

IMP

formal def
graph
E.g.

Θ	→ Upper Bound
Ω	→ Lower Bound
Θ	→ Tight Bound

12/07/2024

* Asymptotic Notations

- Mathematical way of Representing time complexity.

1) Big-Oh - O (order of function)

2) Big-Omega - Ω

3) Theta - Θ

- Common Runtime time complexities are

$O(2^n)$, $O(n^3)$, $O(n)$, $O(\log n)$, $O(n \log n)$, $O(n^2)$

1) Big-Oh

- When we do performance analysis of algorithm without execution no. of time instructions executed is represented by Asymptotic notation.
- (Upper Bound of running time)
- Using Big-Oh we can use longest amount of time taken by algorithm to complete.

Given : two functions $f(n)$ & $g(n)$
 we say that $f(n)$ is $O(g(n))$ if there exists constant $c > 0$ & $n_0 > 0$ such that

$$f(n) \leq c * g(n) \text{ for all } n \geq n_0$$

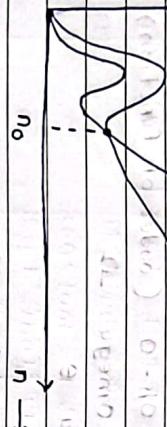
$T(n)$

Example) $f(n) = 1000n = O(n)$

$C * g(n)$

Example) $f(n) = n^2 + 3n + 1 \rightarrow O(n^2)$

$f(n)$ is asymptotically bounded above by $g(n)$ up to constant factor c



$f(n)$ is asymptotically bounded above by $g(n)$ up to constant factor c

$f(n) = n^2 + 3n + 1$ is bounded above by $g(n) = n^2$ for some c

Example)

$$3n^2 - 100n + 6 = O(n^2)$$

$f(n) = n^2 - 100n + 6$ is bounded above by $g(n) = n^2$ for some c

$f(n) = n^2 - 100n + 6$ is bounded above by $g(n) = n^2$ for some c

$f(n) = 3n^2 + 2n^2 + 5n + 1$ is bounded above by $g(n) = 3n^2$ for some c

$f(n) = 3n^2 + 2n^2 + 5n + 1$ is bounded above by $g(n) = 3n^2$ for some c

$f(n) = 3n^2 + 2n^2 + 5n + 1$ is bounded above by $g(n) = 3n^2$ for some c

$f(n) = 3n^2 + 2n^2 + 5n + 1$ is bounded above by $g(n) = 3n^2$ for some c

$f(n) = 3n^2 + 2n^2 + 5n + 1$ is bounded above by $g(n) = 3n^2$ for some c

$f(n) = 3n^2 + 2n^2 + 5n + 1$ is bounded above by $g(n) = 3n^2$ for some c

$f(n) = 3n^2 + 2n^2 + 5n + 1$ is bounded above by $g(n) = 3n^2$ for some c

$f(n) = 3n^2 + 2n^2 + 5n + 1$ is bounded above by $g(n) = 3n^2$ for some c

Big-O Omega Notation (Ω)

Using omega notation we can denote shortest time of algorithm.

The function $f(n)$ is said to be in Omega $g(n)$

If the $f(n)$ is bounded below by some positive constant multiple of $g(n)$ such that

$f(n) \geq c \cdot g(n)$ for all $n \geq n_0$



$f(n)$ is bounded below by $g(n)$ up to constant factor c

$f(n) = n^3$ is bounded below by $g(n) = n^3$ up to constant factor c

$f(n) = n^3$ is bounded below by $g(n) = n^3$ up to constant factor c

$f(n) = n^3$ is bounded below by $g(n) = n^3$ up to constant factor c

$f(n) = n^3$ is bounded below by $g(n) = n^3$ up to constant factor c



$f(n)$ is bounded below by $g(n)$ up to constant factor c

3) Theta Θ

formal definition

$f(n)$ & $g(n)$ are two non-negative integer function

$f(n) = \Theta g(n)$ [Read. f of n is Theta of $g(n)$]

if there exist two positive

such that $c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)$

for all $n \geq n_0$

That is it is bounded from above & below by the same function $g(n)$.

- Example**) $f(n) = 4n + 1$
 $g(n) = n$
- Ci. $n \leq 4n + 1 \leq C_2 \cdot n$
- $C_1 = 4$ $C_2 = 5$
- | n | $c_1 \cdot g(n)$ | \leq | $4n + 1$ | \leq | $c_2 \cdot g(n)$ |
|---|------------------|--------|----------|--------|------------------|
| 1 | 4 | | 5 | | 5 |
| 2 | 8 | | 9 | | 10 |
| 3 | 12 | | 13 | | 15 |
| 4 | 16 | | 17 | | 20 |
- Example**) $f(n) = 2n^2 + n$

c_1

c_2

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

$$c_1 \cdot 0 \leq 2n^2 + n \leq c_2 \cdot n$$

$c_1 = 2$

$$n \quad | \quad c_1 \cdot g(n) \quad | \quad \leq \quad 2n^2 + n \quad \leq \quad (c_2 \cdot g(n))$$

$$1 \quad | \quad 2 \quad | \quad 3 \quad | \quad 3$$

$$2 \quad | \quad 4 \quad | \quad 10 \quad | \quad 6$$

$$3 \quad | \quad 6 \quad | \quad 18 \quad | \quad 9$$

4

8

array ele not equal - Distinct

Let's us denote $C(n)$ no. of time comparison executed.

$$J = 1 \quad u = n - 1$$

Algorithm $\text{MaxElement}(A[0..n])$

Apply formula ;

$$\sum_{i=1}^n i = u - 1 + 1 \quad \text{where } u \leq n \text{ some lower bound}$$

$= n - 1 + 1 = n$

$$\sum_{i=1}^n i = u - 1 + 1 \quad \text{where } u \leq n \text{ some lower bound}$$

Summation formula

$$\sum_{i=1}^n i = u - 1 + 1 \quad \text{where } u \leq n \text{ some lower bound}$$

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

Example) Consider the problem finding the value of largest element from the list of numbers

Algorithm $\text{MaxElement}(A[0..n])$ running time $\Theta(n^2)$

// Determine the value of the largest element in a given array having n elements

Input : An array $A[0..n-1]$ of integer

// Output : the largest value from array $A[0..n-1]$

Algorithm $\text{MaxElement}(A[0..n-1])$

// Input : array

// Output : Array element are distinct or not.

For $i \leftarrow 1$ to $n-1$ do

 if $A[i] > \text{maxvalue}$ then

$\text{maxvalue} \leftarrow A[i]$

return maxvalue

Steps :

two operation :

 i) $A[i] > \text{maxvalue}$

 ii) $\text{maxvalue} \leftarrow A[i]$

Worst Case = Outer loop, inner loop

$$= (n-1) + (i+1) + \dots + n = n^2 - 1$$

$$= n^2 - 1$$

$$i = 0 \quad j = 1 \quad \dots \quad n-1$$

$$i = 0 \quad j = 2 \quad \dots \quad n-2$$

$$i = 0 \quad j = 3 \quad \dots \quad n-3$$

$$i = 0 \quad j = 4 \quad \dots \quad n-4$$

$$i = 0 \quad j = 5 \quad \dots \quad n-5$$

$$i = 1 \quad j = 2 \quad \dots \quad n-1$$

$$i = 1 \quad j = 3 \quad \dots \quad n-2$$

$$i = 1 \quad j = 4 \quad \dots \quad n-3$$

$$i = 1 \quad j = 5 \quad \dots \quad n-4$$

$$i = 2 \quad j = 3 \quad \dots \quad n-2$$

$$i = 2 \quad j = 4 \quad \dots \quad n-3$$

$$i = 2 \quad j = 5 \quad \dots \quad n-4$$

$$i = 3 \quad j = 4 \quad \dots \quad n-3$$

$$i = 3 \quad j = 5 \quad \dots \quad n-4$$

$$i = 4 \quad j = 5 \quad \dots \quad n-4$$

$$i = 5 \quad j = 6 \quad \dots \quad n-5$$

$$i = 6 \quad j = 7 \quad \dots \quad n-6$$

$$i = 7 \quad j = 8 \quad \dots \quad n-7$$

$$\text{Total cost} = n(n-1) + (n-2)(n-1) + \dots + 2(1)$$

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

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

Example) Matrix Multiplication

Algorithm matmul ($A[0..n-1, 0..n-1]$, $B[0..n-1, 0..n-1]$)

// matrix multiplication (.n) and matrix multiplication (.n)

// input : two matrix A & B : n x n

// output : C Matrix containing multiplication of A & B

for i ← 0 to n-1 do

 for j ← 0 to n-1 do

 c[i, j] ← 0

 for k ← 0 to n-1 do

 c[i, j] = c[i, j] + a[i, k] * b[k, j]

- 1) Input Size = $n \times n$
- 2) Order of matrix = $n + (1+1) + (1+1)$
- 3) Basic operation in innermost loop = Addition & Multiply.

$$C(n) = \text{Outerloop} \times \text{innerloop} \times \text{innermost loop}$$

i -for loop $\times j$ -for loop $\times k$ -for loop

$$C(n) = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} \sum_{k=0}^{n-1} (1)$$

$$(1) = (n+1) \times (n+1) \times (n+1)$$

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

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

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

Example) Summation all element

Algorithm Sum (n) : $[a_1, a_2, a_3, \dots, a_n]$ contains multi-sigma

Algorithm Sum (n) : $[a_1, a_2, a_3, \dots, a_n]$ contains multi-sigma

// input : A \rightarrow A parameter out : s : integral

$s \leftarrow 0$ form primitive variable s : integral

for $i=0$ to n do $a_i \rightarrow s$: integral

$s \leftarrow s + a_i$: $i=0 \rightarrow i=n$: integral

return s : $s = a_1 + a_2 + \dots + a_n$

Order of growth = n

Time complexity = $O(n)$

24/07/24

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

Step 1 : Identify $a, b, f(n)$

$$a = 4$$

$$b = 2$$

$$4\sqrt{n} = n^2$$

Step 2 : Compute $\log_b n = \log_2 2 = 1$

$$= 2$$

Step 3 : Compare $f(n)$ with $n^{\log_b b}$

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

Step 4 : Determine case : 2

$$f(n) = n^{\log_b b} = n^2 = n^2$$

Step 5 : Apply master method

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

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

Step 1 : Identify $a, b, f(n)$

$$a = 2$$

$$b = 2$$

Step 2 : Compute $\log_b n = \log_2 2 = 1$

$$= 1$$

Step 3 : Compare $f(n)$ with $n^{\log_b b}$

$$f(n) = n$$

Step 4 : Determine Case : 2 $\Rightarrow T(n) < n^{\log_b b}$

$$f(n) = n \log_2 n = n$$

Step 5 : Apply master method - case 3

$$T(n) = \Theta(n^{\log_b b})$$

Step 1 : Identify $a, b, f(n)$

$$a = 5$$

$$b = 5$$

$$f(n) = n$$

Step 2 : Compute $\log_b n = \log_5 n$

$$= 2$$

Step 3 : Compute $f(n)$ with $= n^{\log_5 5}$

$$= n^5$$

Step 4 : Determine Case : 2

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

Step 5 : Apply Master Method

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

* Substitution method to solve recurrence Relations

Backward 

- One of simplest method for solving simple recurrence relation.

- We solve the recurrence relation for $n=10, 1, 2 \dots$ until we see a pattern then we make a guess and predict the running time.

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

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

$$\textcircled{3} \quad - \quad c + \left(\frac{8}{U} \right) = \left(\frac{4}{U} \right) T$$

$$\text{Substitute } T \left(\frac{\pi}{2} \right) \text{ in eq, (1)}$$

$$L = \left(\frac{d}{dx} \right) + C = \cos x + C$$

$$\tau(n) \approx \tau\left(\frac{n}{4}\right) + 2C_1 \quad (12) \quad (21)$$

$$= T \left(\frac{q}{2\epsilon} \right) + 2C$$

Substitute $T\left(\frac{n}{2}\right)$ in eqn — (3)

$$T(u) = \tau \left(\frac{g}{U} \right)^2 + C + C + C$$

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

$$T(n) = T\left(\frac{n}{24}\right) + 4c \quad \text{--- Ans}$$

$$T(n) = T\left(\frac{n}{25}\right) + 5c$$

$$n = 2k$$

$$\tau(n) = \begin{cases} \tau\left(\frac{n}{2}\right) + c & \text{if } n > 1 \\ 1 & \text{if } n = 1 \end{cases}$$

Backward Substitution *

- Where the value of $n = k$ & $f(n) = k$ & also the $T(n) = n$
- In Backward substitution we do opposite of forward substitution.

$$t(n) = \begin{cases} t\left(\frac{n}{a}\right) + c & \text{if } n > 1 \\ b & \text{if } n = 1 \end{cases}$$

*

Backward Substitution

$$n = 2^k$$

$$\log n = \log 2^k$$

$$= k \log 2$$

$$\log n = k(1) + \left(\frac{0}{k}\right) 1 + \left(\frac{0}{k}\right) 1$$

Put k values

$$\frac{n}{2} = n/2$$

$$\begin{aligned} T(n) &= \begin{cases} 1 & \text{if } n=1 \\ 2T\left(\frac{n}{2}\right) + n & \text{otherwise} \end{cases} \\ & \quad \text{Given } T(n) \text{ we can calculate the value of } T(n/2) \end{aligned}$$

$$\text{Then, } T\left(\frac{n}{2}\right) = T\left(\frac{n}{4}\right) + c \quad \text{Put eqn ①}$$

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

$$\text{Put } n = n/2 \text{ in eqn ①}$$

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

$$T(n) = \left[T\left(\frac{n}{8}\right) + 2c \right] + c$$

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

$$\text{Put } n = n/2 \text{ in eqn ①} \Rightarrow T\left(\frac{n}{8}\right) = (n) 1$$

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

$$T(n) = \left[T\left(\frac{n}{16}\right) + 3c \right] + c$$

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

$$T(n) = T\left(\frac{n}{32}\right) + 4c$$

$$T(n) = T\left(\frac{n}{64}\right) + 4c$$

$$T(n) = T\left(\frac{n}{128}\right) + 4c$$

$$T(n) = T\left(\frac{n}{256}\right) + 4c$$

$$T(n) = T\left(\frac{n}{512}\right) + 4c$$

$$T(n) = T\left(\frac{n}{1024}\right) + 4c$$

$$T(n) = T\left(\frac{n}{2048}\right) + 4c$$

$$T(n) = T\left(\frac{n}{4096}\right) + 4c$$

$$T(n) = T\left(\frac{n}{8192}\right) + 4c$$

$$T(n) = T\left(\frac{n}{16384}\right) + 4c$$

$$T(n) = T\left(\frac{n}{32768}\right) + 4c$$

$$T(n) = T\left(\frac{n}{65536}\right) + 4c$$

$$T(n) = T\left(\frac{n}{131072}\right) + 4c$$

$$T(n) = T\left(\frac{n}{262144}\right) + 4c$$

$$T(n) = T\left(\frac{n}{524288}\right) + 4c$$

$$T(n) = T\left(\frac{n}{1048576}\right) + 4c$$

$$T(n) = T\left(\frac{n}{2097152}\right) + 4c$$

$$T(n) = T\left(\frac{n}{4194304}\right) + 4c$$

$$T(n) = T\left(\frac{n}{8388608}\right) + 4c$$

$$T(n) = T\left(\frac{n}{16777216}\right) + 4c$$

$$T(n) = T\left(\frac{n}{33554432}\right) + 4c$$

$$T(n) = T\left(\frac{n}{67108864}\right) + 4c$$

$$T(n) = T\left(\frac{n}{134217728}\right) + 4c$$

$$T(n) = T\left(\frac{n}{268435456}\right) + 4c$$

$$T(n) = T\left(\frac{n}{536870912}\right) + 4c$$

$$T(n) = T\left(\frac{n}{1073741824}\right) + 4c$$

$$T(n) = T\left(\frac{n}{2147483648}\right) + 4c$$

$$T(n) = T\left(\frac{n}{4294967296}\right) + 4c$$

$$T(n) = T\left(\frac{n}{8589934592}\right) + 4c$$

$$T(n) = T\left(\frac{n}{17179869184}\right) + 4c$$

$$T(n) = T\left(\frac{n}{34359738368}\right) + 4c$$

$$T(n) = T\left(\frac{n}{68719476736}\right) + 4c$$

$$T(n) = T\left(\frac{n}{137438953472}\right) + 4c$$

$$T(n) = T\left(\frac{n}{274877906944}\right) + 4c$$

$$T(n) = T\left(\frac{n}{549755813888}\right) + 4c$$

$$T(n) = T\left(\frac{n}{1099511627776}\right) + 4c$$

$$T(n) = T\left(\frac{n}{2199023255552}\right) + 4c$$

$$T(n) = T\left(\frac{n}{4398046511104}\right) + 4c$$

$$T(n) = T\left(\frac{n}{8796093022208}\right) + 4c$$

$$T(n) = T\left(\frac{n}{17592186044416}\right) + 4c$$

$$T(n) = T\left(\frac{n}{35184372088832}\right) + 4c$$

$$T(n) = T\left(\frac{n}{70368744177664}\right) + 4c$$

$$T(n) = T\left(\frac{n}{140737488355328}\right) + 4c$$

$$T(n) = T\left(\frac{n}{281474976710656}\right) + 4c$$

$$T(n) = T\left(\frac{n}{562949953421312}\right) + 4c$$

$$T(n) = T\left(\frac{n}{1125899906842624}\right) + 4c$$

$$T(n) = T\left(\frac{n}{2251799813685248}\right) + 4c$$

$$T(n) = T\left(\frac{n}{4503599627370496}\right) + 4c$$

$$T(n) = T\left(\frac{n}{9007199254740992}\right) + 4c$$

$$T(n) = T\left(\frac{n}{18014398509481984}\right) + 4c$$

$$T(n) = T\left(\frac{n}{36028797018963968}\right) + 4c$$

$$T(n) = T\left(\frac{n}{72057594037927936}\right) + 4c$$

$$T(n) = T\left(\frac{n}{144115188075855872}\right) + 4c$$

$$T(n) = T\left(\frac{n}{288230376151711744}\right) + 4c$$

$$T(n) = T\left(\frac{n}{576460752303423488}\right) + 4c$$

$$T(n) = T\left(\frac{n}{1152921504606846976}\right) + 4c$$

$$T(n) = T\left(\frac{n}{2305843009213693952}\right) + 4c$$

$$T(n) = T\left(\frac{n}{4611686018427387904}\right) + 4c$$

$$T(n) = T\left(\frac{n}{9223372036854775808}\right) + 4c$$

$$T(n) = T\left(\frac{n}{18446744073709551616}\right) + 4c$$

$$T(n) = T\left(\frac{n}{36893488147419103232}\right) + 4c$$

$$T(n) = T\left(\frac{n}{73786976294838206464}\right) + 4c$$

$$T(n) = T\left(\frac{n}{147573952589676412928}\right) + 4c$$

$$T(n) = T\left(\frac{n}{295147905179352825856}\right) + 4c$$

$$T(n) = T\left(\frac{n}{590295810358705651712}\right) + 4c$$

$$T(n) = T\left(\frac{n}{1180591620717411303424}\right) + 4c$$

$$T(n) = T\left(\frac{n}{2361183241434822606848}\right) + 4c$$

$$T(n) = T\left(\frac{n}{4722366482869645213696}\right) + 4c$$

$$T(n) = T\left(\frac{n}{9444732965739290427392}\right) + 4c$$

$$T(n) = T\left(\frac{n}{18889465931478580854784}\right) + 4c$$

$$T(n) = T\left(\frac{n}{37778931862957161689568}\right) + 4c$$

$$T(n) = T\left(\frac{n}{75557863725914323379136}\right) + 4c$$

$$T(n) = T\left(\frac{n}{151115727451828646758272}\right) + 4c$$

$$T(n) = T\left(\frac{n}{302231454903657293516544}\right) + 4c$$

$$T(n) = T\left(\frac{n}{604462909807314587033088}\right) + 4c$$

$$T(n) = T\left(\frac{n}{1208925819614629174066176}\right) + 4c$$

$$T(n) = T\left(\frac{n}{2417851639229258348132352}\right) + 4c$$

$$T(n) = T\left(\frac{n}{4835703278458516696264704}\right) + 4c$$

$$T(n) = T\left(\frac{n}{9671406556917033392529408}\right) + 4c$$

$$T(n) = T\left(\frac{n}{19342813113834066785058816}\right) + 4c$$

$$T(n) = T\left(\frac{n}{38685626227668133570117632}\right) + 4c$$

$$T(n) = T\left(\frac{n}{77371252455336267140235264}\right) + 4c$$

$$T(n) = T\left(\frac{n}{154742504910672534280470528}\right) + 4c$$

$$T(n) = T\left(\frac{n}{309485009821345068560941056}\right) + 4c$$

$$T(n) = T\left(\frac{n}{618970019642690137121882112}\right) + 4c$$

$$T(n) = T\left(\frac{n}{1237940039285380274243764224}\right) + 4c$$

$$T(n) = T\left(\frac{n}{2475880078570760548487528448}\right) + 4c$$

$$T(n) = T\left(\frac{n}{4951760157141521096975056896}\right) + 4c$$

$$T(n) = T\left(\frac{n}{9903520314283042193950113792}\right) + 4c$$

$$T(n) = T\left(\frac{n}{19807040628566084387900227584}\right) + 4c$$

$$T(n) = T\left(\frac{n}{39614081257132168775800455168}\right) + 4c$$

$$T(n) = T\left(\frac{n}{79228162514264337551600910336}\right) + 4c$$

$$T(n) = T\left(\frac{n}{158456325228528675103201820672}\right) + 4c$$

$$T(n) = T\left(\frac{n}{316912650457057350206403641344}\right) + 4c$$

$$T(n) = T\left(\frac{n}{633825300914114700412807282688}\right) + 4c$$

$$T(n) = T\left(\frac{n}{1267650601828229400825614565376}\right) + 4c$$

$$T(n) = T\left(\frac{n}{2535301203656458801651229130752}\right) + 4c$$

$$T(n) = T\left(\frac{n}{5070602407312917603202458261504}\right) + 4c$$

$$T(n) = T\left(\frac{n}{1014120481462583520640491652308}\right) + 4c$$

$$T(n) = T\left(\frac{n}{2028240962925167041280983304616}\right) + 4c$$

$$T(n) = T\left(\frac{n}{4056481925850334082561966609232}\right) + 4c$$

$$T(n) = T\left(\frac{n}{8112963851700668165123933218464}\right) + 4c$$

$$T(n) = T\left(\frac{n}{16225927703401336320247866436928}\right) + 4c$$

$$T(n) = T\left(\frac{n}{32451855406802672640495732873856}\right) + 4c$$

$$T(n) = T\left(\frac{n}{64903710813605345280985465747712}\right) + 4c$$

$$T(n) = T\left(\frac{n}{129807421627210690561970931495424}\right) + 4c$$

$$T(n) = T\left(\frac{n}{259614843254421381123941862990848}\right) + 4c$$

$$T(n) = T\left(\frac{n}{519229686508842762247883725981696}\right) + 4c$$

$$T(n) = T\left(\frac{n}{1038459373017685524495767451963392}\right) + 4c$$

$$T(n) = T\left(\frac{n}{2076918746035371048991534903926784}\right) + 4c$$

$$T(n) = T\left(\frac{n}{4153837492070742097983069807853568}\right) + 4c$$

$$T(n) = T\left(\frac{n}{8307674984141484195966139615707136}\right) + 4c$$

$$T(n) = T\left(\frac{n}{16615349968282968391932279231414272}\right) + 4c$$

$$T(n) = T\left(\frac{n}{33230699936565936783864558462828544}\right) + 4c$$

$$T(n) = T\left(\frac{n}{66461399873131873567729116925657088}\right) + 4c$$

$$T(n) = T\left(\frac{n}{132922799746263747135458233851314176}\right) + 4c$$

$$T(n) = T\left(\frac{n}{265845599492527494270916467702628352}\right) + 4c$$

$$T(n) = T\left(\frac{n}{531691198985054988541832935405256704}\right) + 4c$$

$$T(n) = T\left(\frac{n}{1063382397970109977083665870810513408}\right) + 4c$$

$$T(n) = T\left(\frac{n}{2126764795940219954167331741621026816}\right) + 4c$$

$$T(n) = T\left(\frac{n}{4253529591880439908334663483242053632}\right) + 4c$$

$$T(n) = T\left(\frac{n}{8507059183760879816669326966484107264}\right) + 4c$$

$$T(n) = T\left(\frac{n}{17014118367521759633338653932968214528}\right) + 4c$$

$$T(n) = T\left(\frac{n}{34028236735043519266677307865936429056}\right) + 4c$$

$$T(n) = T\left(\frac{n}{68056473470087038533354615731872858112}\right) + 4c$$

$$T(n) = T\left(\frac{n}{136112946940174077066709231463745716224}\right) + 4c$$

$$T(n) = T\left(\frac{n}{27222589388034815$$

Generalize eqn :

solve for Big-O notation

$$T\left(\frac{n}{k}\right) = 2T\left(\frac{n}{k}\right) + c \quad \text{Put eqn 1}$$

$$T(n) = T\left(\frac{n}{k}\right) + k.c \Rightarrow \left(\frac{n}{k}\right)^r = (n)^r$$

$$\therefore n = 2k$$

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

$$= T(1) + k.c \quad \text{if } n = 1 \\ = T(1) + k.c \quad \text{if } n = 1$$

①

$$\log n = k \log 2 \quad \text{or} \quad \left(\frac{n}{2}\right)^r = (n)^r$$

①

$$\log n = k \log 2$$

$$T(n) = T(1) + k.c$$

①

Discard constants

$$T(n) = \log_2 n$$

①

Example) $T(n) = 2T\left(\frac{n}{2}\right) + c$ ① $\therefore \sqrt{n} = n^{1/2}$

put eqn ①

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

①

$$T(n) = 4T\left(\frac{n}{4}\right) + 3c$$

..

$$T(n) = 8T\left(\frac{n}{8}\right) + 5c$$

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

put eqn ①

$$T(n) = 8T\left(\frac{n}{16}\right) + 6c$$

$$T(n) = 16T\left(\frac{n}{32}\right) + 4c$$

①

$$\uparrow \quad \text{if } \left(\frac{n}{32}\right)^r = (n)^r$$

①

Discard constants

$$T(n) = \log_2 n$$

①

Example)

$T(n) = 2T\left(\frac{n}{2}\right) + c$ ① $\therefore \sqrt{n} = n^{1/2}$

put eqn ①

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

①

$$T(n) = 4T\left(\frac{n}{4}\right) + 3c$$

..

Example $\tau(n) = \tau\left(\frac{n}{3}\right) + c$ ————— (1)

Put $n = n/3$ in eqⁿ ①

$$\boxed{\tau\left(\frac{n}{3}\right) = \tau\left(\frac{n}{9}\right) + c}$$

eqⁿ (1)

$$\tau(n) = \left[\tau\left(\frac{n}{9}\right) + c \right] + c$$

$$\boxed{\tau(n) = \tau\left(\frac{n}{9}\right) + 2c} \quad \text{--- (2)}$$

Put $n = n/9$

$$\boxed{\tau(n) = \tau\left(\frac{n}{9}\right) + 2c} \quad \text{--- (2)}$$

eqⁿ (1)

$$\boxed{\tau(n) = \left[\tau\left(\frac{n}{9}\right) + 2c \right] + c}$$

$$\boxed{\tau(n) = \tau\left(\frac{n}{81}\right) + 3c}$$

Ans : $\tau(n) = c \cdot \log_3 n + 1$

* Divide & Conquer

Algorithm DC (P)

If P is too small then return soln of P

else

(notisakm) Divide (P) and obtain $P_1, P_2, P_3, \dots, P_n$

where $n \geq 1$

Apply DC to each subproblem

return combine (DC(P_1), DC(P_2), ...)

Recurrence Relation

$$\tau(n) = \begin{cases} g(n) & \text{if } n \text{ is small} \\ T(n_1) + T(n_2) + \dots + T(n_r) + f(n) & \text{otherwise} \end{cases}$$

* Binary Search Recursive Algorithm

Algorithm BinSearch (A, key, low, high)

// This algorithm is to search an element using Binary Search

// Input : An array A & key = (1 to 1000)

// Output : Index of search key if key is found

```
(1) low = 0    (1 to 1000) = (1 to 1000)
    high = n - 1
    mid = (low + high) / 2
    if (key == A[mid]) then return mid
    else if (key < A[mid]) then
        binsearch (A, key, low, mid - 1)
```

31/07/24

$$C_{\text{worst}}(2^k) = \left[C_{\text{worst}}(2^{k-1}) + 1 \right] + 1$$

else

if key \neq arr[mid]

} else if key \neq arr[mid+1]

$$C_{\text{worst}}(2^k) = C_{\text{worst}}(2^{k-1}) + 1$$

Analysis :

$$C_{\text{worst}}(n) = C_{\text{worst}}\left(\frac{n}{2}\right) + 1 \quad \leftarrow \begin{array}{l} \text{if key is at mid} \\ \text{(one comparison)} \end{array}$$

$$C_{\text{worst}}(1) = 1$$

$\Rightarrow C_{\text{worst}}(2^k) = C_{\text{worst}}(2^{k-1}) + 1$

$C_{\text{worst}}(2^k) = C_{\text{worst}}(2^{k-2}) + 1$

$C_{\text{worst}}(2^k) = C_{\text{worst}}(2^{k-3}) + 1$

$C_{\text{worst}}(2^k) = C_{\text{worst}}(2^{k-4}) + 1$

$C_{\text{worst}}(2^k) = C_{\text{worst}}(2^{k-5}) + 1$

$C_{\text{worst}}(2^k) = C_{\text{worst}}(2^{k-6}) + 1$

$C_{\text{worst}}(2^k) = C_{\text{worst}}(2^{k-7}) + 1$

$C_{\text{worst}}(2^k) = C_{\text{worst}}(2^{k-8}) + 1$

$C_{\text{worst}}(2^k) = C_{\text{worst}}(2^{k-9}) + 1$

$C_{\text{worst}}(2^k) = C_{\text{worst}}(2^{k-10}) + 1$

$C_{\text{worst}}(2^k) = C_{\text{worst}}(2^{k-11}) + 1$

$C_{\text{worst}}(2^k) = C_{\text{worst}}(2^{k-12}) + 1$

$C_{\text{worst}}(2^k) = C_{\text{worst}}(2^{k-13}) + 1$

$C_{\text{worst}}(2^k) = C_{\text{worst}}(2^{k-14}) + 1$

Using Backward Substitution,

$C_{\text{worst}}(2^{k-1}) = C_{\text{worst}}(2^{k-2}) + 1$

$$\begin{aligned} C_{\text{worst}}(2^k) &= C_{\text{worst}}(2^{k-1}) + 1 \\ &= C_{\text{worst}}(2^0) + k.3 \\ &= 1 + k.3 \\ \therefore C_{\text{worst}}(2^k) &= 1 + \log_2 n . C \end{aligned}$$

Discard constants

$$C_{\text{worst}}(2^k) = n \log_2 n$$

using $n = 2^k$ then, $\log_2 n = k$

$C_{\text{worst}}(2^k) = n k$

$C_{\text{worst}}(2^k) = C_{\text{worst}}(2^{k-1}) + 1$

$C_{\text{worst}}(2^k) = C_{\text{worst}}(2^{k-2}) + 1$

$C_{\text{worst}}(2^k) = C_{\text{worst}}(2^{k-3}) + 1$

$C_{\text{worst}}(2^k) = C_{\text{worst}}(2^{k-4}) + 1$

$C_{\text{worst}}(2^k) = C_{\text{worst}}(2^{k-5}) + 1$

$C_{\text{worst}}(2^k) = C_{\text{worst}}(2^{k-6}) + 1$

$C_{\text{worst}}(2^k) = C_{\text{worst}}(2^{k-7}) + 1$

$C_{\text{worst}}(2^k) = C_{\text{worst}}(2^{k-8}) + 1$

$C_{\text{worst}}(2^k) = C_{\text{worst}}(2^{k-9}) + 1$

$C_{\text{worst}}(2^k) = C_{\text{worst}}(2^{k-10}) + 1$

$C_{\text{worst}}(2^k) = C_{\text{worst}}(2^{k-11}) + 1$

$C_{\text{worst}}(2^k) = C_{\text{worst}}(2^{k-12}) + 1$

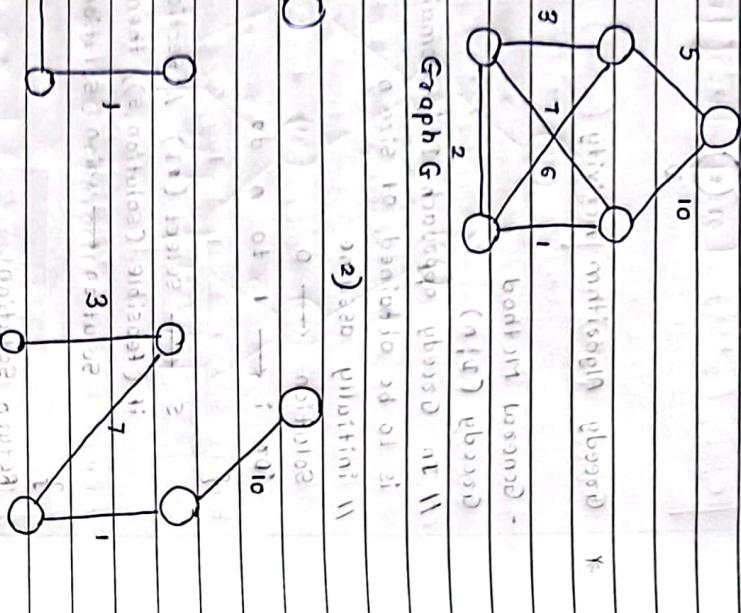
$C_{\text{worst}}(2^k) = C_{\text{worst}}(2^{k-13}) + 1$

$C_{\text{worst}}(2^k) = C_{\text{worst}}(2^{k-14}) + 1$

* Haffman Coding

iv) Greedy method works in stages only on input considered at each time, & based on these input it is considered or decide whether particular input gives optimal solution.

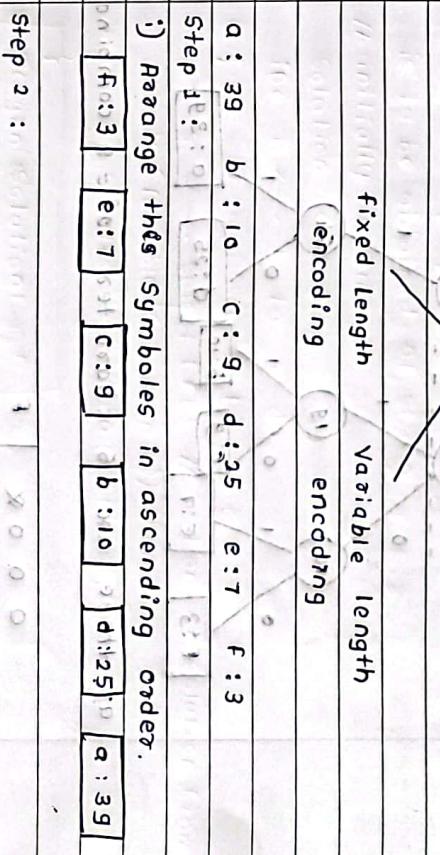
Example)



$$w(\tau_1) = 11$$

$$w(\tau_1) = 21$$

Step 2 : Combination 1



Data	a	b	c	d	e	f	g	h	h	a

↑
8 bits

$11 \times 8 = 88$ bits

- Haffman encoding contain the data in the form of sequence of character
- Create a table having Frequency of occurrences of each character in the given data.
- From this table Haffmans tree is constructed.

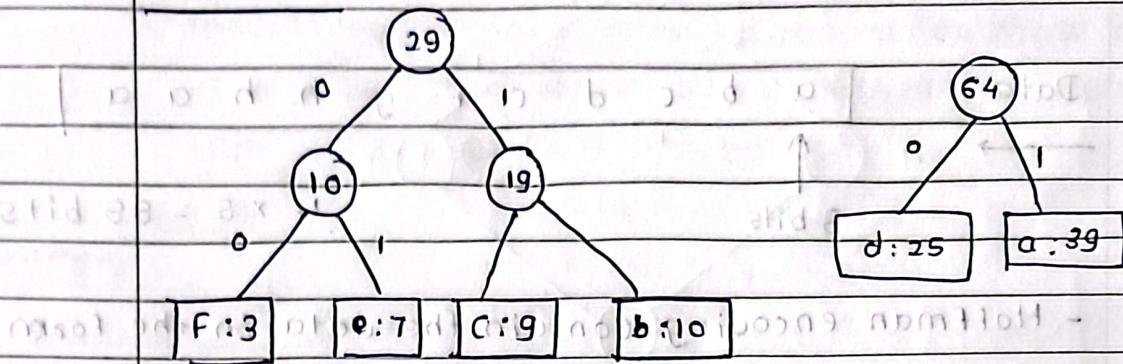
Having fixed length variable length encoding

E.g.) a : 39 b : 10 c : 9 d : 25 e : 7 f : 3

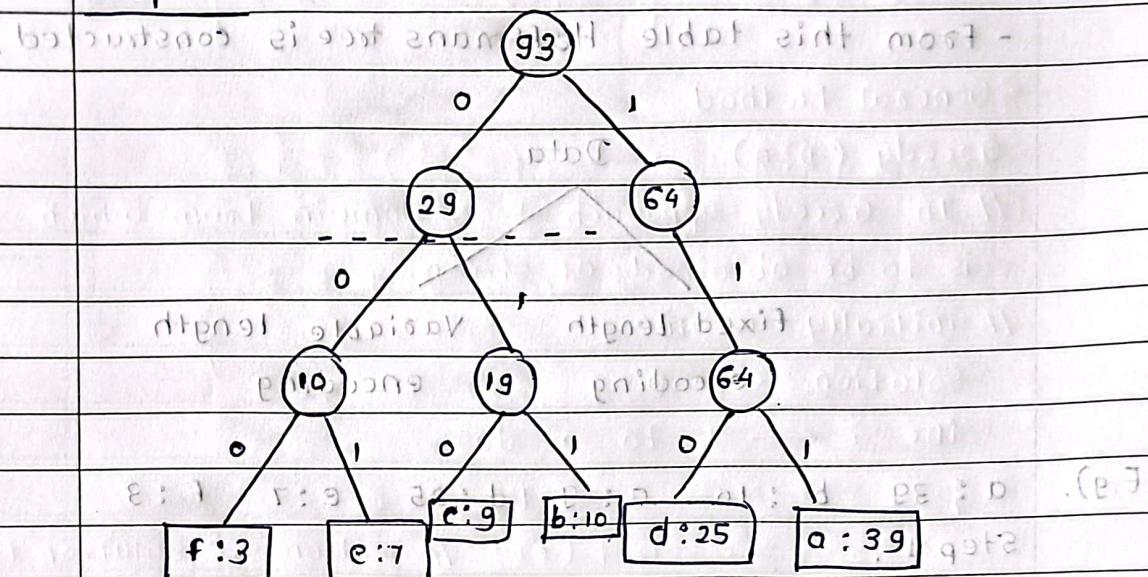
Step 1 :

i) Arrange these Symbols in ascending order.

Step 3 :



Step 4 :



As there are 6 characters $1:2^3 = 8$ combinations so tree

0 0 0 X	f
0 0 10 X	e
(10) 0 1 0	(c)
1 0 0 1 1	b

1 0 0 1 1 0 1 0 (10) 0 1 0 1 0 0 X 0 0 10 X (0 1) 1 0 0 1 1

* Activity selection using greedy approach.

- Greedy algorithm strategies builds up soln piece by piece always choosing the next piece that offers most & obvious and immediate benefits.
 - Greedy algorithm are used for optimization problem having following property.
- At Every step we can make a choice that looks based at the moment. & we get optimal solution to the complete problem.

You are given 'n' activities with their start & finish time. Select the maximum no. of activities that can be performed by single person assuming that a person can only work on single activity at a time.

- Suppose $S = \{1, 2, 3, \dots, n\}$ is the set of proposed activities, the activities share the resource which can be used by only one activity at a time. ex - Tennis court, lecture Hall
- each activity 'i' has starts & finish time where $s \leq f$
 - activities i & j are compatible if the intervals do not overlap
 - The activity selection problem chosen the maximum set of mutually consistent activities.

Given 10 activities : $S = \{A_1, A_2, A_3, \dots, A_{10}\}$ & $F = \{F_1, F_2, \dots, F_{10}\}$

$$S_i = \{1, 2, 3, 4, 6, 7, 8, 9, 9, 11, 12\}$$

$$F_i = \{3, 5, 4, 7, 10, 9, 11, 13, 12, 14\}$$

compute the schedule where the greatest no. of activities take place.

\rightarrow Maximum number of overlapping activities

Step 1 :

Find the start time and finish time of all activities

Activity	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	A_9	A_{10}
Start	1	3	5	6	7	8	9	10	11	12
Finish	3	4	5	7	9	10	11	12	13	14

start = {1, 3, 5, 6, 7, 8, 9, 10, 11, 12} = 2 overlapping activities

Solve the following :

Q.1) $n = 5$ items

w = 3 3 2 5 1

P = 10 15 10 20 8

knapsack capacity = 10 kg

Q.2) Give items a $\{ \text{Value}, \text{weight} \}$ $\{40, 20\}$ $\{30, 10\}$ $\{20, 5\}$

knapsack capacity = 20 find max value of P

Q.3) knapsack capacity = 40 find maximum value of P

assuming items to be divisible & non-divisible

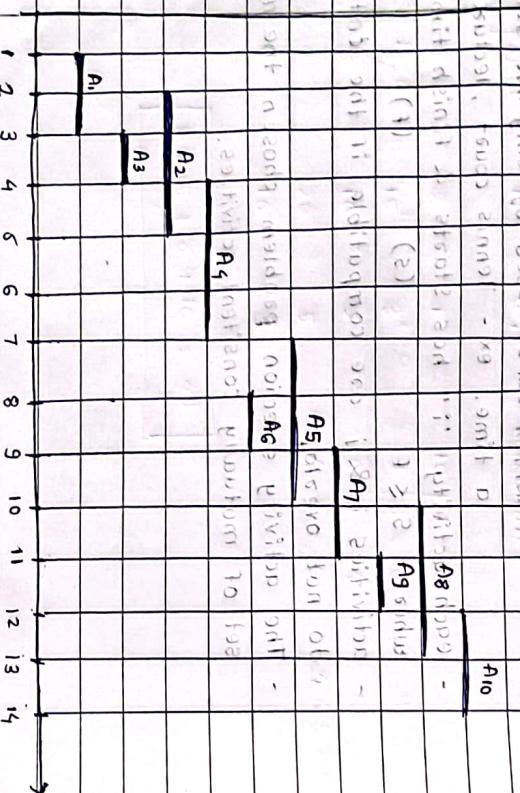
$\{60, 20\}$ $\{50, 25\}$ $\{20, 5\}$

Activity Selection

Start = {1, 3, 0, 5, 8, 5}

end = {2, 4, 6, 7, 9, 9}

A person can perform only one activity at a time.



~~~~ Step 1 : ~~Sort objects by weight~~  $\rightarrow$  1 (1) 2 (2) 3 (3) 4 (4) 5 (5)

Object 1 2 3 4 5 Profit  $\rightarrow$  10 15 20 25 30

Weight 3 3 2 5 10  $\rightarrow$  10 10 10 10 10

Profit/Weight 10/3 15/3 20/3 25/5 30/10  $\rightarrow$  3.3 5 6.7 5 3

$P_i/w_i$  3.3 5 6.7 5 3  $\cancel{4}$   $\cancel{4}$   $\cancel{4}$   $\cancel{4}$   $\cancel{4}$

Profit 10 15 20 25 30  $\rightarrow$  10 10 10 10 10

$w_i$  10 10 10 10 10  $\rightarrow$  10 10 10 10 10

$P_i/w_i$  ratio 10/10 15/10 20/10 25/10 30/10  $\rightarrow$  1 1.5 2 2.5 3

Step 2 : Find  $P_i/w_i$  ratio  $\rightarrow$  1 1.5 2 2.5 3

Step 2 : Arrange in decreasing order  $\rightarrow$  3 2 1 4 5

Step 3 :  $\cancel{1} \cancel{2} \cancel{3} \cancel{4} \cancel{5}$

Obj 5 2 3 4 1 Profit  $\rightarrow$  30 20 15 10 10

Wt 10 10 10 10 10  $\rightarrow$  10 10 10 10 10

$P_i/w_i$  30 20 15 10 10  $\rightarrow$  3 2 1 4 5

Remaining Capacity 20 10 5 0 0  $\rightarrow$  20 10 5 0 0

Profit 30 20 15 10 10  $\rightarrow$  30 20 15 10 10

$P_i/w_i$  Ratio 3 2 1 4 5  $\rightarrow$  3 2 1 4 5

Object 1 2 3 4 5 Profit  $\rightarrow$  10 15 20 25 30

Weight 10 10 10 10 10  $\rightarrow$  10 10 10 10 10

Profit/Weight 10/10 15/10 20/10 25/10 30/10  $\rightarrow$  1 1.5 2 2.5 3

$P_i/w_i$  1 1.5 2 2.5 3  $\cancel{4}$   $\cancel{4}$   $\cancel{4}$   $\cancel{4}$   $\cancel{4}$

Profit 10 15 20 25 30  $\rightarrow$  10 10 10 10 10

$w_i$  10 10 10 10 10  $\rightarrow$  10 10 10 10 10

$P_i/w_i$  ratio 10/10 15/10 20/10 25/10 30/10  $\rightarrow$  1 1.5 2 2.5 3

Step 3 :  $\cancel{1} \cancel{2} \cancel{3} \cancel{4} \cancel{5}$

Obj 5 2 3 4 1 Profit  $\rightarrow$  30 20 15 10 10

Wt 10 10 10 10 10  $\rightarrow$  10 10 10 10 10

$P_i/w_i$  30 20 15 10 10  $\rightarrow$  3 2 1 4 5

Remaining Capacity 20 10 5 0 0  $\rightarrow$  20 10 5 0 0

Profit 30 20 15 10 10  $\rightarrow$  30 20 15 10 10

$P_i/w_i$  Ratio 3 2 1 4 5  $\rightarrow$  3 2 1 4 5

Object 1 2 3 4 5 Profit  $\rightarrow$  10 15 20 25 30

Weight 10 10 10 10 10  $\rightarrow$  10 10 10 10 10

$P_i/w_i$  10/10 15/10 20/10 25/10 30/10  $\rightarrow$  1 1.5 2 2.5 3

$$8 + 15 + 10 + 15 = 48$$

$\rightarrow$  48

Q.3)

Step 1 : Ratio  $P_i/w_i$ 

$$P_i \quad 60 \quad 50 \quad 20$$

$$w_i \quad 20 \quad 25 \quad 15$$

$$P_i/w_i \quad 3 \quad 2 \quad 4$$

Step 2 : Descending order

$$P_i \quad 20 \quad 60 \quad 50$$

$$w_i \quad 5 \quad 20 \quad 15$$

Step 3 :

| $P_i$ | $w_i$ | Remaining Capacity | Profit |
|-------|-------|--------------------|--------|
|-------|-------|--------------------|--------|

$$20 \quad 5 \quad 40 - 5 = 35 \quad 20$$

$$60 \quad 20 \quad 35 - 20 = 15 \quad 60$$

$$50 \quad 15 \quad 15 - 15 = 0 \quad 0$$

$$\text{divisible} = 20 + 60 + 30 = 110$$

$$\text{Non-divisible} = 60 + 20 = 80$$