Course Name : Data Structures & Algorithms

BITS Pilani
Pilani Campus

Bharat Deshpande
Computer Science & Information Systems

What is a program?

- **Algorithm**

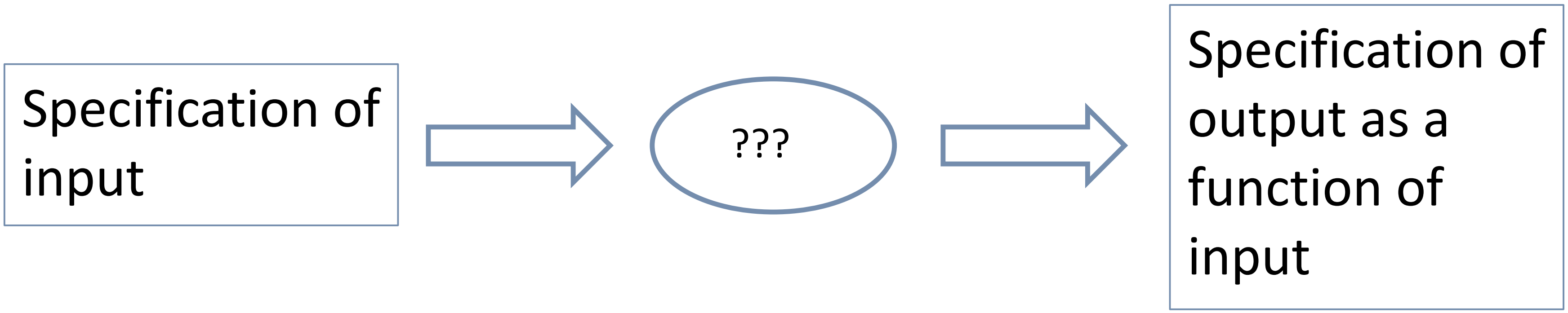
  An algorithm is a step-by-step procedure for solving a problem in a finite amount of time.

- **Data Structures**

  Is a systematic way of organizing and accessing data, so that data can be used efficiently.

  Algorithms + Data Structures = Program

Algorithmic problem

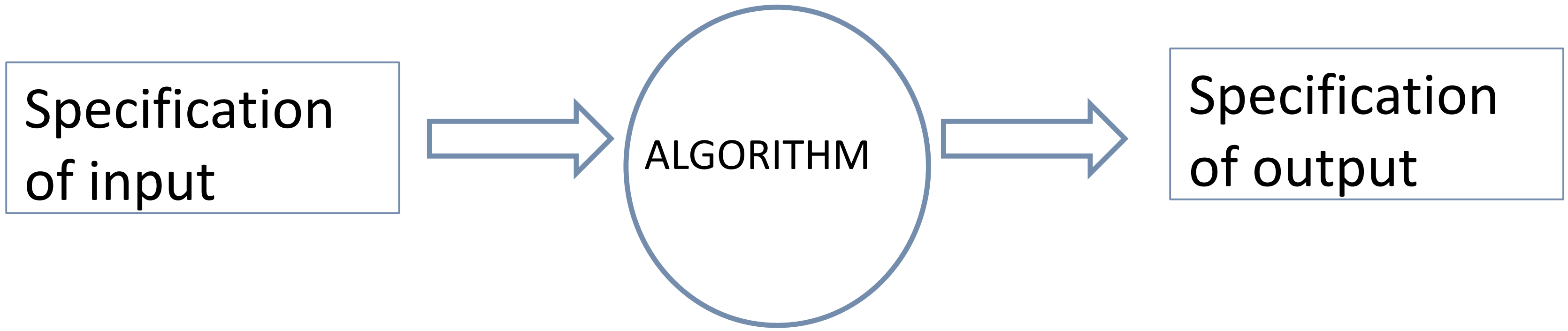


For eg: Sorting of integers

Input Instance : 8,4 ,5,2,10

Output Instance as a permutation of input : 2,4,5,8,10

Algorithmic Solution



- Algorithm describes actions on the input instance.
- Infinitely many correct algorithm for the same problem.

Infinite number of input instances satisfying the specification.

Two key points: Repeatable argument & Correctness

What is good algorithm?

- Resources Used
  - Running time
  - Space used

- Resource Usage
  - Measured proportional to (input) size

Measuring the running time

- Write a program implementing the algorithm.
- Run the program with inputs of varying size and composition.
- Use a method like System.currentTimeMillis() to get an accurate measure of the actual running time.

Limitations of experimental studies

- Implementation is a must.
- Execution is possible on limited set of inputs.
- If we need to compare two algorithms we need to use the same environment

  (like hardware, software etc)

Analytical model to analyze algorithm

- Algorithm should be analyzed by using general methodology.
- This approach uses:
  - High level description of the algorithm.
  - Takes into account all possible inputs.
  - Allows one to evaluate the efficiency of any algorithm in a way that is independent of the hardware and the software environment.

Pseudo-code

- A mixture of natural language and high level programming concepts that describes the main ideas behind a generic implementation of a data structure and algorithms.

Algorithm ***arrayMax(A, n)***
Input: **An array *A of n integers***
Output: **The maximum element of *A***

currentMax* ← *A[0]
for *i* ← *1 to n - 1 do*
  if *A[i] > currentMax then currentMax* ← *A[i]*
return *currentMax*

Pseudo-code

- Is structured than usual prose but less formal than a programming language.
- Expressions
  - Use standard mathematical symbols to describe numeric and Boolean expressions.
  - Uses $\leftarrow$ for assignment.
  - Use = for the equality relationship.
- Method declaration
  - Algorithm name(param1,param2…)

Assumptions

Individual statement considered as “unit” time

- Not applicable for function calls and loops

Individual variable considered as “unit” storage

Often referred to as “algorithmic complexity”

Complexity Example [1]

Example 1 (Y and Z are input)

X = Y * Z;

X = Y * X + Z;

// 2 units of time and 1 unit of storage

// Constant Unit of time and Constant Unit of storage

Complexity Example [2]

Example 2 (a and N are input)

```
j = 0;
while (j < N) do
   a[j] = a[j] * a[j];
   b[j] = a[j] + j;
   j = j + 1;
endwhile;
// 3N + 1 units of time and N+1 units of storage
// time units prop. to N and storage prop. to N
```

Complexity Example [3]

Example 3 (a and N are input)

```
j = 0;
while (j < N) do
    k = 0;
    while (k < N) do
         a[k] = a[j] + a[k];
        k = k + 1;
    endwhile;
    b[j] = a[j] + j;
    j = j + 1;
endwhile;
```

//**???** units of time and **???** units of storage

// time prop. to N^2 and storage prop. to N

Example of sorting

Input Sequence of numbers
$a_1, a_2, a_3, \ldots, a_n$
8,4,6,2,1

→ Sort →

Output a Permutation of input of numbers
$b_1, b_2, b_3, \ldots, b_n$
1,2.4.6.8

Correctness(Requirement for the output)
For any input algorithm halts with the output:

- $b_1 < b_2 < b_3 < \ldots < b_n$
- $b_1, b_2, b_3, \ldots, b_n$ is a permutation of $a_1, a_2, a_3, \ldots, a_n$

Running time of algorithm depends on

- Number of elements n.
- How (partially)sorted they are.

Order Notation

- Purpose
  - Capture proportionality
  - Machine independent measurement
  - Asymptotic growth

    (i.e. large values of input size N)

Motivation for Order Notation

Examples

- $100 * \log_2 N < N$ for $N > 1000$
- $70 * N + 3000 < N^2$ for $N > 100$
- $10^5 * N^2 + 10^6 * N < 2^N$ for $N > 26$

Asymptotic Analysis

- Goal: To simplify analysis of running time of algorithm .eg $3n^2=n^2$.
- Capturing the essence: how the running time of the algorithm increases with the size of the input in the limit.

Asymptotic Notation

- The big O notation

  Definition

  Let f and g be functions from the set of integers to the set of real numbers. We say that $f(x)$ is in $O(g(x))$ if there are constants $\boldsymbol{C} >$ and $\boldsymbol{k}$ such that

  $|\boldsymbol{f(x)}| \leq \boldsymbol{C}\ |\boldsymbol{g(x)}|$, ***w***henever $\boldsymbol{x} >= \boldsymbol{k}$**.**

- This is read as $f(x)$ is ***big-oh*** of $g(x)$

Note: Pair of C and k is never unique.

Order Notation

Examples

$g(n) = 17*N + 5$

$\lim_{n \to \infty} g(n) / f(n) = c$

$\lim_{n \to \infty} (17*N + 5)/N = 17$. The asymptotic complexity is $O(N)$

$g(n) = 5*N^3 + 10*N^2 + 3$

$\lim_{n \to \infty} (5*N^3 + 10*N^2 + 3) / N^3 = 5$. The asymptotic complexity is $O(N^3)$

$g(n) = C1*N^k + C2*N^{k-1} + \ldots + Ck*N + C$

$\lim_{n \to \infty} (C1*N^k + C2*N^{k-1} + \ldots + Ck*N + C) / N^k = C1$.

The asymptotic complexity is $O(N^k)$

$2^N + 4*N^3 + 16$ is $O(2^N)$

$5*N*\log(N) + 3*N$ is $O(N*\log(N))$

1789 is $O(1)$

Linear Search

```
function search(X, A, N)
 j = 0;
while (j < N)
      if (A[j] == X)  return j;
     j++;
 endwhile;
return "Not-found";
```

Linear Search - Complexity

Time Complexity

“if” statement introduces possibilities

- Best-case: O(1)
- Worst case: O(N)
- Average case: ???

Binary Search Algorithm

Assume: Sorted Sequence of numbers

```
low = 1;  high = N;
while (low <= high) do
   mid = (low + high) /2;
   if (A[mid] = = x)  return x;
   else if (A[mid] < x) low = mid +1;
   else high = mid – 1;
endwhile;
  return Not-Found;
```

Binary Search - Complexity

- Best Case
  - O(1)
- Worst case:
  - Loop executes until low <= high
  - Size halved in each iteration
  - N, N/2, N/4, … 1
  - How many steps ?

Binary Search - Complexity

- Worst case:
  - K steps such that $2^K = N$

  i.e. $\log_2 N$ steps is $O(\log(N))$