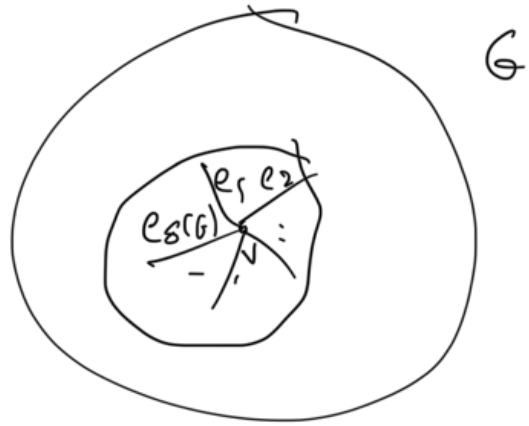
Proof: For any graph G, ✓

$$\underset{\text{Connectivity}}{\kappa(G)} \le \underset{\text{edge-Connectivity}}{\lambda(G)} \underset{\checkmark}{\le} \underset{\text{min-degree.}}{\delta(G)}.$$

Proof:

$\lambda(G) \le \delta(G)$:



Removing $F=\{e_1, \dots, e_{\delta(G)}\}$ from G separates v from the rest of G.

Fix v, with $d(v)=\delta(G)$

$\therefore \lambda(G) \le |F| = \delta(G).$

$\kappa(G) \le \lambda(G)$:

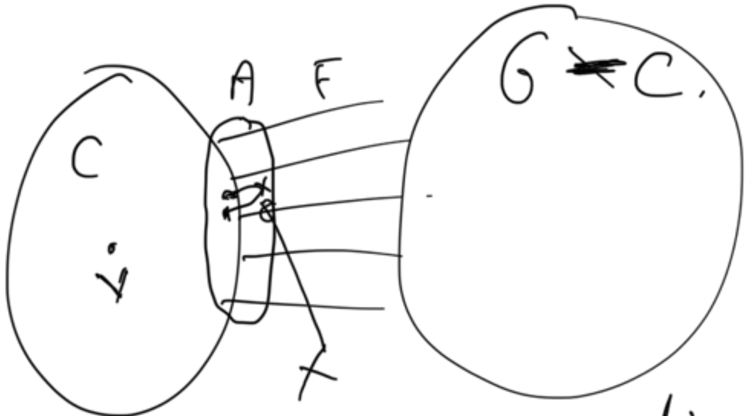
Let F be a set of $\lambda(G)$ edges such that $G-F$ is disconnected (Such an F exists by defn of $\lambda(G)$).

Further such an F is a minimal separating set of edges in G.

Goal: $\boxed{\kappa(G) \le \overset{\lambda(G)}{\overset{\shortparallel}{|F|}}}$

Case 1: G has a vertex v not incident with an edge in F.

Let $C :=$ Component of $G - F$ containing v.



$A :=$ the set of vertices in C adjacent to the edges in F.

Then A separates v from $G - C$

$\therefore$ by defn of $\kappa(G)$,

$\kappa(G) \leq |A|$

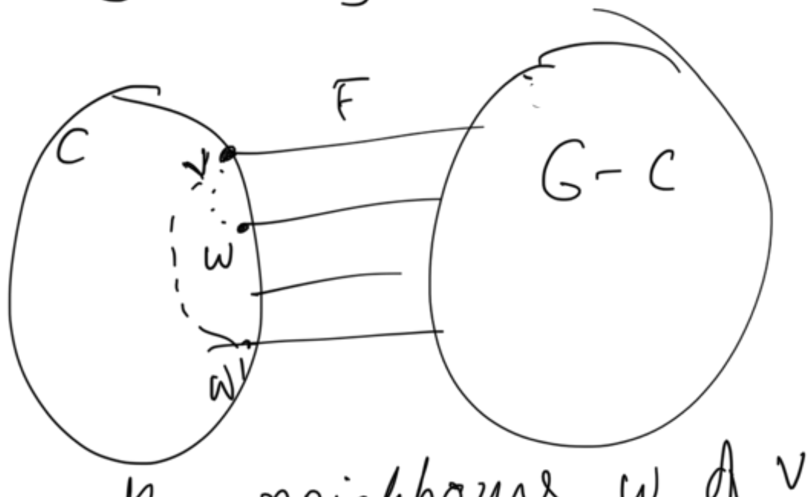
Furthermore no edge in F has both ends in C.
(why? Because of minimality of F).

$\kappa(G) \leq |A| \leq |F| = \lambda(G)$

$\therefore \kappa(G) \leq \lambda(G)$.

Case 2: Every vertex in G is incident to an edge in F.

Now let v be any vertex of G & C the component of G − F containing v.

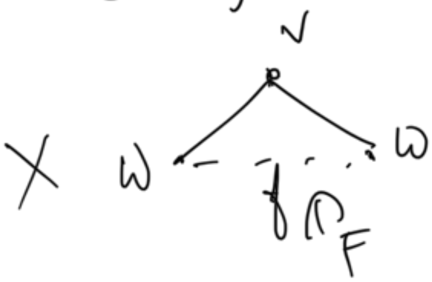


Then the neighbours w of v with $vw \notin F$ lie in [Since F separates

C. and are incident [from $G-C$] to distinct edges in F.

(by minimality of F).

(only:



$\therefore\ d_G(v) \leq |F|$

As $N_G(v)$ separates v from other vertices in G, ||

$\overset{\text{neighbours of } v \text{ in } G}{}$

$$\kappa(G) \leq |N_G(v)| = d_G(v) \leq |F| = \lambda(G),$$

as we need.

unless $\{v\} \cup N_G(v) = V$

Since v was an arbitrary vertex, the only remaining case now is when for every $v \in V$, ||

$N_G(v) = V$.

$\{v\} \cup N_G(v) = V$.

This means G is complete in the remaining case.

In this case,

$$\kappa(G) = \lambda(G) = |G| - 1 .$$

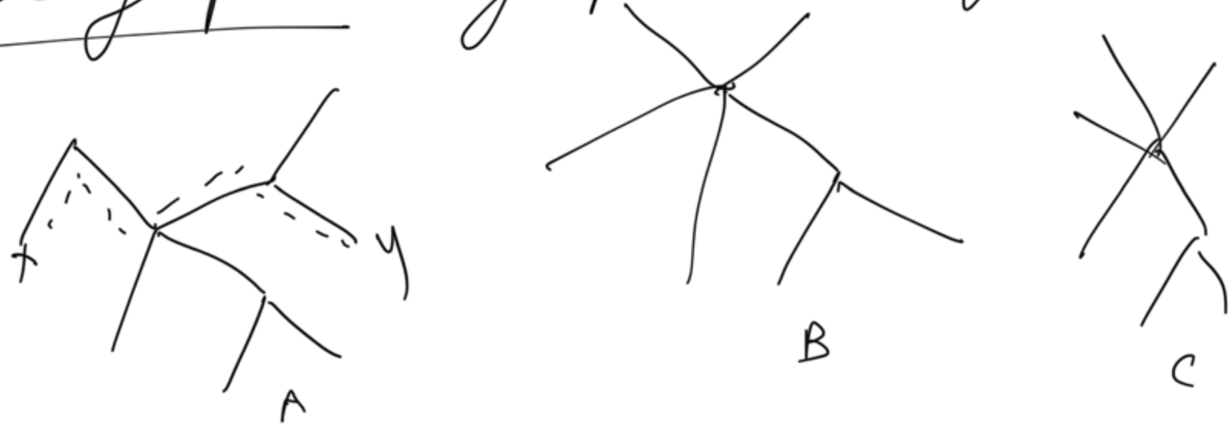
Q.E.D.

K_5

Trees and forests

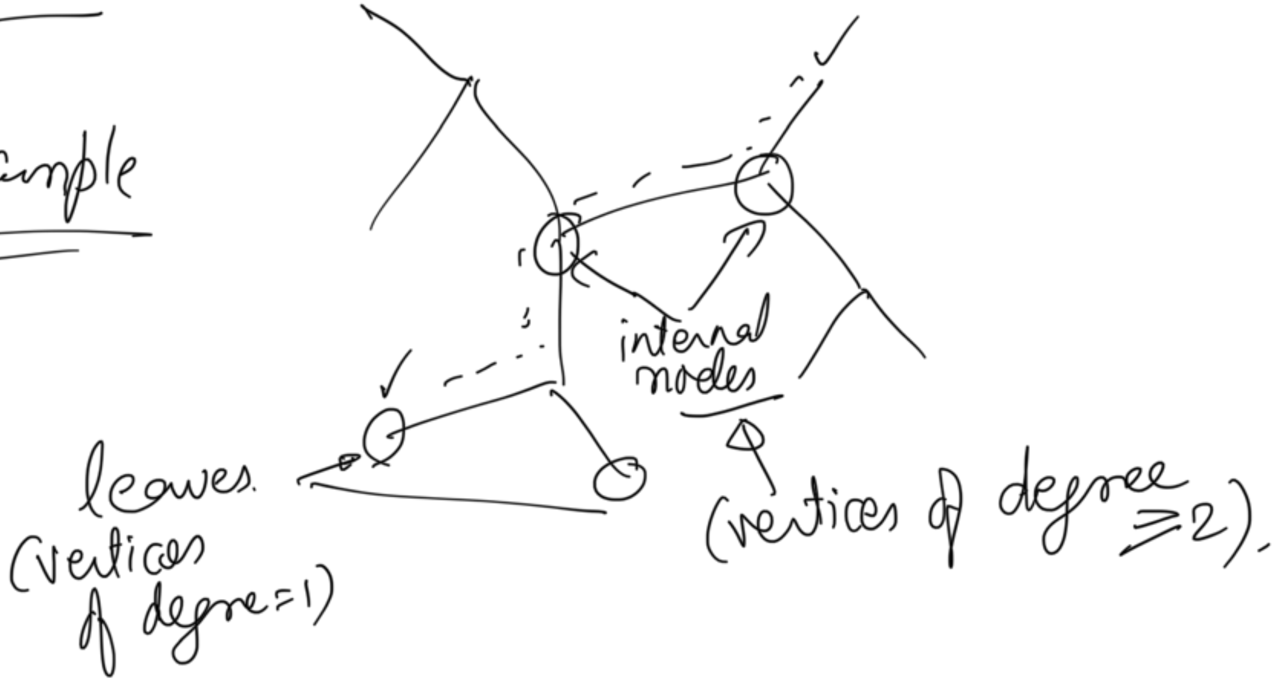
Acyclic graph: A graph without cycles.

A Forest



Tree: A connected acyclic graph.

Example



Prop: 1 The following assertions are equivalent for a graph T.

1) T is a tree.
2) Any two vertices of T are linked by a unique path in T.
3) T is minimally connected: T is connected but $T - \{e\}$ is disconnected for any edge e of T.

4) T is maximally cyclic:
T is acyclic but no
T + (x,y) for any vertices x & y of T.

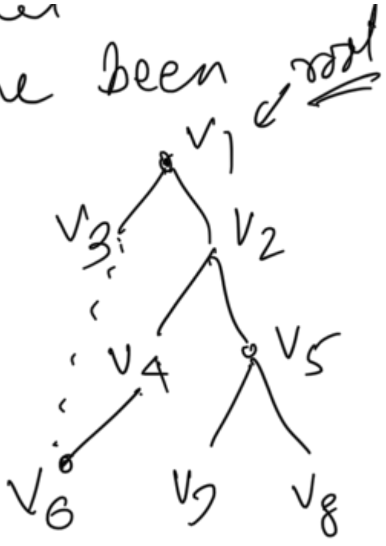
Proof: Exercise.

If T is a tree and x and y are its vertices, we denote by xTy the unique path (cf. Prop 1, (2)) in T connecting x to y.

Corollary 1 (of Prop 1):
The vertices of a tree can be enumerated as $v_1, v_2, \dots, v_n$, $n = |T|$, so that every v_i, $i \geq 2$, has a unique neighbours in $v_1, \dots, v_{i-1}$.

Proof: Let v_1 be any vertex of T.

By induction, assume that v_1 to v_{i-1} have been constructed.



Example

Let v_i be any vertex in $V \setminus \{v_1, \dots, v_{i-1}\}$ which is connected to some vertex in $\{v_1, \dots, v_{i-1}\}$.

[Then v_i cannot have two neibours in $\{v_1, \dots, v_{i-1}\}$, — otherwise there will be a cycle.] Q.E.D.

Corollay 2: A connected graph with n vertices is a tree iff it has $n-1$ edges.

Pf: $\rightarrow$: Let $v_1, \dots, v_n$, $n = |T|$, be the enumeration as per Corolley 1.

$\Rightarrow$ by Corolley 1,

$\#$ edges in $T = n-1$.

$\Longleftarrow$: Let G be a connected graph on n vertices with $n-1$ edges.

Let G' be its spanning tree. [always exists].

$(V(G') = V(G))$

Can be constructed greedily:

Let v_1 be any vertex of G.

By induction assume that $v_1, \ldots, v_{i-1}$ have been constructed.

Since G is connected, there exists some edge e which connects $\{v_1, \ldots, v_{i-1}\}$ to $V \setminus \{v_1, \ldots, v_{i-1}\}$.

Include e in the tree & let v_i = other end of e.

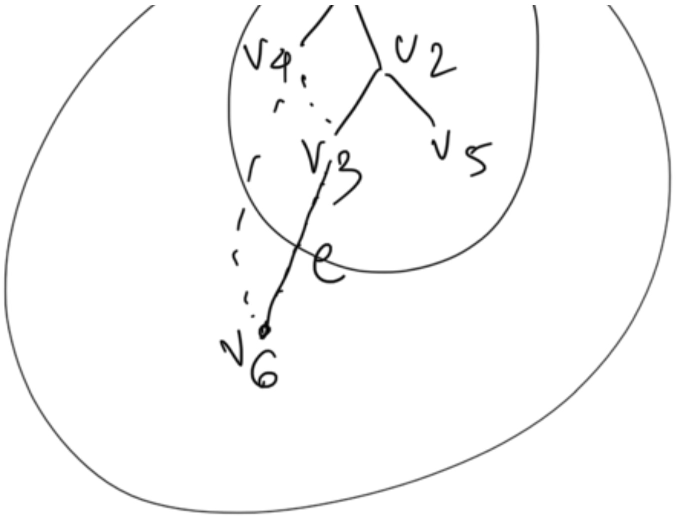
$G' \subseteq G$

& $V(G') = V(G)$

& G' is a tree.

Then G' has $n-1$ edges by $\Longrightarrow$.

Since G also has $n-1$ edges, so $G = G'$.



Q.E.D.

<u>Corollary 3</u>: If T is a tree and G is any graph with $\delta(G) \geq |T|-1$, then T can be embedded in G as its subgraph: $T \underset{\sim}{\subseteq} G$.

embedding as a subgraph.

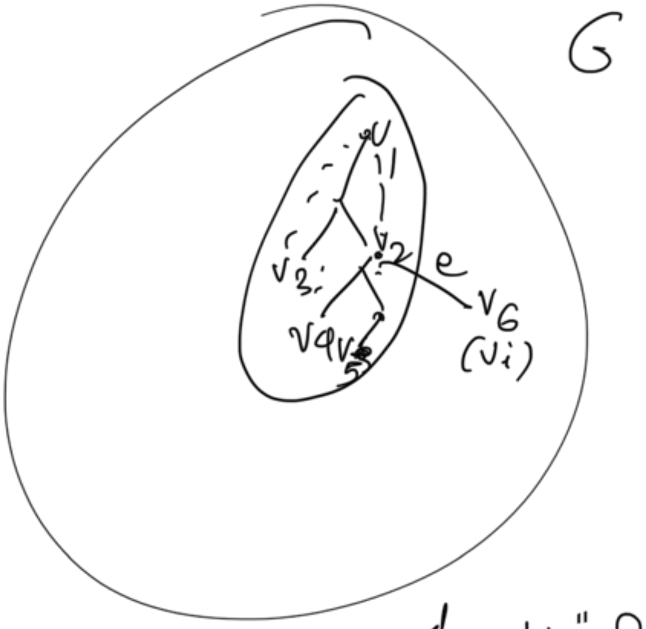
<u>Proof</u>: Let $v_1 .. v_n$ be the enumeration of the vertices of T as per Corollary 1.

By induction assume that

By induction.

$T[v_1, \dots, v_{i-1}] \underset{\sim}{\subseteq} G$.

Let e be the edge $\overset{\text{in } T}{\text{connecting}}$ v_i to some vertex $\underset{v_j}{}$ in $\{v_1, \dots, v_{i-1}\}$.



How do we embed e in G.

neighbours of v_j & $\{v_1, \dots, v_{i-1}\}$

so far $\le i-2$

$\le n-2$

but degree of v in G, $n = |T|$.

by assumption, is $\ge |T| - 1$

$= n-1$

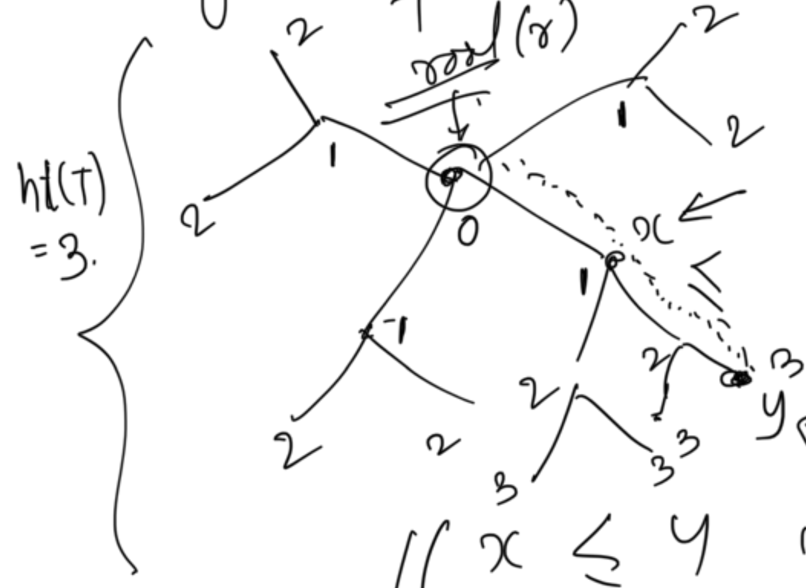
∴ There is some edge e' incident

to v_i in G whose other endpoint is not in $\{v_1, \dots, v_{i-1}\}$.

Let v_i be the other endpoint of e.

Q.E.D

A rooted tree: A tree with a fixed vertex (r), called a root.



T : a rooted tree with root r.

Then given two vertices $x, y \in T$, We say that

$$x \le y \text{ iff } x \in \underbrace{rTy}_{\text{Unique path in } T \text{ from } r \text{ to } y}$$

$$\le \; = \; \le_T$$

Partial order called the tree order for T.

The tree order $\le_T$ defines the height of any vertex in T.

1) Reflexive ($x \le x \ \forall x$)

2) Transitive ($x \le y$ & $y \le z$ then $x \le z$).

3) Antisymmetric: if $x < y$ ($\le$ but $\ne$) then $y \not< x$.

$\text{height}(y) = l(r T y)$.

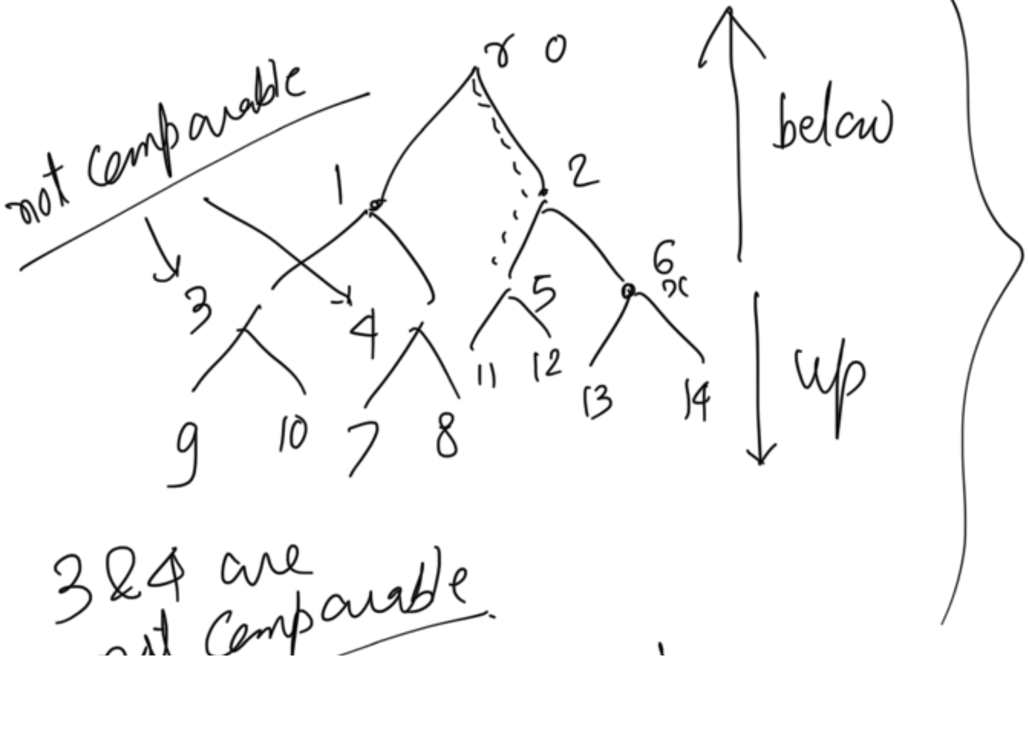
$\uparrow$ T

$\text{ht}(T) = \max\{\text{ht}(y) \mid y \in T\}$

$\uparrow$ height

Exercise: Prove that $\le_T$ is a partial order.

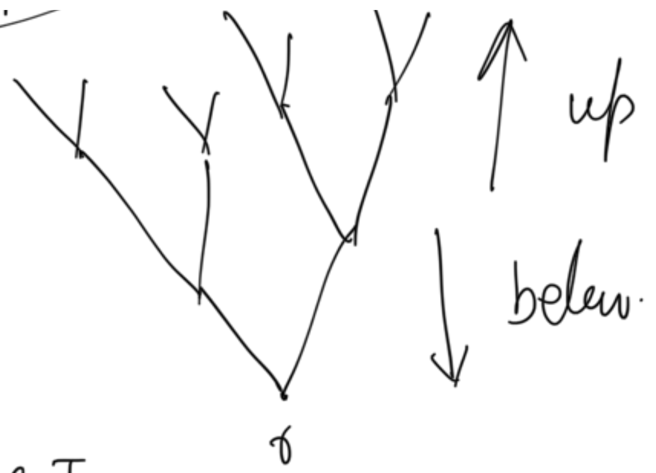
$x < y$: iff $x \le y$ & $x \ne y$. : x is below y.



$[5] = \{0,2,5\}$

$\lceil 2 \rceil = \{2,5,6,11,12,13,14\}$

3&4 are not comparable



Given $y \in T$,

$\downarrow \lceil y \rceil = \{ x \leq y \}$

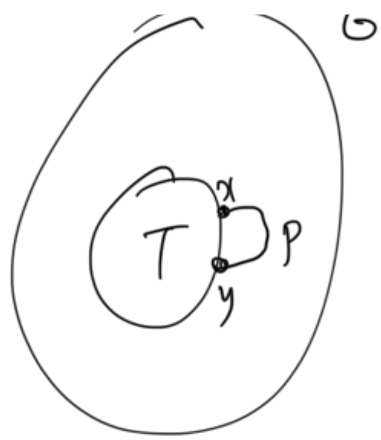
$\downarrow$
down-closure of y

Given $x \in T$,

$\uparrow \lfloor x \rfloor = \{ y \mid y \geqslant x \}$

up-closure $\overset{\uparrow}{T}$

Defn: A rooted tree $T \subseteq G$ is called normal (in G) if the ends of every T-path in G are comparable in the tree order $\leq_T$ of T.



Comparable: either $x \leq y$ or $y \leq x$.

If T spans G ($V(T) = V(G)$)

normality

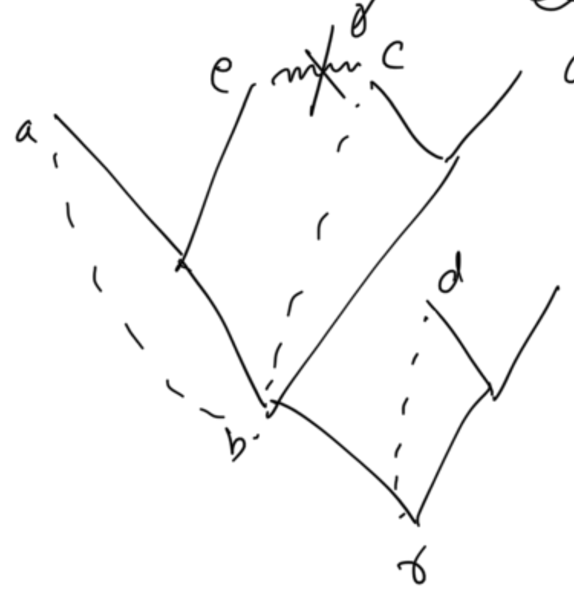
This condition is equivalent to saying that any two vertices of T must be comparable if they are adjacent in G.

Example:

e & c are not comparable in $\leq_T$.

c > b

this kind of an edge is not allowed in G for T to be normal.



\ : T

: : G \ T

a & b are adjacent in G ($a \geq b$).

T: is normal.

$d > r$.

—x—

Does every connected graph have a normal tree?

a normal rooted tree :
can be found by a

Home Work assigned { Yes — Depth-First-Search tree } is normal.

— x —

Properties of normal trees

Lemma 1:

Let T be a normal spanning tree in G.:

(1) Any two vertices x & $y \in T$ are separated in G by the set $\lceil x \rceil \cap \lceil y \rceil$

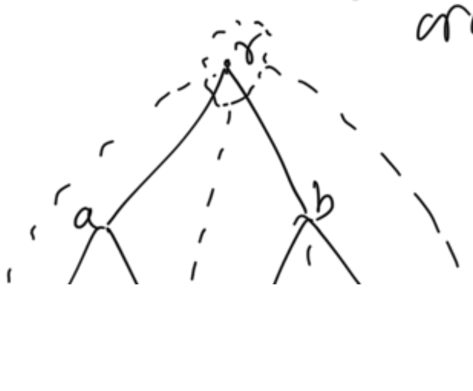
Down-closures.

(2) If $S \subseteq V(T) = V(G)$ & S is down-closed then the components of G-S are spanned by the sets $\lfloor x \rfloor$ with x minimal (in the tree order $\leq_T$) in T-S.

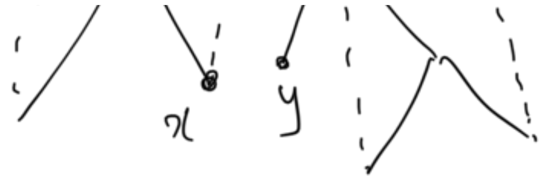
means $S = \{\lceil x \rceil \mid x \in S\}$

up-closure of x normal tree.

Example: for (1)



$\lceil x \rceil = \{x, a, r\}$



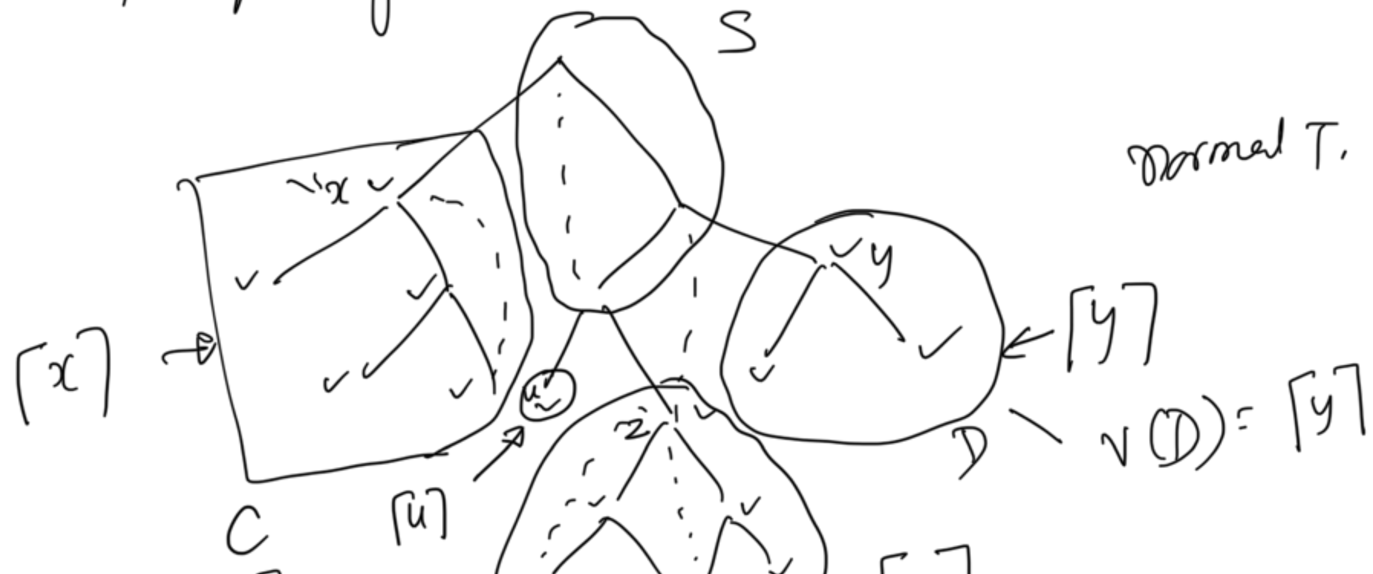
$\lceil x \rceil$

$\lceil y \rceil = \{y, b, r\}$

$\lceil x \rceil \cap \lceil y \rceil = \{r\}$

r separates x & y.

Example for (2):



$v(C) = \lceil x \rceil$

x is minimal not just in v(C) but also in S-T.

Proof of Lemma [Basic ideas].

(1) Let P be any x-y path in G.

claim: P meets $\lceil x \rceil \cap \lceil y \rceil$. ← Exercise

$\implies$ (1).

(2) Let x be any minimal element

in $T-S$. Then:

Claim [Exercise]:

(a) $V(\underbrace{C}_{\substack{\text{Component} \\ \text{of } G-S \\ \text{containing } x}}) = \lfloor x \rfloor$

(b) x is minimal not just in $V(C)$ but also in $S-T$. [use fact that S is down-closed]

(c) Conversely, if x is minimal in $S-T$ then it is also minimal in the component C by $G-S$ to which it belongs ($\underset{\text{by (a)}}{\Longrightarrow} V(C) = \lfloor x \rfloor$)

Claim $\underset{\substack{\text{trivial} \\ \text{check}}}{\Longrightarrow}$ (2).

Q.E.D.

<u>Prop</u>: Every connected graph G contains a normal tree T.

Pf: Fix any vertex of G as a root r. Starting from r perform a depth first search on G.

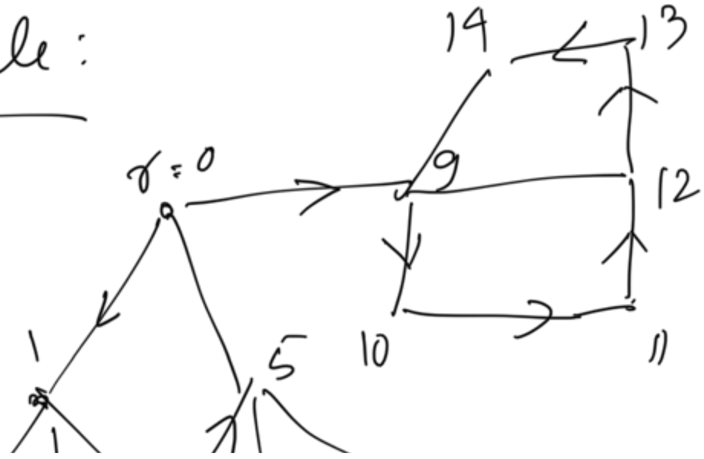
Let $T :=$ Depth-first-search tree of G.:

$e \in G$ belongs to T iff it was traversed while going down in the depth first search.

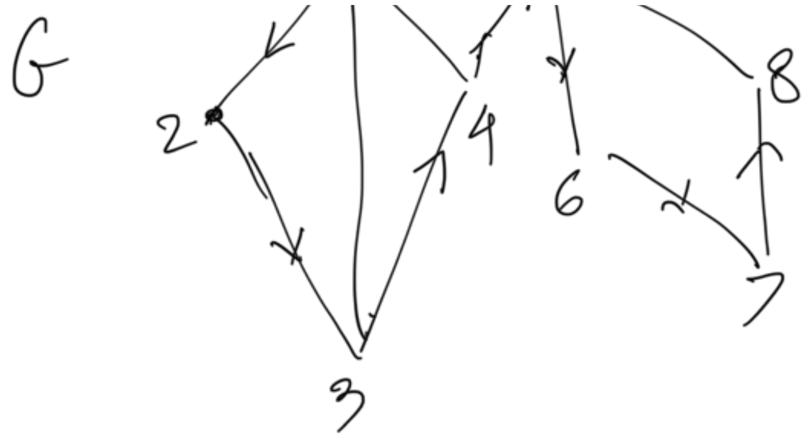
Homework.

Then T is normal. [Exercise] Q.E.D

<u>Example</u>:



Depth-first search D: tree T: Consists of the edges



marked
$\longrightarrow$.

$$\left.\begin{array}{ll}(1,3): & 1 \leq 3\\ (0,5): & 0 \leq 5\\ (1,4): & 1 \leq 4\\ (9,12): & 9 \leq 12\end{array}\right\}\ \underline{\text{Comparable}}$$

$(5,8): \ 5 \leq 8$

$\therefore$ T is normal.

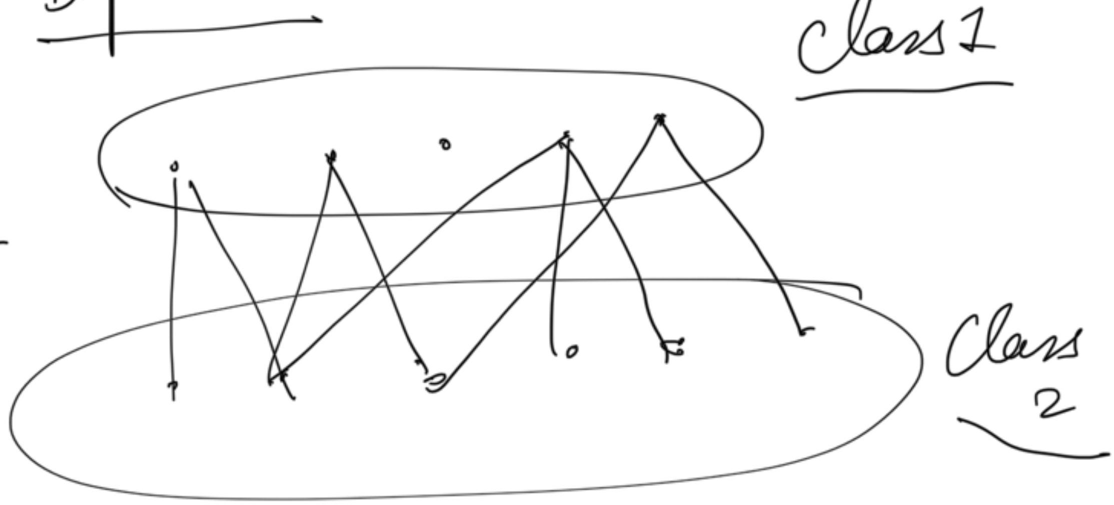
Bipartite graphs and matchings

<u>Defn</u>: A graph $G=(V,E)$ is called <u>r-partite</u> $(r \geq 2)$ if V admits partition into r classes s.t. every edge $e \in E$ has its ends in different classes [This also means

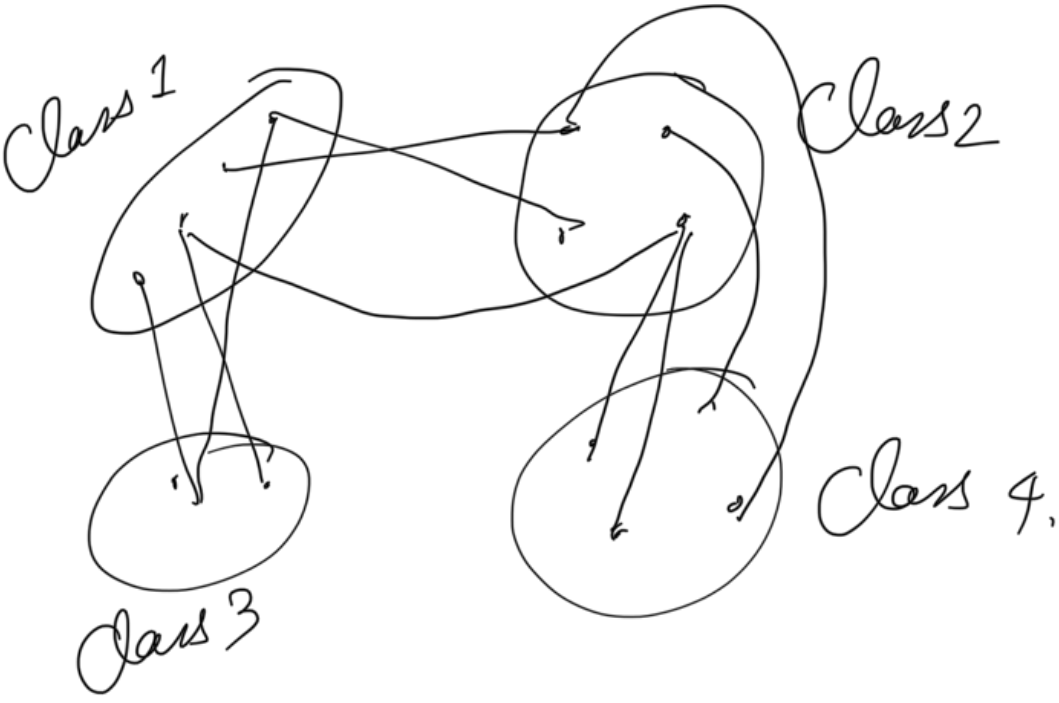
that vertices in the same class cannot be adjacent.

A 2-partite graph is also called bipartite.

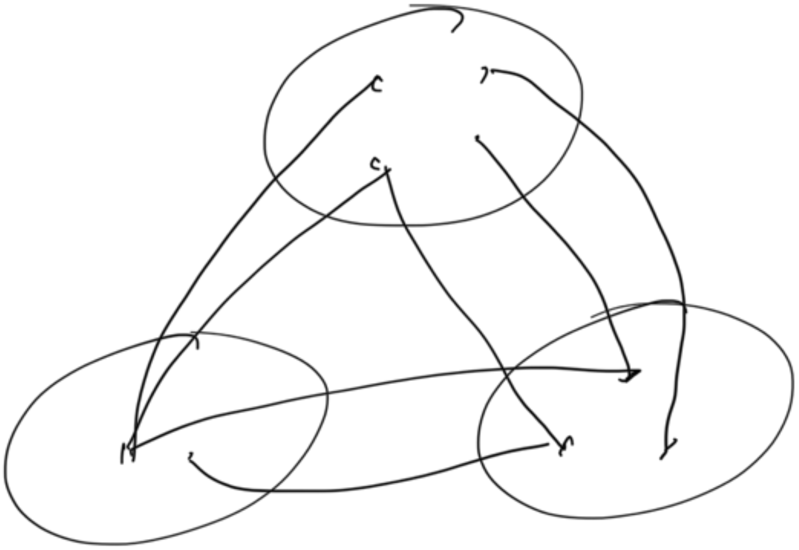
Bipartite:



four-partite:



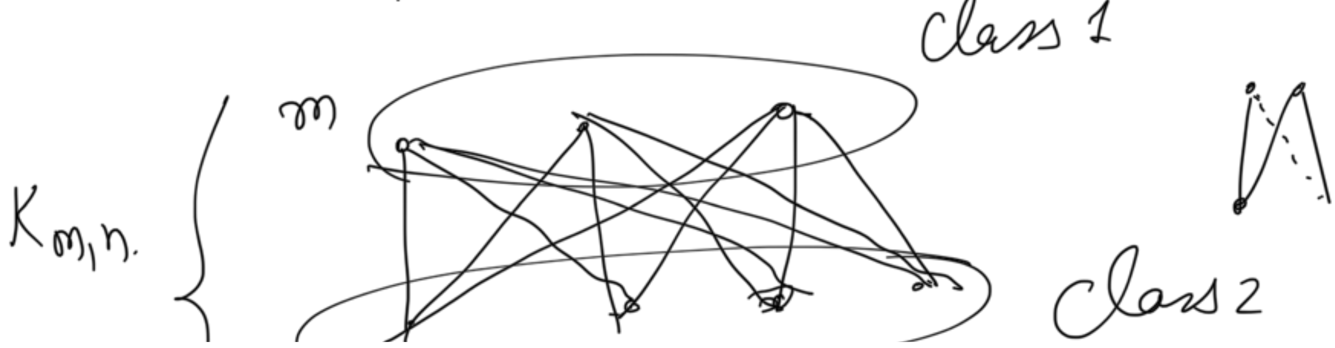
tr- partite .



An r-partite complete graph: is a An r-partite graph in which any two vertices from different partition classes are adjacent.

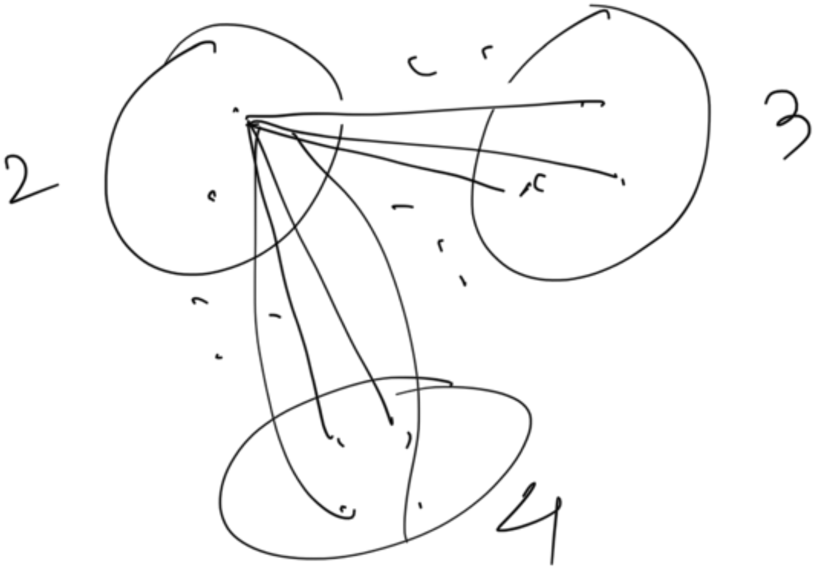
Example:

A bipartite complete graph:



A complete r-partite graph in which the r-components have ~~xx~~ $n_1, \dots, n_r$ vertices is denoted

$$K_{n_1, \dots, n_r}$$



$K_{2,3,4}$.

Properties of bipartite graphs.

Prop: A graph G is bipartite iff it contains no odd cycle. → cycle with odd

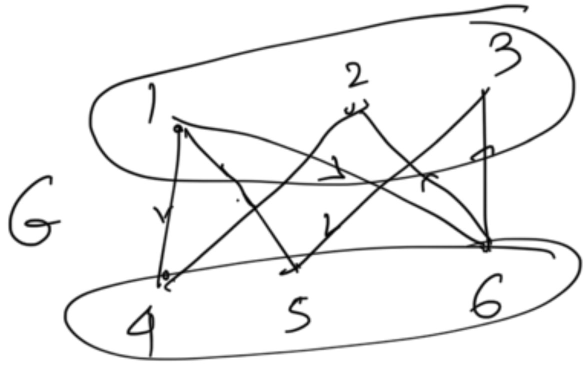
— ⟶ cycle w... length.

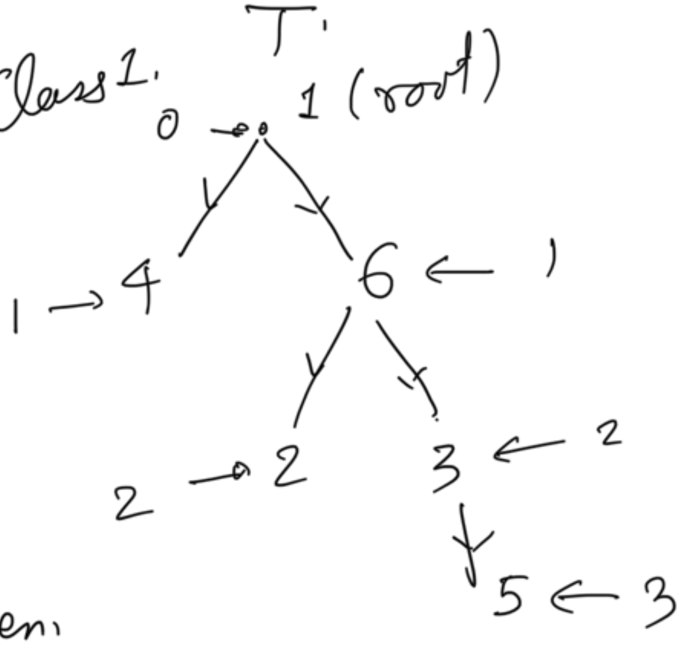
Proof: ⟶ : Clearly a bipartite graph G cannot contain an odd cycle.

⟵ W.l.g Suppose that G is connected.

Let T be a rooted spanning tree of G. (r = root)

Every vertex of T can be given a height





Class 0: Put in class 0 all vertices of T whose heights have parity 0. even

Class 1: Put in class 1

Class 0:

all vertices in ,
whose heights have
parity 1. ~~x~~ odd.

$\{1, 2, 3\}$

Class 1:
$\{4, 6, 5\}$

If there are edges in G between two vertices in the same class:, then we get an odd cycle.

$\therefore$ all edges in G must be between vertices from different classes.

This means G is bipartite. $|V| = n$ $|E| = m$ $= (V, E)$ Q.E.D.

Question: Given a graph G, how fast can we decide if G is bipartite?

Prob: Given a graph G, one can

Prop: Given , decide in $O(n+m)$ time if G is bipartite.

Proof: Fix any vertex r (root) in G & perform depth-first-search.. ($O(n+m)$ time)

(w.l.g assume that G is connected)

T: Depth first search tree in G

Then T is normal. [Home work Exercise]

$O(m+n)$ Now label all vertices of T with even ht as 0. & all vertices of odd ht as 1.

labelling can be done during DFS (depth first search).

Class 0: All vertices with label 0

Class 1: — — — — — — 1

Check For every $e \in E$, $O(m)$

Check: For every $e \in E$, $\Big\}\ O(m)$

Check if its endpoints are
in different classes.

If Yes, then G is bipartite.

If NO, then G is not bipartite.

$\therefore$ The whole algorithm takes
$O(m+n)$ time. Q.E.D.