PES UNIVERSITY

Text Book:
Introduction to the Design and Analysis of Algorithms
Author: Anany Levitin
2nd Edition

Textbook Chapter 2
Section 2.1

Analysis of Algorithm

Investigation of Algorithm's efficiency with respect to two resources time and space is termed as analysis of algorithms.
We need to analyse the algorithms to

- determine the resource requirement(CPU time and memory)
- Compare different methods for solving the same problem before actually implementing them and running the programs.
- To find an efficient algorithm

There are two approaches to determine time complexity

- Theoretical Analysis
- Experimental study

Theoretical Analysis

General Framework to determine time complexity of algorithm

- Measuring an input's size
- Measuring running time
- Finding Orders of growth
- Worst-base, best-case and average efficiency

Time efficiency is represented as function of input size. Time efficiency is determined by counting the number of times algorithms basic operation executes. This is independent of processor speed, quality of implementation, compiler and etc.

Basic Operation: The operation that contributes most to the running time of an algorithm.

For some problems number of times basic operation executes differs for different inputs of same size for such problems we need to do Best, Worst and Average class analysis

Problem	*Input size measure*	*Basic operation*
Searching for key in a list of *n* items	Size of list	Key comparison
Multiplication of two matrices	Dimension of matrix	Elementary multiplication

Order of growth of algorithm's running time is important to compare the performance of different algorithms

Best Worst and Average case Analysis

- Worst case Efficiency
    - Number of times basic operation is executed for the worst case input of size n.
    - The algorithm runs the longest among all possible inputs of size n.
- Best case Efficiency
    - Number of times basic operation is executed for the best case input of size n.
    - The algorithm runs the fastest among all possible inputs of size n.
- Average case Efficiency:
    - Number of times basic operation is executed for random input of size n.
    - NOT the average of worst and best case

Time complexity Analysis of Sequential Search

```
ALGORITHM SequentialSearch(A[0..n-1], K)
    //Searches for a given value in a given array by sequential search
    //Input: An array A[0..n-1] and a search key K
    //Output: Returns the index of the first element of A that matches K or –1 if there are no matching elements
    i ←0
    while i < n and A[i] ǂ K do
        i ← i + 1
    if i < n        //A[I] = K
        return i
    else
        return -1
```

- Worst-Case: Cworst(n) = n
- Best-Case: Cbest(n) = 1

Let **'p'** be the probability that key is found in the list

Assumption: All positions are equally probable

Case1: key is found in the list

$C_{avg,case1}(n) = p*(1 + 2 + \ldots + n) / n = p*(n + 1) / 2$

Case2: key is not found in the list

$C_{avg,\ case2}(n) = (1-p)*(n)$

$\mathbf{C_{avg}(n) = p}(n + 1) / 2 \mathbf{+ (1 - p)}(n)$