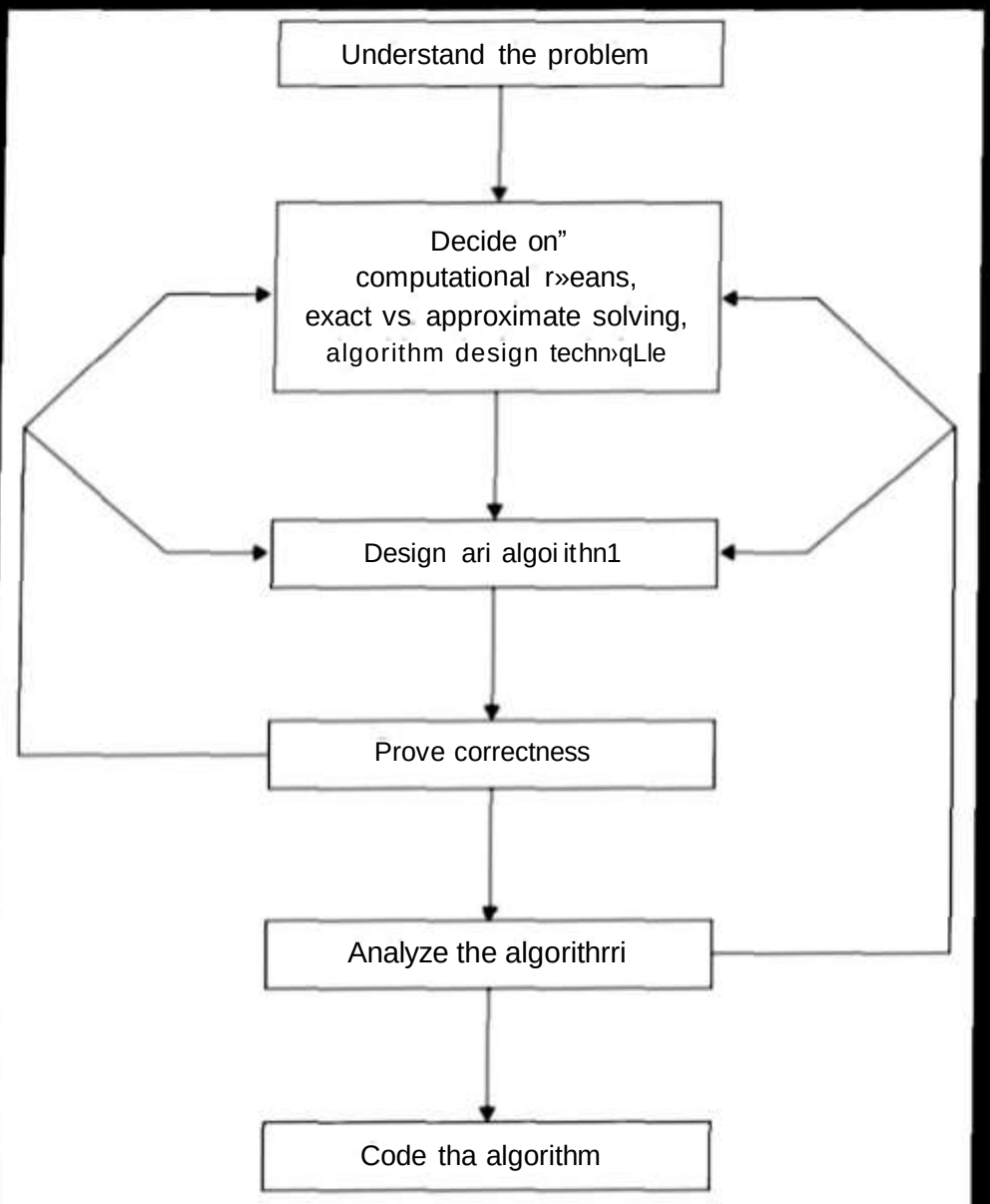


a > b
a > c

a	b	c
5	10	15
15	10	5
5	15	10

ALGORITHMIC PROBLEM SOLVING

Algorithmic problem solving is solving problem that require the formulation of an algorithm for the solution.



MGURE 1.2 Algorithm design and analysis process.

Understanding the Problem

- It is the process of finding the input of the problem that the algorithm solves.
- It is very important to specify exactly the set of inputs the algorithm needs to handle.

g the Cap

p

Approximate Problem Solving

- The next principal decision is to choose between solving the problem exactly or solving it approximately.
- Based on this, the algorithms are classified as exact *algorithm* and *approximation algorithm.* *Example: exact is addition of numbers, approximation is solving a linear equation*

Deciding a data structure:

- Data structure plays a vital role in designing and analysis the algorithms.
- One of the algorithm design techniques also depend on the structuring data specifying a problem's instance
- Algorithm+ Data structure=programs.

Algorithm Design Techniques

❖ An ***algorithm design technique*** (or "strategy" or "paradigm") is a general approach to solving problems algorithmically that is applicable to a variety of problems from different areas of computing.

Methods of Specifying an Algorithm

❖ ***Pseudocode, flowchart, programming language***

Proving an Algorithm's Correctness

❖ Once an algorithm has beer specified, you have to prove its ***correctness***. That is, you have to prove that the algorithm yields a required result for every legitimate input in a finite amount of time.

❖ A common technique for proving correctness is to use mathematical induction because an algorithm's iterations provide a natural sequence of steps needed for such proofs.

Analysing an Algorithm

Efficiency.

Time efficiency, indicating how fast the algorithm runs,

Space efficiency, indicating how much extra memory it uses.