DESIGN AND ANALYSIS OF ALGORITHMS

Algorithm -

Algorithm is set of instructions | Steps to solve problem/task.

Optimal solution -

Based on our requirement, the best solution among all solutions is optimal solution.

Types of Algorithms -

* Searching
* Sorting
* Brute Force
* Greedy Method
* Dynamic programming
* Divide and Conquer
* Recursive

These are different methods through which we can solve a problem.

Algorithm Analysis -

It is used to evaluate the performance of Algorithm.

This is done by
1. Time complexity
2. Space complexity

→ Best case
→ worst case
→ Average case
→ Amortized

Types of Time complexites -

→ Time complexity is defined as in how much ~~types~~ time we solve a problem

→ constant Time complexity - $O(1)$

→ Linear Time complexity - $O(n)$

→ Logarithmic time complexity - $O(\log n)$

→ $O(n\log n)$ Time complexity - $O(n\log n)$

→ Quadratic time complexity - $O(n^2)$

Constant Time complexity - $O(1)$

* No. of steps required to excute the problem is constant.

* Time does not vary if increase in input size.

Ex :- All Arthmetic operation

Linear time complexity - $O(n)$

* No. of steps required to excute the problem depends on input size.

* If input size increases time taken to excute will also increase.

Ex :- Linear Search

Logarthmic time complexity - $O(\log n)$

* problem will be divided into small parts

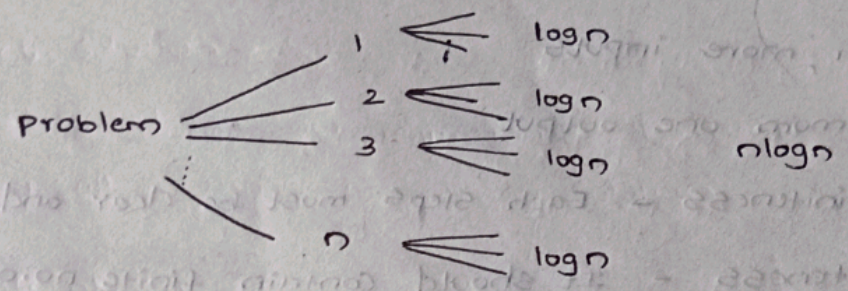
* After each division no. of steps will be reduced to half

Ex :- Binary Search

vivo T2x 5G
Aug 11, 2025, 17:26

O(nlogn) Time complexity - O(nlogn)

* problem will be divided into small parts
* After each division no. of steps required to solve the problem will be reduced to half.
* Colloborating all small parts to complete the excewtion



Ex :- Merge Sort

* this algorithm is efficient for large input

Quadratic Time complexity - $O(n^2)$

* No. of steps required to excewte the problem is double the input size

if n - input size

n^2 - steps required

* This algorithm will be slow for large inputs

Ex :- Bubble Sort

Algorithm -

Finite set of instructions/ steps used to solve, perform a task.

Characterstics -

1. 0 or more inputs
2. Minimum one output
3. Definiteness - Each steps must be clear and unambiguous
4. Finiteness - It should contain finite no. of steps
5. Effectiveness - Each step must be effective.

Issues in the study of Algorithm -

1> How to create Algorithms?

① Divide and conquer
② Greedy
③ Dynamic programming
④ Branch and Bound
⑤ Backtracking

2> How to validate Algorithm?

If Algorithm produce correct result then it is validate and if not we have to modify Algorithm.

3> How to analyze an algorithm?

① Time complexity / computing time
② Storage space / space complexity

To calculate time complexity a) Frequency step co
b) Asympotic note

vivo T2x 5G
Aug 11, 2025, 17:26

4> How to test an algorithm?

① Debugging

② profiling | performance measurement

PERFORMANCE ANALYSIS OF AN ALGORITHM -

We can analyze algorithm using

1. Time complexity

2. space complexity

1. Time Complexity -

Time complexity is in how much time we solve a problem.

a) Best case TC - Minimum amount of time

b) Worst case TC - Maximum amount of time

c) Average case TC - Average amount of time

To calculate time complexity

1> Frequency (or) Step count

2> Asymptotic note

2. Space complexity -

Space complexity is how much space it occupies by a problem.

It is calculated by frequency (or) step count

vivo T2x 5G
Aug 11, 2025, 17:26

Frequency count (or) step count -

Frequency count is no. of times a statement should be executed.

1. For comments, declarating step count is 0.
2. return, Assignment step count is 1.
3. Ignore lower order exponents when higher order exponents are present.

$$3n^{4} + 4n^{3} + 10n^{2} + n^{1} + 100$$

4. Ignore constant multiplier.

Example -

```
Sum (int a[], int n){
    s=0;                        — 1
    for(i=0; i<n; i++)          — n+1
        s = s+a[i];             — n
    return s;                   — 1
}
```

$n=3$

$0<3$, $1<3$, $2<3$, $3<3$ } $4=n+1$

$\Rightarrow 1+n+1+n+1 = 2n+3$

Time complexity $= O(n)$

Space complexity -

$a \rightarrow n$

$n \rightarrow 1$

$s \rightarrow 1$

$i \rightarrow 1$

$S(n) = n+3 = O(n)$

Example -

```
void mat add (int a[ ][ ], int b[ ][ ]){
    int c[ ][ ];
    for (i=0; i<n; i++){        —— n+1
        for (j=0; j<n; j++){    —— n(n+1)
            c[i][j] = a[i][j] + b[i][j]; — n×n
        }
    }
}
```

$\Rightarrow n+1+n^2+n+n^2 = 2n^2+2n+1$

Time complexity $= O(n^2)$

Space complexity -

$a \rightarrow n^2$

$b \rightarrow n^2$

$c \rightarrow n^2$

$i \rightarrow 1$

$j \rightarrow 1$

$n \rightarrow 1$

$3n^2+3 = O(n^2)$

Example -

```
void mat-mul (int a[ ][ ], int b[ ][ ]){
    for (i=0; i<n; i++){        — n+1
        for (j=0; j<n; j++){    —— n(n+1)
            c[i][j] = 0;        — n×n
            for (k=0; k<n; k++){ — n×n×(n+1)
                c[i][j] += a[i][k] * b[k][j]; — n×n×n
            }
        }
    }
}
```

Space complexity :

$a \rightarrow n^2$ $\quad n \rightarrow 1$

$b \rightarrow n^2$ $\quad i \rightarrow 1$

$c \rightarrow n^2$ $\quad j \rightarrow 1$

$k \rightarrow 1$

$S(n) = 3n^2+4$

$O(n^2)$

Time complexity $= n+1+n^2+n+n^2+n^3+n^2+n^3$

$= 2n^3+3n^2+2n+1 = O(n^3)$

vivo T2x 5G
Aug 11, 2025, 17:26

Asymptotic notations -

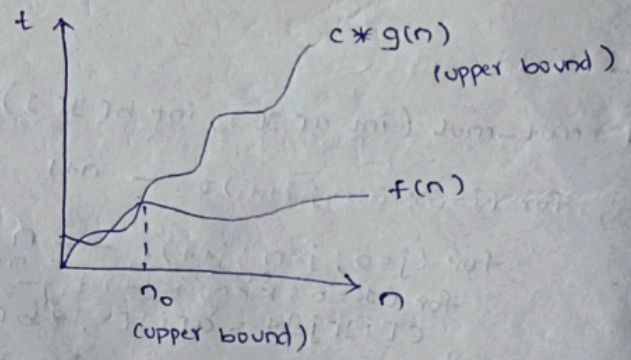
By using Asymptotic notations we can calculate best, worst and average case time complexity.

1. Big oh notation (O)
2. Big omega notation (Ω)
3. Theta Notation (θ)
4. Little oh notation (o)
5. Little omegha notation (ω)

1> Big oh notation (O) -

* Big oh notation mainly represents upper bound of algorithm's running time.
* To calculate maximum amount of time for it's excution i.e worst case Time complexity

Let $f(n)$, $g(n)$ be two non negative functions then $f(n) = O(g(n))$ if there exist 2 +ve constants c, n_0 such that $f(n) \leq c * g(n)$, $\forall\ n > n_0$



Example - $f(n) = 3n+2$ $\quad g(n) = n$

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

$\Rightarrow f(n) \leq c * g(n) \quad \forall\ n > n_0$

$3n+2 \leq c * n$

Let $c=4$

$3n+2 \leq 4n$

vivo T2x 5G
Aug 11, 2025, 17:27

Let $n_0 = 1$ $\quad 5 \leq 4$ ✗

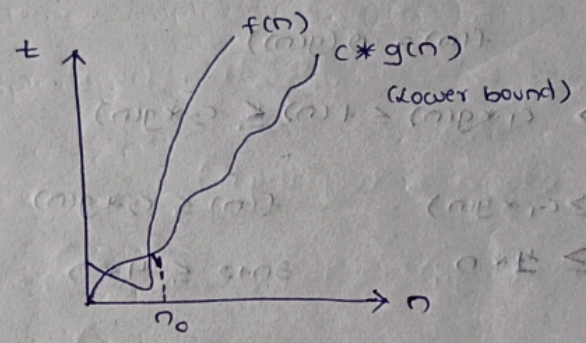
$n_0 = 2$ $\quad 8 \leq 8$ ✓ $\quad$ So, $n_0 \geq 2$

$n_0 = 3$ $\quad 11 \leq 12$ ✓ $\quad$ $n \geq 2$

2> Big omega Notation (Ω) -

* Lower bound of algorithm run time
* To calculate minimum amount of time i.e Best case Time complexity

Let $f(n)$, $g(n)$ be two non negative functions then $f(n) = \Omega(g(n))$ if there exist two +ve constants c, n_0 such that $f(n) \geq c * g(n)$, $\forall n > n_0$



Example -

$f(n) = 3n+2 \quad g(n) = n$

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

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

$3n+2 \geq 1 * n$ $\quad$ Let $c = 1$

Let $n_0 = 1 \rightarrow 5 \geq 1$

$n_0 = 2 \rightarrow 8 \geq 2$ $\quad$ So, $n_0 \geq 1$

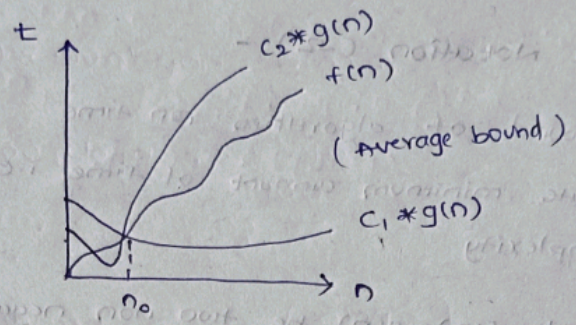
$n_0 = 3 \rightarrow 11 \geq 3$

3> Theta notation (Θ) -

* Average bound of algorithm run time
* To calculate average amount of time i.e average case time complexity.

vivo T2x 5G
Aug 11, 2025, 17:27

Let $f(n), g(n)$ be 2 non negative functions then $f(n) = \Theta(g(n))$ if there exist 3 +ve constants c_1, c_2, n_0 such that $c_1 * g(n) \le f(n) \le c_2 * g(n)$ $\forall$ $n > n_0$



Example –

$f(n) = 3n+2$ $\quad g(n) = n$

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

$\Rightarrow$ $c_1 * g(n) \le f(n) \le c_2 * g(n)$

$f(n) \ge c_1 * g(n)$ $\qquad$ $f(n) \le c_2 * g(n)$

$3n+2 \ge 1 * n$ $\qquad$ $3n+2 \le 4 * n$

$\Rightarrow$ $c_1 = 1$, $c_2 = 4$

$n \le 3n+2 \le 4n$

Let $n_0 = 1$ $\quad 1 \le 5 \le 4$

$n_0 = 2$ $\quad 2 \le 8 \le 8$ $\qquad$ So, $n_0 \ge 2$

$n_0 = 3$ $\quad 3 \le 11 \le 12$ $\qquad$ $n \ge 2$

4> Little oh notation (o) –

Let $f(n), g(n)$ be two non negative functions then $f(n) = o(g(n))$ such that

$$\lim_{n \to \infty} \frac{f(n)}{g(n)} = 0$$

5) Little omega notation (ω) -

Let $f(n), g(n)$ are two non negative functions then $f(n) = \omega(g(n))$ such that

$$\lim_{n\to\infty} \frac{g(n)}{f(n)} = 0 \quad (\text{or}) \quad \lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty$$

a) $f(n) = n \quad g(n) = n^2$

$f(n) = n \quad g(n) = n^2$

n	$f(n)$	$g(n)$
1	1	1
2	2	4
3	3	9
4	4	16

$n_0 = 1$

$f(n) \le g(n)$

$f(n) \le c * g(n)$

$n_0 \ge 1$, c is constant

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

b) $f(n) = 2^n \quad g(n) = n^n$

$f(n) = 2^n \quad g(n) = n^n$

n	$f(n)$	$g(n)$
1	2	1
2	4	4
3	8	27
4	16	256

$n_0 = 2$

$f(n) \le g(n) \quad n \ge 2$

$f(n) \le c * g(n)$

$n \ge 2$, c is constant

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

c) $f(n) = n^{\sqrt{n}} \quad g(n) = 3^{\sqrt{n}\log n}$

$n^{\sqrt{n}} = 3^{\sqrt{n}\log n}$

Apply log on both sides

$\log n^{\sqrt{n}} = \log 3^{\sqrt{n}\log n}$

$\cancel{\sqrt{n}\log n} = \cancel{\sqrt{n}\log n}\log 3$

$1 = \log 3$

vivo T2x 5G
Aug 11, 2025, 17:27

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

d) $100 n \log n = \theta\left(\frac{n \log n}{100}\right)$

$f(n) = 100 n \log n \qquad g(n) = \frac{n \log n}{100}$

$100 \cancel{n \log n} = \frac{\cancel{n \log n}}{100}$

$100^2 = 1$

$100 n \log n = \theta\left(\frac{n \log n}{100}\right)$

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

e) $f(n) = \sqrt{\log n} \qquad g(n) = \log(\log n)$

$\sqrt{\log n} = \log(\log n)$

$\log(\log n)^{1/2} = \log(\log(\log n))$

$\frac{1}{2}\log(\log n) = \underbrace{\log(\log(\log n))}_{\text{smaller value}}$

$f(n) > g(n)$

$f(n) = \Omega\, g(n)$, where $c \neq 1$

Time Complexity –

*

```
a = 1
while (a<b) {
    statm;
    a=a*2;
}
```

$a = 1$

$1 \times 2 = 2$

$2 \times 2 = 2^2$

$2^2 \times 2 = 2^3$

$\vdots$

2^k

To terminate

$a \geq b$

$\therefore a = 2^k$

$2^k \geq b$

$k \geq \log_2^b$

$O(\log n)$

```
* i = 1;
  k = 1;
  while (k < n)
  {
      stat;
      k = k + i;
      i++;
  }
```

i	k
1	1
2	1+1 = 2
3	2+2
4	2+2+3
5	2+2+3+4
⋮	⋮
m	1+2+3+4+...+m

roughly $\frac{m(m+1)}{2}$

$k \geq n$

$\frac{m(m+1)}{2} \geq n$

roughly $m^2 \geq n$ $\quad O(\sqrt{n})$

$m \geq \sqrt{n}$

$\boxed{m = \sqrt{n}}$

```
* while (m != n) {
      if (m > n)
          m = m - n;
      else
          n = n - m;
  }
```

m=16	n=2
14	2
12	2
10	2
8	2
6	2
4	2
2	2

$\frac{16}{2}$

$\frac{3}{2}$

$O(m)$

min time : $O(1)$

max time : $O(n)$

```
* Algorithm Test (n) {
      if (n < 5) {
          Printf ("%d", n); — 1
      }
      else {
          for (i=0; i<n; i++) {
              printf ("%d", i); — n
          }
      }
  }
```

Best : $O(1)$

worst : $O(n)$

$1 < \log n < \sqrt{n} < n < n\log n < n^2 < n^3 < \ldots\ldots < 2^n < 3^n < \ldots < n^n$

Time complexity will be among the above values are multiples of these

Asympotic notations -

1> Big - oh :

$$f(n) \leq c * g(n) \quad \forall \quad n \geq n_0$$

Ex :

$$f(n) = 2n+3$$

Assume some $g(n)$

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

$$2n+3 \leq 10n \quad n \geq 1$$

(or)

Just make n multiply

$$2n+3 \leq 2n+3n$$

$$2n+3 \leq 5n, \; n \geq 1 \rightarrow f(n) = O(n)$$

We can also write

$$2n+3 \leq 2n^2+3n^2$$

$$2n+3 \leq 5n^2, \; n \geq 1 \rightarrow f(n) = O(n^2)$$

$1 < \log n < \sqrt{n} < n < n\log n < n^2 < n^3 < \ldots\ldots < 2^n < 3^n \ldots$

Lower bound — Average bound — upper bound

So, we can write

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

$$f(n) = O(n^2)$$

$$f(n) = O(2^n)$$

2) omegha notation –

$$f(n) \geq c * g(n) \quad \forall \; n \geq n_0$$

Ex: $f(n) = 2n+3$

$$\underset{f(n)}{2n+3} \geq \underset{c}{1} * \underset{g(n)}{n}, \; n \geq 1 \rightarrow f(n) = \Omega(n)$$

We can change $g(n)$ in range of average bound

$$2n+3 \geq 1 * \log n \quad \forall \; n \geq 1 \rightarrow f(n) = \Omega(\log n)$$

Always nearer value is considered among all values in the bound

3) Theta Notation –

$$c_1 * g(n) \leq f(n) \leq c_2 * g(n)$$

Ex: $c_1 * g(n) \leq f(n) \leq c_2 * g(n)$

$f(n) = 2n+3$

$$\underset{c_1}{1} \times \underset{g(n)}{n} \leq \underset{f(n)}{2n+3} \leq \underset{c_2}{5} \underset{g(n)}{n} \qquad \therefore f(n) = \theta(n)$$

Average notation have only one value which acts as barrier between lower & upper bound.

Examples

1) $f(n) = 2n^2+3n+4$

$\rightarrow 2n^2+3n+4 \leq 2n^2+3n^2+4n^2$

$2n^2+3n+4 \leq 9n^2, \; n \geq 1$

$f(n) = O(g(n)) = O(n^2)$

$\rightarrow 2n^2+3n+4 \geq 1 \times n^2$

$f(n) = \Omega(n^2)$

$\rightarrow 1 \times n^2 \leq 2n^2+3n+4 \leq 9n^2$

$f(n) = \theta(n^2)$

vivo T2x 5G
Aug 11, 2025, 17:28

2> $f(n) = n^2 \log n + n$

$$1 \times n^2 \log n \le n^2 \log n + n \le 10\, n^2 \log n$$

$$f(n) = O(n^2 \log n)$$

$$f(n) = \Omega(n^2 \log n)$$

$$f(n) = \Theta(n^2 \log n)$$

3> $f(n) = n!$

$$n! = n \times (n-1) \times (n-2) \times \ldots\ldots \times 3 \times 2 \times 1$$

$$1 \times 1 \times 1 \times 1 \times \ldots\ldots \times 1 \le 1 \times 2 \times 3 \times \ldots \times n \le n \times n \times n \times \ldots \times n$$

$$1 \le n! \le n^n$$

Both sides we have different function. So, no average case.

$$f(n) = \Omega(1)$$

$$f(n) = O(n^n)$$

4> $f(n) = \log n!$

$$\log(1 \times 1 \times 1 \times \ldots\ldots \times 1) \le \log(n \times n-1 \times \ldots\ldots \times 3 \times 2 \times 1) \le \log(n \times n \times n \times \ldots)$$

$$\log 1 \le \log n! \le \log n^n$$

$$\log 1 \le \log n! \le n \log n$$

$$0 \le \log n! \le n \log n$$

$$f(n) = \Omega(1)$$

$$f(n) = O(n \log n)$$

vivo T2x 5G
Aug 11, 2025, 17:28

Properties of Asympotic notations –

→ If $f(n)$ is $O(g(n))$ then $a * f(n)$ is $O(g(n))$

Ex :- $f(n) = 2n^2+5$ is $O(n^2)$

then $7 \cdot f(n) = 7(2n^2+5)$

$= 14n^2+35$ is $O(n^2)$

Same for omega, theta

→ Reflexive property –

If $f(n)$ is given then $f(n)$ is $O(f(n))$

Ex - $f(n) = n^2$ $O(n^2)$

→ Transitive property –

If $f(n)$ is $O(g(n))$ and $g(n)$ is $O(h(n))$

then $f(n) = O(h(n))$

Ex - $f(n) = n$ $g(n) = n^2$ $h(n) = n^3$

if n is $O(n^2)$ and n^2 is $O(n^3)$

then n is $O(n^3)$

→ Symmetric property –

If $f(n)$ is $\theta(g(n))$ then $g(n)$ is $\theta(f(n))$

Ex - $f(n) = n^2$ $g(n) = n^2$

$f(n) = \theta(n^2)$ only for theta

$g(n) = \theta(n^2)$

→ Transpose symmetric –

If $f(n) = O(g(n))$ then $g(n)$ is $\Omega(f(n))$

Ex - $f(n) = n$ $g(n) = n^2$

then n is $O(n^2)$ and

n^2 is $\Omega(n)$

vivo T2x 5G
Aug 11, 2025, 17:28

→ If $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$

$$g(n) \le f(n) \le g(n)$$

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

→ If $f(n) = O(g(n))$ and $d(n) = O(e(n))$

then $f(n) + d(n) = O(\max(g(n), e(n)))$

Ex - $f(n) = n^2 = O(n)$

$d(n) = n = O(n^2)$

$f(n) + d(n) = n^2 + n = O(n^2)$

→ If $f(n) = O(g(n))$ and $d(n) = O(e(n))$ then

$f(n) * d(n) = O(g(n) * e(n))$

vivo T2x 5G
Aug 11, 2025, 17:28

PROBABILISTIC ANALYSIS -

→ Analysis of an algorithm based on probability.

→ An algorithm is said to be randomized in behaviour if the o/p produced is dependent on the random generator.

Hiring problem -

Scenario -

Manager hiring an Assistant with the help of an Agency.

Strategy for selection -

- Agency sends a candidate per day
- Hire the new candidate if he is better than the current Assistant

Algorithm -

```
Hire (N) {
    best = 0 ; // dummy candidate
    for (i=1 to N){
        if ( candidate i  better than  best)
        {
            best = i
            Hire candidate i (Ci)
        }
    }
}
```

Objective : Reduce the cost

$$\boxed{\text{Total cost} = O(nc_i + mc_h)}$$

m - candidates hired

n - candidates interviewed

vivo T2x 5G
Aug 11, 2025, 17:28

Worst case - Agency is sending candidate in increasing order of quality.

C_i	1	2	3	4	5
Rank;	1	3	7	10	15
	✓	✓	✓	✓	✓
	×	×	×	×	

Here,

$$\boxed{\text{Total cost} = nC_h + nC_i} = O(n(C_h + C_i))$$

→ No control over the order in which candidates are sent for interview

1. Randomize i/p distribution
2. Apply probabilistic analysis

Candidates will appear in random order i.e

① Agency will only provide the candidates list

② Manager will call candidates in random order.

As input is randomized,

$$\boxed{\text{Actual Running Time} = \text{Expected running Time}}$$

Indicator Random Variable (IRV)

* Method to convert probabilities to expectations and vice versa

* Indicator variable, $I\{A\}$

$$I\{A\} = \begin{cases} 1 & \text{if event A occurs} \\ 0 & \text{otherwise} \end{cases}$$

Eg: Coin Toss

$$X_h = I\{H\} = \begin{cases} 1 & \text{if } H \text{ occurs} \\ 0 & \text{otherwise} \end{cases}$$

H - occurance of Head

In a flip,

$$E(X_h) = E(I\{H\})$$

$$= 1 \cdot \text{prob}\{H\} + 0 \cdot \text{prob}\{T\}$$

$$= 1 \cdot \frac{1}{2} + 0 \cdot \frac{1}{2}$$

$$E(X_h) = \frac{1}{2}$$

Give a Sample Space S and Event A

Let $X_A = I\{A\}$

$$E(X_A) = \text{prob}\{A\}$$

For n flips,

$$X = \sum_{i=1}^{n} X_i$$

$$E(X) = E\left(\sum_{i=1}^{n} X_i\right)$$

$$= \sum_{i=1}^{n} E(X_i)$$

$$= \sum_{i=1}^{n} \frac{1}{2}$$

$$= \frac{1}{2} + \frac{1}{2} + \dots\dots + \frac{1}{2}$$

$$\boxed{E(X) = \frac{n}{2}}$$

vivo T2x 5G
Aug 11, 2025, 17:28

Analyzing Hiring prblm with IRV -

X - random variable $\left(\begin{array}{c}\text{\# times we hire} \\ \text{a new assistant}\end{array}\right)$

$$X = \sum X_i \qquad i = 1, 2, \ldots, n$$

$$X_i = \begin{cases} 1 & \text{if } i^{th} \text{ candidate hired} \\ 0 & \text{otherwise} \end{cases}$$

$$E(X_i) = \text{prob}\{\text{candidate } i \text{ hired}\}$$

→ i^{th} candidate is hired iff

$$C_i \xrightarrow{\text{better than}} C_1 \text{ to } C_{i-1}$$

$$\text{Prob}\{\text{candidate } i \text{ hired}\} = \frac{1}{i}$$

i.e

$$E(X_i) = \text{prob}\{C_i \text{ hired}\} = \frac{1}{i}$$

$$E(X) = E\left(\sum_{i=1}^{n} X_i\right)$$

$$= \sum_{i=1}^{n} E(X_i)$$

$$= \sum_{i=1}^{n} \frac{1}{i}$$

$$\boxed{E(X) = \log n + O(1)}$$

For +ve integer in Harmonic Series

$$H_n = 1 + \frac{1}{2} + \frac{1}{3} + \ldots\ldots + \frac{1}{n}$$

$$H_n = \sum_{i=1}^{n} \frac{1}{i} = \log n + O(1)$$

We need to hire $\log n$ out of n people interviewed.

$$\boxed{\text{Total cost} = \log n\, C_h + n\, C_i}$$

Amortized Analysis -

→ Amortized Analysis is used to average the running time of operations over a sequence of operations.

We have 3 methods
1. Aggregate Method
2. Accounting Method
3. Potential Method

1) Aggregate Method -

→ Calculate total cost of n operations and then divides by n.

Jan, Feb, Apr, May, July, August, oct, Nov - 50/-

March, June, Sept - 100/-

Dec - 200/-

$$\text{Amortized Cost} = \frac{8 \times 50 + 3 \times 100 + 200}{12} = \frac{900}{12} = 75/-$$

	Jan	Feb	Mar	Apr	May	June	July	Au	S	O	N	D
Actual cost	50	50	100	50	50	100	50	50	100	50	50	200
Amortized cost	75	75	75	75	75	75	75	75	75	75	75	75
Potential cost	25	50	25	50	75	50	75	100	75	100	125	0

Potential cost,

$$\boxed{P(i) = \text{Amortized cost}(i) - \text{Actual cost}(i) + P(i-1)}$$

$P(n) \geq 0$

$P(\text{Jan}) = 75 - 50 + 0 = 25$

$P(\text{Feb}) = 75 - 50 + 25 = 50$

$P(\text{Mar}) = 75 - 100 + 50 = 25$

interviewed.

vivo T2x 5G
Aug 11, 2025, 17:28

2> Accounting Method -

→ Assign assumed different costs (amortized costs) to operations

→ Uses space or save space.

3> potential Method -

$$P(i) = \begin{cases} 0, & n \bmod 12 = 12 \\ 25, & n \bmod 12 = 1,3 \\ 50, & n \bmod 12 = 2,4,6 \\ 75, & n \bmod 12 = 5,7,9 \\ 100, & n \bmod 12 = 8,10 \\ 125, & n \bmod 12 = 11 \end{cases}$$

$$\boxed{\text{Amortized cost}(i) = p(i) + \text{Actual cost}(i) - p(i-1)}$$

Jan = 25 + 50 = 75

Feb = 50 + 50 - 25 = 75

July = 75 + 50 - 50 = 75

vivo T2x 5G
Aug 11, 2025, 17:28

INTRODUCTION TO DIVIDE AND CONQUER ALGORITHM -

→ It is an Algorithm Design Strategy

→ It involves 3 steps

1. Divide : Divide the given problem into no. of subproblems.
2. Conquer : Solve the subproblem independently.
3. Combine : The solutions of subproblems are combined to form a solution to the original problem.

Algorithm -

```
Algorithm DAC(p)
{
    if Small(p) then return S(p);
    else
    {
        Divide p into smaller subproblems
                P1, P2, ....., Pn;
        Apply DAC to each of subproblem
        return combine(DAC(P1), DAC(P2), ....., DAC(P2)).
    }
}
```

Time Complexity -

$$T(n) = \begin{cases} T(1) & \text{is } n=1 \\ aT\left(\frac{n}{b}\right)+f(n) & n>1 \end{cases}$$

a, b are constants and f(n) is to combine solutions

Applications -

1. Linear Search
2. Merge Sort
3. Quick Sort
4. Find max & min element
5. Strassen's Algorithm for matrix multiplication

vivo T2x 5G
Aug 11, 2025, 17:28

Recursion –

→ A function that calls itself is called Recursion

→ It should have termination condition (stopping point), or it may give infinite loop

Recurrence –

→ Recurrence is an equation or inequality that describes the function in terms of its smaller value.

→ Recurrence usually used in Divide and Conquer

Example –

Recurrence for Binary Search : $T(n) = T\left(\frac{n}{2}\right) + 1$

Recurrence of QuickSort : $T(n) \leq 2T\left(\frac{n}{2}\right) + n$

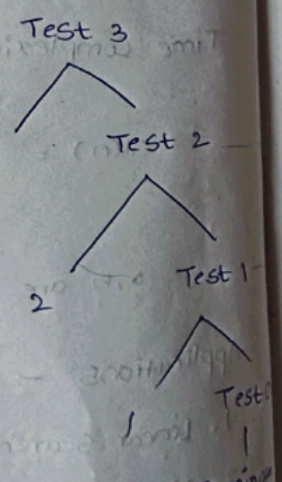
Solving Recurrence Relation –

1. Tracing Tree
2. Substitution Method

1>

```
void Test (int n)  — T(n)
{
  if (n>0){
    printf ("%d",n); — 1
    Test (n-1);  — T(n-1)
  }
}
```

For Test (3):



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

→ It call function (n+1) times

Print values n times

$f(n) = n+1$

$T(n) = O(n)$

Reccurence relation -

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

$$T(n) = \begin{cases} 1 & n = 0 \\ T(n-1)+1 & n > 0 \end{cases} \qquad T(n-1) = T(n-1-1)+1 = T(n-2)+1$$

$\rightarrow$ $T(n) = T(n-1) + 1$ $\qquad T(n-1) = T(n-2)+1$

Substitute $T(n-1)$ in $T(n)$

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

$$T(n) = T(n-2)+2 \qquad T(n-2) = T(n-3)+1$$

Substitute $T(n-2)$ in $T(n)$

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

$$T(n) = T(n-3)+3$$

$\vdots$ for k times

$$T(n) = T(n-k)+k$$

Assume $n-k = 0$ as smaller value

$$n = k$$

$$T(n) = T(0)+n$$

$$T(n) = n+1$$

$\Rightarrow$ $\theta(n)$ as it belongs to Linear class

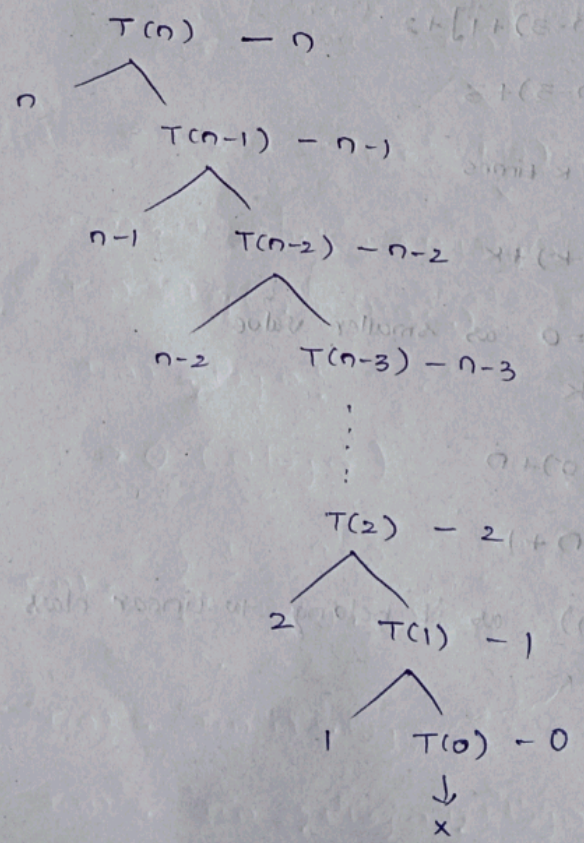
vivo T2x 5G
Aug 11, 2025, 17:29

2>

```
void Test (int n)
  {
  if (n>0)          — 1
   {
   for (i=0; i<n; i++){   — n+1
        printf ("%d", n);  — n
   }
   Test (n-1);  —  T(n-1)
  }
}
```

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

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



$$0 + 1 + 2 + \cdots + n-1 + n = \frac{n(n+1)}{2}$$

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

$$\boxed{T(n) = O(n^2)}$$

$$T(n) = \begin{cases} 1 & n=0 \\ T(n-1)+n & n>0 \end{cases}$$

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

$T(n) = [T(n-2) + n - 1] + n \quad [T(n-1) = T(n-2) + n - 1]$

$T(n) = [T(n-3) + n - 2] + (n-1) + n \quad [T(n-2) = T(n-3) + n - 2]$

$T(n) = T(n-3) + (n-2) + (n-1) + n$

$T(n) = T(n-4) + (n-3) + (n-2) + (n-1) + n$

$\vdots$

$T(n) = T(n-K) + (n-(K-1)) + (n-(K-2)) + \dots\dots + (n-1) + n$

Assume $n - K = 0$

$n = K$

$T(n) = T(0) + (n-n+1) + (n-n+2) + \dots\dots + (n-1) + n$

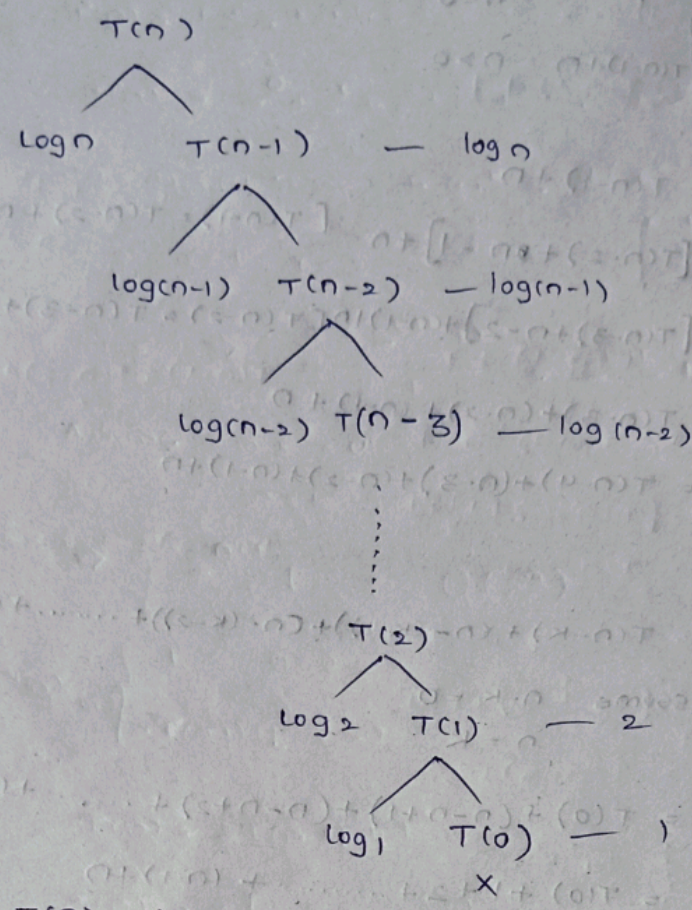
$T(n) = T(0) + 1 + 2 + \dots\dots + (n-1) + n$

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

$$\boxed{T(n) = \Theta(n^2)}$$

3>
```
void Test (int n)
{
 if(n>0){        — 1
    for (i=1;i<n;i=i*2)  — logn +1
      {
      printf("%d",i);  — logn
      }
     Test(n-1);  — T(n-1)
 }
}
```

$T(n) = T(n-1) + \log n$



$$T(n) = \log n + \log(n-1) + \log(n-2) + \dots\dots + 2 + 1$$

$$T(n) = \log(n(n-1)(n-2)\dots\dots 2 \cdot 1)$$

$$T(n) = \log n!$$

$$\Rightarrow O(n \log n)$$

$$T(n) = \begin{cases} 1 & n = 0 \\ T(n-1) + \log n & n > 0 \end{cases}$$

$$T(n) = T(n-1) + \log n$$

$$T(n) = T(n-2) + \log(n-1) + \log n$$

$$T(n) = T(n-3) + \log(n-2) + \log(n-1) + \log n$$

$$\vdots$$

$$T(n) = T(n-K) + \log(n-(K-1)) + \dots\dots + \log(n-1) + \log n$$

vivo T2x 5G
Aug 11, 2025, 17:29

Assume $n-k=0$

$n=k$

$\Rightarrow T(n) = T(n-n) + \log(n-n+1) + \log(n-n+2) + \cdots$

$\cdots\cdots + \log(n-1) + \log n$

$T(n) = T(0) + \log(1) + \log(2) + \cdots\cdots + \log(n-1) + \log n$

$T(n) = 1 + \log(n(n-1)(n-2)\cdots\cdots 2.1)$

$T(n) = 1 + \log n!$

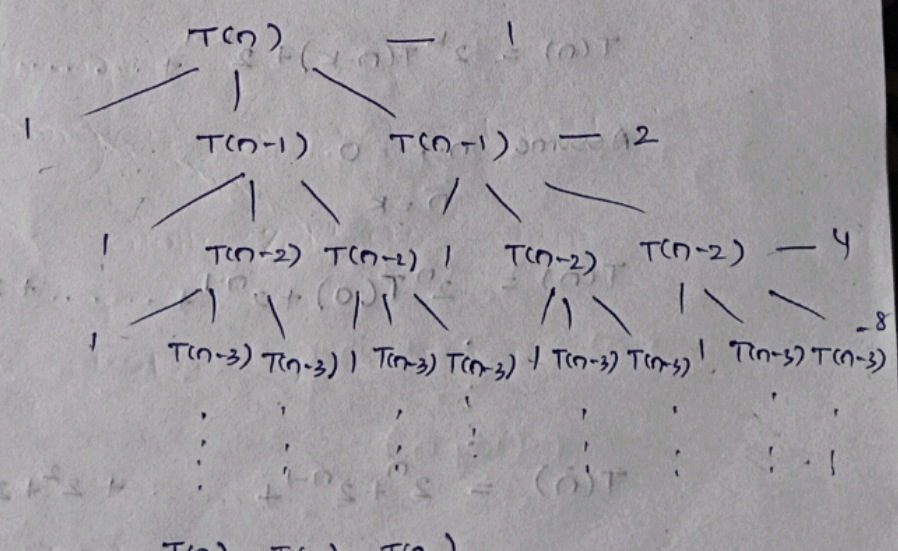
$T(n) = 1 + \log n!$

$\Rightarrow \Theta(n \log n)$

4) Algorithm

```
Test (int n){
    if (n>0){
        printf ("%d", n); — 1
        Test (n-1); — T(n-1)
        Test (n-1); — T(n-1)
    }
}
```

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



vivo T2x 5G
Aug 11, 2025, 17:29

$$T(n) = 1 + 2 + 4 + 8 + \ldots\ldots$$

$$T(n) = 2^0 + 2^1 + 2^2 + 2^3 + \ldots\ldots + 2^k$$

$$a + ar + ar^2 + \ldots\ldots + ar^k = \frac{a(r^{k+1} - 1)}{r-1}$$

$$a = 1 \quad r = 2$$

$$\Rightarrow \frac{1(2^{k+1} - 1)}{2-1} = 2^{k+1} - 1$$

Assume $n - k = 0$

$$n = k$$

$$\Rightarrow 2^{n+1} - 1 = O(2^n)$$

$$T(n) = \begin{cases} 1 & n = 0 \\ 2T(n-1) + 1 & n > 0 \end{cases}$$

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

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

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

$$T(n) = 2^3 T(n-3) + 2^2 + 2 + 1$$

$$\vdots$$

$$T(n) = 2^k T(n-k) + 2^{k-1} + \ldots\ldots + 2^2 + 2^1 + 2^0$$

Assume $n - k = 0$

$$n = k$$

$$T(n) = 2^n T(0) + 2^{n-1} + \ldots\ldots + 2^2 + 2^1 + 2^0$$

$$\vdots$$

$$T(n) = 2^n + 2^{n-1} + \ldots\ldots + 2^2 + 2^1 + 2^0$$

$$T(n) = 2^{n+1} - 1 \Rightarrow \Theta(2^n)$$

Solving

$\rightarrow T(n)$

$T(n$

$\rightarrow$

vivo T2x 5G
Aug 11, 2025, 17:29

Solving Recurrances - Substitution

$$\rightarrow T(n) = \begin{cases} 1 & \text{if } n=1 \\ T(n-1)+n & \text{, otherwise} \end{cases}$$

$$\begin{aligned} T(n) &= T(n-1)+n \\ &= (T(n-2)+(n-1))+n \\ &= T(n-2)+(n-1)+n \\ &= T(n-3)+(n-2)+(n-1)+n \\ &= T(n-4)+(n-3)+(n-2)+(n-1)+n \\ &\vdots \end{aligned}$$

$$T(n) = T(n-k)+(n-(k-1))+(n-(k-2))+\dots\dots+n$$

Assume $n-k=1$

$k=n-1$

$$\begin{aligned} T(n) &= T(n-(n-1)+(n-(n-2))+(n-(n-3))+\dots\dots+(n-1) \\ &\quad +n \\ &= T(1)+2+3+\dots\dots+(n-1)+n \\ &= 1+2+3+\dots\dots+(n-1)+n \\ &= \frac{n(n+1)}{2} = \frac{n^2}{2}+\frac{n}{2} = O(n^2) \end{aligned}$$

$$\rightarrow T(n) = \begin{cases} n * T(n-1) & \text{otherwise} \\ 1 & \text{if } n \le 1 \end{cases}$$

$$\begin{aligned} T(n) &= n * T(n-1) \\ &= n \times [(n-1) \times T(n-2)] \\ &= n \times (n-1) \times [(n-2)\, T(n-3)] \\ &= n(n-1)(n-2)(n-3)\, T(n-4) \\ &= n(n-1)(n-2)(n-3)(n-4)\, T(n-5) \\ &\vdots \end{aligned}$$

$$T(n) = n(n-1)(n-2)(n-3)\dots\dots\dots(n-(k-1))\, T(n-k)$$

vivo T2x 5G
Aug 11, 2025, 17:30

$n-k = 1$

$k = n-1$

$T(n) = n(n-1)(n-2) \ldots\ldots (n-(n-2))\,T(n-(n-1))$

$= n(n-1)(n-2) \ldots\ldots (2)T(1)$

$= n(n-1)(n-2) \ldots\ldots 2 \times 1$

$T(n) = 1 \times 2 \times 3 \times \ldots\ldots \times (n-2)(n-1)n = n! = O(n!)$

$\cancel{= \frac{n(n+1)}{2} = \frac{n^2+n}{2} = O(n^2)}$

$\rightarrow \quad T(n) = \begin{cases} T(n/2) + c & n > 1 \\ 1 & \text{if } n \le 1 \end{cases}$

$T(n) = T(n/2) + c$

$= T\left(\frac{n}{2^2}\right) + c + c$

$= T\left(\frac{n}{2^3}\right) + c + c + c$

$\vdots$

$T(n) = T\left(\frac{n}{2^k}\right) + k \cdot c$

Assume $\frac{n}{2^k} = 1$

$n = 2^k$

$k = \log_2 n$

$T(n) = T\left(\frac{n}{2^{\log n}}\right) + \log n \cdot c$

$= T\left(\frac{n}{n}\right) + \log n \cdot c$

$T(n) = 1 + \log n \cdot c$

$O(\log n)$

$$\rightarrow \quad T(n) = \begin{cases} 2T\left(\frac{n}{2}\right) + n & \text{if } n > 1 \\ 1 & \text{if } n \le 1 \end{cases}$$

$$T(n) = 2T\left(\frac{n}{2}\right) + n$$

$$= 2\left(2T\left(\frac{n}{4}\right) + \frac{n}{2}\right) + n$$

$$= 2^2 T\left(\frac{n}{4}\right) + \frac{2n}{2} + n$$

$$= 2^2\left(2T\left(\frac{n}{8}\right) + \frac{n}{4}\right) + \frac{2n}{2} + n$$

$$= 2^3 T\left(\frac{n}{8}\right) + n + n + n$$

$$\vdots$$

$$T(n) = 2^k T\left(\frac{n}{2^k}\right) + k \cdot n$$

Assume $\frac{n}{2^k} = 1$

$$n = 2^k$$

$$k = \log n$$

$$T(n) = n\,T\left(\frac{n}{n}\right) + \log n \cdot n$$

$$= n\,T(1) + \log n\,(n)$$

$$= n + n\log n \quad = O(n \log n)$$

$$\rightarrow \quad T(n) = \begin{cases} 2T\left(\frac{n}{2}\right) + 2 & \text{if } n > 2 \\ 1 & \text{if } n \le 2 \end{cases}$$

$$T(n) = 2T\left(\frac{n}{2}\right) + 2$$

$$= 2\left(2T\left(\frac{n}{4}\right) + 2\right) + 2$$

$$= 2^2 T\left(\frac{n}{4}\right) + 2^2 + 2$$

$$= 2^2\left(2T\left(\frac{n}{8}\right) + 2\right) + 2^2 + 2$$

$$= 2^3 T\left(\frac{n}{2^3}\right) + 2^3 + 2^2 + 2$$

$$\vdots$$

$$= 2^k T\left(\frac{n}{2^k}\right) + 2^k + 2^{k-1} + 2^{k-2} + \dots + 2^2 + 2$$

vivo T2x 5G
Aug 11, 2025, 17:30

$$= 2^k T\left(\frac{n}{2^k}\right) + \frac{2^k - 1}{2-1}$$

$$= 2^k T\left(\frac{n}{2^k}\right) + (2^k - 1)$$

Assume

$$\frac{n}{2^k} = 2$$

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

$$k+1 = \log n$$

$$k = \log n - 1$$

$$T(n) = \frac{n}{2} T\left(\frac{\frac{n}{2}}{2}\right) + \left(\frac{n}{2} - 1\right)$$

$$= \frac{n}{2} T(2) + \left(\frac{n}{2} - 1\right)$$

$$= \frac{n}{2} + \frac{n}{2} - 1 = \frac{2n}{2} - 1 = n - 1$$

$$T(n) = O(n)$$

$$\rightarrow \quad T(n) = \begin{cases} 8T\left(\frac{n}{2}\right) + n^2 & n > 2 \\ b & \text{if } n \le 2 \end{cases}$$

$$T(n) = 8T\left(\frac{n}{2}\right) + n^2$$

$$= 8\left(8T\left(\frac{n}{4}\right) + \left(\frac{n}{2}\right)^2\right) + n^2$$

$$= 8^2 T\left(\frac{n}{4}\right) + 8\left(\frac{n}{2}\right)^2 + n^2$$

$$= 8^2\left(8T\left(\frac{n}{8}\right) + \left(\frac{n}{4}\right)^2\right) + 8n^2 + n^2$$

$$= 8^3 T\left(\frac{n}{8}\right) + 8^2\left(\frac{n}{4}\right)^2 + 8\left(\frac{n}{2}\right)^2 + n^2$$

$$\vdots$$

$$= 8^k T\left(\frac{n}{2^k}\right) + 8^{k-1}\left(\frac{n}{2^{k-1}}\right)^2 + 8^{k-2}\left(\frac{n}{2^{k-2}}\right)^2 + 8^{k-3}\left(\frac{n}{2^{k-3}}\right)^2 \cdots$$

$$\cdots + 8^2\left(\frac{n}{4}\right)^2 + 8\left(\frac{n}{2}\right)^2 + n^2$$

$$= 8^k T\left(\frac{n}{2^k}\right) + n^2\left(\frac{8^{k-1}}{4^{k-1}} + \frac{8^{k-2}}{4^{k-2}} + \frac{8^{k-3}}{4^{k-3}} + \cdots\right.$$

$$\left. \cdots + \frac{8^2}{4^2} + \frac{8^1}{4} + 1\right)$$

$$= 8^k T\left(\frac{n}{2^k}\right) + n^2\left(2^{k-1} + 2^{k-2} + 2^{k-3} + \dots + \right.$$

$$= 8^k T\left(\frac{n}{2^k}\right) + n^2\left(2^{k-1} - 1\right)$$

Assume, $\frac{n}{2^k} = 2$ $\quad k = 1$

$$2^k = \frac{n}{2}$$

$$k+1 = \log\left(\frac{n}{2}\right)$$

$$k = \log n - 1$$

$$= 2^{3\log n - 3} T\left(\frac{n}{2}\right) + n^2\left(\frac{n}{2}\cdot\frac{1}{2} - 1\right)$$

$$= (2^k)^3 T(2) + n^2\left(\frac{n}{4} - 1\right)$$

$$= \left(\frac{n}{2}\right)^3 6 + n^2\left(\frac{n}{4} - 1\right) = \frac{6n^3}{8} + \frac{n^3}{4} - n^2$$

$$T(n) = O(n^3)$$

$\rightarrow$ $T(n) = T\left(\frac{n}{3}\right) + n$

$$T(n) = T\left(\frac{n}{3}\right) + n$$

$$= T\left(\frac{n}{3^2}\right) + \frac{n}{3} + n$$

$$= T\left(\frac{n}{3^3}\right) + \frac{n}{3^2} + \frac{n}{3} + n$$

$$\vdots$$

$$= T\left(\frac{n}{3^k}\right) + n\left(\frac{1}{3^{k-1}} + \frac{1}{3^{k-2}} + \dots + \frac{1}{3^2} + \frac{1}{3^1} + \frac{1}{3^0}\right)$$

$$\frac{n}{3^k} = 1$$

$$3^k = n$$

$$k = \log_3 n$$

$$= T\left(\frac{n}{3^k}\right) + n\cdot\frac{3}{2}\left(1 - \frac{1}{3^{k-1}}\right)$$

$$= T(1) + n\cdot\frac{3}{2}\left(1 - \left(\frac{1}{3}\right)^{\log_3 n - 1}\right)$$

$$= T(1) + n\cdot\frac{3}{2}\left(1 - \frac{3}{n}\right)$$

$$= 1 + \frac{3n}{2} - \frac{9}{2} \rightarrow O(n)$$

$$\rightarrow \quad T(n) = \begin{cases} 7T\left(\frac{n}{2}\right) + n^2 & n > 2 \\ b & n \le 2 \end{cases}$$

$$T(n) = 7T\left(\frac{n}{2}\right) + n^2$$

$$= 7\left(7T\left(\frac{n}{4}\right) + \left(\frac{n}{2}\right)^2\right) + n^2$$

$$= 7^2 T\left(\frac{n}{2^2}\right) + 7\left(\frac{n}{2}\right)^2 + n^2$$

$$= 7^2\left(7T\left(\frac{n}{2^3}\right) + \left(\frac{n}{2^2}\right)^2\right) + 7\left(\frac{n}{2}\right)^2 + n^2$$

$$= 7^3 T\left(\frac{n}{2^3}\right) + 7^2\left(\frac{n}{2^2}\right)^2 + 7\left(\frac{n}{2}\right)^2 + n^2$$

$$\vdots$$

$$= 7^K T\left(\frac{n}{2^K}\right) + 7^{K-1}\left(\frac{n}{2^{K-1}}\right)^2 + 7^{K-2}\left(\frac{n}{2^{K-2}}\right)^2 + \cdots$$

$$\cdots\cdots 7^2\left(\frac{n}{2^2}\right)^2 + 7\left(\frac{n}{2}\right)^2 + n^2$$

$$= 7^K T\left(\frac{n}{2^K}\right) + n^2\left(\frac{7^{K-1}}{4^{K-1}} + \frac{7^{K-2}}{4^{K-2}} + \cdots + \frac{7^2}{4^2} + \frac{7}{4} + 1\right)$$

$$= 7^K T\left(\frac{n}{2^K}\right) + n^2\left(1 + \frac{7}{4} + \frac{7^2}{4^2} + \cdots + \frac{7^{K-2}}{4^{K-2}} + \frac{7^{K-1}}{4^{K-1}}\right)$$

$$= 7^K T\left(\frac{n}{2^K}\right) + n^2\left(\left(\frac{7}{4}\right)^0 + \left(\frac{7}{4}\right)^1 + \left(\frac{7}{4}\right)^2 + \cdots + \left(\frac{7}{4}\right)^{K-2} + \left(\frac{7}{4}\right)^{K-1}\right)$$

$$T(n) = 7^K T\left(\frac{n}{2^K}\right) + n^2 \frac{4}{3}\left(\left(\frac{7}{4}\right)^{K-1} - 1\right)$$

Assume $\frac{n}{2^K} = 2$

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

$$K = \log n - 1$$

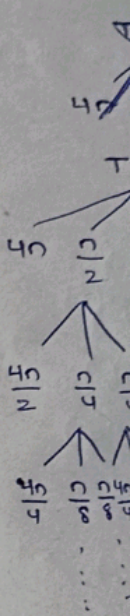
$$T(n) = 7^{\log n - 1} T(2) + n^2 \frac{4}{3}\left(\left(\frac{7}{4}\right)^{\log n - 2} - 1\right)$$

$$= \frac{7^{\log n}}{7} b + n^2 \frac{4}{3} \frac{7^{\log n - 2}}{4^{\log n - 2}} - n^2 \frac{4}{3}$$

$$= \frac{b\, 7^{\log n}}{7} + \frac{4n^2}{3} \frac{4^2}{7^2} \frac{7^{\log n}}{4^{\log n}} - \frac{4n^2}{3} \rightarrow O(n^2)$$

Tree Method -

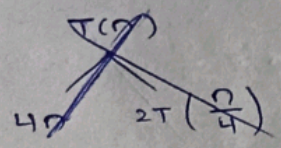
(1) $T(1) = 4$



vivo T2x 5G
Aug 11, 2025, 17:30

Tree Method -

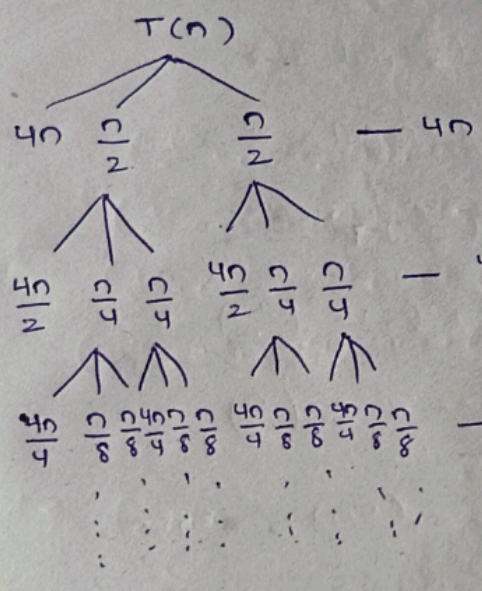
(1) $T(1) = 4 \quad T(n) = 2T\left(\frac{n}{2}\right) + 4n$



$\frac{n}{2^k} = 1$

$2^k = n$

$k = \log_2 n$



$k \cdot 4n = \log_2 n \cdot 4n$

$O(n \log n)$

STRASSEN'S MATRIX MULTIPLICATION -

→ It is an application of Divide & conquer Stra

1. conventional (or) traditional (or) Standard $= O(n^3)$
2. Divide & conquer strategy $= O(n^3)$
3. Strassen's Matrix Multiplication $= O(n^{2.81})$

1. Conventional method of matrix multiplication

$$\overset{A}{\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}} \times \overset{B}{\begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}} = \overset{C}{\begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix}}$$

$C_{11} = A_{11}B_{11} + A_{12}B_{21}$

$C_{12} = A_{11}B_{12} + A_{12}B_{22}$

$C_{21} = A_{21}B_{11} + A_{22}B_{21}$

$C_{22} = A_{21}B_{12} + A_{22}B_{22}$

```
for (i=0; i<n; i++){
    for (j=0; j<n; j++){
        c[i,j] = 0;
        for (k=0; k<n; k++){
            c[i,j] += a[i,k] * b[k,j];
        }
    }
}
```

$$C(i,j) = \sum_{1 \le k \le n} A(i,k) * B(k,j)$$

No. of multiplications $= 8$ $\qquad \log_b^a = \log_2^2 = 3$

No. of additions $= 4$

Time complexity $= O(n^3)$

2. Divide and conquer Stragey of Matrix Multiplication

Here 4x4 matrix is divided into 4 2x2 matrices

$$\overset{A}{\begin{array}{c} \begin{array}{cc} A_{11} & A_{12} \end{array} \\ \left[\begin{array}{cc|cc} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ \hline a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{array}\right] \\ \begin{array}{cc} A_{21} & A_{22} \end{array} \end{array}} \quad \overset{B}{\begin{array}{c} \begin{array}{cc} B_{11} & B_{12} \end{array} \\ \left[\begin{array}{cc|cc} b_{11} & b_{12} & b_{13} & b_{14} \\ b_{21} & b_{22} & b_{23} & b_{24} \\ \hline b_{31} & b_{32} & b_{33} & b_{34} \\ b_{41} & b_{42} & b_{43} & b_{44} \end{array}\right] \\ \begin{array}{cc} B_{21} & B_{22} \end{array} \end{array}}$$

vivo T2x 5G
Aug 11, 2025, 17:30

```
Algorithm MM(A, B, n){
    if (n ≤ 2){
        C11 = A11 × B11 + A12 B21
        C12 = A11 B12 + A12 B22
        C21 = A21 B11 + A22 B21
        C22 = A21 B12 + A22 B22
    }
    else {
        MM(A11, B11, n/2) + MM(A12, B21, n/2)
        MM(A11, B12, n/2) + MM(A12, B22, n/2)
        MM(A21, B11, n/2) + MM(A22, B21, n/2)
        MM(A21, B12, n/2) + MM(A22, B22, n/2)
    }
}
```

$$T(n) = \begin{cases} 1 & \text{if } n \le 2 \\ 8T\left(\frac{n}{2}\right) + n^2 & \text{if } n > 2 \end{cases}$$

$aT\left(\frac{n}{b}\right) + f(n) \rightarrow \log_b^a$

$a = 8$
$b = 2$

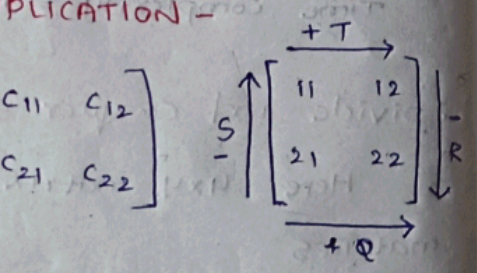
$f(n) = n^2$

$\log_b^a = \log_2^8 = 3\log_2^2 = 3$

$O(n^3)$

STRASSEN'S MATRIX MULTIPLICATION –

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix} = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix}$$



$B_{11}\ A_{11}\ A_{22}\ B_{22}$

$P = (A_{11} + A_{22})(B_{11} + B_{22})$

$Q = B_{11}(A_{21} + A_{22})$

$R = A_{11}(B_{12} - B_{22})$

$S = A_{22}(B_{21} - B_{11})$

$T = B_{22}(A_{11} + A_{12})$

$U = (A_{21} - A_{11})(B_{11} + B_{12})$

$V = (A_{12} \ldots$

$C_{11} = P + S - T + V$

$C_{12} = R + T$

$C_{21} = Q + S$

$C_{22} = P + R - Q + U$

No. of multiplication = 7

Additions = 18

vivo T2x 5G
Aug 11, 2025, 17:31

$$T(n) = \begin{cases} 1 & \text{if } n \le 2 \\ 7T\left(\frac{n}{2}\right) + n^2 & \text{if } n > 2 \end{cases}$$

$$aT\left(\frac{n}{b}\right) + f(n) \rightarrow \log_b^a$$

$a = 7$
$b = 2$
$f(n) = n^2$

$\log_2^7 = 2.81 \rightarrow O(n^{2.81})$

BINARY SEARCH -

→ We apply binary search when any elements are in sorted order

	low ↓							high ↓
	0	1	2	3	4	5	6	7
	10	15	18	30	50	70	90	100

First calculate Array's middle element

mid = (low+high)/2

1. Compare key ele with Array's middle ele
   If key == a[mid] then key ele found

2. If key < a[mid] then search only left part. so update high value
   high = mid - 1

3. If key > a[mid] then search only right part. so update low value
   low = mid + 1

Algorithm -

In/p - Array size (n), Array elements, key element

O/p - key ele is found or not found

1. flag = 0
2. low = 0
3. high = n - 1
4. while low <= high do
   4.1 mid = (low+high)/2
   4.2 if key == a[mid] then
      a. flag = 1
      b break
   4.3 if key < a[mid] then
      a. high = mid - 1

vivo T2x 5G
Aug 11, 2025, 17:31

4.4 if key > a[mid] then

a. low = mid + 1

5 if flag == 1 then

a. print "key ele found"

6. else

b. print "key ele not found"

Time Complexity of binary search :

1. Best case Time complexity : $T(n) = O(1)$

when key is exactly middle element.

2. Worst case Time complexity :

when key is first or last element

$$T(n) = T\left(\frac{n}{2}\right) + 1$$

$$= T\left(\frac{n}{4}\right) + 1 + 1$$

$$= T\left(\frac{n}{8}\right) + 1 + 1 + 1$$

$$\vdots$$

$$T(n) = T\left(\frac{n}{2^K}\right) + K$$

let $2^K = n$ $\quad \frac{n}{2^K} = 1$

$K = \log n$

$$T(n) = T\left(\frac{n}{n}\right) + \log n = T(1) + \log n$$

$$T(n) = 1 + \log n \rightarrow O(\log n)$$

3. Averge case of Time complexity : $O(\log n)$

vivo T2x 5G
Aug 11, 2025, 17:31

MASTER'S THEORM -

Decreasing Function :

For functions of form -

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

where $a > 0$, $b > 0$, $k \geq 0$

case-1 : If $a < 1$ then $T(n) = \Theta(n^k)$

case-2 : If $a = 1$ then $T(n) = \Theta(n^{k+1})$

case-3 : If $a > 1$ then $T(n) = \Theta(a^{n/b} * f(n))$

* $T(n) = 2T(n-2) + n$

$a = 2 \quad b = 2 \quad f(n) = n \quad k = 1$

$a > 1$

$$\cancel{T(n) = \Theta(n^k) \times \Theta(n)}$$

$$T(n) = \Theta(a^{n/b} f(n)) = \Theta(2^{n/2} n)$$

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

$a = \frac{1}{2} \quad b = 1 \quad k = 2$

$a < 1$

$$T(n) = \Theta(n^k) = \Theta(n^2)$$

* $T(n) = T(n-1) + n$

$a = 1, \; b = 1, \; k = 1$

$a = 1$

$$T(n) = \Theta(n^{k+1}) = \Theta(n^2)$$

vivo T2x 5G
Aug 11, 2025, 17:31

Dividing Function -

Function of form -

$$T(n) = aT\left(\frac{n}{b}\right) + f(n) \qquad f(n) = \Theta\left(n^k \log^p n\right)$$

where $a>1, b>1, k \geq 0$, p is a real number

1. If $a > b^k$ then $T(n) = \Theta\left(n^{\log_b^a}\right)$
2. If $a = b^k$ then

   if $p > -1$ then $T(n) = \Theta\left(n^{\log_b^a} \log^{p+1} n\right)$

   if $p = -1$ then $T(n) = \Theta\left(n^{\log_b^a} \log\log n\right)$

   if $p < -1$ then $T(n) = \Theta\left(n^{\log_b^a}\right)$
3. If $a < b^k$ then

   (i) $p \geq 0$ then $T(n) = \Theta\left(n^k \log^p n\right)$

   (ii) $p < 0$ then $T(n) = \Theta\left(n^k\right)$

* $T(n) = 2T\left(\frac{n}{2}\right) + n$

$a = 2 \quad b = 2 \quad k = 1 \quad p = 0$

$a = b^k, \; p = 0$

$$T(n) = \Theta\left(n^{\log_b^a} \log^{p+1} n\right)$$

$$= \Theta\left(n^{\log_2^2} \log n\right)$$

$$T(n) = \Theta(n \log n)$$

* $T(n) = 8T\left(\frac{n}{2}\right) + n^2$

$a = 8 \quad b = 2 \quad k = 2 \quad p = 0$

$a > b^k, \; p = 0$

$$T(n) = \Theta\left(n^{\log_b^a}\right)$$

$$T(n) = \Theta\left(n^3\right).$$

vivo T2x 5G

Aug 11, 2025, 17:31

* $T(n) = T\left(\frac{n}{4}\right) + \log n$

$a = 1 \quad b = 4 \quad k = 0 \quad p = 1$

$a = b^k \quad p = 1$

$T(n) = \Theta\left(n^{\log_b^a} \log^{p+1} n\right)$

$T(n) = \Theta\left(n^{\log_4^1} \log^2 n\right)$

$T(n) = \Theta\left(\log^2 n\right)$

* $T(n) = 3T\left(\frac{n}{3}\right) + \frac{n}{\log n}$

$a = 3 \quad b = 3 \quad k = 1 \quad p = -1$

$a = b^k \quad p = -1$

$T(n) = \Theta\left(n^{\log_b^a} \log\log n\right)$

$T(n) = \Theta\left(n \log\log n\right)$

* $T(n) = 5T\left(\frac{n}{2}\right) + \frac{n^2}{\log n}$

$a = 5 \quad b = 2 \quad k = 2 \quad p = -1$

$a > b^k$

$T(n) = \Theta\left(n^{\log_b^a}\right)$

$T(n) = \Theta\left(n^{\log_2^5}\right)$

* $T(n) = 4T\left(\frac{n}{2}\right) + \log n$

$a = 4, b = 2, k = 0, p = 1$

$a > b^k$

$T(n) = \Theta\left(n^{\log_b^a}\right)$

$T(n) = \Theta\left(n^{\log_2^4}\right)$

$T(n) = \Theta\left(n^2\right)$

* $T(n) = 7T\left(\frac{n}{3}\right) + n^2$

$a = 7, \ b = 3, \ k = 2, \ p = 0$

$a < b^k, \ p = 0$

$T(n) = \Theta\left(n^k \log^p n\right)$

$T(n) = \Theta\left(n^2 \log^0 n\right)$

$T(n) = \Theta\left(n^2\right)$

vivo T2x 5G
Aug 11, 2025, 17:31

Analysis Of Merge Sort

Best case -

$$T(n) = 1$$

$$T(n) = 2T\left(\frac{n}{2}\right) + n$$

where $2T\left(\frac{n}{2}\right)$ is dividing array into 2 parts and n is comparing elements n times

$$T(n) = 2\left(2T\left(\frac{n}{4}\right)\right) + \frac{2n}{2} + n$$

$$= 2\left(2T\left(\frac{n}{8}\right)\right) + \frac{4n}{4} + 2n$$

$$= 8T\left(\frac{n}{8}\right) + 3n$$

$$= 8\left(2T\left(\frac{n}{16}\right) + \frac{n}{8}\right) + 3n$$

$$T(n) = 16T\left(\frac{n}{16}\right) + 4n$$

$$\frac{n}{2^k} = 1$$

$$T(n) = 2^k T\left(\frac{n}{2^k}\right) + kn$$

$$n = 2^k$$

$$k = \log n$$

$$T(n) = nT(1) + n\log n$$

$$T(n) = n + n\log n = O(n\log n)$$

In all cases order of Merge Sort $= O(n\log n)$

vivo T2x 5G
Aug 11, 2025, 17:31

ANALYSIS OF QUICK SORT

Best / Average case -

$$T(n) = 1$$

$$T(n) = 2T\left(\frac{n}{2}\right) + n$$

$$= 2\left(2T\left(\frac{n}{4}\right) + \frac{n}{2}\right) + n$$

$$= 4T\left(\frac{n}{4}\right) + n + n$$

$$= 4\left(2T\left(\frac{n}{8}\right) + \frac{n}{4}\right) + n + n$$

$$= 8T\left(\frac{n}{8}\right) + 3n = 2^3 T\left(\frac{n}{2^3}\right) + 3n$$

$$\vdots$$

$$T(n) = 2^k T\left(\frac{n}{2^k}\right) + kn$$

$$\frac{n}{2^k} = 1$$

$$n = 2^k$$

$$k = \log n$$

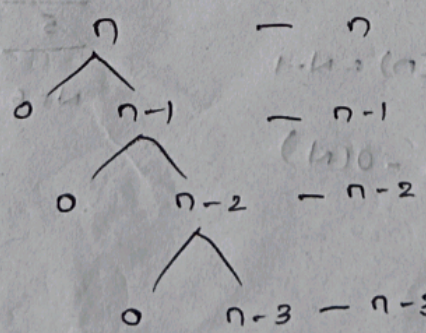
$$T(n) = nT(1) + \log(n)(n)$$

$$T(n) = n + n\log n \longrightarrow O(n\log n)$$

Worst case -

10 20 30 40 50

P P



$\rightarrow O(n^2)$

$$\rightarrow n + (n-1) + (n-3) + \cdots\cdots + 3 + 2 + 1 = \frac{n(n+1)}{2} = \frac{n^2+n}{2}$$

vivo T2x 5G
Aug 11, 2025, 17:32

ANALYSIS OF INSERTION SORT

```
for (int i=1 ; i<N ; i++)
  {
    temp = A[i] ;
    j = i-1 ;
    while (j>=0 && a[j]>temp)
      {
        A[j+1] = A[j] ;
        j-- ;
      }
    A[j+1] = temp ;
  }
```

j	temp				
5	4	10	1	6	2
		i	temp		
4	5	10	1	6	2
			i	i	
1	4	5	10	6	2
					i
1	4	5	6	10	2
1	2	4	5	6	10

Best case –

Array is already sorted

						Comparisons
1	2	4	5	6	10	0
1	2	4	5	6	10	1
1	2	4	5	6	10	1
1	2	4	5	6	10	1
1	2	4	5	6	10	1
1	2	4	5	6	10	1
						5 (N-1)

$T(n) = N-1$

$N = 6$

Best case - $O(N)$

Worst case / Average case –

Array elements are reversly sorted

						Comparisons
10	6	5	4	2	1	0
6	10	5	4	2	1	1
5	6	10	4	2	1	2
4	5	6	10	2	1	3
2	4	5	6	10	1	4
1	2	4	5	6	10	5

No. of comparisons $= 1+2+3+4+5$

$= 1+2+3+\cdots+(N-1)$

$= \frac{(N-1)(N-1+1)}{2} = \frac{N(N-1)}{2}$

$= \frac{N^2-N}{2}$

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

$\rightarrow O(N^2)$

vivo T2x 5G
Aug 11, 2025, 17:32