CPNM Lecture 3 - Algorithms and Flow Charts

Mridul Sankar Barik

Jadavpur University

November 13, 2022

Algorithms I

- An algorithm is a sequence of computational steps that transform the input into the output
- An algorithm acts as a tool for solving a well-specified computational problem
  - The statement of the problem specifies in general terms the desired input/output relationship
  - The algorithm describes a specific computational procedure for achieving that input/output relationship
- Example:
  - The problem of sorting a sequence of numbers into increasing order is formally defined as
    Input: A sequence of n numbers $< a_1, a_2, \ldots, a_n >$
    Output: A permutation (reordering) $< a'_1, a'_2, \ldots, a'_n >$ of the input sequence such that $a'_1 \leq a'_2 \leq \ldots \leq a'_n$
  - There are different algorithms for solving the sorting problem (Ex.- Bubble sort, Insertion sort, Merge Sort, Queick sort etc.)
  - Each of the sorting algorithm can have different implementations (possibly in different programming languages)

Properties of Algorithm

- **Finiteness**: An algorithm must always terminate after a finite number of steps
- **Definiteness**: Each step of an algorithm must be precisely and unambiguously defined.
  - In case of conditionals, explicit handling of all outcomes
  - In case of loop, explicitness about when to stop
- **Input**: An algorithm has zero or more inputs, i.e, quantities which are given to it initially before the algorithm begins.
- **Output**: An algorithm has one or more outputs i.e, quantities which have a specified relation to the inputs.
- **Effectiveness**: All operations to be performed must be sufficiently basic that they can be done exactly and in a finite amount of time.
  - Example of Non Effectiveness: Find exact value of e using the formula

    $$e = 1 + \frac{1}{1!} + \frac{1}{2!} + \frac{3}{3!} + \ldots$$

    Summation of infinite terms requires infinite time

Complexity of Algorithm

- **Time Complexity**: Running time of the program as a function of input size
- **Space Complexity**: Amount of computer memory required during the program execution, as a function of input size

Example of Algorithm

- Example 1: Calculate factorial of a number

```
Algorithm:

Input: N
Output: N!

Step 1: Fact = 1
Step 2: if N > 1
Step 3: Fact = Fact * N
Step 4: N = N - 1
Step 5: goto Step 2
Step 6: Print Fact
```

Expressing Algorithms

- **Natural Languages**: usually verbose and ambiguous
- **Flow Charts**:
  - avoid most (if not all) issues of ambiguity
  - difficult to modify w/o specialized tools
  - largely standardized
- **Pseudo Code**:
  - avoids most issues of ambiguity
  - closely resembles common elements of programming languages
  - does not conform to particular agreement on syntax
- **Programming Language**: tend to require expressing low-level details that are not necessary for a high-level understanding

Flow Charts I

- A flow chart, or flow diagram, is a graphical representation of a process or system that details the sequencing of steps required to generate the desired output

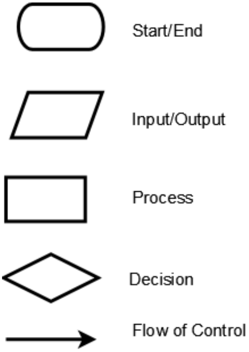
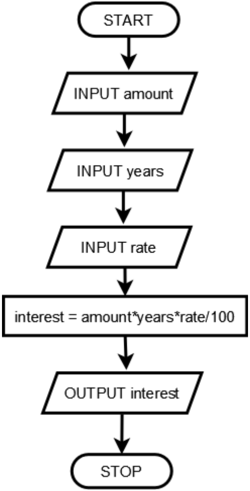


Figure 1: Flow Chart Symbols

Flow Charts II

- Example: Calculate simple interest



Flow Charts III

- Example: Find whether a given number is odd or even