MODULE-1
BASICS OF ALGORITHMS

ALGORITHM:-

An algorithm is a finite step of unambiguous instructions that if followed accomplishes a particular task.

CRITERIA:-

i) Input → 0 or more quantities are externally supplied

ii) Output → Atleast one quantity is produced

iii) Definiteness → Each instruction is clear and unambiguous

iv) Finiteness → The algorithm should terminate after a finite number of steps.

v) Effectiveness → ~~The~~ Every instruction must be very basic and feasible.

ALGORITHM SPECIFICATION

i) // → Comments begin with //, continue till the end of line

ii) {} Blocks are indicated with matching braces { }

iii) Identifiers begin with a variable. The data types of identifiers are not specified.

iv) = → Assignment of the variables is done using the assignment statement.

v) There are 2 boolean values, true and false and can be produced by logical or relational operators.

vi) Elements of a multi-dimensional array are accessed using A[i,j]

vii) Looping statements are employed → for, while, repeat until

viii) Conditional statements are used → if, else if, else if ladder

ix) Input → read
Output → write

x) The algorithm consists of heading and a body.

TIME COMPLEXITY → It is the time taken by the program to execute completely as a matter of inputs.

SPACE COMPLEXITY → It is the space / memory occupied by a program to execute completely.

~~ANAY~~ ANALYSIS FRAMEWORK

BEST CASE EFFICIENCY → The best case efficiency of an algorithm is its efficiency of size n for which the algorithm runs the fastest among all possible inputs of that size.

$\therefore C_{best}(n) = 1$

WORST CASE EFFICIENCY → The worst case efficiency of an algorithm is its efficiency of size n for which the algorithm runs the longest among all possible inputs of that size.

$\therefore C_{best}(n) = n$

AVERAGE CASE EFFICIENCY

We make certain assumptions

i) probability of successful search = p

ii) probability of the first match occurring in the i-th position is same for every i.

$1 \cdot \frac{p}{n} + 2 \cdot \frac{p}{n} + \cdots + i \cdot \frac{p}{n}$ (in case of successful search)

$\frac{p}{n} [1 + 2 + \cdots + n]$

$\frac{p}{n} \cdot \frac{n(n+1)}{2} = \frac{p(n+1)}{2}$

For unsuccessful search,

$n(1-p)$

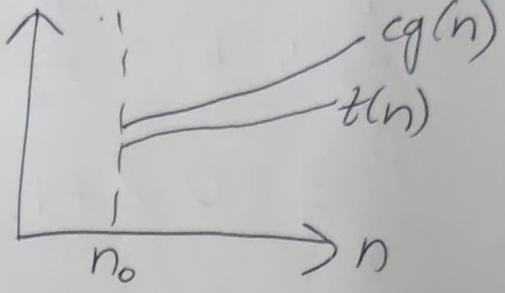
Average case $= \frac{p(n+1)}{2} + n(1-p)$

O-NOTATION

O-

A function $t(n)$ is said to be in $O(g(n))$ if denoted by $t(n) \in O(g(n))$ if $t(n)$ is bounded above by some constant multiple of $g(n)$ for all large n.

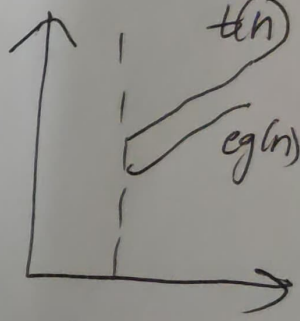
$t(n) \leq cg(n)$ for all $n \geq n_0$



Ω-NOTATION

A function $t(n)$ is said to be in $\Omega(g(n))$ if denoted by $t(n) \in \Omega(g(n))$ if $t(n)$ is bounded below by some constant multiple of $g(n)$ for all large n

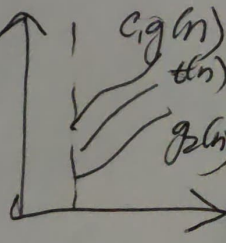
$t(n) \geq cg(n)$ for all $n \geq n_0$



Θ-NOTATION

A function $t(n)$ is said to be in $O(g(n))$, denoted $t(n) \in \Theta(g(n))$, if $t(n)$ is bounded both above and below by some positive constant multiples of $g(n)$ for all large n.

$c_2 g(n) \leq t(n) \leq c_1 g(n)$ for all $n \geq n_0$



MATHEMATICAL ANALYSIS OF NONRECURSIVE ALGORITHMS

$$\sum_{i=l}^{u} ca_i = c \sum_{i=l}^{u} a_i$$

$$\sum_{i=l}^{u} (a_i \pm b_i) = \sum_{i=l}^{u} a_i \pm \sum_{i=l}^{u} b_i$$

$$\sum_{i=l}^{u} 1 = u - l + 1$$

$$\sum_{i=0}^{n} i = \sum_{i=1}^{n} i = 1 + 2 + \cdots + n = \frac{n(n+1)}{2} \approx \frac{1}{2} n^2 \in \Theta(n^2)$$

1) UNIQUE ELEMENT

$$C_{worst}(n) = \sum_{i=0}^{n-2} \sum_{j=i+1}^{n-1} 1$$

$$= \sum_{i=0}^{n-2} [(n-1) - (i+1) + 1]$$

$$= \sum_{i=0}^{n-2} (n-1-i) = \sum_{i=0}^{n-2} (n-1) - \sum_{i=0}^{n-2} i$$

$$= \sum_{i=0}^{n-2} 1 - \frac{(n-2)(n-1)}{2}$$

$$= (n-1)^2 - \frac{(n-2)(n-1)}{2}$$

$$= \frac{(n-1)n}{2} \approx \frac{1}{2} n^2 \in \Theta(n^2)$$

2) MATRIX MULTIPLICATION

$$M(n)=\sum_{i=0}^{n-1}\sum_{j=0}^{n-1}\sum_{k=0}^{n-1}1=\sum_{i=0}^{n-1}\sum_{j=0}^{n-1}n=\sum_{i=0}^{n-1}n^2=n^3=O(n^3)$$

3) ~~NO~~

MATHEMATICAL ANALYSIS OF RECURSIVE ALGORITHMS

1) FACTORIAL

$n! = 1 \cdot \ldots \cdot (n-1) \cdot n = (n-1)! \cdot n$ for $n \geq 1$

$M(n) = M(n-1)+1$

$= [M(n-2)+1]+1 = M(n-2)+2$

$= [M(n-3)+1]+2 = M(n-3)+3$

$= M(n-i)+i$

$= M(n-n)+n$

$= n$

2) TOWER OF HANOI

$M(n) = M(n-1)+1+M(n-1)$

$M(n) = 2M(n-1)+1$

$= 2[2M(n-2)+1]+1 = 2^2 M(n-2)+2+1$

$= 2^2[2M(n-3)+1]+2+1 = 2^3 M(n-3)+2^2+2+1$

$M(n) = 2^i M(n-i) + 2^{i-1}+2^{i-2}+\ldots+2+1 = 2^i M(n-i) + 2^i - 1$

$M(n) = 2^{n-1} M(n-(n-1)) + 2^{n-1} - 1$

$= 2^{n-1} + 2^{n-1} - 1 = 2^n - 1$

IMPORTANT PROBLEM TYPES

* Sorting → Eg → GPA
* Searching → search key
* String processing → characters, string matching
* Graph problems → TSP, coloring, nodes, edges
* Combinatorial problems → no known algos, max/min cost

FUNDAMENTAL DATA STRUCTURES

* ~~ST~~ Stacks
* Queues
* Graphs
* Trees
* Sets
* Dictionaries