I) MAX-MIN problem :-

A:	1	2	3	4	5	6	7	8	9
	12	8	-9	3	0	-6	14	20	5

procedure straight MinMax (A, N, Min, Max)

{
1. max ← min ← A[i];
2. for i ← 2 to n
{
a. if (A[i] > max_i)
max_i = A[i].
else
b. if (A[i] < min_i)
min = A[i];
}
}

Comparison
- Index
- Element

$f(n) = 2(n-1)$
$= 2n - 2$

1) Total no. of element comparison = $2(n-1)$ $O(n)$

2) Total no. of comp

Best case : $(n-1)$ incr/Asc $f(n) = n-1$ $O(n)$

Worst case : $2(n-1)$ DSC/Decr $f(n) = 2(n-1)$ $O(n)$

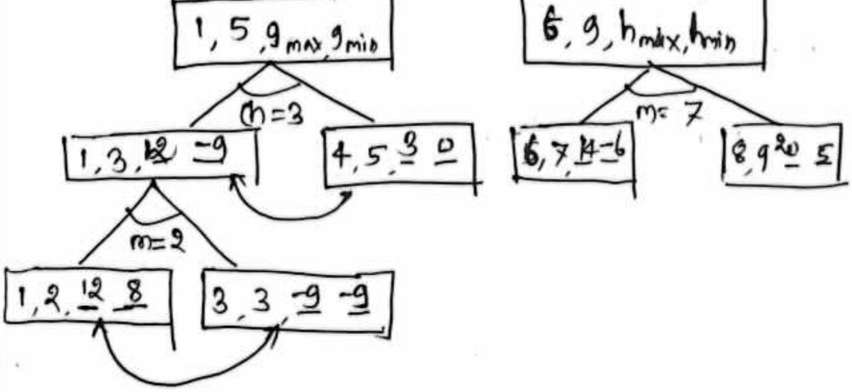
Avg case : Assume that first comparison fails on an avg. for half of given elements

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

$\left(\frac{3n}{2} - 1\right)$

$1 + 9 = 10/2 = 5$

II. DANDC-MAX-MIN :



Let $T(n)$ represent the total no. of element comparisons involved in DANDC-MAXMIN(n); $=2$

$T(n) = a \cdot T(n/b) + f(n)$

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

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

$T(n) = 2T(n/2)+2 \quad -①$

$n = n/2$

$T(n/2) = 2T(n/4)+2 \quad -②$

$T(n/2) = 2 \cdot T(n/4) + 2$

$T(n) = 2 \cdot (2T(n/4)+2) + 2$

$T(n) = 4T(n/4)+4$

$T(n/4) = 2T(n/8)+2$

$4+2$

$\underline{6}$

$= 2^k T(n/2^k) + 2^k$

$= 8T(n/8) + 12 = 2^k T(n/2^k) + 2^1+2^2+2^3+\dots+2^{k-1}$

$\therefore$ $= 2^k T(1) + 2^k$

$T(n/4) = 2 \cdot T(n/8)+2$

$T(n) = 2[4 \cdot T(n/8)+2]+2$

$= 8T(n/8) + k + c$

$2^k T(n/2^k) +$

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

$n = 2^k$

$\underline{k = \log n}$

$O(n)$

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

$T(n) = 4T(n/4) + 6$

$= 2^2 T(n/2^2) + (4+2)$

$= 2^k T(n/2^k) + \sum_{i=1}^{k} 2^i \quad -①$

$= 2^k T(2) + \sum_{i=1}^{k} 2^i$

$2 \cdot 2^k$

2
4
8

2
6
14

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

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

$\log n = k+1$

$\therefore \boxed{k = \log n - 1}$

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

$=$

$\therefore \sum_{i=1}^{k} 2^i = \frac{a(r^n-1)}{r-1} = \frac{2(2^k-1)}{2-1} = 2^{k+1}-2$

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

$n > 2$.

Space Req:

$S(n) = c_2 + (n + \log n)$

$\frac{3n}{2} - 1$

Q. $T(n) = 2 \cdot T(n/2) + 3$

Q. $T(1) = 1$
$T(n+1) = T(n) + \lfloor \sqrt{n+1} \rfloor$
Value of Recurre for $T(m^2)$

* Merge Sort :-

	1	2	3	4	5	6	7	8	9	10
A: <	310	285	179	652	351	423	861	254	450	520 >

B:

Q. $n_1 = 4$ $\quad$ $L_1 : \{2, 4, 6, 10\}$ $\qquad$ $n_2 = 6$ $\quad$ $L_2 = \{3, 5, 9, 15, 20, 25\}$

$n_1 \leq n_2$
$n = n_1 + n_2$

I. Best case:-

$L_1 : \{2\ 5\ 8\ 10\}$ $\qquad$ $L_2 : \{12, 15, 18, 20\ 25\ 30\}$

Best case = 4 comparisons.

$B.C. = n_1$

II. worst case:-

$n_1 = 5$ $\quad$ $L_1 : \{2, 4, 8, 10, 40\}$ $\qquad$ $n_2 = 6$ $\quad$ $L_2 : \{12, 14, 18, 20, 25, 30\}$

$$W.C. = (n_1 - 1) + n_2$$
$$= (n_1 + n_2) - 1$$
$$= (n-1) \text{ comparisons.}$$

n_1

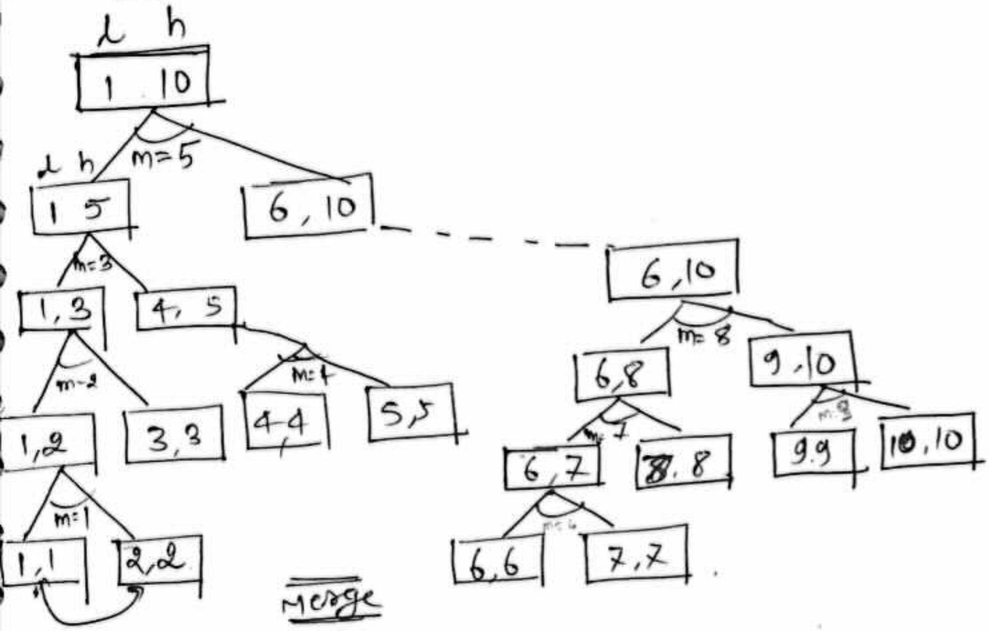
$T(n) = O(n)$

Merge sort :

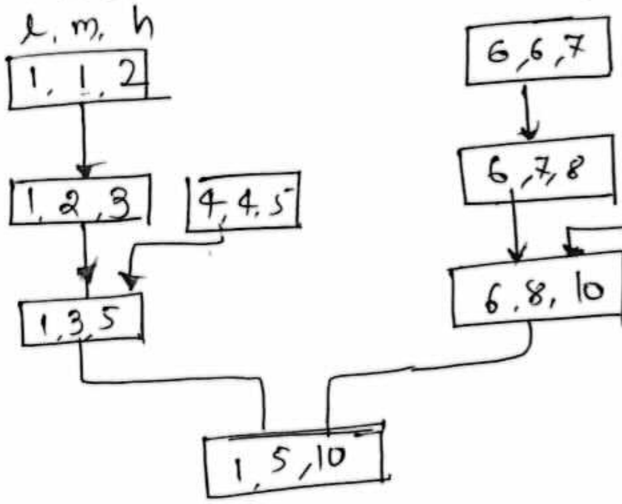
A: 1

B :

Divide



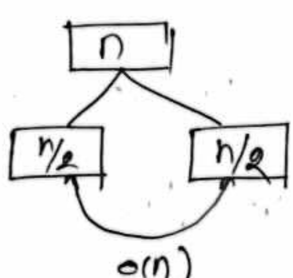
Merge



Let T(n) represent the time complexity of DANDC-MS(n)

$T(n) = c \ , \ n = 1, \ c > 0$

$\quad = 2T(n/2) + bn, \ n > 1, \ b > 0$



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

$T(n/2) = 2T(n/4) + bn/2$

$= 4T(n/4) + 2bn$

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

$= \cancel{2^k T(1) + 2^{k+1} bn}$

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

$\therefore n = 2^K$

$\therefore \underline{k = \log n}$

$\therefore T(n) = n\,T(1) + 2^{\log n - 1} \cdot bn = n\,T(1) + kbn$

$= nc + \frac{2^{\log n}}{2} bn = nc + \log n \cdot bn$

$= nc + \frac{n^2}{2} b = n \cdot c + bn \cdot \log n$

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

$\cancel{T(n) = O(n^2)}$ $\therefore \underline{T(n) = O(n \log n)}$

$\cancel{T(n)} =$

$S(n) = c_1 + \overset{\uparrow}{\underset{A}{n}} + \underbrace{\overset{\uparrow}{\underset{B}{n}} + \underset{\text{stack}}{\log n}}_{Sp}$ (inplace more)

$S(n) = c_1 + 2n + \log n$

$W \cdot S(n) = c_2 + (n + \log n)$

$= \underline{O(n)}$

Note :- $W \cdot S(n) = O(n)$ of Merge sort

$\therefore$ Merge sort is NOT In-place of Algo.

* Bottom-up mergesort :-

- 2-way mergesort

Min. No of comparisons in Bottom-up Merge sort

No of comparisons in Merge sort

$(\frac{n}{8})$ pass3: (4/7) [174 258 285 310 351 361 423 652]

$(\frac{n}{4})$ pass2: (2/3) [174, 285, 310, 652] [258, 351, 361, 423]

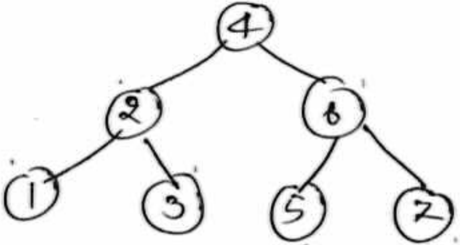
$(\frac{n}{2} \cdot 1)$ pass1: (1/1) [285 310] [174, 652] [351 423] [258 361]

A: [310] [285] [174] [652] [351] [423] [361] [258]

1 2 3 4 5 6 7

* Binary search :- (partial D & C)

1 . . .   n
A [1 . . . 4 . . . 7]

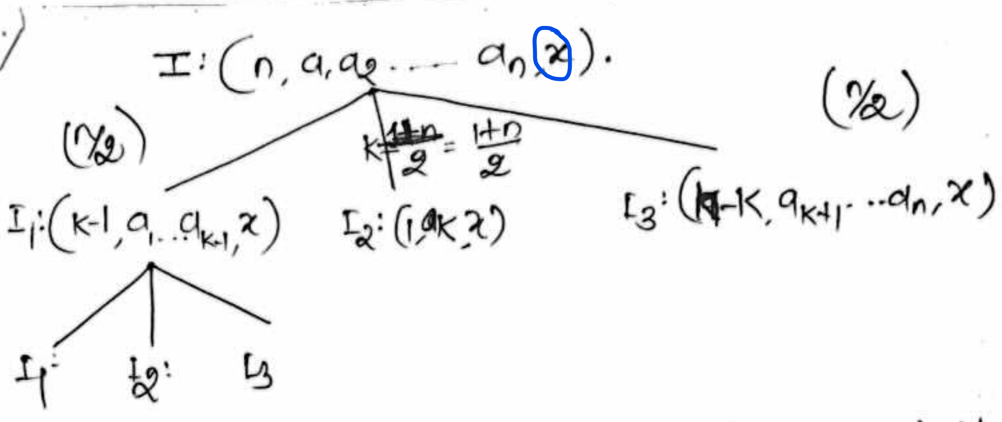


successful

B.S: $O(1)$ W.S: $O(\log n)$

unsuccessful

B.S : W.S : = $\Theta(\log n)$



Let $T(n)$ be represent the time complexity of DANDC-BS (n).

$$T(n) = a \cdot T(n/b) + f(n)$$

$$T(n) = c \quad , n = 1, \; c > 0$$
$$= b + T(n/2) \quad , n > 1$$

BS

$a = 1$

$b = 2$

$f(n) = b$

$$T(n) = T(n/2) + b$$
$$T(n/2) = T(n/4) + b$$
$$T(n) = T(n/4) + 2b$$
$$T(n) = T(n/2^k) + kb$$
$$= T(1) + b\log n$$
$$\boxed{T(n) = c + b\log n}$$
$$\therefore \underline{T(n) = O(\log n)}$$

$$n/2^k = 1$$
$$n = 2^k$$
$$\therefore \underline{K = \log n}.$$

n + log n

$$\therefore S(n) = \underbrace{c_1 + n + \log n}_{} = O(n)$$

Arry   stack

$$W.S(n) = \log n$$
$$= O(\log n)$$

$\therefore$ Binary search is <u>in-place</u>

* Matrix multiplication :-

$A_{n\times n}$, $B_{n\times n}$, $C_{n\times n}$ $\quad n \geq 1$

1) $A + B = C_{n\times n}$

for $i \leftarrow 1$ to n
  for $j \leftarrow 1$ to n
    $C(i,j) = A(i,j) + B(i,j)$ } $O(n^2)$

2) $A * B = C$

for $i \leftarrow 1$ to n
  for $j \leftarrow 1$ to n
    $C(i,j) = 0$
  for $k \leftarrow 1$ to n
    $C(i,j) = A(i,k) * B(k,j)$

$O(n^3)$

$C(i,j) \mathrel{+}= A(i,k) * B(k,j)$

$$A = \begin{array}{cc} A_{11} & A_{12} \end{array}$$

$$A = \begin{bmatrix} 4 & 5 & 8 & 9 \\ 6 & 3 & 2 & 7 \\ 1 & 2 & 4 & 5 \\ 8 & 9 & 2 & 3 \end{bmatrix}_{4\times 4\ (n\times n)} \quad \begin{array}{cc} A_{21} & A_{22} \end{array}$$

$$B = \begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ m & n & o & p \end{bmatrix}_{4\times 4\ (n\times n)} \quad \begin{array}{cc} B_{11} & B_{12} \\ B_{21} & B_{22} \end{array}$$

$$C = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix}_{4\times 4\ (n\times n)}$$

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

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

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

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

$(n \times n)$
$\downarrow$
$(\frac{n}{2} \times \frac{n}{2})$
$\downarrow$
$(\frac{n}{4} \times \frac{n}{4})$
$\downarrow$
$\vdots$
(2×2)

4×4 $\quad$ 2×4

Let $T(n)$ be represent the time complexity of DAND-MAT-MUL $(n \times n)$.

$$T(n) = c \quad , \quad n \le 2 \; ; \; c > 0$$
$$= 8T(n/2) + bn^2 \; , \; n > 2 \; ; \; b > 0$$

Time for multiplication of $(n/2 \times n/2)$

Time for Addn (n) of $(n/2 \times n/2)$

$a = 8 \quad p = 0$

$b = 2 \quad k = 2$

$\theta(n^{\log_b a}) = \theta(n^{\log_2 8})$

$= \theta(n^{\log_2 2^3})$

$= \theta(n^{3 \log_2 2})$

$= \theta(n^3)$

$$T(n) = 8T(n/2) + bn^2 \quad -①$$
$$T(n/2) = 8T(n/4) + b\frac{n^2}{4} \quad -②$$
$$T(n) = 64T(n/4) + 8b\frac{n^2}{4} + bn^2$$
$$= 64T(n/4) + 3bn^2$$
$$= 8^2 T(n/2^2) + (2^2 - 1)n^2$$
$$\vdots$$
$$= 8^k T(n/2^k) + (2^k - 1)n^2$$

$n/2^k = 1$

$\underline{k = \log n}$

$$= 8^k T(1) + (2^{\log n} - 1)bn^2$$
$$= 8^{\log n} T(1) + (2^{\log n} - 1)\, bn^2$$
$$= n^3 + (n-1)bn^2$$
$$= cn^3 + bn^3 - bn^2$$
$$= (c+b)n^3 - bn^2$$
$$T(n) = dn^3 - bn^2$$

$$\underline{T(n) = O(n^3)}$$

* STRASSEN'S MATRIX MULT :-

$$
\left.
\begin{aligned}
P &= (A_{11}+A_{22})\cdot(B_{11}+B_{22}) \\
Q &= (A_{21}+A_{22})\cdot B_{11} \\
R &= A_{11}\cdot(B_{12}-B_{22}) \\
S &= A_{22}\cdot(B_{21}-B_{11}) \\
T &= (A_{11}+A_{12})\cdot B_{22} \\
U &= (A_{21}-A_{11})\cdot(B_{11}+B_{12}) \\
V &= (A_{12}-A_{22})\cdot(B_{21}+B_{22})
\end{aligned}
\right\}
$$

$$
\begin{array}{c|c}
A_1 & A_{12} \\
\hline
A_{13} & A_{14}
\end{array}
$$

$$
\left.
\begin{aligned}
C_{11} &= P+S-T+V \\
C_{12} &= R+T \\
C_{21} &= Q+S \\
C_{22} &= P+R-Q+U
\end{aligned}
\right\}
$$

Time Complexity (STRASSEN)

$$
\left.
\begin{aligned}
T(n) &= c, & n \le 2 \\
&= 7T(n/2)+bn^2, & n > 2
\end{aligned}
\right\}
$$

$a=7$
$b=2$
$p=0$

$k=2$

$\Theta(n^{\log_b a})$

$\Theta(n^{\log_2 7})$

$\Theta(n^{2.81\ldots})$

$T(n) = 7T(n/2)+bn^2$ — ①

$T(n/2) = 7T(n/4)+\frac{bn^2}{4}$ — ②

$T(n) = 49T(n/4)+\left(\frac{7}{4}\right)bn^2+bn^2\left(\frac{7}{4}\right)^0$

$= 7^2T(n/2^2)+bn^2\sum_{i=0}^{1}\left(\frac{7}{4}\right)^i$

$= 7^kT(n/2^k)+bn^2\sum_{i=0}^{k-1}\left(\frac{7}{4}\right)^i$

$$\left(\boxed{\sum_{i=1}^{n} x^i = \frac{x^{n+1}-x}{x-1}} \qquad \sum_{i=1}^{n} < x^{n+1} \qquad \sum_{i=1}^{3} 2^i < 2^4 \right)$$

$\le 7^kT(1)+bn^2\left(\frac{7}{4}\right)^k \ldots$

$\le 7^kc+bn^2\frac{7^k}{4^k}$

$n/2^k = 1$

$k = \log n$

$< 7^{\log n}c + \cancel{bn^2}\frac{7^{\log n}}{4^{\log n}}$

$T(n) \le a.\,7^{\log n} \Rightarrow < a.\,n^{\log_2 7}$

$< a\,n^{(2.81)}$

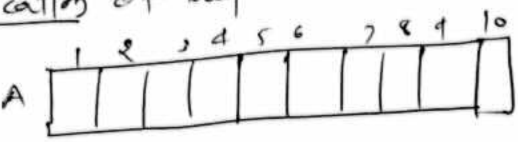
$= O(n^{2.81})$

DAND C $\rightarrow n^3$

S.M $\rightarrow n^{2.81}$

Integer Multiplication :-

Multiplication of large nos:



A[i] = one digit

→ Two integers (large) each having 'n' bits

1) Grammar school approach:-

u : 4586
v : 3583

$$\begin{array}{r} 4\,5\,8\,6 \\ * \; 3\,5\,8\,3 \\ \hline \end{array} \rightarrow O(n^2)$$

2) DANDC :- $n=4$

u : 4586
v : 3583

$m = \lfloor n/2 \rfloor$

$x = u/10^m$

$y = u \% 10^m$

$\therefore \; u = (x \times 10^m + y)$

~~vs (y×10^m +~~

$\therefore \; w = v/10^m$

$z = v \% 10^m$

$v = (w \times 10^m + z)$

$u = 4586$
$v = 3583$

$m = 2$

$x = 45$

$y = 86$

$u = (45 \times 10^2 + 86)$

$\therefore \; u \cdot v = (x \times 10^m + y) \cdot (w \times 10^m + z)$

$$= \underbrace{x \cdot w}_{\text{prod}_1} \cdot 10^{2m} + (\underbrace{xz}_{\text{prod}2} + \underbrace{w \cdot y}_{\text{prod}3}) 10^m + \underbrace{yz}_{\text{prod }4} \quad -(1)$$

$prod1 = x \cdot w$
$prod2 = x \cdot z$
$prod3 = w \cdot y$
$prod4 = yz$

Let $T(n)$ be represent the time complexity to multiply two nos (u, v) of n-bit/digits each

$$T(n) = c \;,\; n = 1$$
$$= 4T(n/2) + bn \;,\; n > 1$$

$$T(n) = 4T(n/2) + bn \quad \text{.①}$$
$$T(n/2) = 4T(n/4) + bn/2 \quad \text{-②}$$
$$T(n) = 16T(n/4) + 3bn$$
$$T(n/4) = 4T(n/16) + bn/4 =$$
$$T(n) \quad 64$$
$$T(n) = 2^{2k}T(n/2^k) + (2^k - 1)bn$$
$$= 4^k T(n/2^k) + (2^k - 1)bn$$

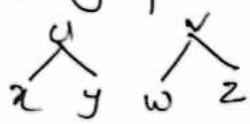
$O(n^2)$

$$n/2^k = \log n$$
$$T(n) =$$

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

* <u>KARATSUBA'S ALGORITHM</u> :- 2-way split

<u>optimize the time</u>



$$u.v = (x \times 10^m + y) \cdot (w \cdot \times 10^m + z)$$
$$= x \cdot w \times 10^{2m} + (xz + wy)10^m + yz$$

Let $(n/2 \text{ size}) \rightarrow r = (x+y)(w+z)$
$$r = x.w + (xz + wy) + yz$$

$$\Rightarrow (xz + wy) = (r - (xw + yz))$$

$$\therefore \; prod_1 = xw$$
$$prod_2 = yz$$
$$prod_3 = (x+y)(w+z) = r$$

$$\therefore \; u.v = (Prod_1 \times 10^{2m} + [prod_3 - (prod_1 + prod_2)] \times 10^m + prod_2$$

$$\therefore \; T(n) = 3T(n/2) + cn = O(n^{\log_2 3}) = \underline{O(n^{1.58})}$$

* Toom & cook optimization :-

3 - split

$$\begin{array}{ccc} & n & \\ n/3 & n/3 & n/3 \end{array}$$

$\therefore \quad T(n) = 9 \cdot T(n/3) + cn$

$\Rightarrow \underline{O(n^2)}$ (using DANDC)

$\therefore \quad T(n) = 8\, T(n/3) + n$

$= O(n^{\log_3 8}) = O(n^{1.8})$ (using KASTRUS optimization.

$\therefore \quad O(n^{\log_3 P}) < O(n^{1.5})$

$\log_3 P < 1.5$

$\underline{P = 5}$ = No. of multiplications in 3-split

$\therefore \quad T(n) = 5\,T(n/3) + cn$

$\Rightarrow O(n^{\log_3 5}) = O(n^{1.4})$

$\therefore$

No. of split	No. of multiplications (optimized)
K = 2	3
K = 3	5
K = 4	7
⋮	⋮
K	(2K−1)

In General,

$$\therefore \quad \boxed{T(n) = (2K-1)\,T(n/K) + cn}$$

* Master Theorem for solving Decrease & Recursion conquer Recurrences subtract & conquer

$$T(n) = c \quad ; \quad n \le 1$$
$$= a.T(n-b) + f(n) \quad , \quad n > 1$$

$a.T(n-b) + n^k$

$(a, b) > 0; \quad k \ge 0; \quad f(n)$ is positive

If $f(n)$ is $O(n^k)$ then

a) $T(n) = O(n^k) \quad , \quad a < 1$ case-I

$= O(n^{k+1}) \quad , \quad a = 1$ case-II

$= O(n^k a^{n/b}) \quad , \quad a > 1$ case-III.

$a T(n-b) + n^k$

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

$a = 1, b = 1, k = 1$

$f(n)$ is $O(n^k)$ $k = 1$

⑤ $T(n) = T(n-1) + c$

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

2) $T(n) = T(n-3) + n^2$ $\longrightarrow O(n^3)$

$a = 1, b = 3, k = 2$

3) $T(n) = 2T(n-1) + 1 \cdot n^0$ c is $O(n^k)$ $k = 0$

$O(1.2^n) \leftarrow O(n^0 \cdot 2^{n/1})$ $O(2^n)$

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

4) $T(n) = 4T(n-1) + n^1$

$n^1 \cdot 4^{n/1}$ $n \cdot 2^{2n}$

$a = 4, b = 1, k = 1$

$O(n.4^n) = O(n.2^{2n})$