1.) Name : Venkatesh.G.D

2.) USN : 2GI19CS175

3.) BE/MBA : BE

4.) Semester : 4

5.) Course Name : Discrete Math

6.) Course Code : 18MATCS41

7.) Name of Colg : KLS GIT

8.) Date & Time : 19-07-21 ; 3:00PM

9.) Mobile No : 9972287030

10.) Signature :

4.) Pigeonhole Principle.

Statement: If "m" pigeons occupy "n" pigeonhole &, $m > n$, then two or more pigeons occupy the same pigeon hole.

Generalised pigeonhole principle

[ ] - Greatest integer function.

Statement: If 'm' pigeons occupy 'n' pigeonholes then atleast one pigeonhole must contain $\left[\frac{m-1}{n}\right]+1$ or more pigeons.

Proof: We prove this principle by method of contradicti

$p \to q \equiv \sim q \to \sim p$

Suppose each pigeon hole contains no more than $\left[\frac{m-1}{n}\right]+1$ pigeons $(\sim q)$

* Total number of pigeons in 'n' pigeonholes is less than equal to

$$n \cdot \left[\frac{m-1}{n}\right] \leq \cancel{m} < \frac{m-1}{n} = m-1$$

* Total number of ~~pigeonholes~~ pigeons in 'n' pigeonholes is less than or equal to $m-1$ $(\sim p)$

This contradicts our fact that 'm' pigeons.

$\therefore$ At least one pigeonhole must contain at least $\left[\frac{m-1}{n}\right]+1$ or more pigeons. $(p \to q)$

$\because (p \to q \equiv \sim q \to \sim p)$

Proof completed.

2.) $f(x) = 3x+4 \quad f: R \to R \to ①$

$g(x) = \frac{1}{3}(x-4) \quad g: R \to R \to ②$

$(g \circ f)(x) = g(f(x)) = g(3x+4)$

let $(3x+4) = x$

$= g(x)$

$= \frac{1}{3}(x-4) \to$ from ②

substitute x back

$= \frac{1}{3}(3x+4-4)$

$= \frac{3x}{3}$

$= x$

$g(f(x)) = x \to ③$

And, $(f \circ g)(x) = f(g(x)) = f\left(\frac{1}{3}(x-4)\right) \to$ from ②

$= f(x')$ let $x' = \frac{1}{3}(x-4)$

$= 3x' + 4$ from ①

Substitute x back

$= 3\left(\frac{1}{3}(x-4)\right) + 4$

$= x-4+4$

$= x.$

$f(g(x)) = x \to ④$

$\therefore$ From ③ & ④ its clear that g is inverse of f

3)

i) $S(6,4)$

$$S(m,n) = \frac{1}{n!}\sum_{k=0}^{n} (-1)^k \, {}^nC_k (n-k)^m$$

$m=6, \ n=4$

$$= \frac{1}{4!}\sum_{k=0}^{n} (-1)^k \, {}^4C_k (4-k)^6$$

$$= \frac{1}{4!}\Big[(-1)^0 \cdot {}^4C_0 (4-0)^6 + (-1)^1 \cdot {}^4C_1 \cdot 3^6 + (-1)^2 \cdot {}^4C_2 \cdot 2^6 +$$

$$(-1)^3 \cdot {}^4C_3 (1)^6 + {}^4C_4 \cdot (0)\Big]$$

$$= \frac{1560}{4!}$$

$$\boxed{= 65}$$

ii) $S(8,5)$

$$= S(m,n) = \frac{1}{n!}\sum_{k=0}^{n} (-1)^k \, {}^nC_k (n-k)^m$$

$$= \frac{1}{5!}\Big[(-1)^0 \cdot {}^5C_0 \cdot (5)^8 + (-1) \cdot {}^5C_1 \, 4^8 + (-1)^2 \, {}^5C_2 \cdot 3^8 + (-1)^3 \cdot {}^5C_3 \cdot 2^8$$

$$+ (-1)^4 \cdot {}^5C_4 \cdot 1^8 + 0\Big]$$

$$= \frac{126000}{5!}$$

$$\boxed{= 1050}$$

5) $a_n - 8a_{n-1} + 16a_{n-2} = 0 \rightarrow (1)$

Order $\rightarrow (n-(n-2)) = 2$

To get characteristic auxiliary eqn of (1)

$a_n = \alpha^2$, $a_{n-1} = \alpha$, $a_n = \alpha^0 = 1$

$\alpha^2 - 8\alpha + 16(1) = 0$

$(\alpha - 4)^2 = 0$

$\alpha_1 = \alpha_2 = 4$

Roots are real & equal

$\therefore$ Solution of (1) is given by,

$a_n = (A_1 + nA_2)\,\alpha_1$

$a_n = (A_1 + nA_2)\,4^n \longrightarrow (2)$

Given $\rightarrow a_2 = 6,\ a_3 = 80$

$a_2 = (A_1 + 2A_2)4^2 = 6$

$A_1 + 2A_2 = \frac{6}{16}$ ---- (ii)

$a_3 = A_1 + 3A_2 = \frac{80}{43}$ ---- (iii)

(iii) - (ii)

$A_2 = \frac{80}{64} - \frac{6}{16}$

$= \frac{56}{64}$

$A_2 = 0.875$

$\therefore A_1 = \frac{6}{16} - 2A_2 = \frac{6}{16} - 2(0.875)$

$A_1 = -1.375$

5) Put A_1, A_2 in (2)

$$a_n = \left(\frac{-11}{8} + n \cdot \frac{7}{8}\right) 4^n$$

$$\boxed{a_n = \left(\frac{7n - 11}{2}\right) 4^{n-1}}$$

---

6)

7)

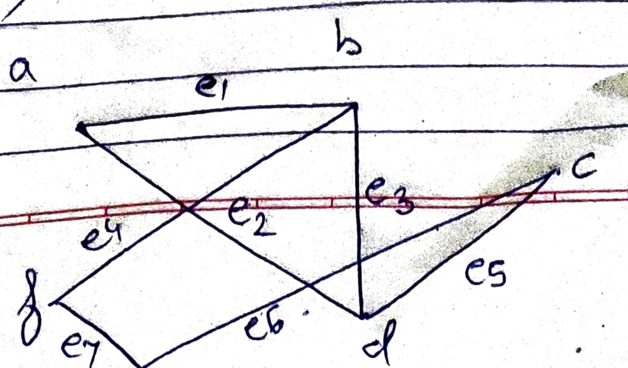
i) Graph: A graph consists of a finite set of vertices V a finite set of edges E & a function that assigns to each edge an ordered or unordered pair (V_1, V_2) where V_1 & V_2 are vertices & graph is denoted by G or G(V, E)

ii) Degree of vertex: Degree of vertex V is denoted by deg(V) or d(V) is the number of edges connected to it.

iii) Pendent vertex: A vertex is said to be pendent vertex or end vertex if its degree 1.

iv) Adjacent vertex: 2 vertices are said to be adjacent vertices if they are connected by an edge

v) $G(V, E) \Rightarrow$



6.) **Divide & Conquer algorithm**

* It works recursively breaking down or dividing a problem into 2 or more sub problems of same or related type until they become simple to solve directly

* A typical D&CA solves a problem using following steps
1) Divide : Divide/break problem into sub problems
2) Conquer : Recursively solve subproblems
3) Combine : Combines solution of subproblems

* The idea of divide & conquer algorithm
Given a problem P of size $n = 2^k$

* Algorithm DAC(P)
- If n is small solve it
else
- divide P into 2 sub probs of size $n/2$
- DAC(P_1) } sol^n of P_1 & P_2 can be formed
- DAC(P_2) }
- Combine solution of sub-problems P_1 & P_2 to solution of P

* Time to solve P

$$\underset{P}{T(n)} = \underset{P_1}{T\left(\frac{n}{2}\right)} + \underset{P_2}{T\left(\frac{n}{2}\right)} + f(n)$$

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

6) Let $f(n)$ be additional cost of combining
In general.

$$T(n) = aT\left(\frac{n}{b}\right) + f(n) \quad \text{--- } (2) \quad (a \leq b).$$

a = Terms in consideration
b = Total no. of divisions

This is the RR for D & C Algorithm