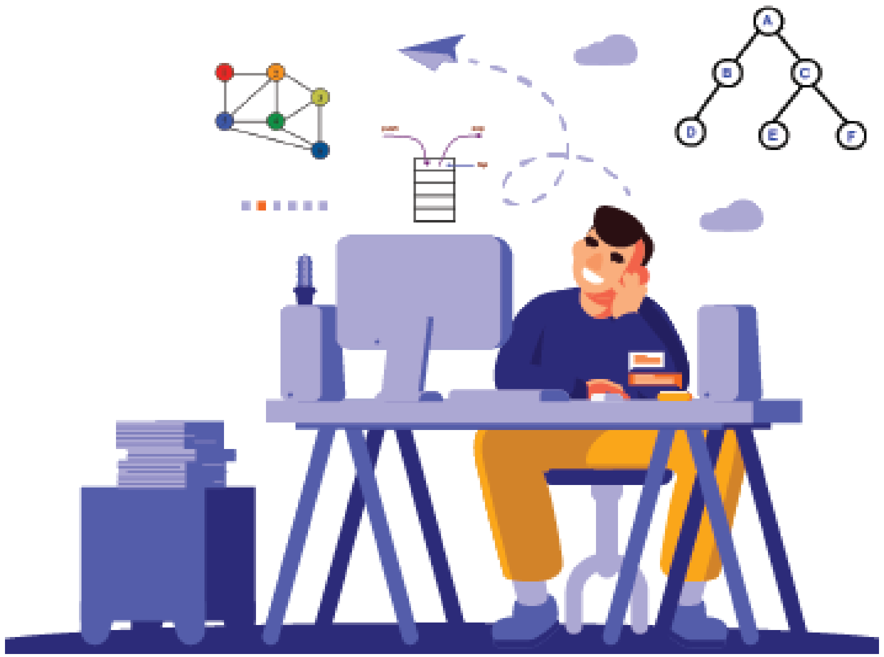
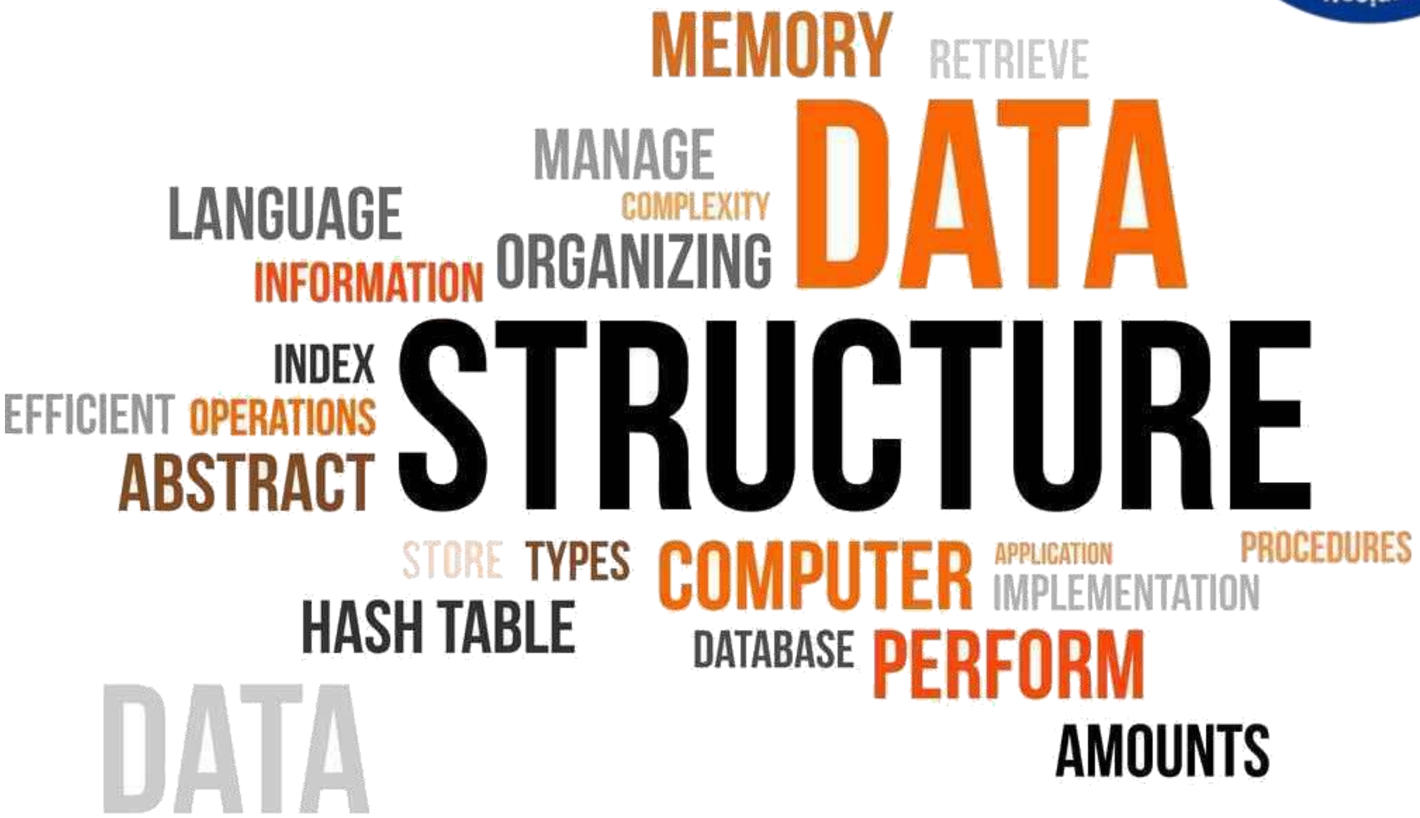
Data Structures

Course code: IT623





Dr. Rahul Mishra
Assistant Professor
DA-IICT, Gandhinagar

Lectures 5

Algorithm analysis and Run time

Algorithmic analysis and runtime

* 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 operations – in sorting and searching algorithms, e.g., the number of comparisons.

* Specifically, key operations are so defined ~~that~~ the time for other operations is much less than or at most proportional to the time for the key operations.

* The space measured by counting the maximum of memory needed by the algorithm.

* " The complexity of an algorithm $\boxed{M}$ is the function $\boxed{f(n)}$ which gives the running time and/or storage space requirement of the algorithm in terms of the size n of the input data.

Algorithmic analysis and runtime

* The storage space required by an algorithm is simply a multiple of the data size n.

* Unless otherwise stated or implied, the term "complexity" shall refer to the running time.

* The running time of an algorithm depends not only on the size n of the input data but also on the particular data.

e.g.

Suppose we are given an English short story "TEXT" and suppose we want to search through TEXT for the first occurrence of a given 3-letter word W. If W is the 3-letter word "the," then it is likely that W occurs near the beginning of TEXT, so $f(n)$ will be small. On the other hand, if W is the 3-letter word "zoo" then W may not appear in TEXT at all, so $f(n)$ will be large.

Algorithmic analysis and runtime

1) Worst case : → the maximum value of $f(n)$ for any possible input

2) Average case: the expected value of $f(n)$

3) Best case: Sometimes, we also consider the minimum possible value of $f(n)$,

* Analysis of average case assumes a certain probabilities distribution for the input data;

↳ one such assumption might be that all possible permutations of an input data set are equally likely.

→ The average case also uses the following concept in probability theory

→ Let the number $n_1, n_2, \ldots, n_k$ occur with respective probabilities $p_1, p_2, \ldots, p_k$

Expected value $\boxed{E = n_1 p_1 + n_2 p_2 + \cdots + n_k p_k}$

Algorithmic analysis and runtime

Example 2.8 Linear Search

Suppose a linear array DATA contains n elements, and suppose a specific ITEM of information is given. We want either to find the location LOC of ITEM in the array DATA, or to send some message, such as LOC = 0, to indicate that ITEM does not appear in DATA. The linear search algorithm solves this problem by comparing ITEM, one by one, with each element in DATA. That is, we compare ITEM with DATA[1], then DATA[2], and so on, until we find LOC such that ITEM = DATA[LOC]. A formal presentation of this algorithm follows.

Algorithm 2.4: (Linear Search) A linear array DATA with N elements and a specific ITEM of information are given. This algorithm finds the location LOC of ITEM in the array DATA or sets LOC = 0.

1. [Initialize] Set K := 1 and LOC := 0.
2. Repeat Steps 3 and 4 while LOC = 0 and K ≤ N.
3. If ITEM = DATA[K], then: Set LOC: = K.
4. Set K := K + 1. [Increments counter.]
   [End of Step 2 loop.]
5. [Successful?]
   If LOC = 0, then:
   Write: ITEM is not in the array DATA.
   Else:
   Write: LOC is the location of ITEM.
   [End of If structure.]
6. Exit.

Algorithmic analysis and runtime

- The complexity of the search algorithm is given by the number **C** of comparisons between
  ITEM and **DATA[K].**

- We seek **C(n)** for the worst case and the average case.

Worst case:
The worst case occurs when **ITEM** is the last element in the array **DATA** or is not there.

$$C(n) = n$$

Accordingly, **C(n) = n** is the worst-case complexity of the linear search algorithm.

Algorithmic analysis and runtime

- We assume that the **ITEM** does appear in the **DATA** and that it is equally likely to occur at any position in the array.

- The number of comparisons can be any of the numbers: **1, 2, 3, …, n**, and each number occurs with the probability **p= 1/n**.

$$C(n) = 1.\ 1/n + 2.\ 1/n + \ldots + n.\ 1/n$$
$$= (1+2+3+ \ldots + n).\ 1/n$$
$$= n(n+1)/2\ .\ 1/n = (n+1)/2.$$

- This agrees with the intuitive feeling that the average number of comparisons needed to find the location of ITEM is approximately equal to half the number of elements in the DATA list.

Algorithmic analysis and runtime

- The complexity of average case of an algorithm is usually much more complicated to analyses than that of the worst case.
- The probabilistic distribution that one assumes for the average case may not actually apply to real situations.
- Thus, unless otherwise stated or implied, the complexity of an algorithm shall mean the function which gives the running time of the worst case in terms of the input size.
- Moreover, the complexity of the average case for many algorithms is proportional to the worst case.

Growth of function

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

* The complexity $\boxed{f(n) \text{ of } M}$ increases as $\boxed{n}$ increases.

* It is usually the rate of increase of $\boxed{f(n)}$ that we want to examine.

* This is usually done by comparing $f(n)$ with some standard function

$$\boxed{\log_2 n, \quad n, \quad n\log_2 n, \quad n^2, \quad n^3, \quad 2^n}$$

n \ $g(n)$	$\log_2 n$	n	$n\log_2 n$	n^2	n^3	2^n
5	3	5	15	25	125	32
10	4	10	40	100	10^3	10^3
100	7	100	700	10^4	10^6	10^{30}
1000	10	10^3	10^4	10^6	10^9	10^{300}

"Rate of Growth of Standard Functions"

Asymptotic Analysis

- ***Dictionary meaning:*** *"Asymptotic function approaches a given value as an expression containing a variable tends to infinity."*
- We are concerned with how the **running time** of an algorithm increases with the size **of the input in the limit,** **as the size of the input increases without bounds.**
- An algorithm that is asymptotically more efficient will be the best choice for all but very small input.
- The notations we use to describe the asymptotic running time of an algorithm are defined in terms of functions whose domain is the set of natural numbers, **N =** {0,1,2,…}.
- They are used for defined worst-case running-time function **T(n),** which usually is defined only on integer input sizes.

Asymptotic Analysis

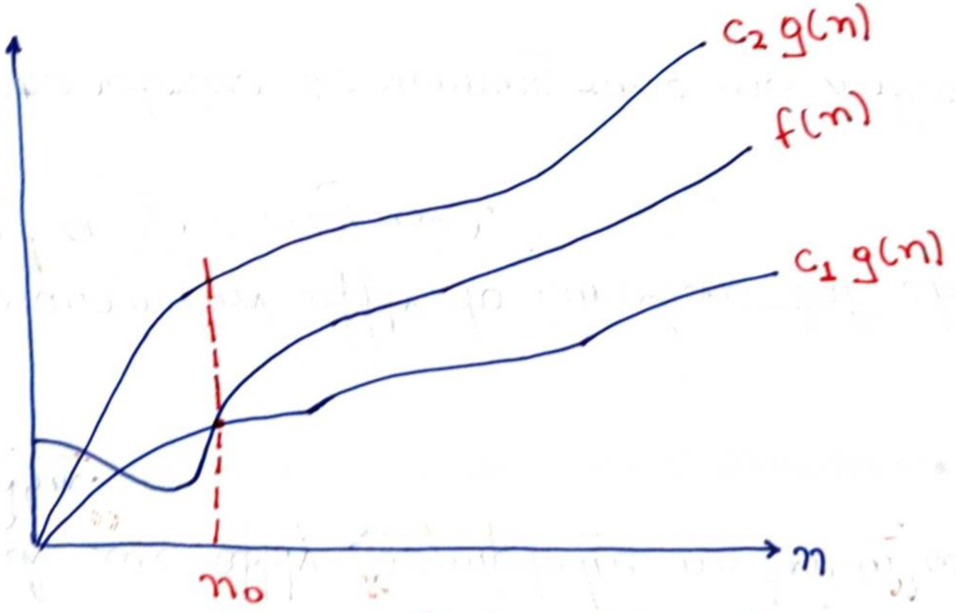
- We might extend the notation to the domain of ***real numbers or alternatively*,** restrict it to a subset of the natural number.
- The function to which we apply "*asymptotic notation*" will usually characterize the running times of the algorithm.
- In addition, the asymptotic notation can apply to functions that characterize some other aspects of the algorithm (the amount of space they used).
- We often wish to make/ characterize the running time no matter what the input.
- *Asymptotic notations are well suited to characterizing running times no matter what the input.*

Asymptotic Analysis: $\theta - notation$

Let us define what this notation mean. For a given function $g(n)$, we denote by $\Theta(g(n))$ the set of functions.

* $\Theta(g(n)) = \{ f(n) :$ there exist positive constants c_1, c_2 and n_0 such that

$0 \le c_1 g(n) \le f(n) \le c_2 g(n)$ for all $n \geq n_0 \}$.



$f(n) = \Theta(g(n))$

Asymptotic Analysis: $\theta - notation$

Θ - Notation

* A function $\boxed{f(n)}$ belongs to the set $\boxed{\Theta(g(n))}$ if there exist positive constants $\boxed{c_1}$ and $\boxed{c_2}$ such that it can be "sandwiched" between $\boxed{c_1 g(n) \text{ and } c_2 g(n)}$, for sufficiently large $\boxed{n}$.
  * → (read as f of n is theta of g of n)

* Since $\Theta(g(n))$ is a set, we can write "$f(n) \in \Theta(g(n))$" to indicate that $f(n)$ is a member of $\Theta(g(n))$.

* Instead, we will usually write "$f(n) = \Theta(g(n))$" to express the same notion.

* An intuitive picture of functions $\boxed{f(n) \text{ and } g(n)}$, where $f(n) = \Theta(g(n))$.

* For all values of n at and to the right of n_0, the value of $f(n)$ lies at or above $\boxed{c_1 g(n)}$ and at or below $\boxed{c_2 g(n)}$.

* $g(n)$ is an asymptotically tight bound for $f(n)$.