

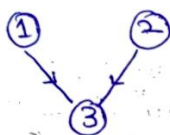
What is a graph?

A graph G is an ordered pair $G = (V, E)$ where V is a ~~Group~~ (finite) set of elements and E is set of 2-subsets of V .

Ex: $V = \{1, 2, 3\}$ (vertices)
 $E = \{\{1, 2\}, \{2, 3\}, \{1, 3\}\}$

Simple graphs: no loop
 no multiple edges

directed graph: Each edge has a direction associated to it



$(1, 3)$ $(2, 3)$

→ These brackets for ordered pair

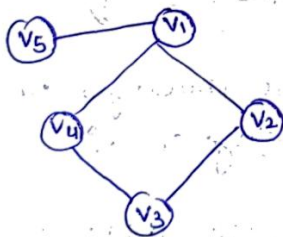
Note: $|V|$ = order of the graph ✓
 $|E|$ = size of the graph ✓

Examples of Graph

$G = (V(G), E(G))$
 vertices edges

$\{u, v\} \in E$
 $uv = vu$ } undirected graph

Ex: $V = \{v_1, v_2, v_3, v_4, v_5\}$
 $E = \{v_1 v_2, v_2 v_3, v_3 v_4, v_4 v_1, v_1 v_5\}$

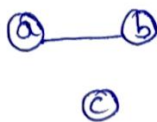


$e = uv$

u and v are ends of the edge e u and v are adjacent (neighbours)

if $e_1 = uv$ and $e_2 = uw$ [i.e. they have a common end]
 then e_1 and e_2 are adjacent

Ex: $V = \{a, b, c\}$
 $E = \{ab\}$



Eg: of disconnected graph.

Families of Graph

K_1



K_2



A complete graph K_n on n vertices is a simple graph with an edge b/w every pair of vertices



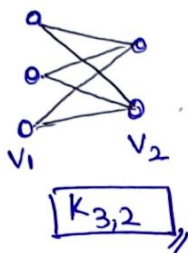
$$|E| = {}^n C_2$$

Empty graph

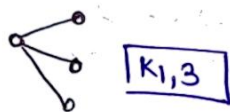
- o o $|V| \neq \text{null}$
- o o $|E| = \text{Empty}$

This is an empty graph on 4 vertices

Bipartite Graph:- A graph whose vertex set can be partitioned into 2 sets V_1 and V_2 such that every edge $uv \in E$ has $u \in V_1$ and $v \in V_2$



↓
Complete bipartite Graph: It has every possible edge between the 2 sets of vertices $K_{n,m}$.



if any of the parameters n or m is 1 then that graph is called a Star graph

↓
unbalanced



Path: A path (P_n) is a graph whose vertices can be arranged in a sequence



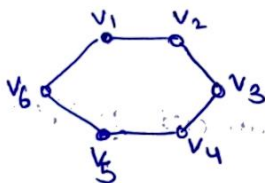
$V_1, V_2, \dots, V_n$
Such that the edge set is

$$E = \{V_i V_{i+1} \mid i=1, 2, \dots, n-1\}$$

Cycle graph

A cycle graph (C_n) is a graph whose vertices can be arranged in a cyclic sequence $(v_1, v_2, \dots, v_n)$ such that the edge set is

$$E = \underbrace{\{V_i V_{i+1} \mid i=1, 2, \dots, n-1\}}_{\text{Path}} \cup \underbrace{\{V_n V_1\}}_{\text{for cycle}}. \text{ For cycle } \# \text{ of vertices } n \geq 3$$



Cycle C_6

$n=3$

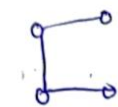


Triangle C_3, K_3

Connected Graph

A graph G is connected if, for every pair of distinct vertices $u, v \in V$, there is a path from u to v in G .

Eg:



connected graph



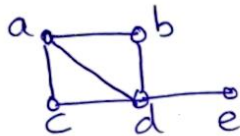
disconnected graph



disconnected graph

Regular Graph

Neighbourhood of a graph



Let v be a vertex in a graph G .
The (Open) neighbourhood of v in G is

$$N_G(v) = \{u \mid uv \in E(G)\}$$

closed Neighbourhood

$$N_G[v] = N_G(v) \cup \{v\}$$

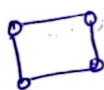
$$N_G(a) = \{b, c, d\}$$

$$N_G[a] = \{a, b, c, d\}$$

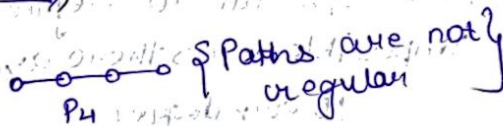
The degree of a vertex $v \in V(G)$ is

$$\deg_G(v) = |N_G(v)|$$

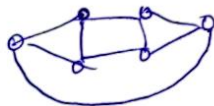
A graph G is α -regular if $\deg_G(v) = \alpha$ for all $v \in V(G)$



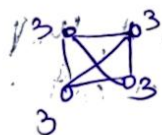
Cycles are 2-regular



Paths are not regular



3-regular (cubic)



K_4 is a 3-regular (cubic)

$$K_n \rightarrow (n-1) \text{ regular}$$

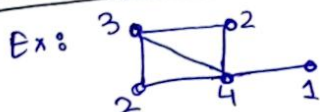
min-degree: ($\delta(G)$)

max-degree: ($\Delta(G)$)

In α -regular graph

$$\delta(G) = \Delta(G) = \alpha$$

Theorem 1: The sum of the degrees is twice the number of edges



$$1+2+2+3+4 = 12$$

$$|E| = 6$$

Theorem: In any graph G [undirected]

$$\sum_{v \in V} \deg(v) = 2|E(G)|$$

Proof

Let $e \in E(G)$ be any edge. Then e has 2 ends say u and w (So, $e = uw$).

When we sum the degrees of the vertices, edge e gets counted twice
once in $\deg(u)$ term &
once in $\deg(w)$ term

Similarly every edge gets counted twice.

Corollary: In any graph there are an even number of odd degree vertices

Proof: Let $G = (V, E)$

Suppose $V = \{v_1, \dots, v_n\}$

Previous theorem $\Rightarrow \underbrace{\deg(v_1) + \dots + \deg(v_n)}_{\text{must be even}} = \underbrace{2|E(G)|}_{\text{even}}$

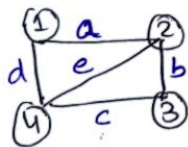
So there cannot be an odd number of odd degrees

Q Find a graph with 10 degree 2 vertices and one degree 3 vertex.

Impossible. as there are 1 odd degree vertices and 10 even degrees

$$\underbrace{|V_1 \dots + v_{10}|}_{\text{Even}} + \underbrace{|v_{11}|}_{\text{odd}} = \text{odd} \neq 2|E| \in \text{even.}$$

Representing Graphs using matrices



Adjacency matrix

	1	2	3	4
1	0	1	0	1
2	1	0	1	1
3	0	1	0	1
4	1	1	1	0

$$A_{ij} = \begin{cases} 1 & \text{if } ij \in E \\ 0 & \text{otherwise} \end{cases}$$

rows & columns represent vertices

row sum or column sum = degree of vertex that the row represents

Incidence matrix

↑ degree.

$$M = \begin{matrix} & \begin{matrix} a & b & c & d & e \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \end{matrix} & \begin{bmatrix} 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \end{bmatrix} \end{matrix}$$

rows represent vertices columns represent edges

$$m_{ij} = \begin{cases} 1 & \text{if vertex } i \text{ belongs to edge } j \\ 0 & \text{otherwise} \end{cases}$$

row sum = degree

column sum = 2

note: if the graph contain
Self loop in that case

$$m_{ii} = 2 //$$

Problem Set

Q1 let G be a simple graph and $m = |E(G)|$. Show that $m \leq \binom{n}{2}$ and determine when equality holds.

Notations: $\binom{n}{k} = {}^nC_k = \frac{n!}{(n-k)!k!} = \# \text{ of } k\text{-subsets of a set of size } n$

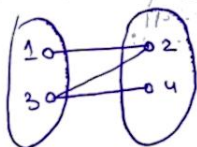
$$\binom{n}{2} = {}^nC_2 = \# \text{ of 2 subset of an } n\text{-set}$$

= Size of the set of all possible edges in a simple graph on n vertices

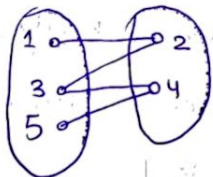
$\therefore m \leq \binom{n}{2}$ and $m = \binom{n}{2}$ when G is the complete graph K_n

Q2 Prove that every path is bipartite

P_4 : 1-2-3-4



P_5 : 1-2-3-4-5



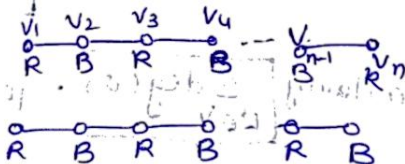
Proof: let P_n be the path $[v_1, v_2, \dots, v_n]$ where $E(P_n) = \{v_i v_{i+1} \mid i=1, \dots, n-1\}$

Define $X = \{v_i \mid i = \text{odd}\}$

$Y = \{v_i \mid i = \text{even}\}$

odd

even



In either case, every edge of P_n is of the form $v_i v_{i+1}$ where i and $i+1$ are of different parties

$\therefore$ every edge has one end in X and the other end in Y .

3.] For $k=0,1,2$ characterise the k -regular graphs

$k=0$

• •

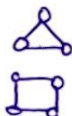
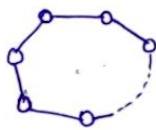
$\underbrace{\circ \circ \circ \dots \circ}_{n \text{ vertices}}$
0 edges

A graph is 0-regular $\iff$ it is an empty graph.

$k=1$ $\begin{array}{c} \circ - \circ \\ \circ - \circ \\ \circ - \circ \end{array} \} k_2$

A graph is 1-regular $\iff$ it is the disjoint union of k_2 's

$k=2$



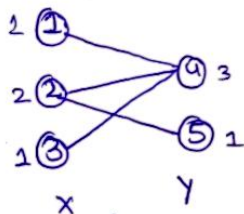
A graph is 2-regular $\iff$ it is disjoint union of cycles of any lengths

4.] Let G be a bipartite graph with partite sets X and Y

Prove that:

$$\sum_{v \in X} \deg(v) = \sum_{v \in Y} \deg(v)$$

Ex:



$$3+1=4$$

$$\# \text{ edges} = 4$$

$$1+2+1=4$$

Proof: $\sum_{v \in X} \deg(v) = |E(G)|$ since each edge that is incident with a vertex $v \in X$ has its other end in Y so is counted exactly once in the sum

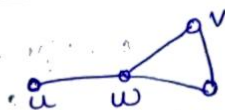
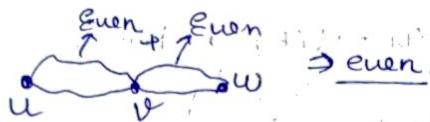
$$\sum_{v \in X} \deg(v)$$

Similarly $\sum_{v \in Y} \deg(v) = |E(G)|$

5] True / False

If $u, v, w \in V(G)$ and there is an even length path from u to v and there is an even length path from v to w then there is an even length path from u to w . : False

Counter Example



There is an even length path from v to w and no even length path from u to w .

Isomorphisms

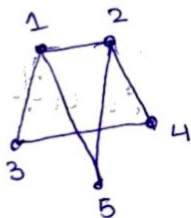
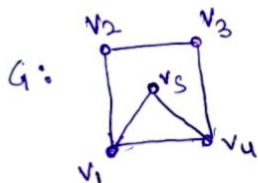
Two simple graphs G and H are isomorphic if there is a bijection

$$\theta: V(G) \rightarrow V(H)$$

which preserves adjacency and non-adjacency

$$uv \in E(G) \iff \theta(u)\theta(v) \in E(H)$$

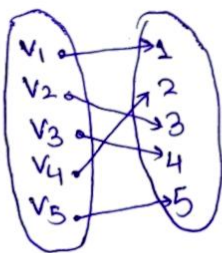
Notations: $G \cong H$ means G and H are isomorphic



$$\theta: \begin{pmatrix} v_1 & v_2 & v_3 & v_4 & v_5 \\ 1 & 3 & 4 & 2 & 5 \end{pmatrix}$$

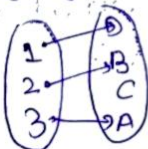
Consider edge $v_1v_2 \in E(G)$

$$\theta(v_1)\theta(v_2) = 13 \in E(H)$$

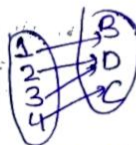


$$V(G) \rightarrow V(H)$$

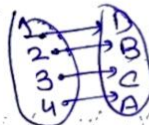
A bijection is a map from a set A to a set B that is one-to-one (injective) and onto (surjective)



one to one
(unique image)
Injective

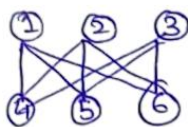


All elements of
Domain are
involved.
onto
Surjective

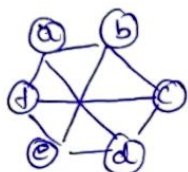


one-to-one and
Onto
Surjective +
Injective $\equiv$
Bijective

Isomorphic or not?



6 vertices
9 edges
3-regular
Bipartite



6 vertices
9 edges
3-regular
Bipartite

- Same # of vertices
- Same # of edges
- Structural similarities or differences
 - ↳ R-regular
 - ↳ Bipartite

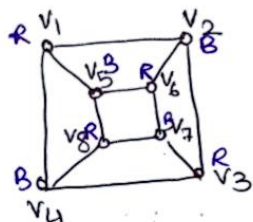
• Define $\theta: V(G_1) \rightarrow V(G_2)$ as follows

$$\theta: \begin{pmatrix} \overset{R}{1} & \overset{R}{2} & \overset{R}{3} & \overset{B}{4} & \overset{B}{5} & \overset{B}{6} \\ \hline \underset{R}{d} & \underset{R}{e} & \underset{R}{f} & \underset{B}{c} & \underset{B}{a} & \underset{B}{b} \end{pmatrix}$$

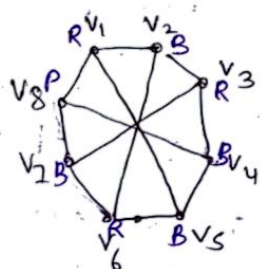
$$\sigma: \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ \hline a & c & e & d & f & b \end{pmatrix}$$

θ is onto $\because$ each vertex in H gets mapped to something in G_1
 θ is one-one $\because$ no two vertex of G_1 map to same vertex of G_2
 $\therefore \theta$ is bijection

$14 \in E(G_1)$
 $\theta(1)\theta(4) = \boxed{d c} \in E(G_2)$ $\therefore$ true



8 vertices
12 edges
3-regular graph
Bipartite



8 vertices
12 edges
3-regular
not Bipartite
mutually non adjacent vertices
 $\{V_1, V_7, V_4\}$
count: 3

mutually non adjacent vertices

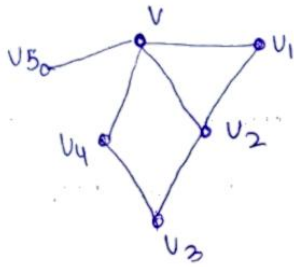
$\{V_1, V_8, V_3, V_6\}$
count: 4

Therefore non isomorphic

$\therefore G_1 \neq H$

Neighbours of a vertex

$$N_G(v) = \{u \in V(G) \mid uv \in E(G)\}$$

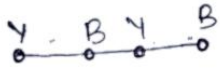


$$N_G(v) = \{u_1, u_2, u_4, u_5\}$$

$$d(v) = 4 = |N_G(v)|$$

$$\delta(G) = \min \{d(v) \mid v \in G\}$$

$$\Delta(G) = \max \{d(v) \mid v \in G\}$$



Is P_4 Bipartite?

method:

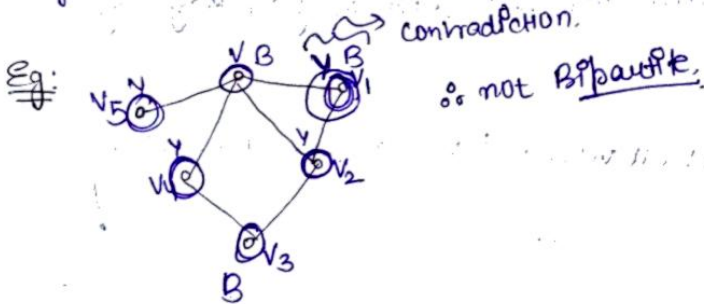
→ start from any vertex v , and colour it colour 1

→ Colour neighbours of v , with colour 2

→ Proceed by colouring all neighbours of an already coloured vertex with the opposite colour

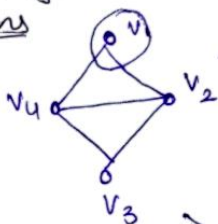
Note: If the Graph is bipartite we can complete this colouring with no contradiction

If it is not bipartite then at some point a vertex will get both colours.

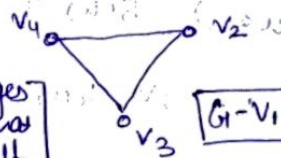


Spanning and induced Subgraphs

Subgraphs



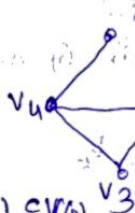
vertex deletion
[remove all edges incident with this vertex as well]



$$G - v_1$$

induced Subgraph

Edge-deletion



$$G \setminus e$$

spanning sub graph representation

A Graph F is Subgraph of Graph G iff

- $V(F) \subseteq V(G)$
- $E(F) \subseteq E(G)$

If F is subgraph of G we write $F \subseteq G$ & we can say F is contained in G or G contains F .

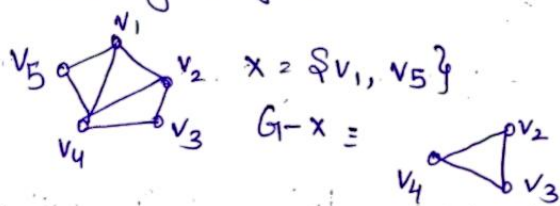
Spanning Subgraph

Subgraph obtained only by edge deletions, in other words the vertex set of the subgraph is the entire vertex set of the original graph.

Induced Subgraph

Subgraph obtained only by vertex deletion is called an induced subgraph.

Graph G and X is the set of vertices that are deleted, the resulting subgraph is $G - X$.



[we are interested in the set of vertices]

$$V = V \setminus X$$

$$V = \{v_2, v_3, v_4\}$$

$G[V]$ = Graph induced by the vertices in V .

The graph with vertex set V and edge set consists of all edges of G which have both ends in V .

Theorem: Let G be a graph in which all vertices have degree at least 2. Then G contains a cycle.

Case 1: If G is not simple

→ loop

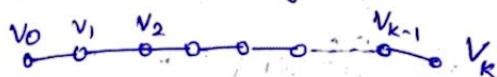
$$u \text{ loop } \delta(u) = 2$$

→ parallel edges

$$u \text{ --- } v \text{ --- } u, \delta(u) = \delta(v) = 2$$

Case 2: Assume G is a simple graph

Proof: let P be a longest path in G where $P = [v_0, v_1, v_2, \dots, v_{k-1}, v_k]$



Since $d(v_k) \geq 2$

There is a vertex $v \in V(G)$ where $v \neq v_{k-1}$ and it is adjacent to v_k

If v does not belong to the path P $v \neq v_0, v_1, \dots, v_{k-2}$, then we have a longer path in G

$$P' = [v_0, v_1, \dots, v_{k-1}, v_k, v]$$

But this is impossible as P is the longest path.

Then v must be one of the vertices of path P [other than v_{k-1}]

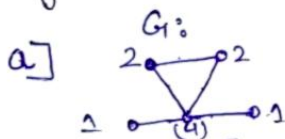
$$v = v_i \text{ for some } i \text{ where } 0 \leq i \leq k-2$$

So, Then there is a cycle in the graph

$(v_i, v_{i+1}, \dots, v_{k-1}, v_k)$ is a cycle in graph G .

Problems

1) Are the following pairs of graphs isomorphic or not? If they are give the isomorphism. If not, explain why?



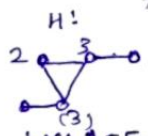
$$|V(G_1)| = 5$$

$$|E(G_1)| = 5$$

of vertex with $\deg(v) = 2$

vertex with $\deg(v) = 2$

$$\Delta \deg = 4$$



$$|V(H_1)| = 5$$

$$|E(H_1)| = 5$$

vertex with $\deg(v) = 2$

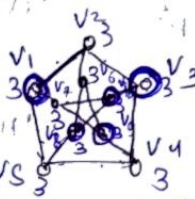
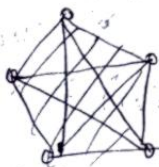
vertex with $\deg(v) = 1$

$$\Delta \deg = 3$$

$$G_1 \not\cong H_1$$

$\therefore$ not isomorphic

b)

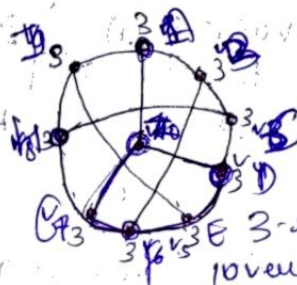


3-regular

10 vertices

15 edges

non adjacent set: $\{v_1, v_3, v_5\}$



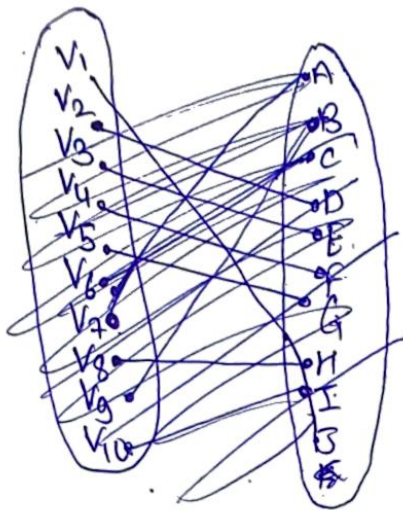
3-regular

10 vertices

15 edges

$\{v_1, v_3, v_5, v_7, v_9\}$

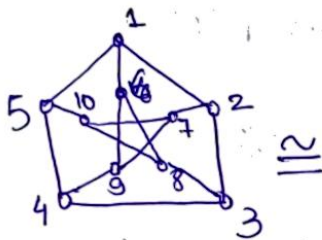




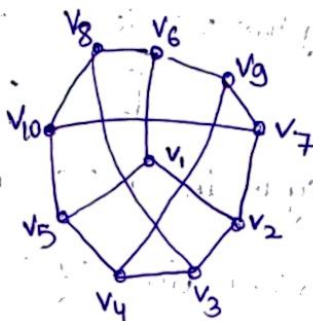
EG: (12345)

(681079)

16, 27, 38, 49, 5(10)



$\cong$



2] Show that any isomorphism between 2 graphs maps each vertex to a vertex of the same degree.

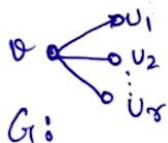
$\sigma: V(G_1) \rightarrow V(H)$ (bijection) such that
 $uv \in E(G_1) \iff \sigma(u)\sigma(v) \in E(H)$

Proof

Suppose σ is an isomorphism from G_1 to H

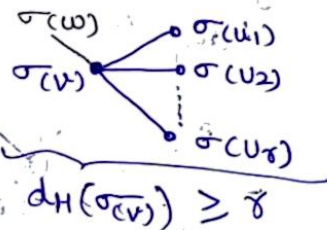
Let $v \in V(G_1)$ say $r = d_{G_1}(v)$

Let $u_1, u_2, \dots, u_r$ be the neighbours of v in G_1



$vu_i \in E(G_1)$
for $i=1, 2, \dots, r$

$\xrightarrow[\text{Graph}]{\text{isomorphic}}$



$d_H(\sigma(v)) \geq r$

If $\sigma(w) \in V(H)$ with $\sigma(v)\sigma(w) \in E(H)$ then $vw \in E(G_1)$ which is not true

So, $d_H(\sigma(v)) = r$, Thus v gets mapped to $\sigma(v)$ which has the same degree as v

3] Prove that C_n is bipartite iff n is even

Proof:

Statement 1: C_n is bipartite iff n is even

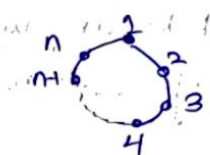
Proof:

* Statement 1: if n is even then C_n is bipartite.

Proof:

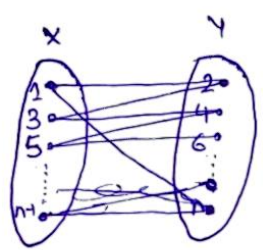
Suppose n is ~~odd~~ even.

Label the vertices of C_n as $(1, 2, 3, \dots, n)$



$X = \{i | i \text{ is odd}\} \Rightarrow \text{Colour Blue}$

$Y = \{i | i \text{ is even}\} \Rightarrow \text{Colour Yellow}$



$\therefore$ we have partition the vertices into X and Y .
 $\therefore C_n$ is a bipartite graph with partite set X and Y if n is even.

$\therefore$ Hence proved.

* Statement 2: if C_n is bipartite then n is even

Proof by Contrapositive: if n is not even $\rightarrow$ then C_n is not bipartite.
 i.e. if n is odd $\rightarrow$ then C_n is not bipartite.

Proof:

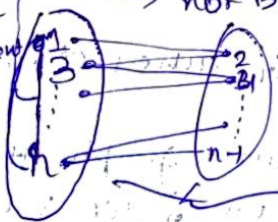
let $n = \text{odd}$.

Label the vertices of C_n as $(1, 2, \dots, n)$

$X = \{i | i \text{ is odd}\} \rightarrow \text{Colour Blue}$

$Y = \{i | i \text{ is even}\} \rightarrow \text{Colour Yellow}$

$\therefore n$ is odd $\rightarrow$ not bipartite.



$\therefore n$ is present in X same as vertex 1 which are neighbours and are having same colour: $\therefore$ not bipartite

decomposition not

Principle of mathematical Induction

Let $P(n)$ be a statement involving the integer n .

IF $\textcircled{S}$ the statement is true then $n=1$ and
Basis $\rightarrow$ whenever the statement is true for $n=k$ it is also true for $n=k+1$

then the statement is true for all the integers $n \geq 1$

4] let $n \geq 4$ be an even integer, show that there exists a 3-regular graph of order n

any 3-regular graph must have even order i.e. even number of vertices

$$\sum_{v \in V(G)} d(v) = 2|E| = \text{even}$$

Even # of vertices of odd degree

Proof

Basis $n=4$ there exist a 3-regular graph

 K_4 is a 3-regular graph of order 4.

To show: (Induction Hypothesis)

If there is a graph that is 3-regular of order k where k is even, then there is a 3-regular graph of order $k+2$

Induction Hypothesis:

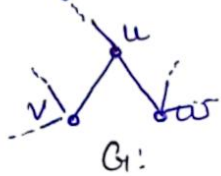
Suppose there is a 3-regular graph of order k where k is even.

Note: using this prove that there exist a 3-regular graph for Order $(k+2)$.

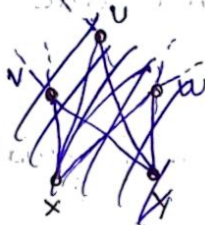
manipulate G until we get G' where order of $G' = k+2$

let $u \in V(G)$, since G is 3-regular graph therefore the $d(u) = 3$ i.e. it has 3 neighbours, let $v, w \in V(G)$ which are neighbours of u it means $\boxed{uv, uw \in E(G)}$

Diagrammatically



III



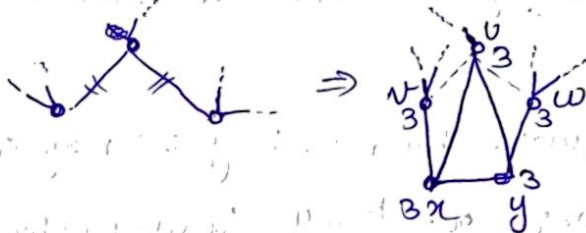
construct a new graph G_1' from G_1 as follows

1) add 2 new vertices, x and y to G_1

$$\therefore |V(G_1')| = |V(G_1)| + 2 = (k+2)$$

2) remove the edges uv, uw and connect u to x and y and v to x and y and w to x and y i.e. vx, ux, vy, wy, uy .

$$E(G_1') = (E(G_1) \setminus \{uv, uw\}) \cup \{ux, vx, uy, vy, wy, wy\}$$

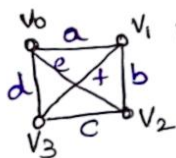


G_1' is also 3 regular graph of order $k+2$

$\therefore$ By the PMI, the result holds for even integers $n \geq 4$

Walk Trail and Paths

1] Walk: A walk in a graph G_1 is a sequence $W = v_0 e_1 v_1 e_2 v_2 \dots v_{k-1} e_k v_k$ whose terms alternate b/w vertices and edges (not necessarily distinct) such that $v_{i-1} v_i = e_i$ for $1 \leq i \leq k$



$$W = v_0 a v_1 b v_2 c v_3 d v_1 a v_0 e v_2 c v_3 d v_0$$

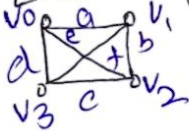
NOTE: when G_1 is simple, we write the walk by indicating the vertices only.

$$W: v_0 v_1 v_2 v_3 v_1 v_0 v_2 v_3 v_0$$

length of the walk: number of edges of the walk.

2] Trail: it is a walk such that all edges are distinct.

[edge not repeated]
[vertices may be repeated]



$$T: v_1 v_2 v_0 v_1 v_3$$

$$\# \text{ edges} \leq \text{length of trail}$$

3] Path: path is a walk such that all the vertices and edges are distinct.

$$P: [v_1, v_2, v_3, v_4]$$

length of path: # of edges

Note:

A walk is closed if the initial and terminal vertices are identical

If there is a walk from x to y we may call it an xy-walk

Another notation: $x \xrightarrow{k} y$ is a walk of length k

if length is not known: then,

$x \xrightarrow{*} y$ is a walk from x to y

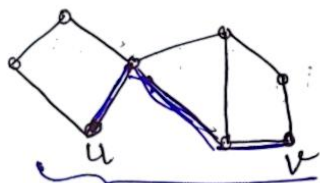
Distance b/w vertices ($d_G(u, v)$)

length of the shortest path b/w u and v

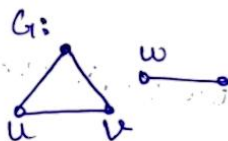
If there is a walk and hence a path from u to v in G

let

$$d_G(u, v) = \min \{k \mid u \xrightarrow{k} v\} \text{ distance b/w } u \text{ and } v$$



$$d_G(u, v) = 3$$



$d_G(u, w) = \infty$ by convention when there is no walk $u \xrightarrow{*} w$

Connected Graph

A graph G is connected if $d_G(u, v) < \infty$ for all $u, v \in G$ and disconnected otherwise

The maximal connected subgraphs of G are its connected components
 $C(G) = \#$ of connected components of G

If $C(G) = 1$ then G is connected.

maximality condition means that a subgraph $H \subseteq G$ is connected subgraph and for every $v \in V(G)$ $v \notin H$ $G - [V(H) \cup \{v\}]$ is disconnected.

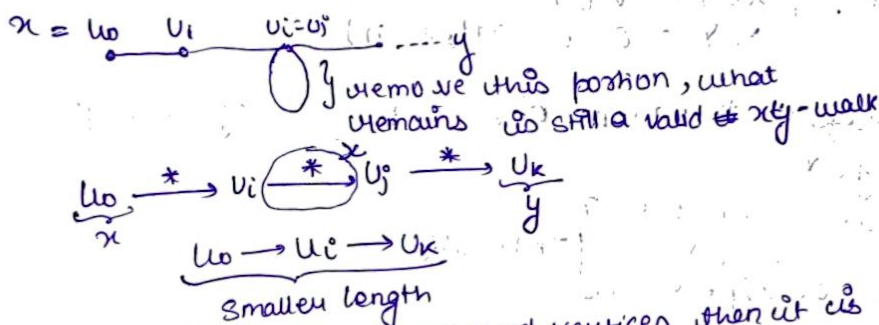
5.] If $x, y \in V(G)$ are distinct vertices of G then every xy -walk in G contains an xy -path in G .

Proof:

Let $w: x = \underbrace{u_0, u_1, u_2, \dots, u_k}_{xy\text{-walk}} = y$

If w has no repeated vertices then w is in fact an xy -path.

Suppose w has a repeated vertex $v_i = v_j$ ($i < j$)



- if it has no repeated vertices then it is the xy-path
- otherwise repeat the process.

6.] A graph G is bipartite iff it contains no odd cycle.

$\Rightarrow$ G is bipartite $\rightarrow G$ has no odd cycle [1st part]

Proof By contradiction

Suppose G is bipartite.

Suppose G_1 is bipartite.
So $V(G_1) = X \cup Y$ and $xny = \emptyset \quad \forall e \in E(G_1) \quad e = xy \text{ where } x \in X, y \in Y$

- Suppose G has at least 1 odd cycle $(v_1, v_2, \dots, v_n)$ where $n = \text{odd}$

Suppose G has n vertices.
Let $C = (v_1, v_2, \dots, v_n)$ where n is odd. i.e. C is an odd cycle.

$v_1 \in X$ then $v_2 \in Y$

$$v_i \in \begin{cases} X & i = \text{odd} \\ Y & i = \text{even} \end{cases}$$
$$\circ \circ \forall n \in X \text{ as } n = \text{odd}$$

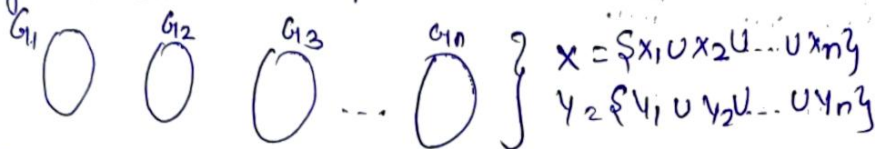
$n \in X$ as $n = \text{odd}$
now (v_1, v_n) belongs to X and $v_n v_1 \in E(G)$
 $\therefore G$ is not bipartite

$\in E(G)$
 $\therefore G$ is not bipartite

$\Leftarrow$] no odd cycles $\Rightarrow$ bipartite

Suppose all cycles in G have even length

* If G is disconnected, then any cycle in G is contained in one of the connected components.



It suffices to show the claim for connected graphs.
 Assume G is connected.

Let $v \in V(G)$

$X = \{x \mid d_G(v, x) \text{ is even}\}$ $d_G(v, u) = \text{distance b/w } v \text{ \& } u$

$Y = \{y \mid d_G(v, y) \text{ is odd}\}$

$\because G$ is connected, $X \cup Y = V(G)$
 $X \cap Y = \emptyset$ as (odd, even)

Need to show any edge in the graph G , $e \in E(G)$ is of the form
 $e = xy$ $x \in X, y \in Y$

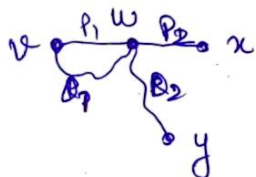
* Need to show that $\left. \begin{array}{l} xy \in E(G) \\ x, y \in X \\ \text{or } x, y \in Y \end{array} \right\} \text{Impossible}$

Let $x, y \in X$ $\therefore$ goal: to show $xy \notin E$ [Proof by contradiction]

Let $P: v \xrightarrow{*} x$ be a shortest path from v to x .

$Q: v \xrightarrow{*} y$ be a shortest path from v to y

$\because xy$ belongs to same set $X \therefore P, Q$ have same parity [odd/even]



Let $w = \text{last common vertex of } P \text{ \& } Q$

$P_1: v \xrightarrow{*} w$ $P_2: w \xrightarrow{*} x$ $P = P_1 P_2$

$Q_1: v \xrightarrow{*} w$ $Q_2: w \xrightarrow{*} y$ $Q = Q_1 Q_2$

$\because P$ and Q are shortest paths, it follows that
 $|P_1| = |Q_1|$

Then P_2 and Q_2 have the same parity.

Then the path $x \xrightarrow{P_2^{-1}} w \xrightarrow{Q_2} y$ has even length
 $\hookrightarrow \begin{array}{l} P_2, Q_2 \text{ odd} + \text{odd} = \text{even} \\ P_2, Q_2 \text{ even} + \text{even} = \text{even} \end{array}$

we are trying to prove that $x, y \in X$ then $xy \notin E(G)$

So, let $xy \in E(G)$ then $P_1 \cup P_2$ together with edge xy will form a cycle of odd length. [odd length cycle]

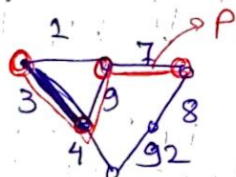
now we know every cycle is of length: even

so $xy \notin E$

The same holds if $[xy \in Y] \Rightarrow xy \notin E(G)$

The Shortest Path Problem

Edge weighted Graph: A graph G together with $\alpha: E(G) \rightarrow \mathbb{N}$
 $\{0, 1, 2, \dots\}$
 weight function



$\alpha(uv)$ called the length of uv

$$\alpha(P) = \sum_{e \in P} \alpha(e) = 1 + 3 + 9 + 8 + 9 + 2 = 32$$

Shortest Path Problem

Given a connected graph G with weight function

$\alpha: E(G) \rightarrow \mathbb{N}$, find for given $u, v \in V(G)$

$$d_G^\alpha(u, v) = \min \{ \alpha(P) \mid P: u \rightsquigarrow v \}$$

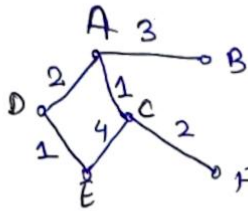
For connected edge weighted graph G and a source vertex $u \in V(G)$.
 find the minimum weight of a path from u to every other vertex in G .
 let $\alpha: E(G) \rightarrow \mathbb{N}$ be the weight function.

Dijkstra's Algorithm

- 1] Set $S = \emptyset$ which will store the vertices of G for which a shortest path has been found.
- 2] Set $t(u) = 0$ and $t(v) = \infty$ for all $v \in V(G)$ $v \neq u$ and add u to S .
- 3] Let w be the newest member of S .
 For each $v \notin S$ $v \in N_G(w)$ Set
 $t(v) = \min \{ t(v), t(w) + \alpha(vw) \}$
- 4] Pick any $w \notin S$ with minimum $t(w)$ and add w to S Repeat steps

Solution: for each $v \in S$, $t(v)$ is the minimum weight of a path in G with weight function α .

$\alpha: E(G) \rightarrow \mathbb{N}$



Source vertex: A

1.] $S = \emptyset$

A	B	C	D	E	F
0	∞	∞	∞	∞	∞
N	N	N	N	N	N

A →

A	B	C	D	E	F
0	3	1	2	∞	∞
N	A	A	A	N	N

C →

A	B	C	D	E	F
0	3	1	2	5	3
N	A	A	A	C	C

D →

A	B	C	D	E	F
0	3	1	2	3	3
N	A	A	A	D	C

B →

A	B	C	D	E	F
0	3	1	2	3	3
N	A	A	A	D	C

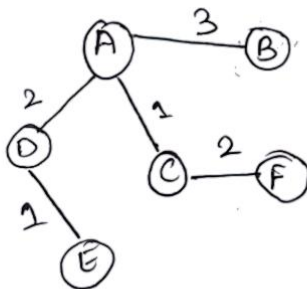
E →

A	B	C	D	E	F
0	3	1	2	3	3
N	A	A	A	D	C

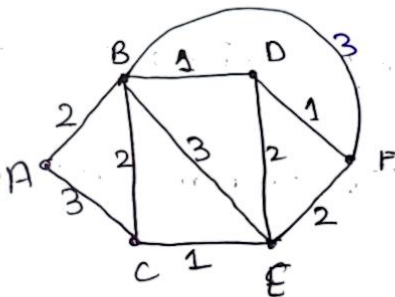
F →

A	B	C	D	E	F
0	3	1	2	3	3
N	A	A	A	D	C

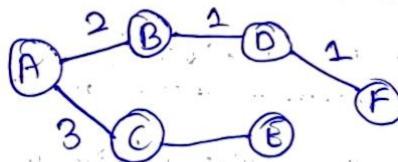
A C D B E F

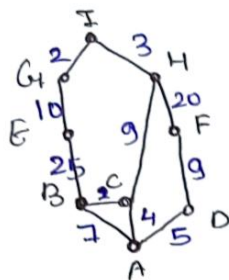


A	B	C	D	E	F
0	3	1	2	3	3
N	A	A	A	D	C



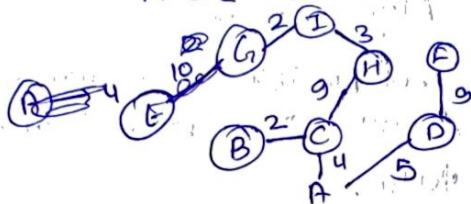
A	B	C	D	E	F
0	3	1	2	3	3
N	A	A	A	D	C





A	0							
B	∞	7A						
C	∞	25B	4A					
D	∞	∞	9C	5A				
E	∞	∞	∞	∞	2G			
F	∞	∞	∞	14D	14G	3H		
G	∞	∞	∞	∞	∞	14D	3H	
H	∞	∞	13C	13D	13G	16F	16I	
I	∞	∞	∞	∞	∞	∞	∞	2E

$A \rightarrow C \rightarrow D \rightarrow B \rightarrow H \rightarrow F \rightarrow G \rightarrow I \rightarrow E$



walk: path $u \xrightarrow{*} v$ is closed if $u = v$

trail: is a walk with no repeated edges

Euler trail: a trail that visits every edge once

Euler tour: an Euler trail that is closed. [if a graph has a

Euler tour it is called Eulerian]

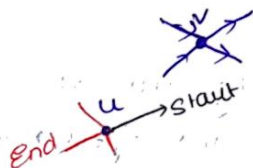
video-24

Theorem: A connected graph G is Eulerian $\iff$ Every vertex of G has even degree.

$\Rightarrow$ Suppose G is a connected Eulerian Graph
let $w: u \xrightarrow{*} u$ be a Euler tour
let $v \neq u$ of G that occurs k times in w [Euler tour]

Then $dc(w) = 2k$

Also $dc(w) = \text{even}$



$\Leftarrow$ let G be a non-trivial connected graph whose vertices all have Even degree

let w be a longest trail in the graph G

$w: v_0 e_1 v_1 e_2 v_2 \dots e_{l-1} v_{l-1} e_l v_l$

$w: v_0 \xrightarrow{l} v_l$ [longest trail from v_0 to v_l]

$\because$ all edges incident with v_l are used in w , and v_l has even degree, $v_l = v_0$
 $\hookrightarrow$ otherwise w could be extended to a longer trail.

if $v_e \neq v_o$ and v_e occurs k times in w then $d(v_e) = 2(k-1) + 1$

Thus w is a closed trail

Suppose w is not a Euler trail

$\therefore G$ is connected there is an edge

$$f = v_i u \in E(G) \setminus E(w)$$



then $f e_{i+1} \dots e_2 e_1 \dots e_i$
 $\hookrightarrow$

Alonger walk
 $\downarrow$
 not possible as w was longest walk
 $\therefore$ our assumption that w is not a Euler tour is false.

8) ** video 24

Theorem: A connected graph has an Euler trail $\iff$ it has at most 2 vertices of odd degree

* A graph has even number of odd degree vertices

Proof

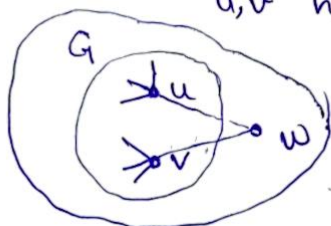
$\Rightarrow$ If G has an Euler trail $u \xrightarrow{*} v$ then as in the proof of the previous theorem, all $w \notin \{u, v\}$ has even degree $\Rightarrow$ either 0 or 2 vertices of odd degree.

$\Leftarrow$ Suppose at most 2 vertices with odd degree

$\circ$ If no vertices have odd degree then there is an Euler tour

$\circ$ If 2 vertices of odd degree

u, v have odd degree in G



Add new vertex w and new edge uw, vw

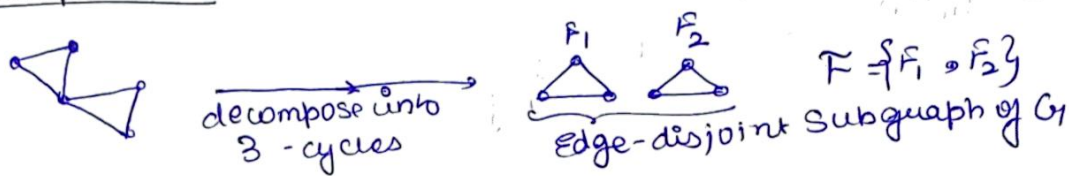
New graph H has all vertices of even degree

$\therefore H$ has an Euler tour

take $w \xrightarrow{*} w$

The portion of the Euler tour with w removed (uw, vw) is an Euler trail from $u \xrightarrow{*} v$

Decompositions

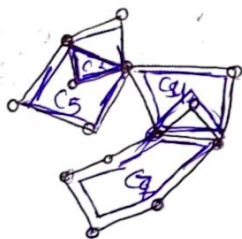


A decomposition of a graph G_1 is a family $\mathcal{F}$ of edge-disjoint subgraphs of G_1 such that

$$\bigcup_{F \in \mathcal{F}} E(F) = E(G_1)$$

If every graph of $\mathcal{F}$ is a cycle then the decomposition is called a cycle decomposition.
If every graph of $\mathcal{F}$ is a path then the decomposition is called a path decomposition.

NOTE: For every simple graph, there is a trivial path decomposition.



$\{C_3, C_4, C_5, C_7\} \equiv$ cycle decomposition

Even Graph: Graph whose all vertices have even degrees.

Video: 26

Theorem: A Graph G_1 has cycle decomposition $\longleftrightarrow G_1$ is even

$\Rightarrow$ Suppose there is a cycle decomposition of G_1

Take any $v \in V(G_1)$

either v is isolated, $d(v) = 0$

or v belongs to some # of cycles in the decomposition
let K .



$$deg(v) = 6$$

Here $K = 3$

$$deg(v) = 2K$$

Even no.

$\therefore$ For all $v \in V(G_1)$ $deg(v) = \text{even}$

$\Leftarrow$ Aim: G_1 is even $\Rightarrow G_1$ has a cycle decomposition
Induction on $|E(G_1)|$

Basis: $|E(G)| = 0$

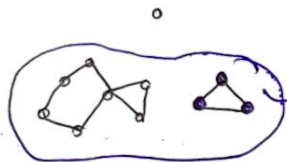
$\emptyset = \{\}$
Empty Set [Trivial Set]

Inductive Hypothesis

Suppose that for all even graphs on $< m$ edges there exists a cycle decomposition.

Take G to be even & $|E(G)| = m$ ~~even~~

let $X =$ the set of $V(G)$ with degree > 0



$$F = G[X]$$

an even graph

In F all vertices have degree ≥ 2

$\Rightarrow$ There is a cycle in F , call it C .

Theorem

G with degree $\geq 2 \rightarrow$ contains cycle

Take $G' = G \setminus E(C)$

G' is even graph with $< m$ edges

$\therefore$ by inductive hypothesis G' has a cycle decomposition call it C'

now $G = G' \cup C$ is a cycle decomposition of original graph G .

Conclusion

Eulerian $\iff$ Even graph $\iff$ Cycle decomposition

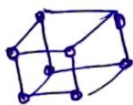
Euler tour can have repeated vertices but not edges

decompose $E(G)$ into $\{C_1, C_2, \dots, C_k\}$ of any lengths

Hamiltonian Graphs

Hamiltonian path: path that covers every vertex once
 Hamiltonian cycle: cycle that covers every vertex once

If $\exists$ Hamilton cycle the graph is Hamiltonian



Hamiltonian cycle can be converted into Hamiltonian path by removing one edge.

If G is Hamiltonian then any supergraph G' $G \subseteq G'$ where G' is obtained by adding new edges b/w non adjacent vertices of G then G' is also Hamiltonian

Clearly C_n is Hamiltonian (C_n)

$C_n = G_1$

$K_n = G_1$
 (Supergraph)

$\therefore K_n$ is Hamiltonian.



Lovasz Conjecture

Every finite connected Vertex transitive graph has a Hamilton path.

Automorphism: An automorphism of a graph G is an isomorphism from G to itself. $\alpha: V(G) \rightarrow V(G)$
 $u \rightarrow v$

Note Vertices $u, v \in V(G)$ are similar if there is an automorphism α such that $\alpha(u) = v$

Vertex Transitive Graph It is a graph in which all vertices are similar.



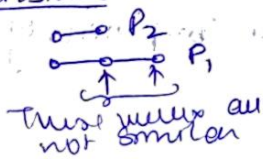
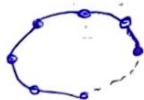
v_1 and v_2
 $\alpha: v_1 \rightarrow v_2$
 $v_2 \rightarrow v_1$
 $v_3 \rightarrow v_3$

$v_1, v_2 \in E(G) \implies \alpha(v_1), \alpha(v_2) \in E(G)$
 $v_2, v_1 \in E(G)$

$v_1, v_3 \in E(G)$

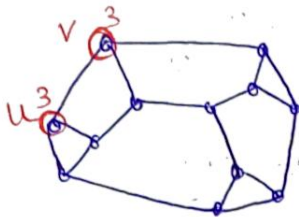
$\alpha(v_1), \alpha(v_3) = v_2, v_3 \in E(G)$

Vertex-transitive



These vertices are not similar

Vertex-transitive graphs are regular.



3-regular graph
not vertex-transitive
 u, v are not similar.

u lies on 3 cycles, 4 cycle
and several larger cycles

vertex v does not lie
on 3 cycle.
 $\therefore$ not similar

Lovasz conjecture: Every finite connected vertex transitive graph has a Hamilton path

To date: no known example of a finite connected vertex transitive graph with no Hamilton path.

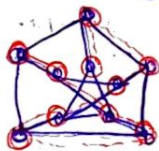
$\hookrightarrow$ In fact, these almost always have a Hamilton cycle.
 $\hookrightarrow$ there are 5 examples that don't have Hamilton cycle

2-path graph



Trivial case $\rightarrow$

Petersen graph



* Coxeter graph

* triangle-replaced Coxeter graph

* triangle-replaced Petersen's graph

video 31

10 Lemma: If G_1 is Hamiltonian, then for every non-empty subset $S \subseteq V(G_1)$, $c(G_1 - S) \leq |S|$

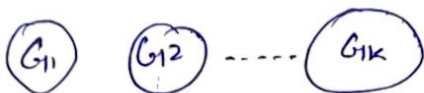
$\downarrow$
of connected components of $G_1 - S$

Proof: G_1 is Hamiltonian, let $\emptyset \neq S \subseteq V(G_1)$ and let $w \in S$ and $C: w \rightarrow w$ be a Hamilton cycle of G_1

Assume $c(G_1 - S) = k$ connected components and let $\{G_1, G_2, \dots, G_k\}$ connected components of $(G_1 - S)$.

if $k=1$ $|S| \geq 1$ $\checkmark$ trivial
 so $k > 1$

Give C an orientation



let u_i be the last vertex of C that belongs to G_i . let v_i be the next vertex of C after u_i

v_i is not going to be in G_i by the definition of u_i as u_i is the last vertex of G_i and if v_i is in G_i in that case u_i is not the last vertex of G_i , then it will be contradictory.

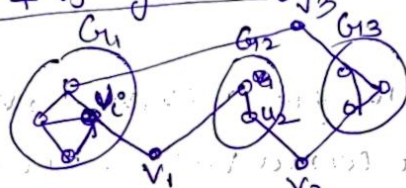
$$\therefore v_i \notin V(G_i)$$

$$v_i \notin V(G_j)$$

$\therefore$ the components are mutually disjoint

$$\therefore \boxed{V_i \in S} \rightarrow \text{removed set.}$$

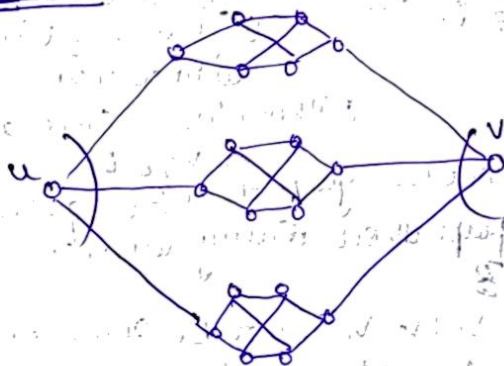
$$v_i \neq v_j \text{ for } i \neq j$$



for each $i=1, \dots, k$ there is a $v_i \in S$ $|S| \geq k$

$$\boxed{k = c(G-S) \leq |S|}$$

Example



$$S = \{u, v\} = 2$$

$$c(G-S) = 3$$

$$3 > |S|$$

lemma not satisfied

$\downarrow$
 no hamilton cycle.

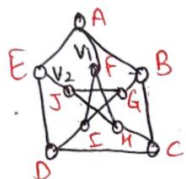
if the condition satisfies there is no guarantee that there exist a hamilton cycle.

Ex: Person graph

$\emptyset = S = V(G)$ satisfies
 $c(G-S) \leq |S|$ but do not have hamilton cycle.

Problems

1.] Find a Hamilton path in Peterson Graph



$F \rightarrow A \rightarrow B \rightarrow C \rightarrow D \rightarrow E \rightarrow J \rightarrow H \rightarrow I \rightarrow G \equiv \text{Path}$

2.] Prove that peterson graph has no Hamilton Cycle

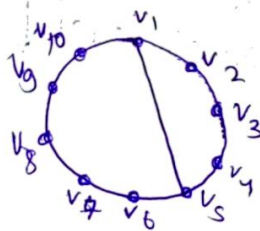
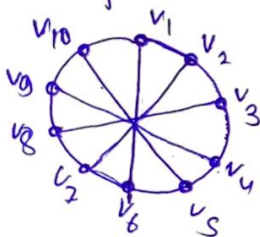
Observe: The peterson graph has no cycle on ≤ 4 vertices
minimum cycle length = 5

Proof: (By contradiction)

suppose C is a Hamilton cycle of the peterson graph.

10 cycle, 15 edges

The peterson graph consists of the 10-cycle $C = (v_1, v_2, \dots, v_{10})$ together with 5 "chord" edges that connect vertices not already adjacent in C .



$v_1 v_5 \in E$

There is a chord that joins vertices at distance ≤ 5 in C , These vertices must be at distance 4 in C .

Without loss of generality $v_1 v_5 \in E$

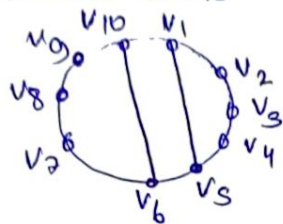
$\therefore$ peterson graph is a 3 regular graph then every vertex is of degree 3

$\therefore$ vertex v_6 must be connected to one other vertex

now v_6 cannot be connected to v_2, v_8, v_9 and v_4, v_3 as this will generate 3 cycle and 4 cycle.

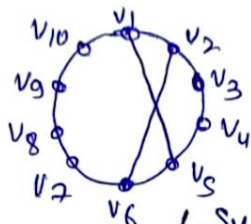
$\therefore$ v_6 options available are v_2 and v_{10} .

Case 1: $v_6 v_{10}$



$\{v_1, v_5, v_6, v_{10}\} = C_4$
 $\hookrightarrow$ cycle of length 4
 so discarded.

Case 2: $v_6 v_2$



$\{v_1, v_5, v_6, v_2\} = C_4$
 Cycle of length 4.

$\circ \circ v_6$ cannot be incident with any chord edge w/o making a cycle of length ≤ 4 . $\circ \circ$ There is contradiction as Peterson graph cannot have ~~any~~ cycle of length ≤ 4 .

Hence Our assumption was incorrect ~~that~~ there is no Hamiltonian cycle in Peterson graph.

hence. ~~fact~~

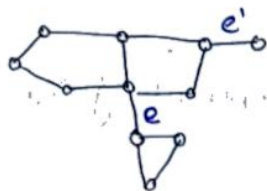
Hence, Peterson graph is not Hamiltonian.

3] Show that $K_{m,n}$ is Hamiltonian $\iff n=m \geq 2$

Proof

$\Rightarrow$ let $K_{m,n}$ have bipartition X, Y where $|X|=n$ $|Y|=m$
 let C be a Hamilton cycle of $K_{m,n}$
 Each cycle in $K_{m,n}$ has even length

Bridge Edges



bridge

$G \setminus e$ has 2 components

$G \setminus e'$ has 2 components

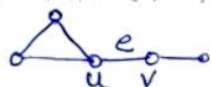


$$c(G) = 2$$

$$c(G \setminus e) = 3$$

An edge $e \in E(G)$ is a bridge of G , if $G \setminus e$ has more connected components than G .

In particular, if G is connected then e is a bridge if and only if $G \setminus e$ is disconnected.



$e = uv$ is a bridge $\Leftrightarrow u, v$ are in different connected components of $G \setminus e$

Theorem: An edge $e \in E(G)$ is a bridge iff e is not in any cycle of G .

$\Rightarrow$ Contrapositive: e is on a cycle then e is not a bridge

Say e is on a cycle C

$G \setminus e$ u and v are in the same connected component

$(e = uv) \therefore e$ is not a bridge.

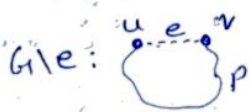
$\Leftarrow$ Contrapositive: e not a bridge $\Rightarrow e$ is on a cycle

$e = uv$ is not a bridge

then $G \setminus e$ there is a path from u to v

(u and v are in the same component)

$P: u \xrightarrow{*} v$



In G

$$eP: u \xrightarrow{*} v \xrightarrow{*} u$$

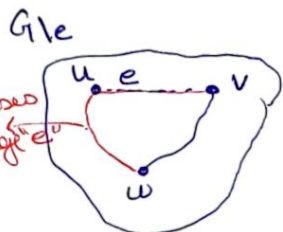
G there is a cycle that contains e

Lemma: let e be a bridge in a connected graph G_1 . Then $c(G_1 - e) = 2$

Proof : $e = uv$

e is a bridge $\Rightarrow u$ and v are in different components of $G - e$

$$\therefore C(G_1 e) \geq 2$$



let $w \in V(G_1)$

G_1 connected $\Rightarrow \exists P: u \xrightarrow{*} v$ in G_1

If P does not use edge e then P is the path from w to v in $G \setminus e$

If P uses edge e then $P: w \xrightarrow{*} u \xrightarrow{e} v$ so there is a path from w to u in $G \setminus e$.

$\therefore$ we can say every vertex present in graph G is connected either to u or v . $\therefore$ we can have at most 2 components.

$$\therefore C(U \setminus e) = 2$$

Trees

A graph with no cycle is acyclic

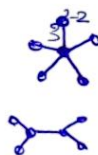
A tree is a connected ^{acyclic} graph

A forest is an acyclic graph.

Corollary: A connected graph is a tree $\iff$ all of its edges are bridges.

Cayley's Formula: There are n^{n-2} trees on a vertex set V of n elements

n	1	2	3	4	5	6	7	...	16
# non-morphine deals	1	1	1	2	3	6	11		19320



Let T be a tree. If $v \in V(T)$ and $d(v) = 1$ then v is a leaf

Fact: Every tree T with $|V(T)| \geq 2$ has at least 2 leaves.

↳ Take a maximal path T .

Each end-vertex of this path is a leaf.

Theorem: A graph G is a tree $\leftrightarrow G$ is acyclic and $|E(G)| = |V(G)| - 1$
 $m = n - 1$

$\Rightarrow$ G is a tree
 $\therefore$ acyclic [by definition]

Proof of theorem formula $m = n - 1$

Base case $n = 1$ $\bullet$ $m = 0$

Induction hypothesis: Suppose all trees of order n have $n - 1$ edges ($n \geq 1$)

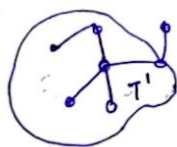
Let T be a tree of order $n + 1$

$v \in V(T)$ with degree 1 (leaf)

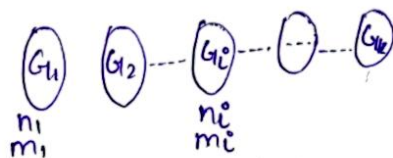
$$T' = T - v$$

$\therefore T'$ has $n - 1$ edges

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



$\Leftarrow$ Suppose G is acyclic and $m = n - 1$



k connected components

acyclic } G_i is a tree
 connected } $m_i = n_i - 1$

$$m = \sum_{i=1}^k m_i = \sum_{i=1}^k n_i - 1 = n - k$$

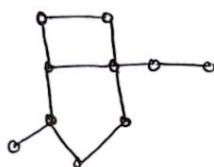
$$m = n - k$$

from our assumption $m = n - 1 = n - k$
 $\therefore k = 1$ $\therefore$ 1 connected component

Theorem: A graph G_1 is a tree $\iff G_1$ is acyclic and $|E(G_1)| = |V(G_1)| - 1$
 $m = n - 1$

Corollary: A forest F of order n with k connected components has $m = n - k$ edges.

Spanning tree of a graph G_1 : Spanning subgraph of G_1 that is a tree touches all of the vertices of G_1



Every connected graph G_1 has a spanning tree
 $\hookrightarrow$ if G_1 is a tree
 $\hookrightarrow$ if G_1 is not a tree

$$m - (m - (n - 1)) = n - 1$$

of edges that need removing

removing edges of cycles until only bridges remain

Corollary: Every connected graph has $m \geq n - 1$

Theorem: A graph G_1 is a tree $\iff G_1$ is connected and has $m = n - 1$

Proof: $\Rightarrow$ G_1 is a tree $\Rightarrow$ connected

$\hookrightarrow$ previous theorem $\Rightarrow m = n - 1$

$\Leftarrow$ G_1 is connected and has $m = n - 1$ edges

Suppose C is a cycle in G_1

$e \in C$

$G_1 \setminus e \leftarrow$ still connected

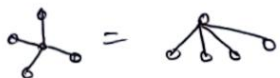
$G_1 \setminus e$ has $n - 2$ ($\Rightarrow \Leftarrow$)

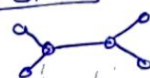
G_1 is acyclic $\therefore G_1$ is a tree.

<u>Trees</u>	connected	☒	☒	☒
	acyclic	☒	☒	☒
	$m \geq n - 1$	☒	☒	☒
		True	True	True

Types of trees:

Path:  P_5

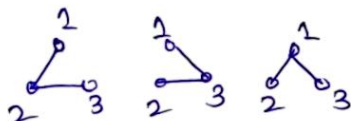
Star:  =  Bipartite graph $(K_{1,5})$

Double star: 

Caterpillar: if we remove all the leaves we will get a path. also path is a spine.

Cayley's Formula

There are n^{n-2} labelled trees of order n



Proof Idea: labelled tree T $\leftrightarrow$ prufer sequence S on n elements of length $n-2$
Correspondence

How many such sequence?
 $n \times n \times n \times \dots \times n = n^{n-2}$

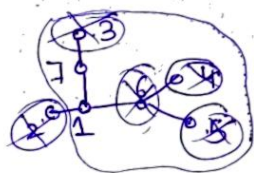
1, n, 4, 4, ...
 $n-2$

Labelled

$T \rightarrow$ Prufer sequence S

- Find a leaf of T with smallest label
- add the neighbour to S
- delete this leaf

Repeat until our tree is K_2 



$S = ()$

$S = (1)$

$S = (1, 7)$

$S = (1, 7, 6)$

$S = (1, 7, 6, 6)$

$S = (1, 7, 6, 6, 1)$

Prufer sequence.

NOTE: why $|S| = n-2$

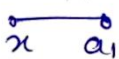
- no leaf get appended to S

- Every vertex v is added to S a total of $\deg(v)-1$ times

n vertices, edges $\Rightarrow m = n-1$

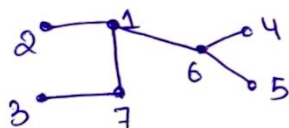
$$\# \text{ of terms in } S = \sum_{v \in V(T)} (\deg(v) - 1) = \sum_{v \in V(T)} (\deg(v)) - \sum_{v \in V(T)} 1 = \frac{\sum \deg v}{2} - n = \frac{2m}{2} - n = m - n = (n-1) - n = -1$$

Prüfer sequence S of length $(n-2)$ using symbols $1, \dots, n \rightarrow$ labelled tree T ,

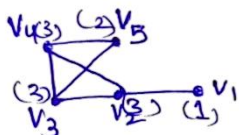
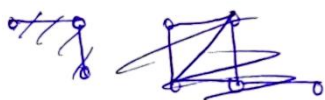
- $S = (a_1, a_2, \dots, a_{n-2})$ $a_i \in \{1, \dots, n\}$
- Find the smallest element $x \in \{1, \dots, n\}$ $x \notin S$
 $\hookrightarrow$ join x to 1st element of S 
- $\rightarrow$ delete a_1 from S , delete x from $\{1, \dots, n\}$
- Find smallest $y \in L \setminus \{x\}$ not in S .
 $\hookrightarrow$ join y to the 1st element of $S \setminus \{a_1\}$
 $\rightarrow$ delete a_2 from S
 $\rightarrow$ delete y from L
- Continue until 2 items remain in L
 $\hookrightarrow$ join these 2 via an edge

$$S = (x, x, x, x) \quad |S| = 5 \quad \therefore n = 7$$

$$L = \{1, x, x, x, x, 7\}$$



Degree Sequences of Graphs



degree sequence: $1, 3, 3, 3, 2$
 or $2, 3, 3, 3, 1$
 or $1, 2, 3, 3, 3$
 etc.

A sequence $d_1, d_2, \dots, d_n$ of non-negative integers is called a degree sequence of a graph G if the vertices of G can be labelled $v_1, v_2, \dots, v_n$ such that $\deg(v_i) = d_i$ for all $i = 1, \dots, n$.

Given a graph G , it is easy to find a degree sequence.

But given a sequence of non-negative integers, when is it a degree sequence of some graph?

If a sequence $s: d_1, \dots, d_n$ is a degree sequence of some graph then the sequence s is called a graphical sequence.

Obvious conditions:

$\sum_{i=1}^n d_i$ must be even

$d_i \leq n-1$ for all $i=1, \dots, n$

Although these conditions are necessary they are not sufficient

Ex: 3, 3, 3, 1

$n=4$

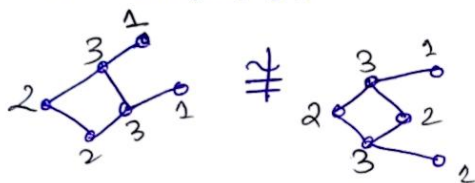
$\sum d_i = 10$ ✓ even

$d_i \leq 3$ ✓

But these sequence is not graphical

Observation: A graphical sequence may be a degree sequence for more than one graph

Ex: 3, 3, 2, 2, 1, 1



Necessary and sufficient condition for a sequence to be graphical:

Theorem (Havel and Hakimi):

A sequence $s: d_1, d_2, \dots, d_n$ of non-negative integers with $d_1 \geq d_2 \geq \dots \geq d_n$ and $n \geq 2, d_1 \geq 1$ is graphical iff

the sequence $s_1 = \underbrace{d_2-1, d_3-1, \dots, d_{d_1+1}-1}_{d_1 \text{ terms}}, d_{d_1+2}, \dots, d_n$ is graphical

Ex:

Ex: 3, 3, 3, 1 is graphical $\iff 2, 2, 0$ is graphical

$\iff 1, -1$ is not graphical false

Ex: 3, 3, 3, 2, 2, 1 is graphical $\iff 2, 2, 1, 2, 1$ is graphical

Put in order

2, 2, 2, 1, 1

$\iff 1, 1, 1, 1$ is graphical

$\iff 0, 1, 1$ is graphical

$1, 1, 0$ is graphical

$0, 0$ is graphical ✓

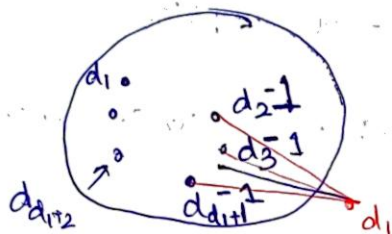
Proof of Havel-Hakimi Theorem

⇒ Suppose S is graphical.

Then by definition $\exists$ a graph G_1 of order $n-1$ with degree sequence S .

Hence we can label $V(G_1)$ with $v_2, v_3, \dots, v_n$ such that

$$\deg(v_i) = \begin{cases} d_i - 1 & \text{for } i=2, 3, \dots, d_1+1 \\ d_i & \text{for } i=d_1+2, \dots, n \end{cases}$$



we can construct a new graph G as follows:

- Start with graph G_1
- add a new vertex v_1
- add d_1 new edges $v_1 v_i$ for $i=2, 3, \dots, d_1+1$

Thus in the graph G we have

$$\deg(v_i) = d_i \text{ for } i=1, 2, \dots, n$$

- The sequence $S: d_1, d_2, \dots, d_n$ is graphical

⇒ Suppose S is a graphical sequence

so there exist graph of order n with degree sequence " S ".

Among all such graphs let G be one such that

$V(G) = \{v_1, \dots, v_n\}$ with $\deg(v_i) = d_i$ for $i=1, \dots, n$ and the sum of degrees of the vertices adjacent to v_1 is maximum.

Claim: v_1 is adjacent to vertices having degrees $d_2, d_3, \dots, d_{d_1+1}$.

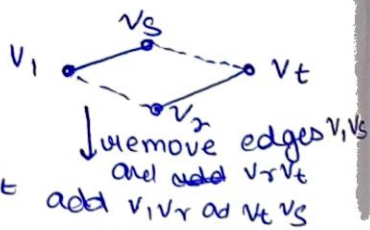
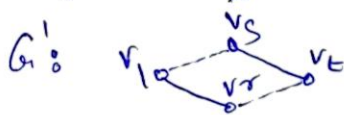
↳ Proof

suppose the claim is not true.

Then $\exists$ vertices v_r and v_s with $d_r > d_s$ such that v_1 is adjacent to v_s & v_1 is not adjacent to v_r .

- $d_r > d_s \exists$ a vertex v_t such that

v_t is adjacent to v_r
& v_t is not adjacent to v_s



degree sum

The edge switching results in a graph G_1 having the same degree sequence as G but the sum of the degrees of the vertices adjacent to v_1 in G_1 is larger than that in G .

This contradicts the choice of G_1 ($\Rightarrow \text{contradiction}$)

So v_1 is adjacent to vertices having degrees $d_2, d_3, \dots, d_{d_1+1}$

Thus the graph $G - v_1$ has degree sequence

$$S_1: d_2-1, d_3-1, \dots, d_{d_1}-1, d_{d_1+1}, \dots, d_n$$

$\therefore S_1$ is graphical

Degree sequence of a Tree

connected
acyclic

Theorem: A sequence $d_1, d_2, \dots, d_n$ of n +ve integers $n \geq 2$ is a degree sequence of a tree of order n if and only if

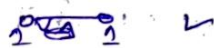
$$\sum_{i=1}^n d_i = 2n-2$$

Proof: $\Rightarrow$ obvious since $\sum_{i=1}^n d_i = 2|E| = 2(n-1) = 2n-2$

$\Leftarrow$ Suppose $d_1, d_2, \dots, d_n$ satisfies $\sum_{i=1}^n d_i = 2n-2$

We claim that $\exists$ a tree with this degree sequence.
We prove this by induction.

Base: $n=2$ so, $d_1+d_2=2$
 $d_1=d_2=1$



Inductive Hypothesis: Suppose the result is true for $n=k$ for some $k \geq 2$

Goal: show the result is true for $n=k+1$

Let $S: d_1, d_2, \dots, d_{k+1}$ be a sequence of $k+1$ +ve integers such that

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

WLOG suppose we have ordered the sequence so that

$$d_1 \geq d_2 \geq \dots \geq d_{k+1}$$

We have $k+1$ +ve integers which sum to $2k$. so it follows that
 not all of these integers are equal to 1 i.e. $d_1 \geq 2$

• At least 2 of the integers ≥ 1 i.e. $d_k = d_{k+1} = 1$

Now consider the sequence

$$s': d_1 - 1, d_2, d_3, \dots, d_k$$

This is a sequence of k +ve integers which sum to $(2k-2)$

Inductive hypothesis $\Rightarrow$ $\exists$ a tree T' of order k with $V(T') = \{v_1, \dots, v_k\}$
 such that $\deg(v_1) = d_1 - 1$ $\deg(v_i) = d_i$ for $i = 2, \dots, k$

Let T be the tree obtained from T' by adding a new vertex v_{k+1}
 and the edge $v_1 v_{k+1}$

$$\begin{aligned} \text{So } \deg(v_1) &= d_1 & \deg(v_i) &= d_i \text{ for } i = 2, \dots, k \\ \deg(v_{k+1}) &= 1 = d_{k+1} \end{aligned}$$

• $d_1, d_2, \dots, d_{k+1}$ is a degree sequence of the tree T .

Specific degrees in Trees

Theorem: let T be a non-trivial tree with $\Delta(T) = k$ and let $n_i =$

$n_i = \#$ vertices of degree i in T for $i = 1, 2, 3, \dots, k$

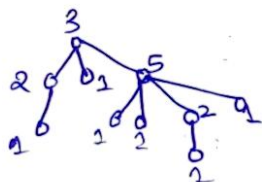
$$\text{Then } n_1 = n_3 + 2n_4 + 3n_5 + \dots + (k-2)n_k + 2$$

Recall every non-trivial tree has at least 2 leaves

$$\begin{aligned} n_1 &= 6 & n_2 &= 2 \\ n_3 &= 1 & n_4 &= 0 & n_5 &= 1 \end{aligned}$$

$$\begin{aligned} RHS &= n_3 + 2n_4 + 3n_5 + 2 \\ &= 1 + 0 + 3 + 2 = 6 \in n_1 \end{aligned}$$

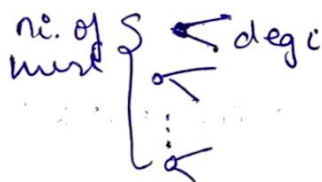
Ex:



Proof: Suppose T is a non-trivial tree and let $|V(T)| = n$ and

$|E(T)| = m$ then $m = n - 1$

$$\text{Also, } \sum_{i=1}^k i \cdot n_i = \sum_{v \in V(T)} \deg(v) = 2m = 2(n-1) = 2n-2$$



$$= 2\left(\sum_{i=1}^k n_i\right) - 2$$

$$\therefore \sum_{i=1}^k i \cdot n_i = 2 \sum_{i=1}^k n_i + 2 \geq 0$$

$$\sum_{i=1}^k (i-2) n_i + 2 \geq 0$$

$$-n_1 + 0n_2 + n_3 + 2n_4 + 3n_5 + \dots + (k-2)n_k + 2 \geq 0$$

$$n_1 = (n_3 + 2n_4 + 3n_5 + \dots + (k-2)n_k) + 2$$

All trees are subgraphs

Theorem: Let T be a tree of order k and G a graph with $\delta(G) \geq k-1$. Then G contains a subgraph isomorphic to T .

Ex:



T has order 4

Any graph with $\min \deg \geq 3$ has subgraph isomorphic to T

Ex: T has order 5



Any graph with $\min \deg \geq 4$ has a subgraph isomorphic to T

Proof Induction on k

Base: $k=1$

The only tree of order 1 is K_1 and this is a subgraph of any graph.

$k=2$

The only tree of order 2 is K_2 and this is a subgraph of any non-empty graph i.e. with $\min \deg \geq 1$

Inductive Hypothesis: Assume for each tree T' of order $k-1$ ($k \geq 3$) and every graph H with $\delta(H) \geq k-2$ that H contains subgraph

isomorphic to T' .

Let T be a tree of order k and G a graph with $\delta(G) \geq k-1$

Let v be a leaf of T

$$\text{So } \deg_T(v) = 1$$

Let $u \in V(T)$ such that $uv \in E(T)$

Then $T-v$ is a tree of order $k-1$ and G has $\delta(G) \geq k-1 > k-2$

So Inductive Hypothesis $\Rightarrow G$ contains a subgraph F isomorphic to $T-v$.

Let $u' \in V(F)$ that corresponds to u in $T-v$

$\therefore T-v$ has order $k-1$ so does F .

Thus there are only $k-2$ vertices in F other than $u' \Rightarrow \deg_F(u') \leq k-2$

$\therefore \exists$ a vertex $w \in V(G) \setminus V(F)$ such that $u'w \in E(G)$

$$\therefore (\delta(G) \geq k-1)$$

$$\therefore F \cup \{w\} \cong T$$

Subgraphs of regular graph

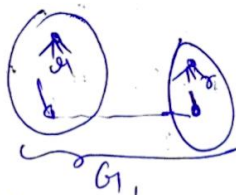
Theorem (König) : Every graph G with $\Delta(G) = r$ is an induced subgraph of an r -regular graph

Proof: If G is r -regular, then we're done.

Suppose G is not r -regular, i.e. $\delta(G) < r$

Let G' be another copy of G and join corresponding vertices of G and G' with an edge if they had degree $< r$

Call the resulting graph G_1 .

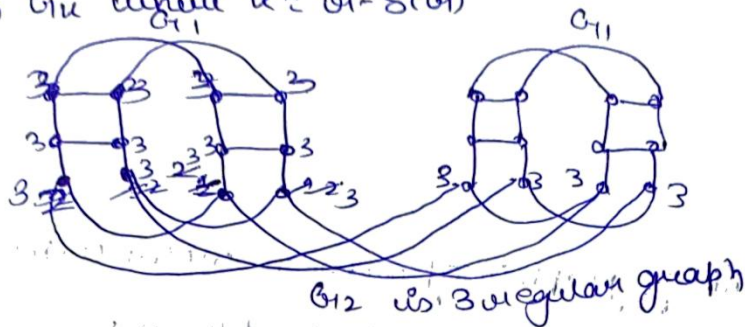


If G_1 is r -regular then we're done

$\therefore G$ is an induced subgraph of G_1

If not, continue the procedure until arriving at an α -regular graph G_{i+1} where $k = \alpha - \delta(G_i)$

Ans:



Q Is every graph G with $\Delta(G) = \alpha$ a spanning subgraph of an α -regular graph.

Ans: NO



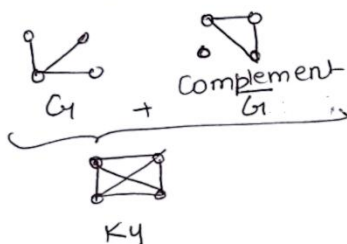
$$\Delta(G) = 4$$

$$|V(G)| = 5$$

Can there exist a 3-regular graph on 5 vertices?
No, $\because$ every graph has an even # of odd degree vertices

Complement of a graph

Ex:



Let $G = (V, E)$ be a graph of order n . The complement graph $\bar{G}$ is the graph with

$$V(\bar{G}) = V(G)$$

$$E(\bar{G}) = E(K_n) \setminus E(G)$$

$$|E(\bar{G})| = \binom{n}{2} - m$$

where $m = |E(G)|$

$$\binom{4}{2} = 2$$

A graph G is called self-complementary if $G \cong \bar{G}$

G_1 : $\bar{G}_1$: $4C_2 \equiv 6$ and $G_1, \bar{G}_1$ each have 3 edges

G_2 : $\bar{G}_2$: $5C_2 \equiv 10$ and $G_2, \bar{G}_2$ each have 5 edges
 $G \cong \bar{G} \cong G_5$

Q Is there is a self complementary graph of order 6?
 $\hookrightarrow$ no $\because 6C_2 = \frac{6 \times 5}{2} = 15 \rightarrow$ odd and this cannot be equally divided b/w G and $\bar{G}$.

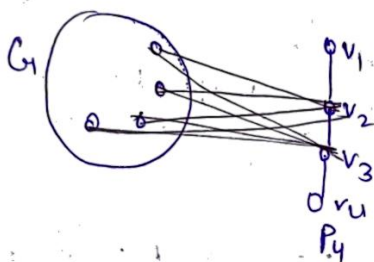
Fact: Self complementary graph of order n must have $\frac{1}{2} n(n-1)$ edges

$\Rightarrow$ If G of order n is self complementary then $n(n-1)$ must be divisible by 4 so, $n \equiv 0$ or $1 \pmod{4}$.

4	✓
5	✓
6	x
7	x
8	✓
9	✓

$$\begin{array}{r} 2 \\ 8 \times 7 \\ \hline 4 \end{array}$$

Given a self complementary graph G_1 of order n we can construct a self complementary graph G of order $n+4$ as follows:



Add $P_4 = [v_1, v_2, v_3, v_4]$ and join v_1 to v_2 and v_3 to all the vertices of G_1 .

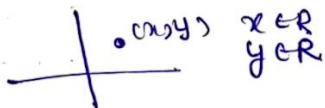
Cartesian Product of Graphs

Recall: The Cartesian product of 2 sets A and B is
 $A \times B = \{(a, b) \mid a \in A \text{ and } b \in B\}$

Ex: $A = \{x, y\}$, $B = \{1, 2, 3\}$

Then $A \times B = \{(x, 1), (x, 2), (x, 3), (y, 1), (y, 2), (y, 3)\}$

Ex: Euclidean plane: $\mathbb{R} \times \mathbb{R}$



$\mathbb{R}$ is a set of real numbers.

Suppose G_1 and H are graphs with $V(G_1) = \{u_1, u_2, \dots, u_m\}$
 and $V(H) = \{v_1, v_2, \dots, v_n\}$ ↙ can have different set of vertices

Then $G_1 \times H$ is the graph with vertex set

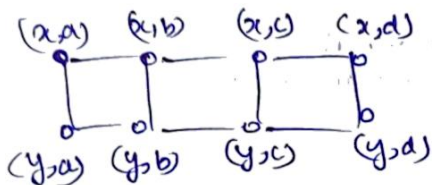
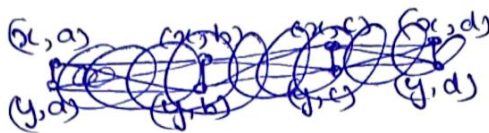
$$V(G_1 \times H) = V(G_1) \times V(H) \\ = \{(u_i, v_j) \mid u_i \in V(G_1), v_j \in V(H)\}$$

and e is an edge of $G_1 \times H$ iff $e = (u_i, v_j)(u_k, v_\ell)$

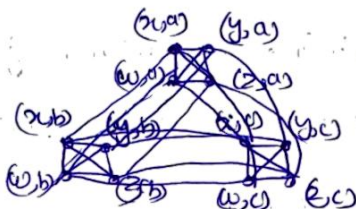
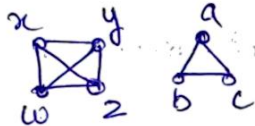
where either:

- 1.) $i = k$ and $v_j v_\ell \in E(H)$
- or 2.) $j = \ell$ and $u_i u_k \in E(G_1)$

Ex 1 $K_2 \times P_4$



Ex 2 $K_4 \times K_3$

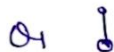


Hypercube (or n-cube) : Q_n is the graph

$$Q_n = \underbrace{K_2 \times K_2 \times \dots \times K_2}_{n \text{ times}}$$

e.g. $Q_1 = K_2$

$$Q_n = Q_{n-1} \times K_2$$

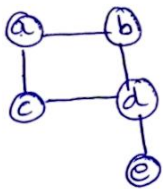


Hamilton path in an appropriately labelled hypercube is equivalent to a Gray code [used in computer science for encryption and error correction]

Terminology : maximum w.r.t some quantity vs maximal w.r.t. inclusion

An independent set in a graph G is a set of vertices $S \subseteq V(G)$ such that no 2 vertices of S are adjacent in G .

Ex:



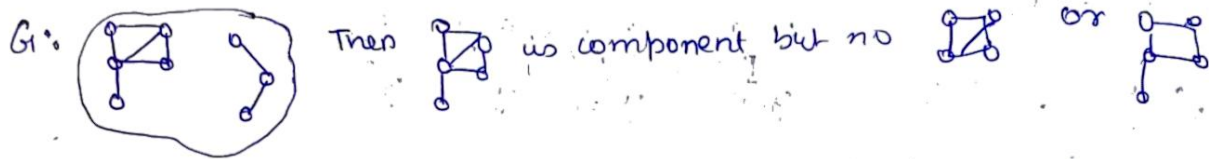
$S_1 = \{a, d\}$ is an independent set and is maximal
 $\because$ no other vertex can be added to S_1 while maintaining the property of being independent.

$S_2 = \{b, c, e\}$ is a maximal independent set for the same reason but it is also maximum

because there is no larger independent set in this graph.

Note: $S_3 = \{b, c\}$ is independent but maximal but not maximum

A component of a graph is maximal connected subgraph



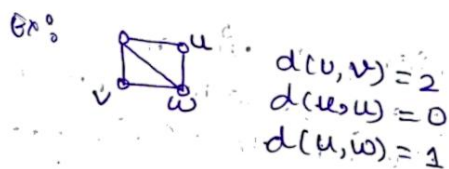
Note:  it is a component bcoz it is maximal (not maximum)

Eccentricity, Radius and Diameter

Let u, v be vertices in a graph G . The distance from u to v is the length of a shortest path from u to v in G and is denoted $d(u, v)$

Sometimes denoted $d_G(u, v)$ for clarity

If G is disconnected and u, v are different components we say $d(u, v) = \infty$



The distance is a metric space, $d: V \times V \rightarrow \mathbb{Z}^+ \cup \{\infty\}$

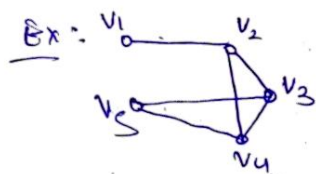
1) $d(u, v) \geq 0$ and $d(u, v) = 0$ iff $u = v$

2) $d(u, v) = d(v, u)$

3) $d(u, v) + d(v, w) \geq d(u, w)$

Eccentricity: Eccentricity of a vertex $v \in V(G)$ is

$$e(v) = \max \{d(u, v) \mid u \in V(G)\}$$



$$\begin{aligned} e(v_1) &= 3 & e(v_4) &= 2 \\ e(v_2) &= 2 & e(v_5) &= 3 \\ e(v_3) &= 2 \end{aligned}$$



$$\begin{aligned} e(a) &= 1 \\ e(b) &= e(c) = e(d) = 2 \end{aligned}$$

Note: $e(v) = 1 \iff v$ is adjacent to all other vertices

$$\text{diam}(G) = \max \{e(v) \mid v \in V(G)\}$$

$$\text{rad}(G) = \min \{e(v) \mid v \in V(G)\}$$

If $e(v) = \text{diam}(G)$ then v is a peripheral vertex. The set of all such vertices make the periphery of G .

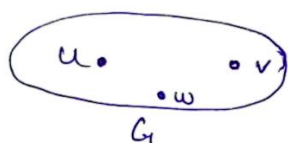
If $e(v) = \text{rad}(G)$ then v is a central vertex

The set of all such vertices make the centre of G

Fact: $\text{rad}(G) \leq \text{diam}(G) \leq 2 \text{rad}(G)$
 By definition.

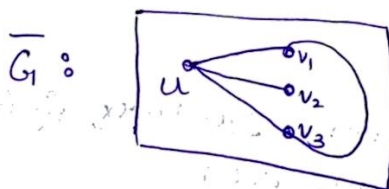
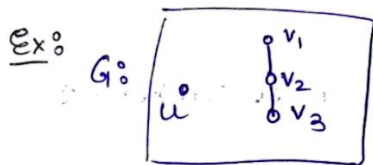
$$\text{diam}(G) \leq 2 \text{rad}(G)$$

Proof: Let $u, v \in V(G)$ such that $d(u, v) = \text{diam}(G)$



Let w be a central vertex of G
 Then $d(u, v) \leq d(u, w) + d(w, v)$
 $\leq 2 \text{ec}(w)$
 $\leq 2 \text{rad}(G)$

If G is a disconnected graph then $\bar{G}$ is connected and $\text{diam}(\bar{G}) \leq 2$

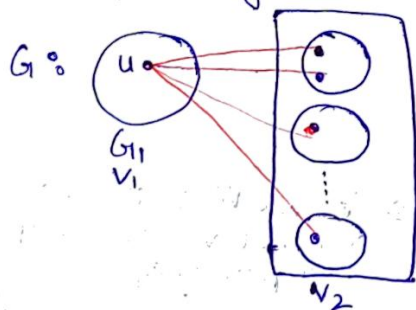


$$\text{ec}(u) = 1$$

$$\text{ec}(v_1) = \text{ec}(v_2) = \text{ec}(v_3) = 2$$

Proof:

Let G be a disconnected graph and let G_1 be one of the connected components of G .



$$\text{Let } v_1 \in V(G_1)$$

$$V_2 = V(G_2) \setminus V(G_1)$$

$$\text{Let } u \in V_1$$

Then for every $v \in V_2$

$$uv \notin E(G)$$

$$\therefore uv \in E(\bar{G}) \quad \text{i.e. } d_{\bar{G}}(u, v) = 1$$

And for every $v_1, v_2 \in V_2$

$$d_{\bar{G}}(v_1, v_2) \leq 2$$

$$\text{And for every } u, v_2 \in V_1 \quad d_{\bar{G}}(u, v_2) \leq 2$$

$$\therefore \max \{ e_{\bar{G}}(v) \mid v \in V(\bar{G}) \} \leq 2$$

$$\text{i.e. } \text{diam}(\bar{G}) = 2$$

Every Graph G_1 is centre of some connected graph.

Let G_1 be any graph

We construct a connected graph H as follows:

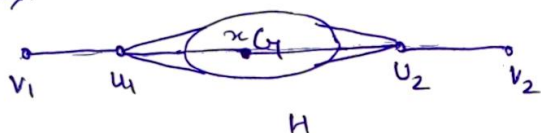
Add 2 new vertices u_1 and u_2 to G_1 Call these u_1 and u_2 and add all edges

$$u_1x \quad \forall x \in V(G_1)$$

$$\text{and all edges } u_2x \quad \forall x \in V(G_1)$$

Next add 2 new vertices v_1 and v_2 and edges v_1u_1 and v_2u_2

In



In the new graph H we have

$$e_H(v_1) = 1 \quad \text{and} \quad e_H(v_2) = 1$$

$$e_H(u_1) = 3 \quad \text{and} \quad e_H(u_2) = 3$$

$$\text{and } e_H(x) = 2 \quad \forall x \in V(G_1)$$

$$\therefore \text{rad}(H) = 2 \quad \text{and} \quad \text{cen}(H) = \{v \in V(H) \mid e_H(v) = \text{rad}(H)\}$$

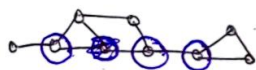
$$= V(G_1)$$

Cut Vertices

A vertex $v \in V(G_1)$ is called a cut-vertex if $G_1 - v$ has more components than G_1 i.e. $c(G_1 - v) > c(G_1)$

Notation: $c(G_1) = \#$ of connected components of G_1 (Sometimes $k(G_1)$ is used)

Find the cutvertices



A characterisation of cut vertices:

Theorem: A vertex $v \in V(G_1)$ is a cut-vertex of $G_1 \iff \exists u, w \in V(G_1) \quad u, w \neq v$
such that v is on every u - w path of G_1

Proof

Assume G is connected

$\Rightarrow$ Let $v \in V(G)$ be a cut-vertex then $G-v$ is disconnected.

Let u, w be vertices in different components of $G-v$

So there cannot exist u to w path in $G-v$

But G is connected so $\exists$ u to w paths in G

$\therefore$ all such path, must through vertex v

$\Leftarrow$ Suppose $\exists u, w \in V(G)$ $u, w \neq v$ such that v lies on every u to w path.

Then removing v means there cannot exist any u to w path in $G-v$. $\therefore G-v$ is disconnected.

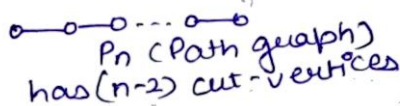
$\therefore v$ is a cut-vertex



How many cut-vertices can a connected graph have?



no cut-vertices in complete graph



Theorem: Every non-trivial connected graph has at least 2 vertices that are not cut-vertices

Proof: (By contradiction)

Suppose the theorem is ^{NOT} true

Then $\exists$ a non-trivial connected graph with at most 1 non cut-vertex
(i.e. every vertex except possibly 1 is a cut-vertex)

Let $u, v \in V(G)$ with $d(u, v) = \text{diam}(G)$

At least one of u, v is a ^{cut}-vertex $\Rightarrow$ say v

So $G-v$ is disconnected.



Let w be a vertex in a different component of $G-v$ than u

Then every u to w path contains v

$\therefore d(u, w) > d(u, v) = \text{diam}(G)$

$\hookrightarrow$ contradiction

$\therefore$ theorem is true

NOTE:

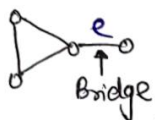
No peripheral vertex is a cut-vertex

Bridge

A Bridge in a graph G_1 is an edge $e \in E(G_1)$ such that

$$C(G_1 - e) > C(G_1)$$

↑
no. of connected components



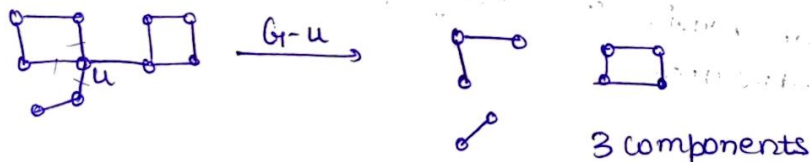
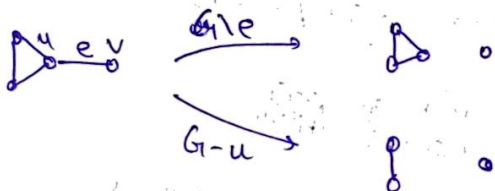
In fact if e is a bridge then

$$C(G_1 - e) = C(G_1) + 1$$

Theorem: $e \in E(G_1)$ is a bridge $\iff e$ is not on any cycle of G_1 .

Theorem: $e \in E(G_1)$ is a bridge $\iff \exists u, w \in V(G_1)$ such that e is on every u - w path in G_1 .

whenever $e = uv$ is a bridge and $\deg(u) \geq 2$ then $u = \text{cut vertex}$.



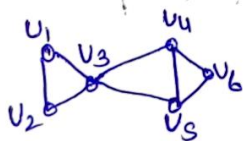
The only connected graph that has a bridge but has no cut vertices is K_2 or P_2



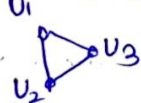
2-connected Graph

A non-trivial connected graph is called non-separable or 2-connected if it has no cut-vertices

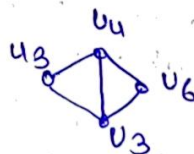
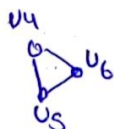
Block: maximal non-separable sub-graph



Blocks:



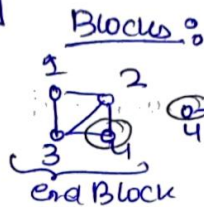
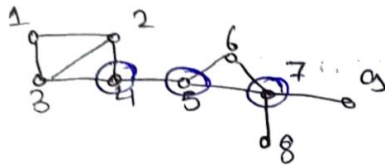
Notice



is non separable but not maximal non-separable

Theorem: If G is a graph with at least 1 cut-vertex, then at least 2 of the block of G contain exactly 1 cut vertex.

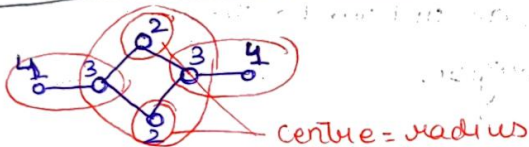
These are called End-Blocks



• Centre of a graph G is

$$\text{cen}(G) = \{v \in V(G) \mid e(v) = \text{rad}(G)\}$$

Theorem: The centre of every connected graph is in a single block.



Proof [By contradiction]

Suppose the theorem is false then $\exists$ a connected graph G where $\text{cen}(G)$ does not lie in a single block of G .

Thus G has a cut-vertex $v \in V(G)$ such that $G-v$ contains components G_1 and G_2 which each contain vertices of $\text{cen}(G)$.

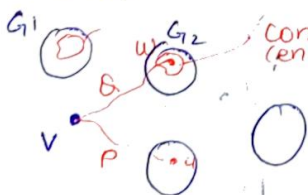
Let $u \in V(G)$ such that $d_G(u, v) = e(v)$ and let P be a shortest $u-v$ path.

At least one of G_1 & G_2 has no vertices of P (say G_2)

now let $w \in V(G_2) \cap \text{cen}(G)$ and let Q be a shortest $v-w$ path.

Now $P \cup Q$ forms a $u-w$ path which necessarily has length $d(u, w)$. So $e(w) > e(v)$ but $w \in \text{cen}(G)$ ($\Rightarrow \Leftarrow$)

But acc. to $e(w)$ has to be min. but $e(w) > e(v)$ contradiction

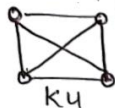


Planar Graph

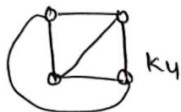
A planar graph is a graph that can be drawn in the plane w/o any edge crossing

Plane Graph

A planar graph that has been drawn in the plane w/o edge crossing.

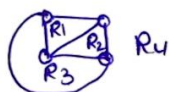


[planar Graph]

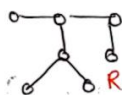


[Plane Graph]

A plane graph can divide the plane into regions

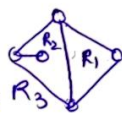
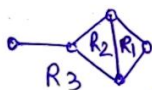


Every plane graph has an unbounded region called the exterior region.



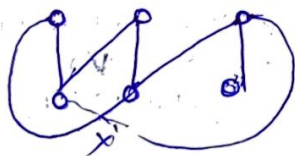
$R_1 \Rightarrow$ unbound region

The boundary of a region in a plane graph is the set of vertices & edges that "outline" it



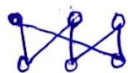
different
(as plane graphs)
[Boundaries are different]

Is $K_{3,3}$ planar?

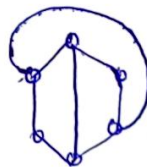


Informal proof:

Notice that $K_{3,3}$ has a Hamilton Cycle



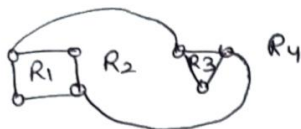
draw
w/o edge
crossing



stuck.

2 remaining edges need to connect vertices on opposite sides of the cycle.

By Region hole Principle we have 2 edges both going across the middle of the Hamilton cycle but both going across outside of the Hamilton cycle. $\therefore$ There will be an edge crossing.



$$n=7, m=9, r=4$$

$$n-m+r=7-9+4=2$$

$$n-m+r=2$$



$$n=7, m=8, r=3$$

$$n-m+r=7-8+3=2$$

$$n-m+r=2$$

Euler formula for connected graph.

Theorem: If G is a connected plane graph with n vertices, m edges and r regions, then $n-m+r=2$ Euler Formula

Proof: induction on m

Base: $m=0$

Then $G=K_1$, so $\left. \begin{matrix} n=1 \\ m=0 \\ r=1 \end{matrix} \right\} n-m+r=1-0+1=2$ It is true for $m=0$

Inductive Hypothesis:

Suppose the theorem is true for all connected plane graphs with $< m$ edges (where $m \geq 1$)

Now consider a connected plane graph G on m edges, n vertices, r regions

• If G is a tree then $m=n-1$ and $r=1$, so

$$n-m+r=n-(n-1)+1$$

$$=n-n+1+1$$

$$=2 \text{ satisfies}$$

• If G is not a tree then G has a cycle c

let e be an edge of cycle c : e is not a bridge

$\therefore G_1$ is connected graph and planar and has n vertices $m-1$ edges

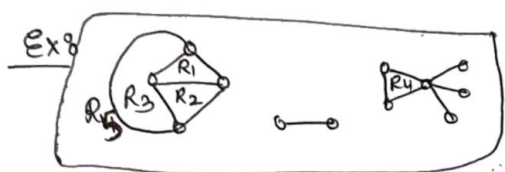
By inductive hypothesis the theorem holds for G_1 as it is planar, connected with $m-1 < m$ edges. $r-1$ regions

$\therefore$ the formula $n-(m-1)+(r-1)=2$

$$n-m+r=2$$

Corollary: If G is a plane graph with $c(G)$ connected components then

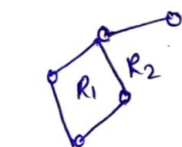
$$n - m + r = 1 + c(G)$$



$$\begin{aligned} n &= 12 \\ m &= 13 \\ r &= 5 \\ c(G) &= 3 \end{aligned}$$

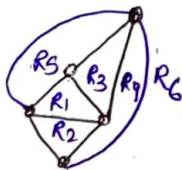
$$\begin{aligned} n - m + r &= 12 - 13 + 5 \\ &= 5 - 1 = 4 = 1 + c(G) \\ &= 1 + 3 \\ &= 4 \end{aligned}$$

True

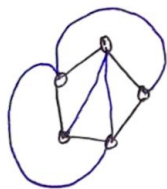


This is a plane graph

keep adding edges
But keep it
Planar



maximal plane graph
(also maximal planar graph)
aka triangulation



This is triangulation

Theorem: If G is a maximal planar graph with n vertices ($n \geq 3$) and m edges then $m = 3n - 6$

Proof: Take a plane drawing of G with r regions. The boundary of every region is triangle.

$$\sum_{\text{all regions } R} (\# \text{ edges on boundary } R) = 3r$$

$= 2m$ Each edge is on the boundary of 2 regions.

$$\begin{aligned} \therefore 2m &= 3r \\ r &= \frac{2m}{3} \end{aligned}$$

Euler's Formula $\Rightarrow n - m + r = 2$

$$n - m + \frac{2m}{3} = 2$$

$$3n - 3m + 2m = 6$$

$$3n - m = 6$$

$$\boxed{m = 3n - 6}$$

Corollary: If G is a planar graph with $n \geq 3$ vertices and m edges then $m \leq 3n - 6$

Proof:

Add enough edges to G so that the resulting graph G' is maximal planar and has m' edges

Then $m' = 3n - 6$

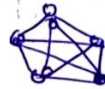
$$m \leq m'$$

$$\therefore \boxed{m \leq 3n - 6}$$

Fact: K_5 is not planar

Proof: K_5 has 5 vertices
10 edges

$$n = 5 \\ m = 10$$



If K_5 is planar, then $m \leq 3n - 6$

$$m \leq 3 \cdot 5 - 6 = 9$$

$m \leq 9$ but $m = 10 \therefore K_5$ is not planar.

Lemma: If G is a planar graph with n vertices and m edges and no triangles, then $\boxed{m \leq 2n - 4}$

Proof: Consider a plane drawing of G .

Add edges while keeping the resulting graph a plane graph with no triangles, until we get G' which is maximal w.r.t. this property.

So G' has m' edges ($m' \geq m$) and.

$$\underbrace{\sum_{R \text{ of } G'} (\# \text{ edges on boundary of } R)}_{\substack{\text{all regions} \\ R \text{ of } G'}} \geq 4r$$

G' has ≥ 4 edges on the boundary of each region

$$2m \geq 4r$$

$$r \leq \frac{2m}{4} = \frac{m}{2}$$

$$r \leq \frac{m}{2}$$

$$G' \text{ is a planar graph} \Rightarrow 2 = n - m' + r \leq n - m' + \frac{m'}{2}$$

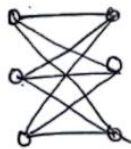
$$2 \leq \frac{2n - m'}{2}$$

$$4 \leq 2n - m'$$

$$\boxed{m \leq m' \leq 2n - 4}$$

Fact: $K_{3,3}$ is not planar

Proof $K_{3,3}$ $n=6$
 $m=9$



$K_{3,3}$ is bipartite and it has no odd cycles

In particular, $K_{3,3}$ has no triangles

If $K_{3,3}$ is planar then $m \leq 2n - 4 = 2 \cdot 6 - 4 = 8$

but $m=9$ $\therefore K_{3,3}$ cannot be planar.

Conclusion

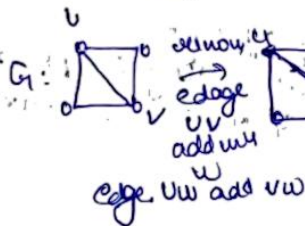
K_5 and $K_{3,3}$ are non planar graphs

Note:

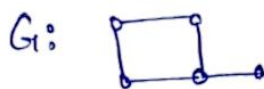
If a graph G contains K_5 or $K_{3,3}$ as a subgraph then G is not planar.

Sub-division

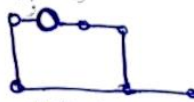
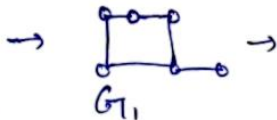
An elementary subdivision of a non-empty graph G is a graph obtained from G by removing an edge $e=uv$ and adding a new vertex w and new edges uw and vw .



A sub-division of a graph G is a graph obtained from G by a sequence of zero or more elementary sub-divisions.



G_1 is subdivision of itself



subdivision of G_2

G_3



G_4

Fact: Any subdivision H of a graph G is planar iff G is planar

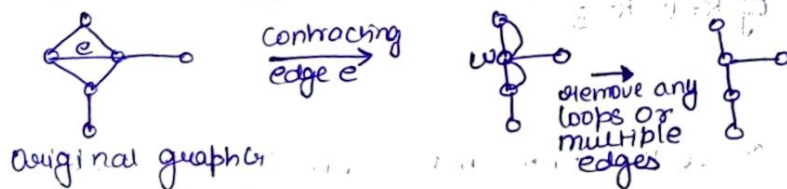
Fact: If a graph G is subdivision of K_5 or $K_{3,3}$ then G is non planar

Fact: If a graph G contains a subgraph that is a subdivision of K_5 or $K_{3,3}$ then G is non-planar.

Kuratowski's Theorem:

A graph G is Planar $\iff G$ contains no subgraph which is a subdivision of K_5 or $K_{3,3}$

Edge Contraction



Let G be a graph and $e \in E(G)$ with $e = uv$.

Suppose $w \notin V(G)$. Contracting edge e in G results in the graph G/e obtained from G by:

- removing edge e
- replacing vertices u and v with a new single vertex w
- vertex w is adjacent to the neighbours of u and to the neighbours of v .

Graph minor

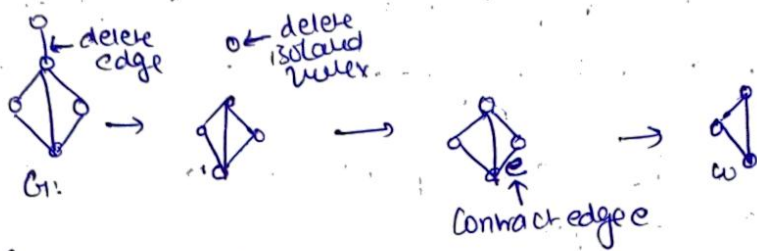
A graph H is called a minor of G , if H can be produced from G by successive application of these reductions:

- deleting an edge
- contracting an edge
- deleting an isolated vertex.

Note: Every graph is minor of itself.

Every graph that is isomorphic to a minor of G is also called a minor of G .

Ex:



G_1 has K_3 minor

Wagner's Theorem

A graph is planar $\iff$ it has no K_5 or $K_{3,3}$ minor.

Kuratowski's Theorem

A graph is planar $\iff$ it has no subgraph that is sub-division of K_5 or $K_{3,3}$.

Notes:

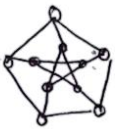
$\rightarrow$ A subdivision of H can be converted into an H -minor by contracting all but one edge in each path formed by the subdivision process.

Subdivision of K_3 . 

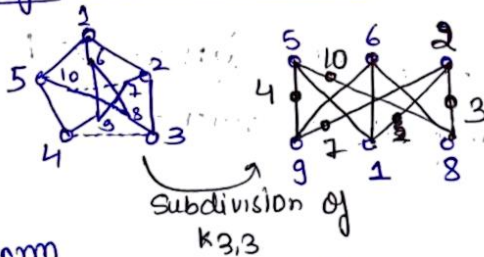
$\rightarrow$ But an H -minor cannot always be converted into a subdivision of H .

* For K_5 and $K_{3,3}$: If G has ≥ 1 of these as a minor, then it has ≥ 1 of these as a subdivision.

Is Petersen graph planar?



Proof 1: Kuratowski's Theorem



Petersen graph contains subgraph that is subdivision of K_5 and $K_{3,3}$.

Proof 2: Wagner's Theorem

Petersen graph has a K_5 minor



Perform edge contractions on edges e_1, e_2, e_3, e_4, e_5

The resulting graph is isomorphic to K_5

By Wagner's theorem, the Petersen graph is non-planar.

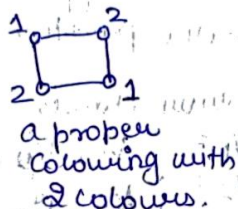
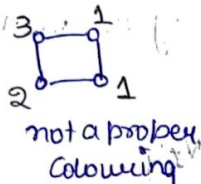
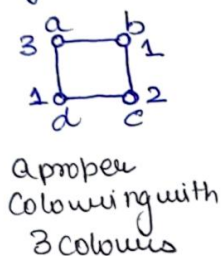
Vertex Colouring

Let G be a graph with vertex set $V(G)$.

A (proper) vertex colouring of G is a labelling of the vertex set

$$f: V(G) \rightarrow \{1, 2, 3, \dots, k\}$$

where the labels are called colours and no 2 adjacent vertices get the same colour.

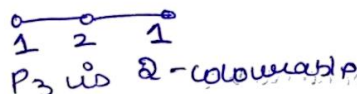
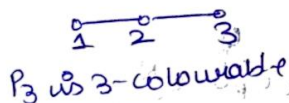


In most cases, "colouring" means "proper vertex colouring".

A k-colouring of a graph G is a colouring of G using k colours.

If G has a k -colouring then it is k -colourable.

Ex:



The chromatic number of a graph G , denoted $\chi(G)$ is the smallest k such that G is k -colourable.

Ex: $\chi(P_3) = 2$

If $\chi(G) = k$, we say that graph G is k-chromatic.

Facts:

- If G has n vertices, then $\chi(G) \leq n$
- $\chi(G) = 1 \iff G$ has no edge
- $\chi(C_{2n}) = 2$ and $\chi(C_{2n+1}) = 3$
- $\chi(K_n) = n$
- If H is a subgraph of G then $\chi(G) \geq \chi(H)$

Let $G = (V, E)$

A k -colouring of G partitions the vertex set V into k sets $V_1, V_2, \dots, V_k$ where each set V_i is an independent set

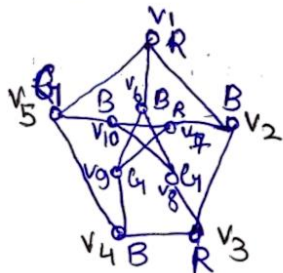
↳ no 2 vertices in the set are adjacent.

This means

$$V = V_1 \cup V_2 \cup \dots \cup V_k$$

and $V_i \cap V_j = \emptyset$ for all $i \neq j$

The independent sets $V_1, \dots, V_k$ are called colour classes



$\text{Colour 1} = R \Rightarrow$ forms the independent set
 $\text{Colour 2} = B \Rightarrow$ forms the independent set
 $\text{Colour 3} = B \Rightarrow$ forms the independent set
 also the partitions of the vertex

Colour classes

$$V_1 = \{v_1, v_2, v_3, v_4, v_5\}$$

$$V_2 = \{v_6, v_7, v_8, v_9, v_{10}\}$$

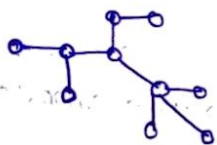
$$V_3 = \{v_1, v_2, v_3, v_4, v_5\}$$

2-colorable graph

Fact: Every tree with at least 2 vertices is 2-chromatic.

Proof: let T be a tree with $|V(T)| \geq 2$

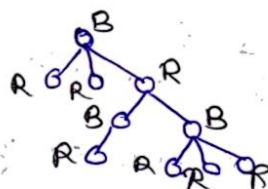
let $v \in V(T)$ and consider T to be rooted at v .



$\leadsto T$ rooted at v :

$$X = \{u \in V(T) \mid d(v, u) \text{ even}\}$$

$$Y = \{u \in V(T) \mid d(v, u) \text{ odd}\}$$



let v be coloured with colour 1.

let the neighbours of v be coloured with colour 2

let the neighbours of those vertices be colour 1

Continue

In a tree, there is a unique path b/w any 2 vertices.

now along any path in T , the vertices alternate colours.

so no pair of adjacent vertices receive the same colour.

$\therefore$ This is a 2-colouring of T and $\chi(T) = 2$

$$\because |V(T)| \geq 2$$

We have

G is a tree $\Rightarrow \chi(G) = 2$

What about the converse?

$\chi(G) = 2 \Rightarrow G$ is a tree? No.

Ex:



$\chi(C_4) = 2$

Theorem: $\chi(G) = 2 \Leftrightarrow G$ is bipartite (for nontrivial graph)

[Recall: G is bipartite $\Leftrightarrow G$ has no odd cycle]

Proof:

$\Rightarrow$ Aim to show: $\chi(G) = 2 \Rightarrow G$ is bipartite

Contrapositive: G is not bipartite $\Rightarrow \chi(G) > 2$

Suppose G is not bipartite

Then G has an odd cycle C_{2l+1} and hence $\chi(G) \geq \chi(C_{2l+1}) = 3$

$\Leftarrow$ Aim to show: G is bipartite $\Rightarrow \chi(G) = 2$

Suppose G is bipartite. Then $V(G) = X \cup Y$ such that every edge of G is of the form $e = xy$ where $x \in X$ and $y \in Y$

The colour all the vertices of X colour 1 and all vertices of Y colour 2.

Theorem:

For every graph G , $\chi(G) \leq \Delta(G) + 1$

Proof: By induction on n (the # of vertices)

Base: $n = 1$
 $G = K_1$
 $\chi(G) = 1$
 $\Delta(G) = 0$

Inductive hypothesis: Assume the result holds for every graph with $n-1$ vertices. ($n \geq 2$)

Let G be a graph on n vertices. Let $v \in V(G)$

Now $G-v$ is a graph on $(n-1)$ vertices

Inductive hypothesis: $\chi(G-v) \leq \Delta(G-v) + 1$

So $G-v$ can be coloured with at most $\Delta(G-v) + 1$ colours.

Note: $\deg_G(v) \leq \Delta(G)$

The neighbours of v use up $\leq \Delta(G)$ colours

Case 1: $\Delta(G) = \Delta(G-v)$

Then there is at least one colour of the $\Delta(G-v)+1 = \Delta(G)+1$ colours not used by the neighbours of v , so v can be coloured with that colour. This gives $(\Delta(G)+1)$ -colouring of G , so $\chi(G) \leq \Delta(G)+1$

Case 2: $\Delta(G) \neq \Delta(G-v)$

Then $\Delta(G-v) < \Delta(G)$

using a new colour for v we will have a $(\Delta(G-v)+2)$ -colouring of G

$\therefore \Delta(G-v)+2 \leq \Delta(G)+1$, we have $\chi(G) \leq \Delta(G)+1$

$$\chi(G) \leq \Delta(G)+1$$

Note:

$\chi(K_n) = n$
 $\Delta(K_n) = n-1$ } Equality in the bound

$\chi(C_{2k+1}) = 3$
 $\Delta(C_{2k+1}) = 2$ } Equality in the bound.

odd cycles and complete graphs are the only graphs that have equality in this bound.

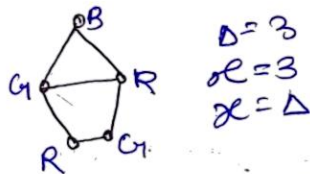
$G \neq K_n$
 $G \neq C_{2k+1}$ } $\chi(G) \leq \Delta(G)$ Brook's Theorem

Brook's Theorem:

STATEMENT

If G is a connected simple graph and it is neither an odd cycle nor a complete graph then $\chi \leq \Delta$

Ex:



Proof

Let G be a k -chromatic graph $\chi(G) = k$

$G \rightarrow$ a connected simple graph

$G \rightarrow$ is not an odd cycle graph

G is not a complete graph

$G \rightarrow k$ critical graph $[k=1, 2 \Rightarrow \text{complete graph}]$

$\therefore k \neq 1, 2 \quad (1 < k < \infty)$

Concurrency Control

- ↳ Lock Based Protocol
- ↳ Timestamping Protocol

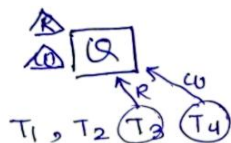
- ↳ Replication with weak consistency.
- ↳ Deadlock Handling

Q: Data item

R - Timestamp(Q) = TSCT₃ : Q last read by T₃

W - Timestamp(Q) = TSCT₁ : Q last written by T₁

Timestamp $\left\{ \begin{array}{l} \text{Logical System Clock} \\ \text{System Counter} \\ \text{Logical Counter} \end{array} \right.$



Operation: T_i → Read(Q)

- 1) If $(TSCT_i) < W\text{-timestamp}(Q)$
then T_i needs to read a value of Q that was already overwritten, the read request will be rejected.
T_i is Rollback.

- 2) If $(TSCT_i) \geq W\text{-timestamp}(Q)$
It is executed
R-Timestamp = $\max(R\text{-TS}(Q), TSCT_i)$
max, as Simultaneous Read operation allowed

Operation: T_i → write(Q)

- 1) $TSCT_i < R\text{-Timestamp}(Q)$

- It is Reject
- T_i is rollback

- 2) If $TSCT_i < W\text{-Timestamp}(Q)$

- It is Reject
- T_i is Rollback

- 3) Else

Executed

$W\text{-Timestamp}(Q) = TSCT_i$

// not max as we cannot allow Simultaneous write.

The rollback transactions will be restarted with new Timestamp.

How to globally assign Transaction Id.

Site Address	Counter
--------------	---------

not followed

Counter	Site Address
---------	--------------

followed

5101
0 (51)

2101
0 (21)

5601
0 (56)

8101
0 (81)

7901
0 (79)

0151
0 (51)

0121
0 (21)

0156
0 (56)

0181
0 (81)

0179
0 (79)

always highest

Browsed towards site.

Problem of slow clock

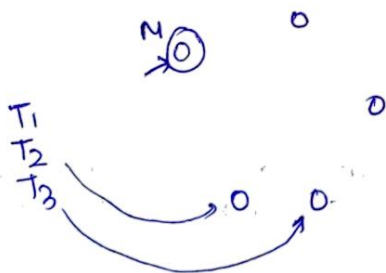
→ Define within each site S_i , a logical clock (LC_i) will generate the unique local timestamp.

→ Require that Site (S_i) advance its Logical clock whenever a request is received from T_i with a Timestamp (x, y) and x is greater than the current value of LC_i .
Counter Site

→ Set S_i advance its logical clock to $(x+1)$.

Note: Logical clock is a Counter
Physical clock is normal Clock.

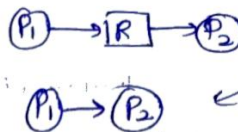
Replication of weak consistency [Employees master-slave Replication]



Deadlock Handling

RA G: [Resource Allocation Graph]

WFG:



Remove all resources from RA G to get WFG

Deadlock Detection

→ Single Instance Resource
↓ RA G → have cycle
↓
deadlock.

→ In Distributed System



Construct Global WFG.

2 methods for Deadlock Detection

→ Centralised Approach [One Coordinator]

→ Distributed Approach

Responsible for WFG (Global)
↓
Identify
↓
Take necessary steps.

- Continuous
- Periodic
- Transfer on Request

3 Approaches with which state information are transferred from local site to coordinator.

Example: T_1, T_2, T_3 — 3 Transaction
 R_1, R_2, R_3 — 3 Resources
S1 S2 } [Continuous transfer]

Step 1: $T_1 \rightarrow R_1$ [Allocated]

Step 2: $T_2 \rightarrow R_2$ [Allocated]

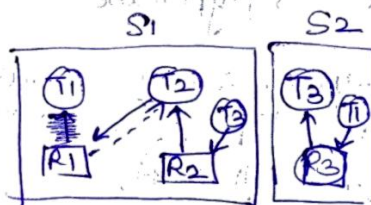
Step 3: $T_3 \rightarrow R_3$ [Allocated]

Step 4: $T_2 \rightarrow R_1$ [wait]

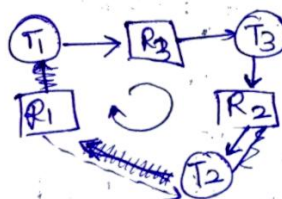
Step 5: $T_3 \rightarrow R_2$ [wait]

Step 6: T_1 releases R_1 [allocated to T_2]

Step 7: $T_1 \rightarrow R_3$ [wait]



Central Coordinator



S1

S2 CC

S2 is closer to CC

7 reaches at CC before 6

leads to FALSE CYCLE

ADSA Parallel Algorithms [Parallel Architecture and Algorithms] Hwang & Briggs

Parallel Algorithm — m.J. Quinn.

Programming point of view

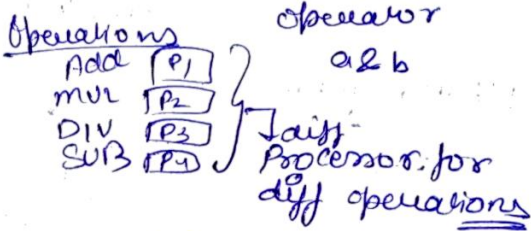
→ all operators must be associative [right or left]

eg. → unary operator [considered as right associative]
→ Binary operators [←———→ left associative]

→ Postfix Evaluation.

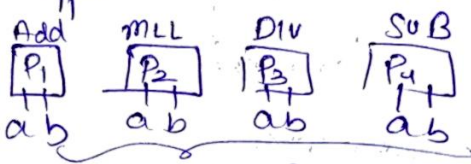
$a + b * c / d - e$

→ parallelly ⇒



$1+2+3+4+5+6$

3 7 11

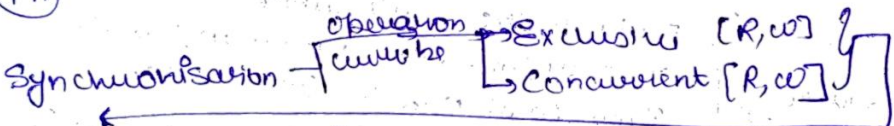


Synchronised

↓ using

Parallel RAM aka PRAM

- P1
- P2
- ...
- Pn



Concept → PRAM Algo

mesh → hypercube

- 4 architecture
- SISD
 - MISD
 - SIMD
 - MIMD

$1+2+3+4+5+6+7+8+9+10$

P1 P2 P3 P4

how to redeem

$3 + 7 + 11 = 21$

now 11 to next Processor

37

10

11



Wraparound

But 3-critical graph are odd cycle graph. $\therefore K \neq 3 \Rightarrow K \geq 4$

(case i) let G has a vertex cut $\{u, v\}$

"Let G be a K -critical graph with a 2-vertex cut $\{u, v\}$. Then $d(u) + d(v) \geq 3K - 5$.

$$2\Delta \geq d(u) + d(v) \geq 3K - 5 \geq 2K - 1$$

$$2\Delta \geq 2K - 1$$

$$K \leq \Delta$$

$$\boxed{\chi \leq \Delta} \text{ as } \chi = K$$

$$\boxed{\chi(G) \leq \Delta(G)}$$

(Case ii): let us assume that, G is 3-connected.

$\therefore G$ is not complete

"If G is simple and connected but not complete then G has 3 vertices u, v and w such that $uv, vw \in E$ and $uw \notin E$ "

$$\text{let } \{v_1, v_2, v_3, v_4, \dots, v_v\} = V$$

$$\text{let us set } u = v_1 \text{ and } w = v_2$$

$$G - \{u, w\}$$

$v_3, v_4, v_5, \dots, v_v$ be any ordering of the vertices of $G - \{u, w\}$ such that each v_i is adjacent to some v_j with $j > i$.

This can be achieved by arranging the vertices of $G - \{u, w\}$ in non-increasing order of their distance from v .

we can a Δ -colouring of G .

assign colour 1 to $v_1 = u$ and then successively colour

$v_3, v_4, \dots, v_v$ each with first available colour from the list $\{1, 2, 3, \dots, \Delta\}$

By the constructions of the sequence $v_1, v_2, \dots, v_v$ each vertex v_i

$1 \leq i \leq v-1$ is adjacent to at most $\Delta-1$ colours, and thus

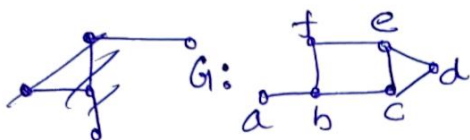
that one of the colours $1, 2, 3, \dots, \Delta$ will be available

Finally, $\therefore v_v$ is adjacent to 2 vertices of color 1, it is adjacent to at most $\Delta-2$ other colors. It can be assign one of the colours $2, 3, \dots, \Delta$. $\therefore \chi(G) \leq \Delta(G)$

Networks and flows

Vertex Separating Set

Let S be a vertex cut of G and let u and v be vertices from distinct components of $G - S$. Then S is a $u-v$ separating set.



$$S = \{e, c, b\}$$

Ford-Fulkerson Algorithm for maximum flow Problem

Problem:

Given a graph which represents a flow network where every edge has a capacity.

Also given 2 vertices

Source s and sink t in the graph. Find out the maximum possible flow from s to t with following constraints.

(a) Flow on edge doesn't exceed the given capacity of the edge.

(b) Inflow = Outflow for every vertex except s and t .

Algorithm:

Ford Fulkerson Algorithm

The following is simple idea of the algorithm.

1] Start with a initial flow as 0.

2] while there is an augmenting path from source to sink
Add this path flow to flow.

3] Return flow.

Terminologies

* Residual graph: It's a graph which indicates additional possible flow.

If there is such path from source to sink then there is a possibility to add flow.

* Residual capacity: It is original capacity of the edge minus flow.

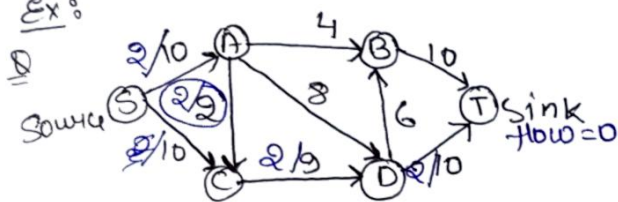
* minimal cut: Also k/a bottleneck capacity, which decides maximum possible flow from source to sink through an augmented path.

Augmenting path

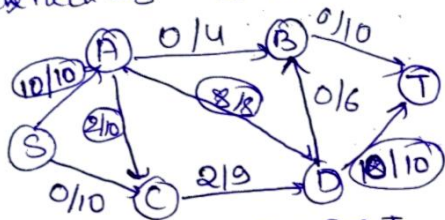
Augmenting path can be done in 2 ways:-

- 1> non-full forward edges
- 2> non-empty backward edges

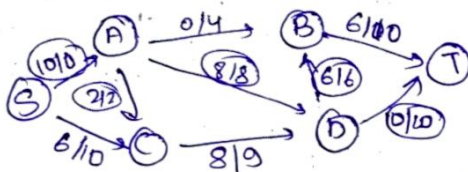
Ex:



Bottleneck = 8 $S \rightarrow A \rightarrow D \rightarrow T$



Bottleneck = 6 $S \rightarrow C \rightarrow D \rightarrow B \rightarrow T$



no more augmenting path possible $\therefore$ max

Augmenting Paths	Bottle neck Capacity
$S \xrightarrow{10} A \xrightarrow{2} C \xrightarrow{2} D \xrightarrow{8} T$	2
$S \xrightarrow{8} A \xrightarrow{8} D \xrightarrow{8} T$	8
$S \xrightarrow{10} C \xrightarrow{7} D \xrightarrow{6} B \xrightarrow{10} T$	6

NPhard and NP complete

Polynomial time ✓

- Linear Search - n
- Binary Search - $\log n$
- Insertion Sort - n^2
- merge sort - $n \log n$
- Matrix multiplication - n^3
- n^4
- n^{100}

Polynomial time

Non-deterministic

Algorithm Nsearch(A, n, key)

$j = \text{choice}()$ $\rightarrow$ non deterministic $O(1)$

if (key = A[j])

write(j);

Success() $\rightarrow$ non deterministic $O(1)$

$j = \text{write}(0);$

$j = \text{failure}()$ $\rightarrow$ non deterministic $O(1)$

$O(1) \leftarrow O(n^{100})$

Exponential time

0/1 Knapsack - 2^n

Travelling Salesman Problem - 2^n

Sum of subset S - 2^n

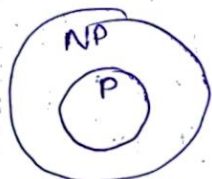
Graph coloring - 2^n

Hamilton cycle - 2^n

P class :
 set of deterministic algorithm which takes polynomial amount of time

NP class

set of non-deterministic algorithm which require more than polynomial time.



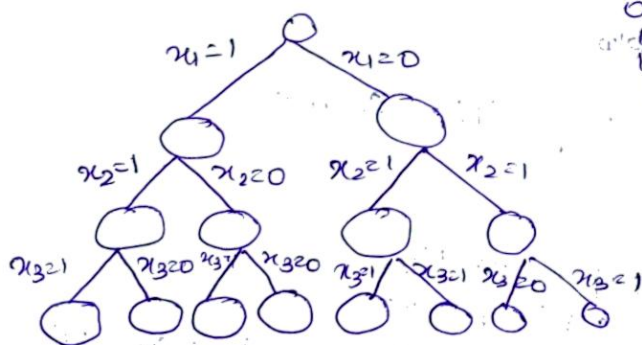
CNF-Satisfiability $\rightarrow 2^n$ { Base Problem }

$$x_i = \{x_1, x_2, x_3\}$$

Eg: 1
 $CNF = (x_1 \vee \overline{x_2} \vee x_3) \wedge (\overline{x_1} \vee x_2 \vee \overline{x_3})$

C1 C2

$\therefore$ this is also exponential time problem



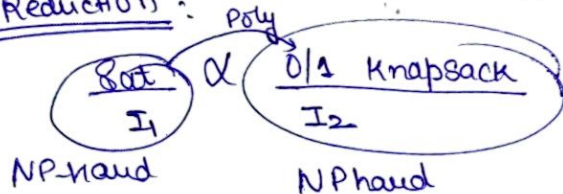
x_1	x_2	x_3
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

of entries $= 2^3$
 i.e. 2^n for n variables

0/1 Knapsack $n=3$
 $\langle 0/1 \ 0/1 \ 0/1 \rangle$ $m=8$

Let's say CNF-Satisfiability Problem is NP-Hard Problem.

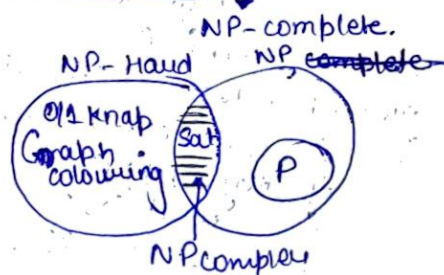
How to show other problems are related to CNF-Satisfiability
Using Reduction :



$$Sat \propto L \uparrow \text{NP-hard}$$

$$Sat \propto L_1 \uparrow \text{NP-hard} \quad \text{and} \quad L_1 \propto L_2 \uparrow \text{NP-hard}$$

CNF-Satisfiability
 Non-Deterministic algorithm that
 for NP-Hard $\rightarrow$



Cook's Theorem

The satisfiability problem (SAT) is NP-complete.

What is SAT?

A propositional logic formula ϕ is called satisfiable if there is some assignment to its variable that makes it evaluate to true.

E.g.
 $\rightarrow p \wedge q$ is satisfiable $p=1, q=1$

$\rightarrow p \wedge \neg p$

3SAT

A language $3SAT = \{ \phi \mid \phi \text{ is satisfiable 3-CNF formula} \}$

3-CNF

$\rightarrow$ CNF with every clause having 3 literals

E.g.: $(x \vee y \vee z) \wedge (\neg x \vee \neg y \vee z)$

Cook's Theorem (Proof outline)

Theorem: SAT is NP complete.

Proof consist of 2 step

- 1] Convert the execution of a polynomial time NDTM to a bunch of well formed formulae such that formula satisfy iff machine accept input.
- 2] Show the sum of lengths of formula is polynomial in the size of problem.

• NPhard(CL) - can polynomially reduce any NP problem to L

• NP complete - $L \in NP$

• $L \in NP$ $\rightarrow$ NDTM for L that runs in polynomial time.

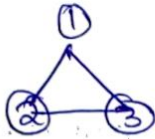
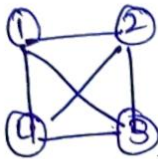
• An NDTM is the only model we have for NP problems.

• SAT $\in NP$

• Therefore, if we can polynomially reduce and an arbitrary polynomial NDTM to SAT, it means we have

Proven SAT is NP-complete.

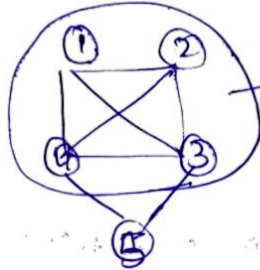
Clique - Decision Problem [CDP]



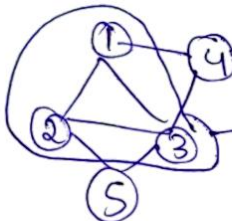
what is a clique

complete graph

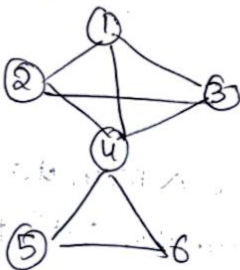
$$\begin{aligned} |V| &= n \\ |E| &= \frac{n(n-1)}{2} \end{aligned}$$



subgraph of a graph which is complete
↓
clique



Complete subgraph



$$\begin{aligned} K=4 \\ K=3 \\ K=2 \end{aligned}$$

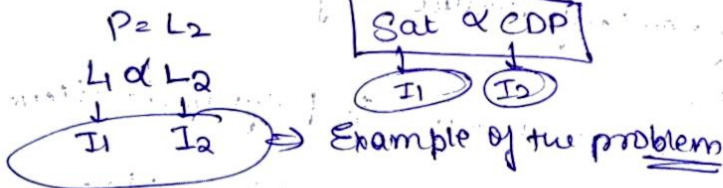
⇒ maximal size of clique

4

Finding a clique of k size + not in the graph
- Decision Problem
Finding maximum clique of the graph
- optimisation Problem

now we need to prove that CDP is NP-Hard Problem

Before that : General method to prove any Problem NP-Hard,



Sat $\leq$ CDP
 x_1, x_2, x_3

$$F = (x_1 \vee x_2) \wedge (\bar{x}_1 \vee \bar{x}_2) \wedge (x_1 \vee x_3)$$

$\quad \quad \quad C_1 \quad \quad \quad C_2 \quad \quad \quad C_3$

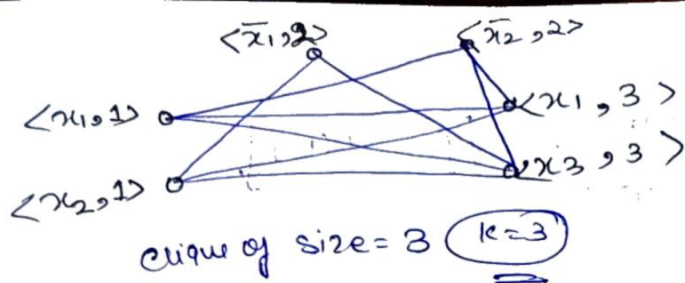
Vertices: all the literals of each clause
Edges:

$G: V = \{ \langle a, i \rangle \mid a \in C_i \}$

$E = \{ \langle \langle a, i \rangle, \langle b, j \rangle \rangle \mid i \neq j \text{ and } b \neq \bar{a} \}$

$F = \bigwedge_{i=1}^k C_i$

1) do not add edges b/w the same clause vertices
2) do not connect clause to its negation



Asymptotic Notation

$$1 < \log n < \sqrt{n} < \underbrace{n}_{\text{Average bound}} < n \log n < n^2 < n^3 < \dots < 2^n < 3^n < \dots < n^n$$

lower bound upper bound

- O (Big-oh) upper Bound
- Ω (Big-omega) lower Bound
- Θ (theta) Average bound

Big-Oh

The function $f(n) = O(g(n))$ iff $\exists$ +ve constant C and n_0 such that $f(n) \leq C * g(n) \forall n \geq n_0$

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

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

or

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

$$\boxed{2n + 3 \leq 5n} \quad n \geq 1$$

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

$$2n + 3n \leq 5n^2$$

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

Tightest upper bound

$$\begin{aligned} f(n) &= O(n^2) \\ f(n) &= O(n^3) \\ f(n) &= O(2^n) \end{aligned}$$

$$f(n) = O(\log n) \quad \text{False}$$

Omega Notation (Ω)

The function $f(n) = \Omega(g(n))$ iff $\exists$ +ve constant C and n_0 such that $f(n) \geq C * g(n) \forall n \geq n_0$

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

$$2n + 3 \geq 1 * \log n \quad \forall n \geq 1$$

$$\begin{aligned} \uparrow & \quad \uparrow \\ f(n) & \geq C * g(n) \end{aligned}$$

Tightest lower bound

$$\begin{aligned} f(n) &= \Omega(n) \\ f(n) &= \Omega(\log n) \\ f(n) &= \Omega(n^2) \quad \text{X} \end{aligned}$$

Theta notation

The function $f(n) = \Theta(g(n))$ iff $\exists$ +ve constants C_1, C_2 and n_0 such that $C_1 * g(n) \leq f(n) \leq C_2 * g(n)$

Eg: $f(n) = 2n + 3 \quad n \leq 2n + 3 \leq 5n$

$$\begin{aligned} f(n) &= \Theta(n) \\ f(n) &= \Theta(n^2) \quad \text{X} \end{aligned}$$

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

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

$f_{\text{ans}} = O(n^2)$

$$\boxed{2n^2 + 3n + 4 \leq 9n^2} \quad n \geq 1$$

$\uparrow \quad \uparrow$
 $c \quad g(n)$

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

$$\boxed{\Omega \text{ cm}^2}$$

$$1 \cdot n^2 \leq 2n^2 + 3n + 4 \leq 9n^2$$

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

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

$$n^2 \log n \leq n^2 \log n + n \leq 10 n^2 \log n$$

$$O(n^2 \log n) \quad \Omega(n^2 \log n)$$

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

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

$$1 \times 1 \times \dots \times 1 \leq 1 \times 2 \times 3 \times \dots \times n \leq n \times n \times n \times \dots \times n$$

$$1 \leq n! \leq n^n$$

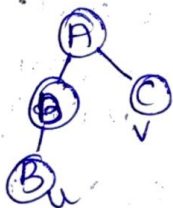
$$O(n^2) \quad O(2n)$$

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

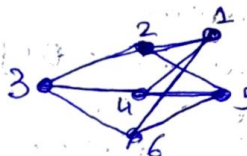
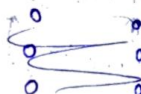
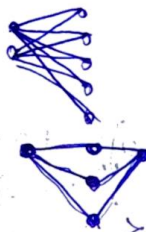
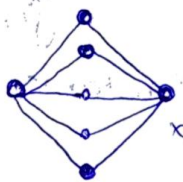
$$\log(1 \times 2 \times 3 \times \dots \times n) \leq \log(n \times n \times \dots \times n)$$

$$\log 1 \leq \log n \leq \log n$$

$O(n \log n)$ $O(2C)$



A, B, C

True

1-2-3-4-5
6

NP Hard

A Problem X is NP

NP Problems

Set of problems whose soln are hard to find but easy to verify and are solved by non-deterministic m/c in polynomial time

→ soln hard to find but easy to verify

→ solved by non deterministic m/c in polynomial time.

NP-Hard (x)

↓

it is a NP-Complete problem y

$y \xrightarrow[\text{to}]{\text{reducible}}$ x in P.T.

SAT, 0/1 Knapsack

NP-complete (x)

↓
it there is an NP problem y such that y is reducible to x in polynomial time

$y \rightarrow x$

Vertex cover Problem
Hamiltonian problem

CDP: To find if a clique of size k exist or not.

Cook's theorem says that SAT is a NP Complete Problem $\Rightarrow$ SAT $\rightarrow$ CDP
 NP Hard

$P = L_2$

S and C are related to C

NO $\leftarrow S$ and C

SAT is

By Cook's theorem SAT(S) is NP-Complete

$\therefore NP \propto S$

$$F_2 = (x_1 \vee x_2) \wedge (\bar{x}_1 \vee \bar{x}_2) \wedge (x_1 \vee x_3)$$

SAT is NP complete

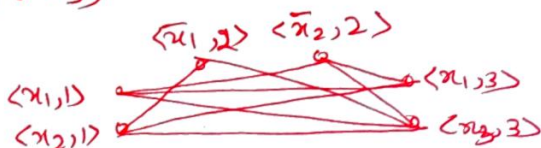
~~CDP~~ SAT $\propto$ CDP m.p.

$$F_2 = \underbrace{(x_1 \vee x_2)}_{C_1} \wedge \underbrace{(\bar{x}_1 \vee \bar{x}_2)}_{C_2} \wedge \underbrace{(x_1 \vee x_3)}_{C_3}$$

$G: V = \{ \langle a, i \rangle \mid a \in C_i \}$
 $E = \{ \langle a, i \rangle \langle b, j \rangle \mid i \neq j \text{ and } b \models \bar{a} \}$

$F_2 = \bigwedge_{i=1}^K C_i$
 $\downarrow$
 $\bar{x}_1, x_3$

$$F = (x_1 \vee x_2) \wedge (\bar{x}_1 \vee \bar{x}_2) \wedge (x_1 \vee x_3)$$



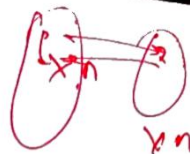
$\bar{x}_1, x_2, x_3$
FOIL
 F_2 TRUE

$G(V, E)$

$V = \{ \langle a, i \rangle \mid i \in C_i \}$
 $E = \{ \langle a, i \rangle \langle b, j \rangle \mid i \neq j \text{ and } b \models \bar{a} \}$

k clause in SAT
now we have to get
max clique size k

G is bipartite $\rightarrow G$ has no odd cycle



$$X \cap Y = \emptyset$$

$$X \cup Y = V(G)$$

G

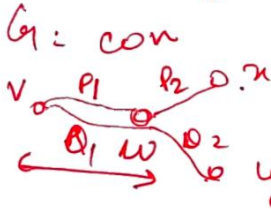
$$C = v_1, v_2, \dots, v_n \rightarrow n = \text{odd}$$



odd cycle
 $G \rightarrow$ Bipartite

G has no odd cycle $\rightarrow$ Bipartite
Cyclicity.

(a)



$$X = \{x \mid d_G(x, v) = \text{odd}\}$$

$$Y = \{y \mid d_G(v, y) = \text{even}\}$$

$$xy \in E \quad xy \notin E$$

$$P: v \rightarrow x$$

$$Q: v \rightarrow y$$

$$w = \text{com} \rightarrow$$



$$P = P_1 P_2$$

$$Q = Q_1 Q_2$$

Sub

— N.L.H.C

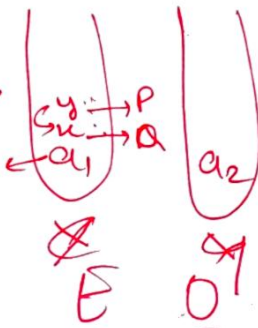
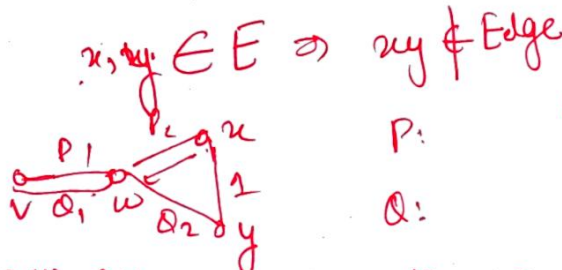
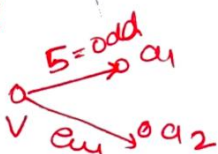
$$v \xrightarrow{P_1} w \quad w \xrightarrow{P_2} x$$

$$v \xrightarrow{Q_1} w \quad w \xrightarrow{Q_2} y$$

$$|P_1| = |Q_1|$$

$$x, y \in X$$

$$P, Q$$



$$P = Q \text{ path}$$

$$P = P_1 + P_2$$

$$Q = Q_1 + Q_2$$

$$\text{odd} + \text{odd} = \text{even}$$

$$\text{even} + \text{even} = \text{even}$$

$$x \rightarrow w \rightarrow y = \text{even length } |P_1| = |Q_1|$$

$$xy \in E(G)$$

$$\text{even} + 1 = \text{odd}$$

K-Dimensional Tree

→ A KD tree is a Binary ^{search} tree where data in each node is a k-dimensional point in space. It is a space partitioning DS for organising points in k-dimensional space.

→ Points left to this space are represented by left subtree.
Points right to this space are represented by right subtree.

→ In 2 dimensional case

- the root node would be at x-aligned plane
- the children of root node would be at y-aligned plane
- the grandchildren of root node would be again at x-aligned plane

→ At depth D, A = aligned plane is calculated as

$$A = D \bmod k$$

For $k=2$ $A=0 \rightarrow x\text{-plane}$
 $A=1 \rightarrow y\text{-plane}$

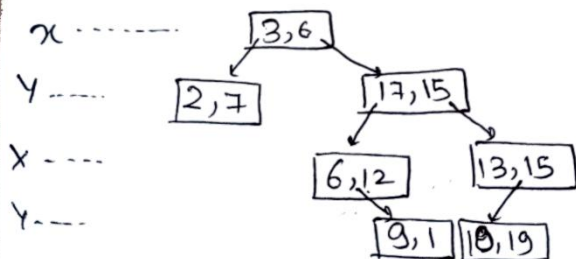
→ If the root node is aligned in plane A, then the left subtree will contain all points whose coordinates in that plane are smaller than of root node.

→ Similarly right subtree will contain all points whose coordinates in that plane are ~~smaller~~ greater or equal to that of root node

Ex: construction of a tree

Consider following points in 2-D plane

(3,6) (2,7) (17,15) (6,12) (9,1) (10,19) (13,15)



1) compare with x of root node
then move accordingly

2) If already node exist then then compare with next coordinate of the candidate with next level and continue level by level

Steps are same for any value of k

Space Partitioning

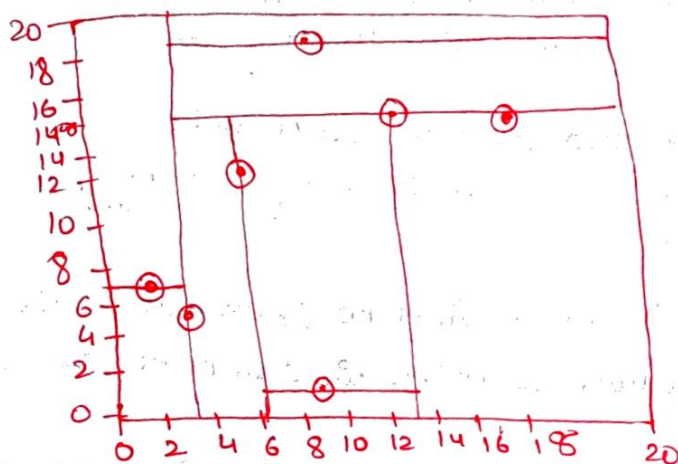
- 1) Point (3,6) divides the space into 2 parts by the line $x=3$
- 2) Point (2,7) divides the space left to $x=3$ into 2 parts by the line $y=7$
- 3) Point (17,15) divides the space right to $x=3$ by the line $y=15$

4) Point $(6, 12)$ divides the space below $y=15$ and to the right of $x=3$ by the line $x=6$.

5) Point $(13, 15)$ divides the space below $y=15$ and to the right of the line $x=6$ by the line $x=13$.

6) Point $(9, 1)$ divides the space between $x=3$, $x=6$ and $y=15$ by the line $y=1$.

7) Point $(10, 19)$ divides the space to the right of $x=3$ and above $y=15$ by the line $y=19$.



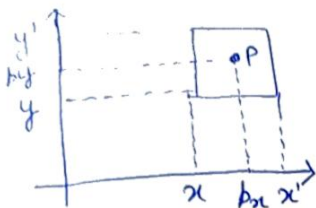
In General

Let P be a set of n points in the plane. Let us assume that no 2 points in P have the same x -coordinate and no 2 points have the same y -coordinate. This restriction is not very realistic, can be overcome with some tricks.

A 2D rectangular range query on P asks for the points from P lying inside a query rectangle, $[x:x'] \times [y:y']$.

A point $p = (p_x, p_y)$ lies inside this rectangle iff $p_x \in [x:x']$ and $p_y \in [y:y']$.

We would say that a 2D rectangular range query is composed of 2 1-D subqueries, one on the x -coordinate of the points and one on the y -coordinate.



Algorithm Building Kd tree (P, depth)

Input: A set of points P and the current depth.

Output: The root of a kd tree storing P.

1.) if P contains only one point

2.) then return a leaf storing this point.

3.) ~~else~~ if depth is even

then split P into 2 subsets with a vertical line ~~and~~ ^{ex l} through the median x-coordinate of points in P. Let P_1 be the set of points to the left of ~~it~~ ^{ex l} or on ~~it~~ ^{ex l} and let P_2 be the set of points to the right of ~~it~~ ^{ex l}.

5.) else split P into two subsets with a horizontal line l through the median y-coordinate of the points in P. Let P_1 be the set of points below l or on l and let P_2 be the set of points above l .

6.) $V_{\text{left}} \leftarrow \text{Build Kd tree } [P_1, \text{depth}+1]$

7.) $V_{\text{right}} \leftarrow \text{Build Kd tree } [P_2, \text{depth}+1]$

8.) Create a node v storing l , make V_{left} the left child of v , and make V_{right} the right child of v .

9.) return v .

Complexity: $T(n) = \begin{cases} O(1) & \text{if } n=1 \\ O(n) + 2T(n/2) & \text{if } n>1 \end{cases}$

It is $O(n \log n)$

Algorithm Search Kd tree $[V, R]$

Input: The root of (a subtree of) a kd tree and a range R.

Output: All points of leaves below v that lie in the range.

- 1.] if v is a leaf
- 2.] then Report the point stored at v if it lies in R
- 3.] else if region $[LC(v)]$ is fully contained in R
- 4.] then Report Subtree ($LC(v)$)
- 5.] else if region $[LC(v)]$ intersects R
- 6.] then Search Kd tree ($LC(v), R$)
- 7.] if region $[RC(v)]$ is fully contained in R
- 8.] then Report Subtree ($RC(v)$)
- 9.] else if region $[RC(v)]$ intersects R
- 10.] then Search Kd tree ($RC(v), R$)

Note:- The Subroutine Report Subtree (v) which traverses the Subtree rooted at a node v and reports all the points stored at its leaves.

$LC(v) \rightarrow$ left child of a node v ,
 $RC(v) \rightarrow$ right child of a node v .

Binomial Heap

- $\rightarrow$ Heaps are frequently used to implement priority queues
- $\rightarrow$ A priority queue is a collection of elements such that each element has an associated priority. In this kind of queue, element to be deleted is the one with highest (or lowest) priority. At any time, an element with arbitrary priority can be inserted into the queue.
- $\rightarrow$ A max^(min) tree is a tree in which the key value in each node is no smaller (larger) than the key values in its children (if any).
- $\rightarrow$ A max heap is a Complete Binary tree that is also a max tree
- $\rightarrow$ A min heap is a Complete Binary tree that is also a min tree.

Binomial Heap

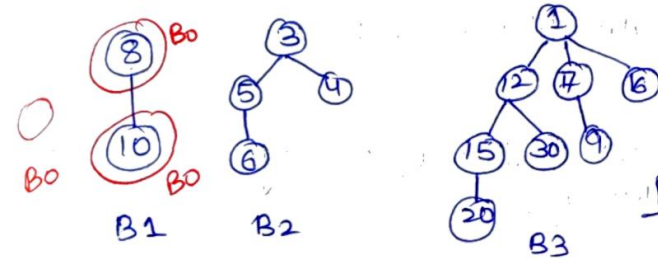
- $\rightarrow$ A binomial heap is a data structure that supports the functions like
 - insert
 - delete min (delete max)
 - meld
- $\rightarrow$ Here, if we amortise (spread out) the part of the cost of the expensive operations over the inexpensive one, then the amortised complexity of

of an individual operation is either $O(1)$ or $O(\log n)$ depending on the type of operation.

→ A min binomial heap is the collection of min trees;
A max binomial heap is the collection of max trees;

• Here we shall explicitly discuss on min binomial heaps and we will refer them as B-heaps.

• The following figure shows an example of a B-heap that is made up of 3 min trees.



Single node: B_0
 $B_k = 2 B_{k-1}$ that are linked together.

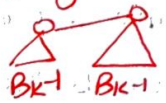


Figure 1:

B-heaps are represented using nodes that have the following data members:
degree, child, link and data.

1) degree: The degree of a node is the number of children it has.

2) The child data members are used to point to any one of its children (if any);

3) link data member: is used to maintain singly linked list of siblings.

Note: All the children of the node form the singly linked circular list and the nodes points to one of these children.

4) Additionally, the roots are min trees that comprises a B-heap are linked to form a singly linked circular list.

The B heap is then pointed at by a single pointer min to the min tree root with smaller key.

Ex: Corresponding to the Fig 1, the B heap is represented by as:

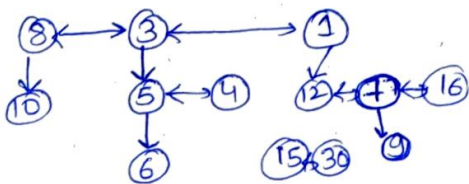


Fig. 2

Insertion into a Binomial tree

An element x may be inserted into a B-heap by first putting x into a new node and then inserting this node into a singular circular linked list pointed at by min.

The pointer min is reset to this new node only if min is 0 or the key of x is smaller than the key in the node pointed at by min. It takes $O(1)$ time.

Melding 2 Binomial Heaps ($\log n$)

In this case we meld the top circular lists of each into a single circular list.

The new B-heap pointer is the min pointer of one of the 2 trees, depending on which has the smaller key.

Since one single comparison is sufficient, complexity is $O(1)$.

Deletion of min Element ($\log n$)

If min is 0, then the Binary heap is empty and a deletion cannot be performed.

Assume that min is not 0.

min points to the node that contains the min element. This node is deleted from its circular list. The new Binary Heap consists of the remaining min trees and the sub-min trees of the deleted root.

Ex: Refer Fig1 and the following ones after deletion.

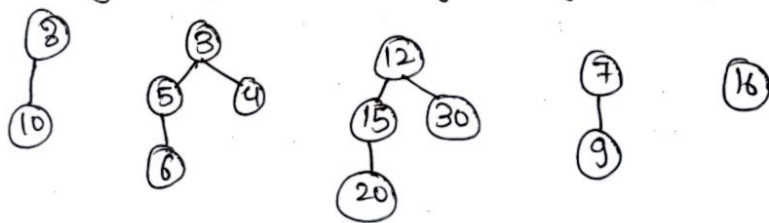


Fig3

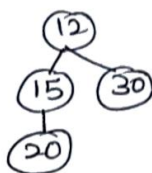
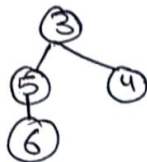
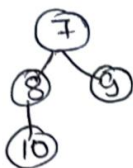
Before forming the circular list of min tree roots, we repeatedly join together pairs of min trees that have the same degree (the degree of non-empty min tree is the degree of its root).

The min tree joining has been done by making the min tree whose root has a larger key a subtree of the other [This is broken arbitrarily].

Ref. Figure-3

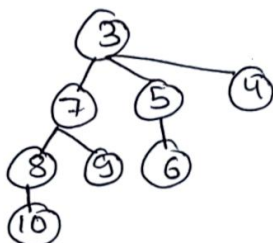
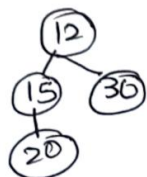
Ex:

Step 1:



(16)

Step 2



(16)

Fibonacci-Heap

The Fibonacci-Heap is a data structure that supports the 4 Binary Heap operations:-

- Get-min • Insert • Delete-min and • merge.

and following addition operations

1) Delete :- Delete the element in the specified node. we refer this delete operation as arbitrary delete ($O(\log n)$).

2) Decrease Key :- Decrease the key/priority of a specified node by the given positive amount ($O(1)$).

Note: B-heaps are special case of Fibonacci heaps.

Deletion from F-heaps

To delete an arbitrary node b from a F-heap, the following steps are done

1) If $(min = b)$ then do a delete-min otherwise do step 2, 3, and 4.

2) Delete b from its doubly linked list.

Combine the doubly linked list of b's children with the doubly linked list pointed at by min into a single doubly linked list.

Trees of equal degree are not joined as in a delete-min operation.

4) Dispose of node "b".

Ex: Ref. Figure 1.

Deletion of node containing 12. Following figure illustrates it.

[The subtrees are just separated]

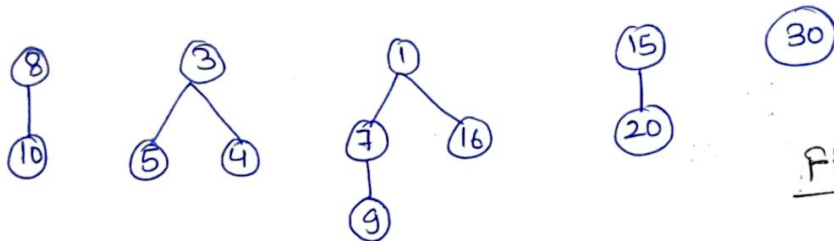


Figure 5

Decrease key

Following steps are done to decrease the key in node b.

- 1.) Reduce the key in b.
- 2.) If b is not a min tree root and its key is smaller than that in its parent, then delete from its doubly linked list and insert it into the doubly linked list of min tree roots.
- 3.) change min to point to b if the key in b is smaller than that in min.

Ex: Ref Figure 1. To decrease the Key 15 by 4

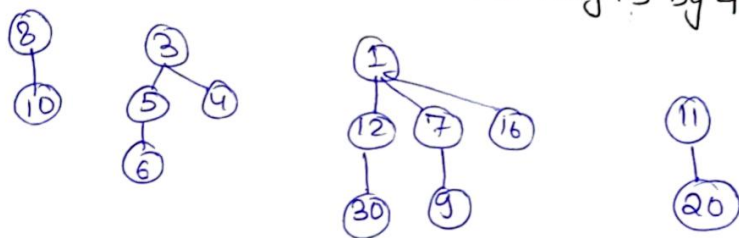


Figure-6

Matrix Operations

Linear Equations Solver

- * It lies at the heart of scientific computing.
- * Efficient algorithms for working with matrices have many practical applications.
- * One important issue that arises in practice is numerical stability.
- * Some floating point representation and/or computation may lead to numerically unstable due to round off errors considered for the limited precision.

Solving the System of Linear Equations

The linear equations with n unknowns $x_1, \dots, x_n$

$$\sum_{j=1}^n a_{ij}x_j = b_i, \quad i=1, \dots, n \quad \text{--- (1)}$$

In matrix notation in letting $A = (a_{ij})$ $x = (x_i)$ & $b = (b_i)$
Equation (1) can be represented as

$$Ax = b$$

If A is **non singular** it poses A^{-1} and
 $x = A^{-1}b$ is the solution vector

Uniqueness

Let x' be the solution apart from x .

$$\text{Then } Ax = Ax' = b$$

$$x = Ix = (A^{-1}A)x = A^{-1}(Ax) = A^{-1}(Ax') = (A^{-1}A)x' = x'$$

$$\therefore x = x'$$

Sufficient condition

A is non singular or equivalent

That is rank of A i.e. $\rho(A) = \#$ of unknowns

Underdetermined system

no. of equations is less than the no. of unknowns [infinite/no solⁿ]

Overdetermined system

If no. of equations exceeds the no. of unknown, and may not have any solution.

In case 3 instability may occur

The other approach : LUP decomposition.

This is faster and numerically stable.

The idea behind LUP decomposition is to find 3 matrices L , U and P

$$\text{Such that } PA = LU$$

- Where
- $\circ L =$ Unit lower Δ matrix
 - $\circ U =$ Unit upper Δ matrix
 - $\circ P =$ Permutation matrix.

Note :- Every non singular matrix A possesses such a decomposition.

$$U = \begin{pmatrix} u_{11} & u_{12} & \dots & u_{1n} \\ 0 & u_{22} & \dots & u_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & u_{nn} \end{pmatrix}, L = \begin{pmatrix} l_{11} & 0 & 0 & \dots & 0 \\ l_{21} & l_{22} & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ l_{n1} & l_{n2} & l_{n3} & \dots & l_{nn} \end{pmatrix}$$

$$u_{ij} \neq 0, i > j$$

It is unit for all $u_{ii} = 1$

$$l_{ij} \geq 0, i < j$$

It is unit if $l_{ii} = 1$ for i .

P matrix has exactly one 1 in each row or column and 0's elsewhere
one example.

$$P = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

Note: By the help of P, we can permute (rearrangement) of elements of $\vec{x}$.

Taking appropriate action we may have

$$PAx = Pb$$

$$LUx = Pb$$

$$Ly = Pb \quad \text{where } Ux = y$$

By forward substitution solve for then by back substitution solve for x .

∵ P is invertible, then the relation $PA = LU$

can be written as

$$P^{-1}PA = P^{-1}LU$$

$$\boxed{A = P^{-1}LU}$$

Hence the vector x is our solution to $Ax = b$:

$$Ax = P^{-1}LUx$$

$$= P^{-1}Ly$$

$$= P^{-1}Pb$$

$$\boxed{Ax = b}$$

Forward and Backward Substitution.

Algorithm

→ Permutation matrix P is represented by array π .

LUP solve (L, U, π, b)

$n = L$ rows

let x be new vector of length n

for $i = 1$ to n

$$y_i = b[\pi[i]] - \sum_{j=1}^{i-1} L_{ij} y_j$$

for $i = n$ down to 1

$$x_i = \left(y_i - \sum_{j=i+1}^n U_{ij} x_j \right) / U_{ii}$$

return x

Example

$$\begin{pmatrix} 1 & 2 & 0 \\ 3 & 4 & 4 \\ 5 & 6 & 3 \end{pmatrix} x = \begin{pmatrix} 3 \\ 7 \\ 8 \end{pmatrix}$$

$$A = \begin{bmatrix} 1 & 2 & 0 \\ 3 & 4 & 4 \\ 5 & 6 & 3 \end{bmatrix} \quad b = \begin{bmatrix} 3 \\ 7 \\ 8 \end{bmatrix}$$

method LUP decomposition

$$L = \begin{bmatrix} 1 & 0 & 0 \\ 0.2 & 1 & 0 \\ 0.6 & 0.5 & 1 \end{bmatrix} \quad U = \begin{bmatrix} 5 & 6 & 3 \\ 0 & 0.8 & -0.6 \\ 0 & 0 & 2.5 \end{bmatrix} \quad P = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

Verify $PA = LU$

$$Ly = Pb$$

$$\begin{bmatrix} 1 & 0 & 0 \\ 0.2 & 1 & 0 \\ 0.6 & 0.5 & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} 8 \\ 3 \\ 7 \end{bmatrix}$$

$$y = \begin{pmatrix} 8 \\ 1.4 \\ 1.5 \end{pmatrix}$$

Now $Ux = y$ given

$$\begin{bmatrix} 5 & 6 & 3 \\ 0 & 0.8 & -0.6 \\ 0 & 0 & 2.5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 8 \\ 1.4 \\ 1.5 \end{bmatrix}$$

$$x = \begin{bmatrix} -1.4 \\ 2.2 \\ 0.6 \end{bmatrix}$$

Computing LU decomposition

$$A = LU$$

Algorithm

$n = A$: rows

let L and U be $n \times n$ matrices initialises U with 0's below the diagonal.

Initialise L with 1's on the diagonal and 0's above the diagonal.

for $k = 1$ to n

$$U_{kk} = a_{kk}$$

for $i = 1$ to $k+1$ to n

$$d_{ik} = a_{ik} / U_{kk}$$

$$U_{ki} = a_{ki}$$

for $i = k+1$ to n

$$a_{ij} = a_{ij} - d_{ik} U_{kj}$$

return L and U

Ex:
$$\begin{bmatrix} 2 & 3 & 1 & 5 \\ 6 & 13 & 5 & 19 \\ 2 & 19 & 10 & 23 \\ 4 & 10 & 11 & 31 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 3 & 1 & 0 & 0 \\ 1 & 4 & 1 & 0 \\ 2 & 7 & 7 & 1 \end{bmatrix} \begin{bmatrix} 2 & 3 & 15 \\ 0 & 4 & 2 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

$\quad \quad \quad L \quad \quad \quad U$

Computing LUP decomposition

LUP-decomposition (A)

$n = A$ rows

let $\pi[1 \dots n]$ be a new array

for $i = 1$ to n

$\pi[i] = i$

for $k = 1$ to n

$p = 0$

for $i = k$ to n

if $|a_{ik}| > p$

$p = |a_{ik}|$

$k' = i$

if $p = 0$

error "singular matrix"

exchange $\pi[k]$ with $\pi[k']$

for $i = k+1$ to n

$a_{ik} = a_{ik} / a_{kk}$

for $j = k+1$ to n

$a_{ij} = a_{ij} - a_{ik} a_{kj}$

SearchKdTree(v, R) $\rightarrow$ Root of Kd-tree / Root of subtree of Kd-tree
 $\rightarrow$ range R .

1) if (v itself is a leaf)

return if (v lies in range R) $\rightarrow$ return the point stored at v

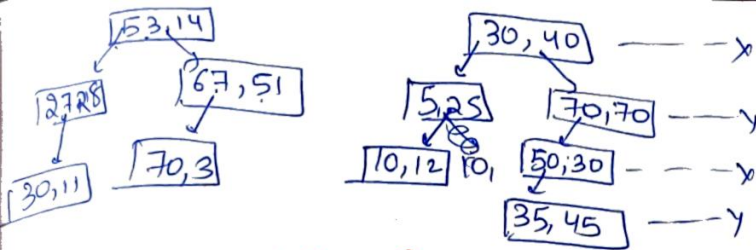
2) else if (region (LC) fully contained in R)
 reportSubtree(LC)

else if (region(LC) intersects with R)

$VRCL = R \cap LC$
 reportSubtree($VRCL$)

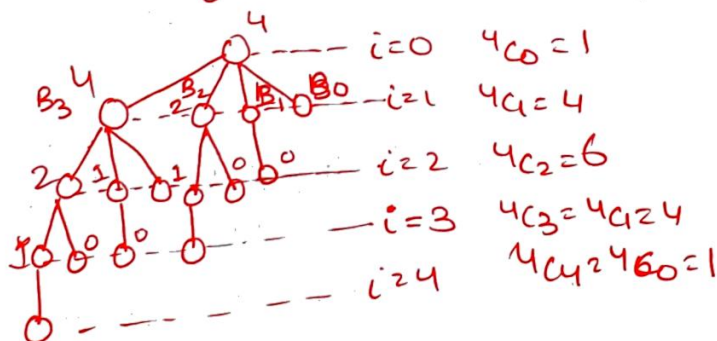
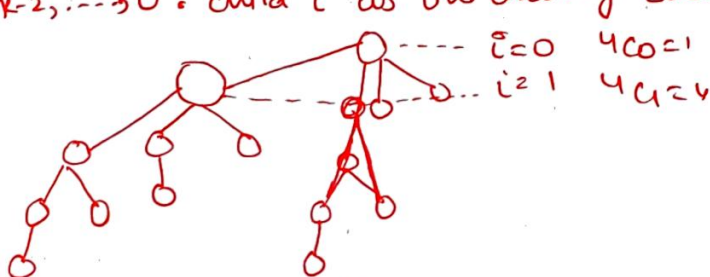
if (region $VRCL$ contained in R) reportSubtree($VRCL$)

else if (region $VRCL$ intersects with R) ~~reportSubtree($VRCL$)~~ R



Properties of Binomial Heap:

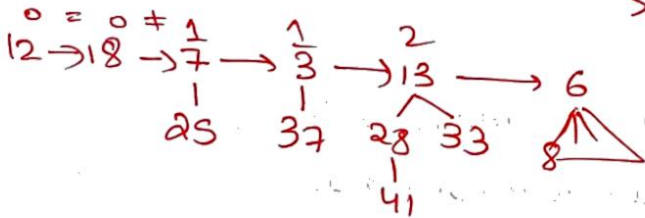
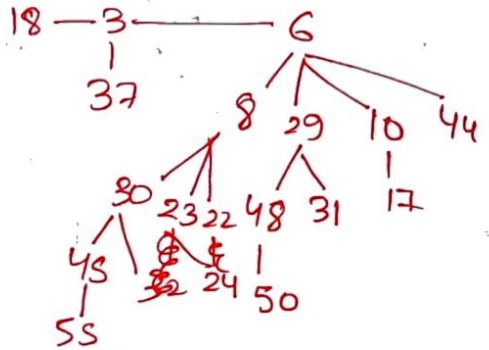
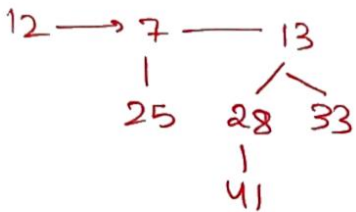
- 1) There 2^k nodes
- 2) The height of the tree is k
- 3) There are exactly kC_i nodes at depth i for $i=0, 1, \dots, k$
- 4) The root has degree k , which is greater than that of any other node.
- 5) If i is the children of the root are numbered from left to right by $k-1, k-2, \dots, 0$. child i is the root of subtree B_i



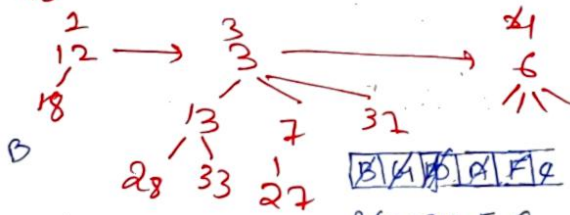
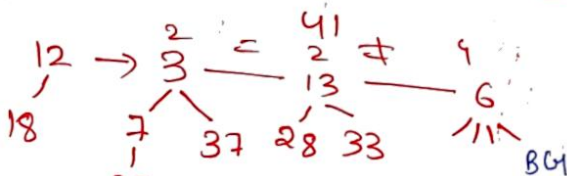
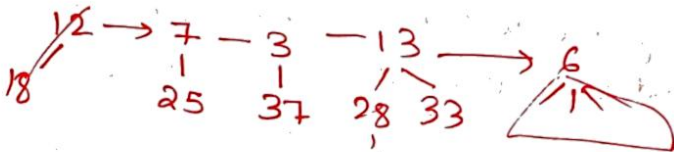
$$\frac{2^{k \times 3}}{2}$$

A Binomial heap is a set of binomial trees that satisfies the following Binomial heap properties.

- Each ~~Binomial~~ Binomial tree of H obeys the min/max of heap properties
- For a non-negative integer k , there is at most 1 binomial tree B_k in H .



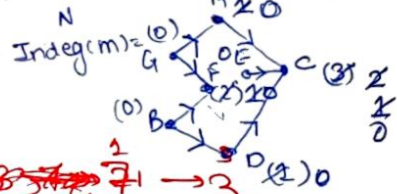
Stack: J
 Pop and print n neighbours
 Print J
 Stack: DK
 Print K Stack: D L F G
 Print G Stack: D E C
 Print C Stack: D E F
 Print F Stack: D G
 Print G Stack: D
 Print D Stack: Empty



K G C F E D

S w/o cycle (graph)

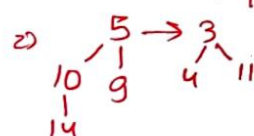
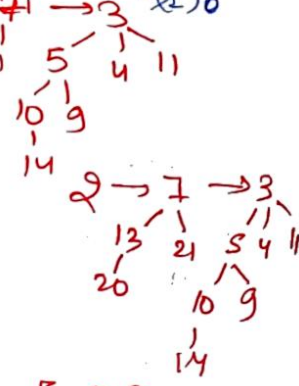
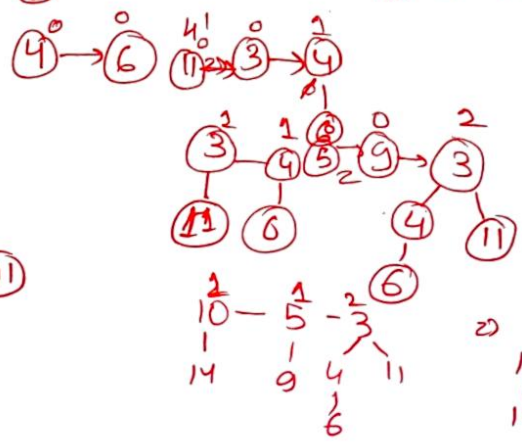
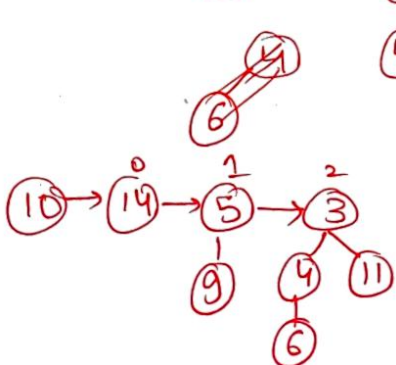
$U < V \mid U \text{ to } V \text{ as}$
 $U \neq U \quad V \neq U$
 $U < V \quad V < U$
 $U < V \quad V < W \quad U < W$

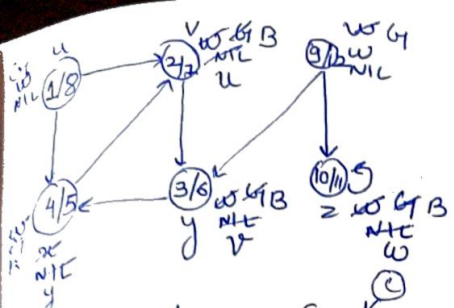


4, 6, 3, 11, 9, 5, 14, 10, 21, 7, 13, 20, 2

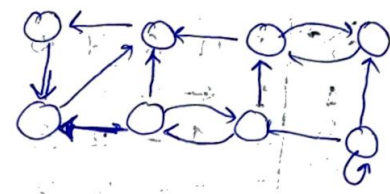
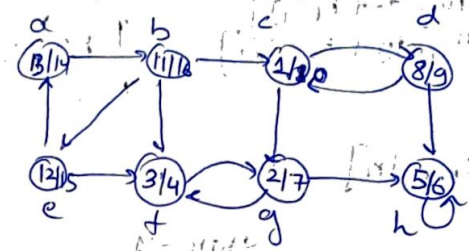
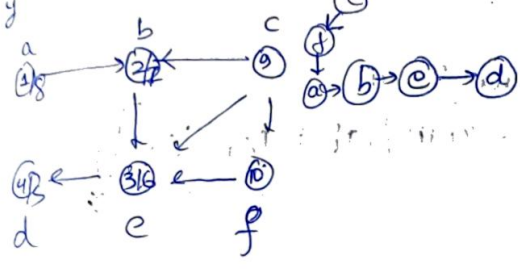
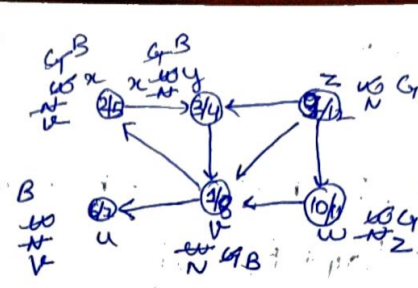
④

④



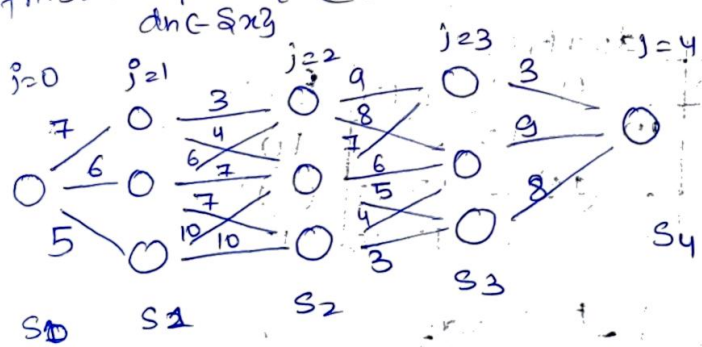


time
0
1



It uses mathematical
An optimal sequence of decisions has the property that whatever the
initial state and decision are, the remaining decisions must
constitute an optimal decision sequence with regard to the
state resulting from the 1st decision.

$$f_n(s) = \text{opt}_{d_n \in S_n} \{ r(d_n) + f_{n-1}(T(s, d_n)) \}$$



Define 3 variables

State variable: represent node of the graph

Decision variable: the arc length

Stage variable: $j = 0, 1, 2, 3, 4, 5$

Developing function Equation

Backward approach

At Stage 4

$$F_4(s_4) = \min [3 + F_3(1), 9 + F_3(2), 8 + F_3(3)]$$

$$= \min [d_4 + F_3(s_3)]$$

$$F_1(s_1) = \min [d_1] = d_1$$

$$F_2(s_2) = \min [d_2 + F_1(s_1)]$$

$$F_3(s_3) = \min [d_3 + F_2(s_2)]$$

in general Term

$$F_i(s_i) = \min [d_i + F_{i-1}(s_{i-1})]$$

$$S_{n-1} = T[S_n, d_n]$$

State S₀

$s_1 \backslash d_1$	7	6	5
1	1	-	-
2	-	1	-
3	-	-	1

State S₁

$s_2 \backslash d_2$	3	4	6	7	10
1	1	2	-	-	-
2	-	1	-	2	3
3	-	-	-	2	3

State S₂

$s_3 \backslash d_3$	3	4	5	6	7	8	9
1	-	-	-	-	2	-	1
2	-	3	-	2	-	1	-
3	3	-	2	-	-	-	-

State S₃

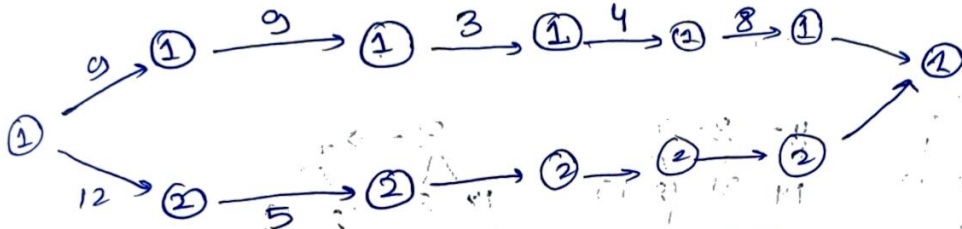
$s_4 \backslash d_4$	3	9	8
1	1	2	3

j	s_1, d_1	$F_1(s_1)$
1	1, 7	7
	2, 6	6
	3, 5	5

j	$s_2 \backslash d_2$	$F_1(s_1)$	$d_2 + F_1(s_1)$	$F_2(s_2)$
		3 4 6 7 10	3 4 6 7 10	
2	1	7 - 6 - -	10 - 12 - -	10
	2	- 7 - 6 5	- 11 - 13 15	11
	3	- - - 6 5	- - - 13 15	13

j	$s_3 \backslash d_3$	$F_2(s_2)$	$F_2(s_2) + d_3$	$F_3(s_3)$
		3 4 5 6 7 8 9	3 4 5 6 7 8 9	
	1	- - - - 11 - 10	- - - - 18 - 19	18
3	2	- 13 - 11 - 10 -	- 17 - 17 - 18 -	17
	3	13 - 11 - - -	16 - 16 - - -	16

9	d4 s4	F2(s3)			F3(s3)+d4			F4(s4)
		3	9	8	3	9	8	
4	1	(18)	17	16	(21)	26	24	(21)



$$z = y_1^2 + y_2^2 + y_3^2$$

$$y_1 + y_2 + y_3 = 10$$

$$y_1, y_2, y_3 \geq 0$$

$$s_3 = y_1 + y_2 + y_3 = 10$$

$$s_2 = s_3 - y_3 = y_1 + y_2$$

$$s_1 = y_1 = s_2 - y_2$$

function for all state.

$$z_{\min} = F_3(s_2) = \min[y_3^2 + F_2(s_2)]$$

$$F_2(s_2) = \min[y_2^2 + F_1(s_1)]$$

$$F_1(s_1) = \min[y_1^2] = (s_2 - y_2)^2$$

$$\frac{d}{dy_2} [y_2^2 + s_2 - y_2] = 2y_2 - 1 = 0$$

$$2y_2 = 1 \Rightarrow y_2 = \frac{1}{2}$$

$$F_2(s_2) = \frac{1}{4} \frac{s_2^2}{4} + \left(s_2 - \frac{s_2}{2}\right)^2 = \frac{s_2^2}{4} + \frac{s_2^2}{4} = \frac{s_2^2}{2}$$

$$F_3(s_3) = y_3^2 + \frac{s_2^2}{2} \Rightarrow y_3^2 + \frac{1}{2}(s_3 - y_3)^2$$

$$2y_3 + \frac{1}{2}(s_3 - y_3)(-1) = 0$$

$$2y_3 - s_3 + y_3 = 0$$

$$s_2 = s_3 - y_3 \Rightarrow \frac{2 \times 10}{3}$$

$$\frac{3y_3^2 - s_3}{3} \Rightarrow \boxed{y_3 = \frac{10}{3}}$$

$$\boxed{y_2 = \frac{10}{3}}$$

$$\boxed{y_1 = \frac{10}{3}}$$

$$18 \rightarrow 24$$

$$19 \rightarrow 7 \rightarrow 8$$

$$24$$

$$7 \rightarrow 8$$

$$19 \rightarrow 24$$

$$3 \rightarrow 7$$

$$18 \rightarrow 19$$

$$24$$

$$14 \rightarrow 8 \rightarrow 3 \rightarrow 7$$

$$31$$

$$18$$

$$19$$

$$24$$

$$8 \rightarrow 3 \rightarrow 7$$

$$14$$

$$31$$

$$18$$

$$19$$

$$24$$

$$3 \rightarrow 7$$

$$8$$

$$31$$

$$18$$

$$19$$

$$24$$

$$6 \rightarrow 2 \rightarrow 3$$

$$6$$

$$11$$

$$7$$

$$8$$

$$31$$

$$18$$

$$19$$

$$14$$

$$24$$

$$9 \rightarrow 2 \rightarrow 3 \rightarrow 0$$

$$4$$

$$11$$

$$7$$

$$8$$

$$31$$

$$18$$

$$19$$

$$14$$

$$24$$

$$3$$

$$\begin{array}{r} -18- \\ -19- \\ -14- \\ \hline 6 \\ \hline 57 \end{array}$$