DATA STRUCTURES USING ‘C’

Lecture-06

Data Structures

Introduction to DS
Definition

- An algorithm is a finite sequence of instructions each of which has a clear meaning and can be performed with a finite amount of effort in a finite length of time.

Structure of Algorithms

- **Input step**
- **Assignment step**
- **Decision step**
- **Repetitive step**
- **Output step**

Algorithm of addition:

987+		987+		987+
76	=>	76+	=>	76+
		Carry1		Carry1
------------------		-------------------		----------------
(Carry1) 3		(Carry1) 63		1063

Properties of Algorithm

- **Finiteness**
  - An algorithm must terminate after a finite number of steps.
- **Definiteness**
  - Steps must be precisely defined.
  - No ambiguity in steps.
- **Generality**
  - It must be generic enough to solve all problem of same class
- **Effectiveness**
  - The steps of operations must be basic
  - Not too much complex
- **Input-Output**
  - It must have initial and precise inputs.
  - Output may be generated both at intermediate and final steps.

Data Structure and Algorithms

- Data Structure
  - A data structure is a way of organizing data that considers not only the items stored, but also their relationship to each other.
- The design of an efficient algorithm for the solution of the problem needs the use of appropriate data structure.
- The program which satisfy all the properties of algorithm is not enough for efficient implementation of algorithm.

Cont…

- It is important to arrange the data in well structured manner to prepare efficient algorithm.
- Thus, for the design of efficient solution of a problem, it is essential that algorithm goes hand in hand with appropriate data structure.

Efficiency of Algorithm

- One problem can be solved in many ways then to choose the best one among them we required to measure the performance of algorithm.
- The performance of algorithm can be measured by two main parameter:
  - Time
  - Space

Efficiency of Algorithm (Cont…)

- **Empirical or posterior testing approach**
  - Implement the complete algorithms and execute them for various instances of the problems.
  - The time taken for execution of the programs is noted.
  - Algorithm taking less time is considered as the best among all.
  - Its disadvantage is that it depend on various factors like –
    - ✓ machine on which it is executed.

Efficiency of Algorithm (Cont…)

- ✓ Programming language with which it is implemented
- ✓ Skills of a programmer

➢ **Theoretical or apriori approach**

- Mathematically determine the resources such as time and space needed by algorithm in form of a function of a parameter related to the instance of the problem considered.
- This approach is entirely machine, language and program independent.
- It allows to study the efficiency of the algorithm on any input size instance.

Asymptotic Notation

- Apriori analysis uses asymptotic notations to express the time complexity of algorithms.
- Asymptotic notations are meaningful approximations of functions that represent the time and space complexity of a program.

Asymptotic Notation (Cont…)

- **Big O notation**
  - $f(n)=O(g(n))$ (read: f of n is big oh of g of n), if there exists a positive integer n_0 and a positive number c such that $|f(n)| \le c|g(n)|$, for all $n \ge n_0$.
  - It shows upper bound of a function.

f(n)	g(n)	
$16n^3+12n^2+12n$	n^3	$f(n) = O(n^3)$
$34n - 90$	n	$f(n) = O(n)$
56	1	$f(n) = O(1)$

Asymptotic Notation (Cont…)

- **Omega notation**
  - $f(n)=\Omega(g(n))$(read: f of n is omega of g of n), if there exists a positive integer n_0 and a positive integer c such that $|f(n)|\geq c|g(n)|$, for all $n\geq n_0$.
  - Here g(n) indicate the lower bound of the function f(n).

f(n)	g(n)	
$16n^3+12n^2+12n$	n^3	$f(n) = \Omega(n^3)$
$34n - 90$	n	$f(n) = \Omega(n)$
56	1	$f(n) = \Omega(1)$

Asymptotic Notation (Cont…)

- **Thita notation**
  - f(n)= ϴ(g(n))(read: f of n is thita of g of n), if there exists a positive integer n_0 and a positive integer c_1 and c_2 such that $c_1|g(n)| \leq |f(n)| \leq c_2|g(n)|$, for all $n \geq n_0$.
  - Here g(n) indicate the upper bound as well as lower bound of the function f(n).

f(n)	g(n)	
$16n^3+12n^2+12n$	n^3	f(n) = ϴ(n^3)
34n – 90	n	f(n) = ϴ(n)
56	1	f(n) = ϴ(1)

Asymptotic Notation (Cont…)

- **Little oh notation**
  - $f(n)=o(g(n))$ (read: f of n is little oh of g of n) if $f(n)=O(g(n))$ and $f(n) \neq \Omega(g(n))$.

f(n)	g(n)	
$18n^3 + 9$	n^3	$f(n) = o(n^3)$ because $f(n) = O(n^3)$ and $f(n) \neq \Omega(n^3)$

Average, Best and Worst Cases

- The time complexity of an algorithm is dependent on parameters associated with the input/output instances of the problem.
- Many times the input size is only used to calculate the complexity, in such cases if input size is larger then execution time will be larger.
- But all the time it is not appropriate to consider only the size of input for calculating complexity.

Cont…

- Sometimes, the complexity is also depends on the nature of input.
- For example, consider the following data for sequentially searching the first even number in the list.

Input data	Case
-1, 3, 5, 7, -5, 11, -13, 17, 71, 9, 3, 1, -23, 39, 7, 40	Worst
6, 11, 25, 5, -5, 6, 23, -2, 26, 71, 9, 3, 1, -23, 39, 7	Best
-1, 3, 11, 5, 7, -5, -13, 16, 11, 25, 5, -5, 6, 23, -2, 7	Average

- **Worst case:**
  - The input instance for which algorithm takes the maximum possible time is called the worst case.
  - The time complexity in such a case is called worst case time complexity.
- **Best case:**
  - The input instance for which algorithm takes the minimum possible time is called the best case.
  - The time complexity in such a case is called best case time complexity.
- **Average case:**
  - All input instances which are neither of a best case nor of a worst case are categorized as average case.
  - The time complexity of the algorithm in such cases is referred to as the average case complexity.