UNIT-I Introduction to Algorithms

complexity

Space complexity Time complexity

Algorithm,
Input, output,
Definiteness,
Finiteness,
Effectiveness

$S(P) = C + S_p(I)$

C — All variables/ Constants

$S_p(I)$ — Stack / Array / formal Parameters

$T(P) = C + T_p(I)$

Count method / Table method / depends on System processes etc.

⇒ Step / Instruction
n / 1

```
void printmatrix(int matrix[][MAX_SIZE], int rows,
                 int cols)
{
    int i, j;
    for(i=0; i<rows; i++)
    {
        for(j=0; j<cols; j++)
        {
            printf("%d", matrix[i][j]);
        }
        printf("\n");
    }
}
```

$O(n^2)$

$S_p = n^2 + 0$

$T_c = n^2 + 0$

0	–	0
0	–	0
0	–	0
1	n	n
–	n	n
–	–	–
0	0	0
=	=	=

n^2+0

– – – n – n – 1 – – – –

```
Void Sum(int a[], int n)                    | n
{                                           |
    int i;                                  | 1
    int tempsum = 0;                        | 1
    for(i=0; i<n; i++)                      | n
    {                                       |
        tempsum += a[i];                    |
    }                                       |
    return tempsum;                         | 1
}                                           |
                                            | 2n + 3
```

```
Void Sum(int a[], int n)                    |
{                                           |
    if(n)                                   | n
    {                                       |
        return Sum(a, n-1) + Sum(a, n)      | n
    }                                       | 1
}                                           | 1
                                            | 2n+2
```

=) Amortized

a) Aggreg

b)

T(n)/n

=) Amortize

if a>b then ___

if b>a then ___
else ___

for i=1 to n do
{
}

while(n>0) do
{
}

a := b

<, ≤, ≥, >, ≠, =

A[i,j]

Pseudo code conventions | Algorithm Name(list of parameters)

// | { } | node = record

{
    datatype1 data1;
    datatypen datan;
    node *link;
}

=) Amortized Analysis:- Loan payment (EMI)/month, making Big things into Small by dividing.

a) ~~Aggregate~~ Analysis:-

Similar data type's
No Balancing done
Average of Time Complexities

push	0.18
pop	0.18
multipop	0.18

b) Accounting method:- Prepaid Credit

Different data type's
Balancing done

$T(n)/n$

Push	0.3
Pop	0.3
multi-pop	0.7

Excess charge $= 0.4$

Push	0.4	Push	0.43
Pop	0.4	Pop	0.43
multi-pop	0.5	multi-pop	0.43

c) potential method:-

The whole process Balancing done at a time.

To show multipop as less as Expensive,

Taking one process.

Push +10₹, Pop +10₹, multipop +10₹

⇒ $O(1)$ - push | $O(n)$

$O(1)$ - pop | $O(0)$

$O(n)$ - multi pop, | $O(0)$, Avg - $O(n)$

Avg - $O(n)$

Substitution, Iteration, Recursive-tree, The master

* $T(n) = T(\frac{n}{2} + 1)$ Substitution / $O(\log n)$ | $T(n) \leq c\log n$

$T(n) \leq c\log(n)$

$T(n) \leq c\log(\frac{n}{2}) + 1$

$T(n) \leq c(\log n - \log_2 2) + 1$

$T(n) \leq c\log n - c + 1$

$T(n) \leq c\log n$ for $c \geq 1$

* $T(n) = T(n-1)$ Iteration | $T(n) = T(n-1)$

$I_2 = 2[2T(n-2)] = 2^2T(n-2)$ | $2^iT(n-i)$

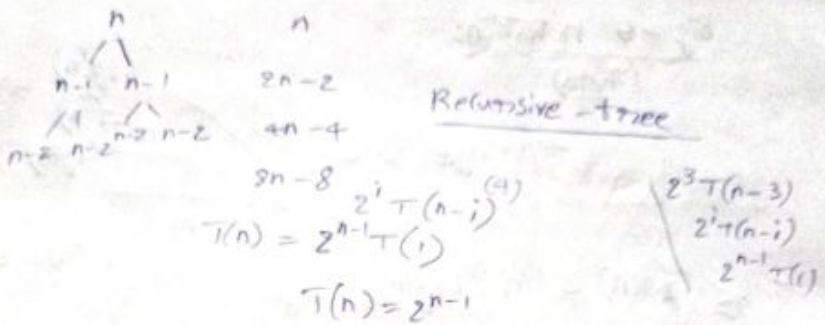
$I_3 = 4[2T(n-3)] = 2^3T(n-3)$ | $n-i=1$, $n-1=i$

$I_4 = 8[2T(n-4)] = 2^4T(n-4)$ | $2^{n-1}T(1)$

$T(n) = 2^iT(n-i)$ | $n-i=1$, $n-1=i$

$T(n) = 2^{n-1}T(1)$

$T(n) = 2^{n-1}$



* $T(n) = aT\left(\frac{n}{b}\right) + f(n)$ — The master

a – no. of sub problems

n – size of the problem

$\frac{n}{b}$ – size of each sub problem

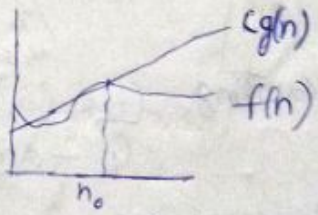
f(n) – work done outside the function call.

$T(n) = aT\left(\frac{n}{b}\right) + f(n)$

⇒ Big–O notation;–

worst-case,

$f(n) \leq c(g(n)$



$f(n) = O(g(n))$

$10n^2 + 4n + 2 \leq 11n^2 \quad / \quad n \geq 5$

Big –Ω notation;– (omega)

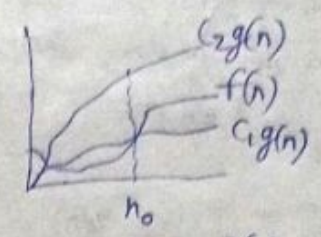
Best-case,

$f(n) \geq c(g(n)$

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

Big-θ notation:-
(Theta)

Average case



$0 \le c_1 g(n) \le f(n) \le c_2 g(n)$ $\quad f(n) = \theta(g(n))$

O(1), O(log n), O(n), O(n log n), O(n^c), O(c^n), O(n!)

O(1)
O(n)
O(n²)
O(n³)
O(2^n)
O(log n)
O(n log n)

Aggregate method

$\sum_{1 \le i \le n} p(i) = \sum_{1 \le i \le n} (\text{amortized}(i) - \text{actual}(i) + p(i-1))$

Accounting

Credit = An op. amortized cost - actual cost

$\bar{c}_i$ - amortized cost

c_i - Actual cost

$\sum_{i=1}^{n} \bar{c}_i \ge \sum_{i=1}^{m} c_i$

Total Credit $\Rightarrow \sum_{i=1}^{n} \bar{c}_i - \sum_{i=1}^{m} c_i \ge 0$

Push, - O(1) (O(n))
Pop, - O(1)
multipop, - min(k, n) ← size
↓ Elements

push(x) - 2
pop() - 0
multipop() - 0

push | y → 1+1 | pop use y
x → 1+1 | use x
↓ cost ↓ credit