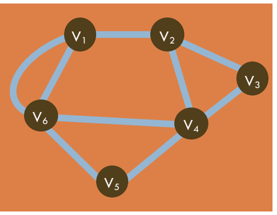


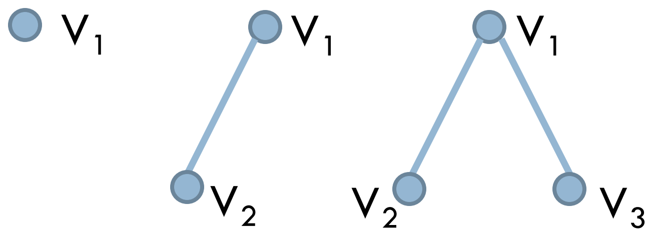
Module 3

Trees

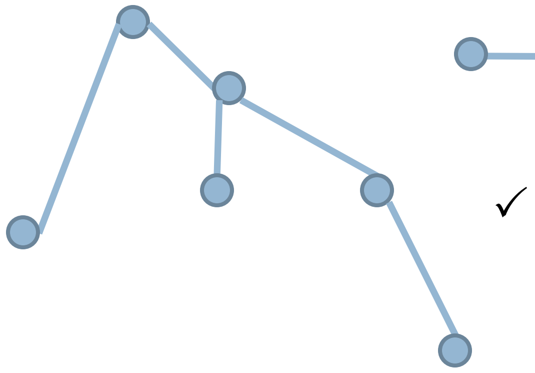
Contents

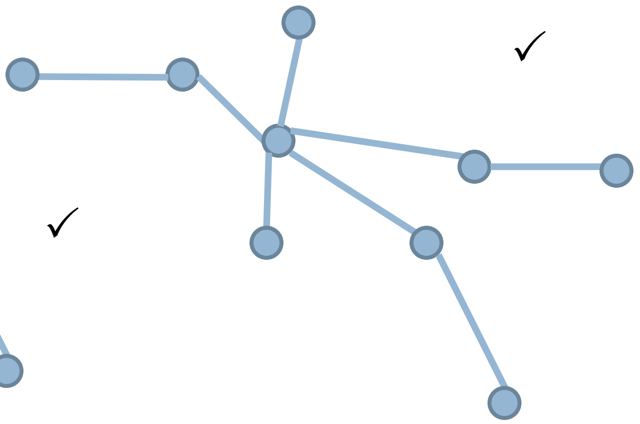
What is a Tree ?

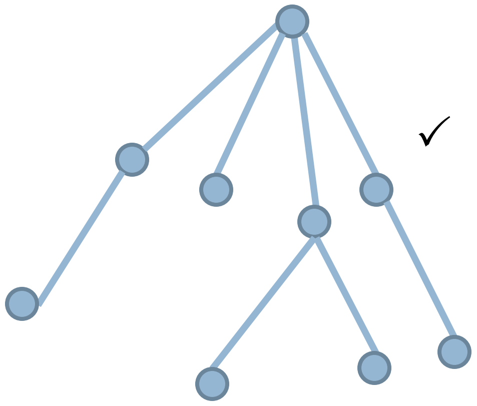
- A tree is a connected graph, without any circuits
- It implies that a tree has to be a simple graph, as self loops & parallel edges form circuits
- A single vertex can also be considered as a tree
- Eg :
  - A tree with 1 vertex
  - A tree with 2 vertices
  - A tree with 3 vertices

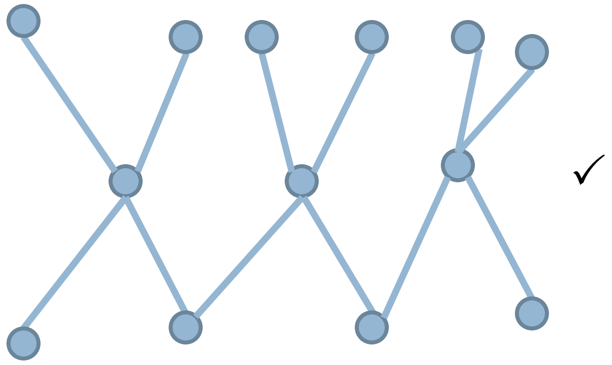


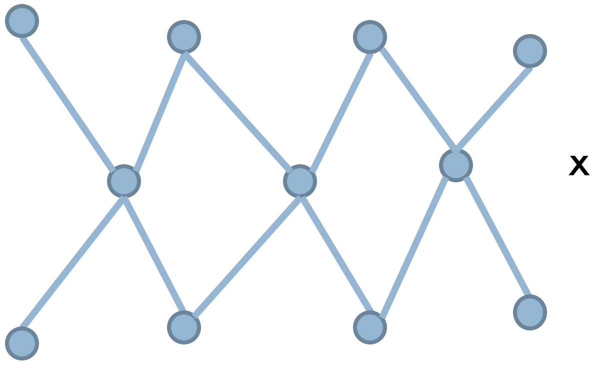
Examples











- Trees can be used to represent & manipulate real life instances like
  - The genealogy of a family
  - A river with its tributaries & sub tributaries
  - Sorting of mail according to zip code
- Minimally connected
  - A graph G is said to be minimally connected if removal of any one edge from G, leaves it disconnected

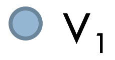
Properties of trees

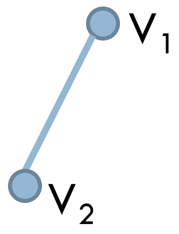
1) A tree is a connected graph & circuit-less
2) There is exactly one path between every pair of vertices of a tree
3) A tree with n vertices will have (n-1) edges
4) A tree is a minimally connected graph
5) A tree will have at least 2 pendant vertices
6) A tree can have only 1 or 2 centers

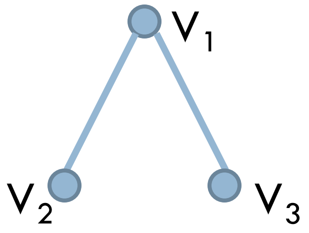
- Theorem 14: There is one and only one path between every pair of vertices in a tree, T
- Proof:
- Since T is a connected graph, there must exist at least one path between every pair of vertices in T
- Now, suppose that between two vertices, say v_i & v_j, there exists two district paths
- Then union of there two paths will result in a circuit; But since T is a tree, it cannot contain any circuits
- Hence there could be no 2 distinct paths between any pair of vertices in T
- Hence the theorem

- Theorem 15: If, in a graph G, there is one & only one path between every pair of vertices, G is a tree
- Proof:
- Since there is at least one path between every pair of vertices in G, we can say that G is connected
- Now, if the graph has a circuit in it, there must be two vertices v_i & v_j such that there exists two distinct paths between v_i & v_j
- But since there is only one path between every pair of vertices, G cannot contain any circuit
- Since G is connected & contains no circuit, we can say that G is a tree

- Theorem 16: A tree with n vertices has n-1 edges
- Proof: By induction we prove that the theorem is true
- For n=1
  - No. of edges = n-1= 1-1= 0
  - Which is true as we can see in the figure
- For n=2
  - No. of edges = n-1 = 2-1=1
  - Which is also true
- For n=3
  - No. of edges = n-1 = 3-1=2
  - true
- Now we assume that the theorem is true for all trees with no. of vertices up to (n-1)







- We need to prove that the theorem holds for a tree with n vertices
- Consider a tree with n vertices; Remove an edge e_k from the tree; Let v_i & v_j be the end vertices of e_k
- Removal of e_k will disconnect the graph into 2 components say T_1 and T_2 each of which is again a tree
- Let the component T_1 contain n_1 vertices; then no. of edges in $T_1 = (n_1-1)$

- Let the component T_2 contain n_2 vertices; then no. of edges in $T_2 = (n_2-1)$
- Since n_1 and $n_2 < n$, the theorem holds, as per our assumption
- Also $n_1+n_2 = n$ (total no. of vertices in the graph)
- Now, add back the deleted edge e_k so that the graph again becomes connected
- The resulting no. of edges in the graph is given by $(n_1-1)+(n_2-1)+1 = n_1+n_2-1-1+1 = n-1$
- Hence a tree with n vertices will have (n-1) edges

- Theorem 17: Any connected graph with n vertices and (n-1) edges is a tree
- Proof:
- The minimum no. of edges required to make a connected graph with n vertices is (n-1)
- In such a graph, removal of any 1 edge leaves the graph disconnected
- If the graph had a circuit, there would be at least one edge, removal of which do not make the graph disconnected. But there is no such edge at all. Hence the graph is circuit-less
- Since the graph is connected & circuit-less, it is a tree

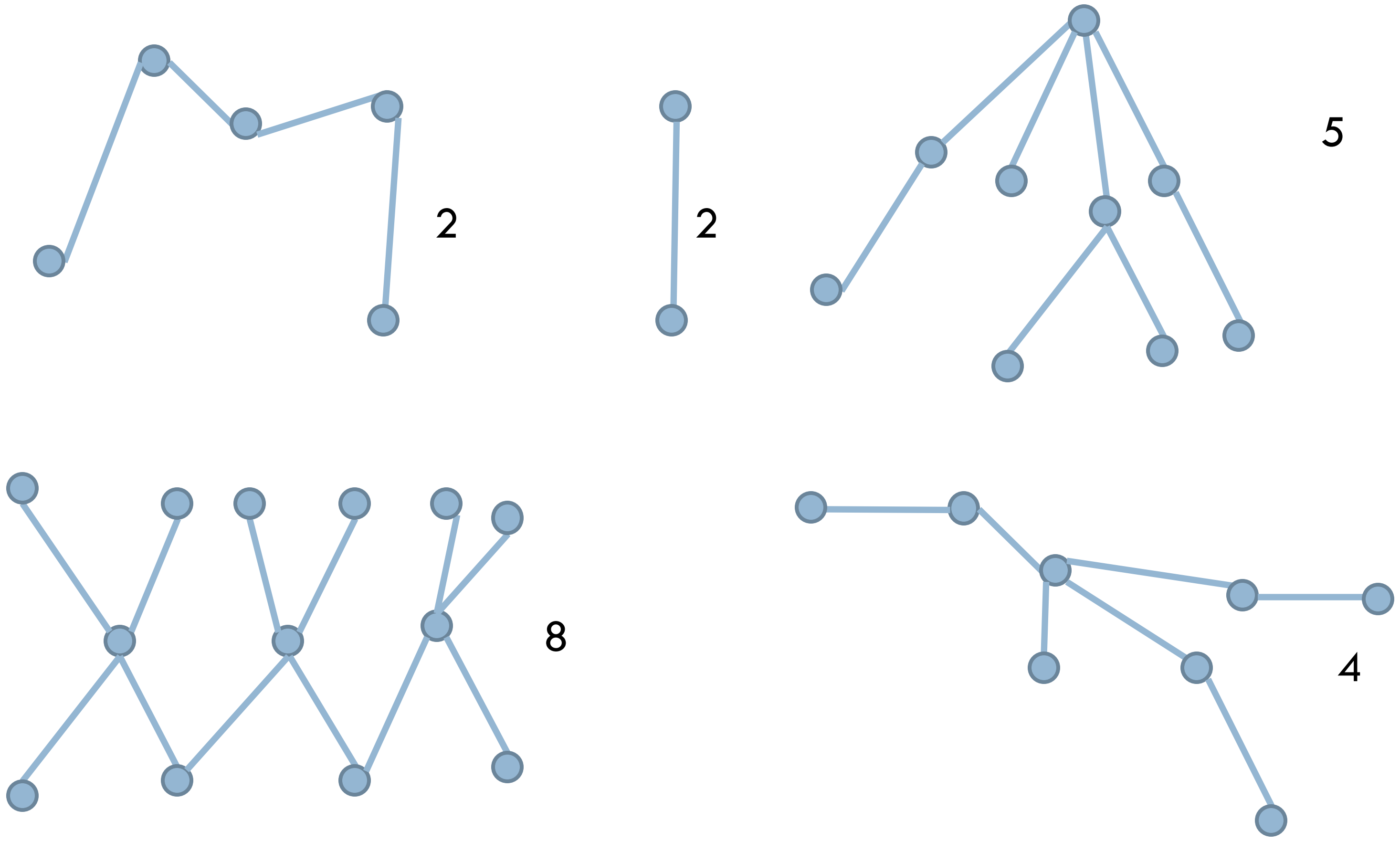
- Theorem 18: A graph is a tree if and only if it is minimally connected
- Proof:
- Suppose the graph is minimally connected
- A minimally connected graph, cannot contain any circuits because if it contained a circuit, then there must be at least one edge removal of which do not make the disconnected
- But all the edges of the graph are such that, removal of any edge, leaves the graph disconnected
- Hence the graph has no circuits & it is connected. So it must be a tree

- Conversely: Let the graph be a tree
- Since it is a tree, it is connected & circuit-less
- And also there is only & only one path b/w every pair of vertices
- Removal of any edge from a path leaves the graph disconnected, hence we can say that the graph which is a tree, is always minimally connected

- Theorem 19: A graph G with n vertices & (n-1) edges & no circuits is connected
- Proof:
- Suppose there exists a graph with n vertices, n-1 edges & no circuits, but it is disconnected
- Then G may contain 2 or more circuit-less components
- Now our disconnected graph G has total n vertices & n-1 edges
- Now add an edge e_k between vertex v_i from one component & vertex v_j from another component
- Adding of this edge will not create circuit as v_i &v_j were from different components & there wasn't any path between them
- Now, our graph must contain n vertices & n edges, which is not possible for a tree. Hence the graph G must be connected

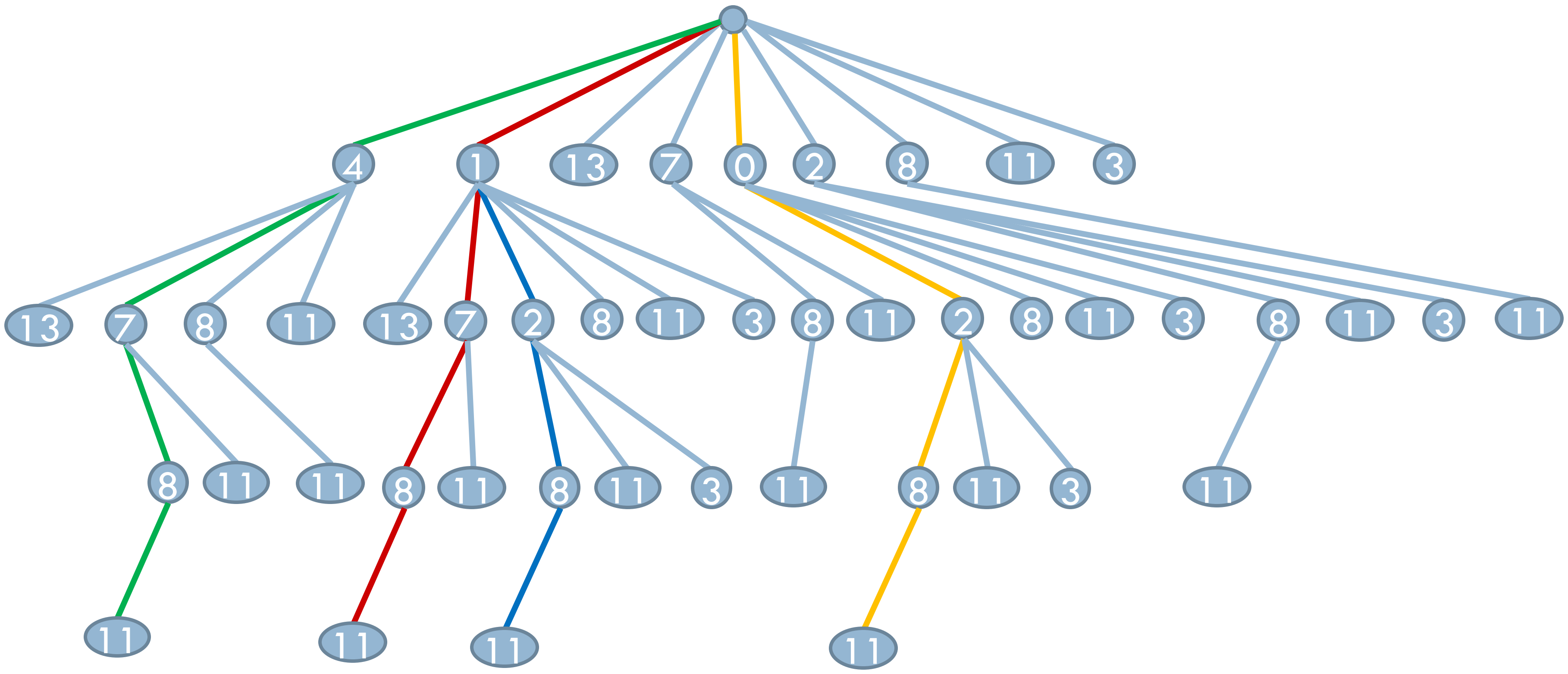
- Theorem 20: In any tree, there must be at least 2 pendant vertices
- Proof:
- A tree with n vertices will have (n-1) edges
- As each edge contributes 2, to the total degree of the graph, total degree of the tree is 2(n-1) = 2n-2
- Which shows all n vertices may take degree 2 each, except 2. Two of the vertices will get only degree1 each
- Hence there will be at least 2 pendant vertices for any tree

Examples



An application of trees

- Problem: Given a sequence of integers, no two of which are the same, find the largest monotonically increasing subsequence in it
- Solution: Suppose if the given sequence is 4,1,13,7,0,2,8,11,3
- Represent it by a tree where each sequence number in the sequence is a descendant vertex of a root vertex
- Now, the path from the start vertex to a particular vertex, v gives the monotonically increasing subsequence ending in v


4
1
13
7
0
2
8
11
3

- For the given sequence, the length of the largest such subsequence is four
- And there are 4 different subsequence, each of length four
- They are (4,7,8,11) , (1,7,8,11) , (1,2,8,11) & (0,2,8,11)
- Such trees are called data trees by computer programmers

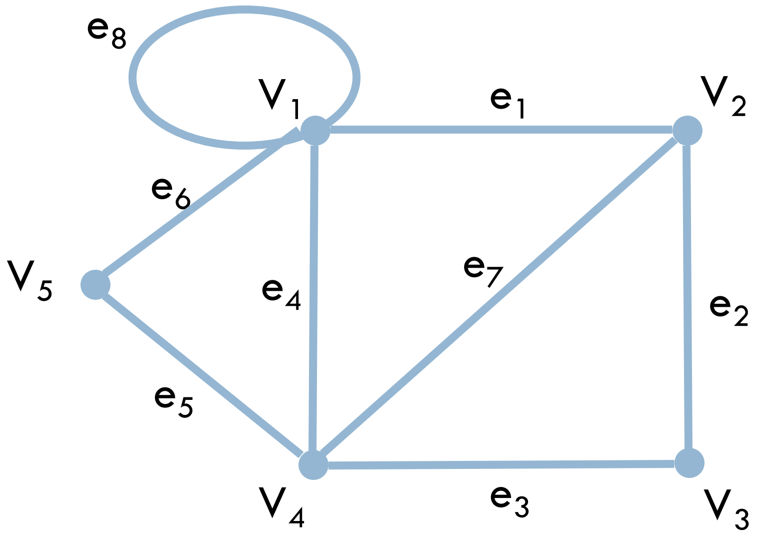
- Q. Find the largest monotonically increasing subsequence of the given sequence 5, 0, 1, 7, 6, 8, 2, 9, 4, 3

Path length

- Length of a path: Length of the path between any two vertices of a graph is simply the no. of edges included in the path
- Shortest path: If more than one path is present between any 2 vertices, then the path with the minimum no. of edges is the shortest path.
- Distance: Distance between any 2 vertices, v_i & v_j is represented as d(vi&vj) and is given by the length of the shortest path between them
- In a normal connected graph, to find the distance b/w any 2 vertices, we have to enumerate all available paths & pick the one with the shortest length.
- However in a tree, there is only path b/w every pair of vertices; so distance between any 2 vertices is just the length of the path b/w them

Example: distance b/w two vertices

- Eg: In graph G, available paths between v_5&v_2 are
  - $v_5\ e_6\ v_1\ e_1\ v_2$
  - $v_5\ e_5\ v_4\ e_7\ v_2$
  - $v_5\ e_6\ v_1\ e_8\ v_1\ e_1\ v_2$
  - $v_5\ e_5\ v_4\ e_3\ v_3\ e_2\ v_2$
  - $v_5\ e_5\ v_4\ e_4\ v_1\ e_1\ v_2$
  - $v_5\ e_5\ v_4\ e_4\ v_1\ e_8\ v_1\ e_1\ v_2$
  - $v_5\ e_6\ v_1\ e_4\ v_4\ e_7\ v_2$
  - $v_5\ e_6\ v_1\ e_4\ v_4\ e_3\ v_3\ e_2\ v_2$



- Two shortest paths are available between v_5&v_2
- Each contain two edges
- Hence $d(v_5,v_2) = 2$

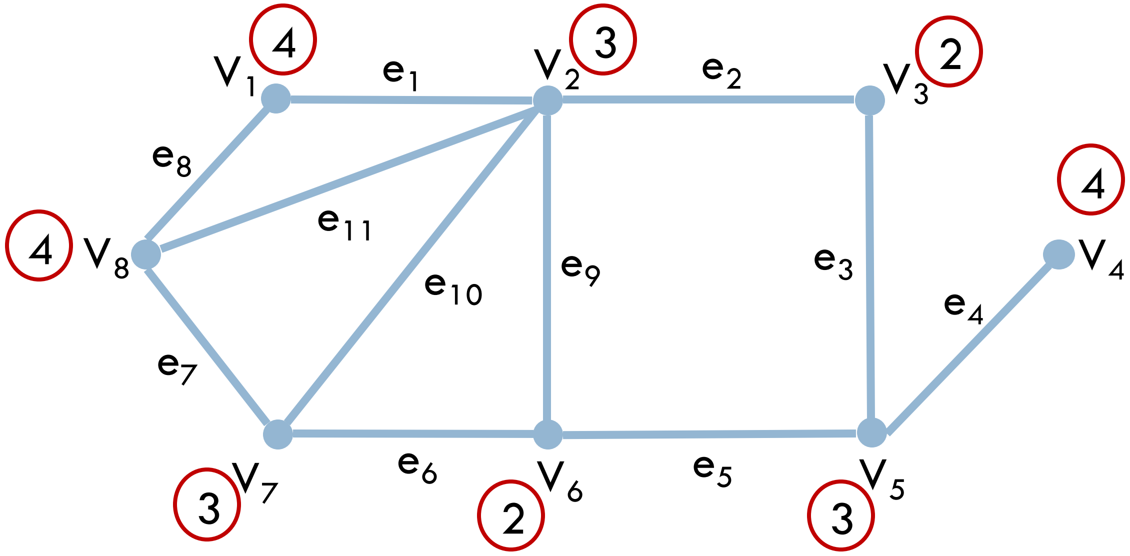
Metric

- A function f(x,y) of two variables is called a metric if it satisfies the following 3 conditions
  1) Non-negativity: f(x,y)>=0
     f(x,y)=0 iff x=y
  2) Symmetry: f(x,y)= f(y,x)
  3) Triangle inequality: f(x,y)<= f(x,z)+ f(z,y) for any z

- Theorem 21: The distance between the vertices of a connected graph is a metric
- Proof:
- Checking the conditions of a metric for the distance between any 2 vertices of a connected graph
- Non negativity: Distance b/w any 2 vertices is never negative. Distance from one vertex to itself is zero (simple graph)
- Symmetry: Distance from one vertex to another vertex will be the same in the reverse direction (undirected graph)
- Triangle inequality: Since distance b/w any 2 vertices $d(v_i \& v_j)$ is the shortest path between them, there cannot be any shorter path that goes through some vertex v_k such that $d(v_i \& v_j) <= d(v_i \& v_k) + d(v_k \& v_j)$
- Hence distance between the vertices of a connected graph is a metric

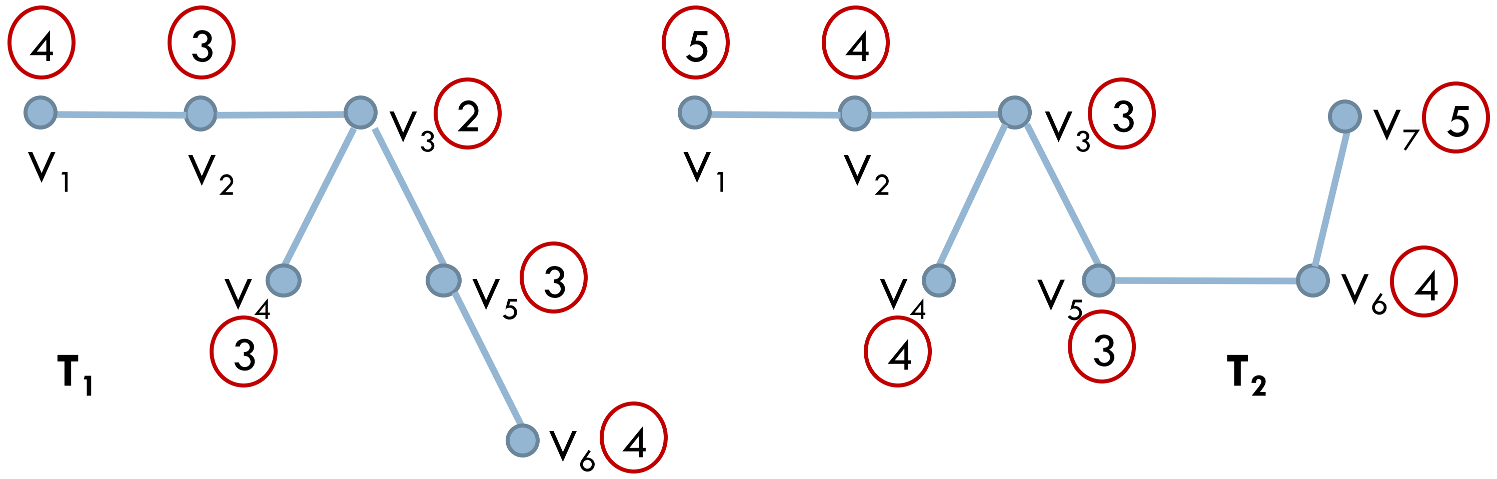
Eccentricity of a vertex

- Eccentricity E(v) of a vertex v in a graph G is the distance from v to the farthest available vertex in G
- $E(v) = \max d(v \& v_i)\ \forall$ vertex v_i of G
- Eg:
  - $E(v_1)=4$
  - $E(v_2)=3$
  - $E(v_3)=2$



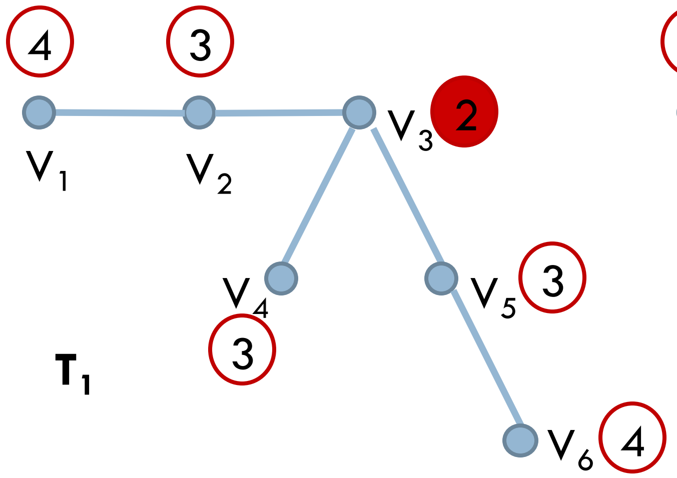
Center of a tree

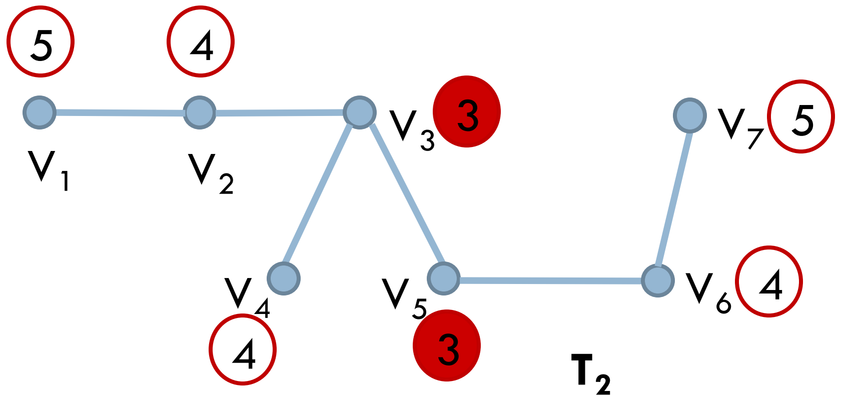
- The vertex with the minimum eccentricity is considered as the center of the tree
- Some trees may have 2 centers - bi-centered
- Every tree will have either one or two centers



Radius of a tree

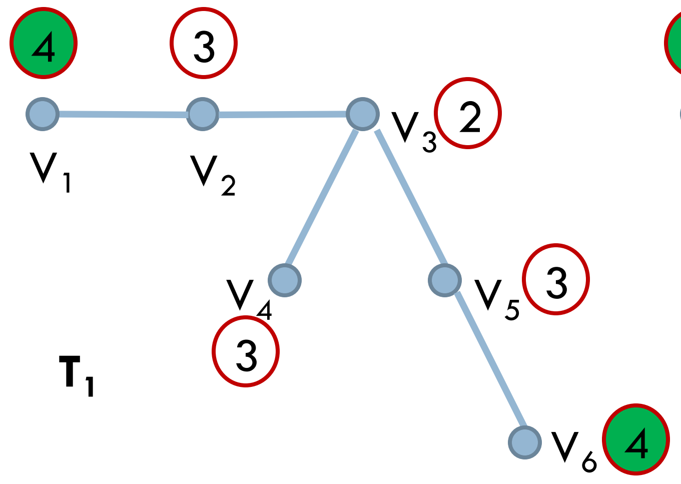
- The eccentricity of the center of the tree is regarded as the radius of the tree
  - In tree T_1, radius is 2
  - In tree T_2, radius is 3

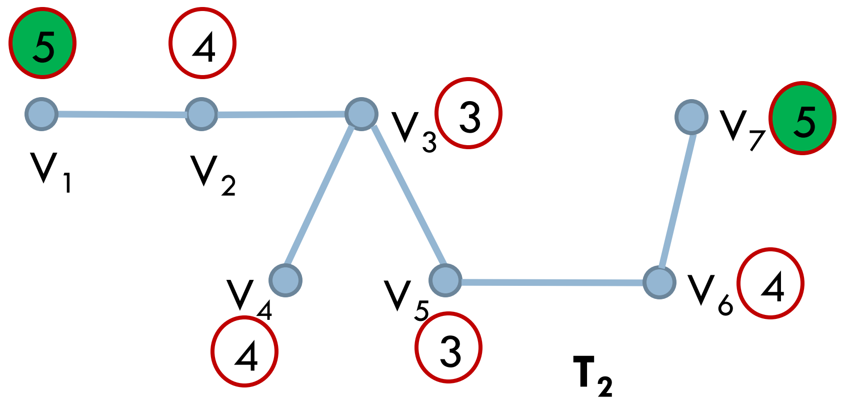




Diameter of a tree

- The length of the longest path in the tree is the diameter of the tree
- It is not necessary that the diameter be twice its radius
  - In tree T_1, diameter is 4 (radius is 2)
  - In tree T_2, diameter is 5 (radius is 3)



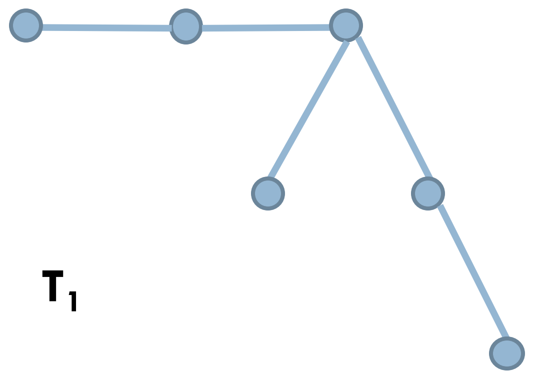


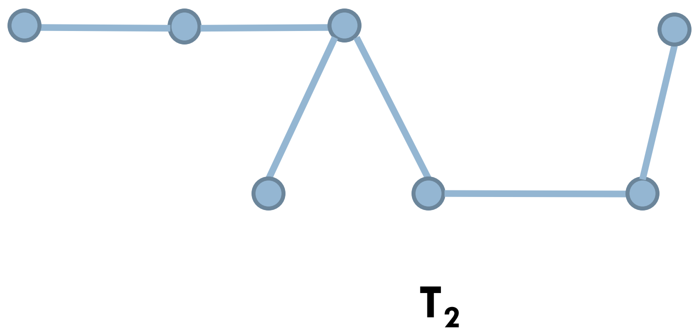
- Theorem 22: Every tree has either one or two centers
- Proof:
- Consider a tree with n vertices where n>2
- A tree must have 2 or more pendant vertices
- Delete all pendant vertices from T. The resulting graph is still a tree, say T'. Deletion of the pendant vertices, will reduce the eccentricities of all remaining vertices by one. Hence the center of the graph will remain the same. From T' again remove all pendant vertices, to obtain another tree T''. Again T'' has the same vertex as its center
- Containing this process will finally end up with a single vertex or a single edge (2 vertices)

- Single vertex implies that the tree had 1 center which is the one finally left
- Single edge implies that the tree had 2 centers which are the end vertices of the edge left
- Hence any tree can have only 1 or 2 centers

- <u>Corollary:</u> If a tree has 2 centers, then both of them must be adjacent

Examples



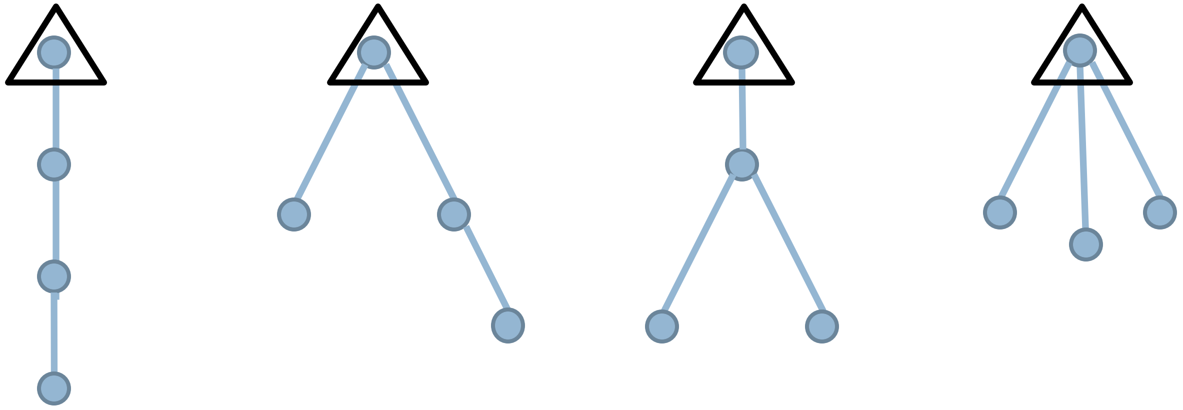


1 vertex is left,
which is the center
of the tree

1 edge is left,
end vertices of
which are the
centers of the tree

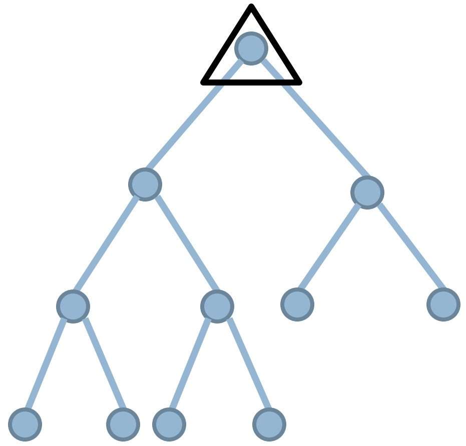
Rooted trees

- A tree in which one vertex (called the root) is distinguished from all others is called a rooted tree
- In a rooted tree, the root vertex is usually marked within a ∆
- Eg: All possible rooted trees with four vertices



Binary trees

- They are a special class of rooted trees where the root vertex is of degree 2 & all remaining vertices are of degree either 1 or 3
- Type of vertices in a binary tree:
  - Root vertex: The only vertex with degree 2
  - Pendant vertices: The vertices that are of degree 1
  - Internal vertices: The vertices that are of degree 3 (Non-pendant vertices)

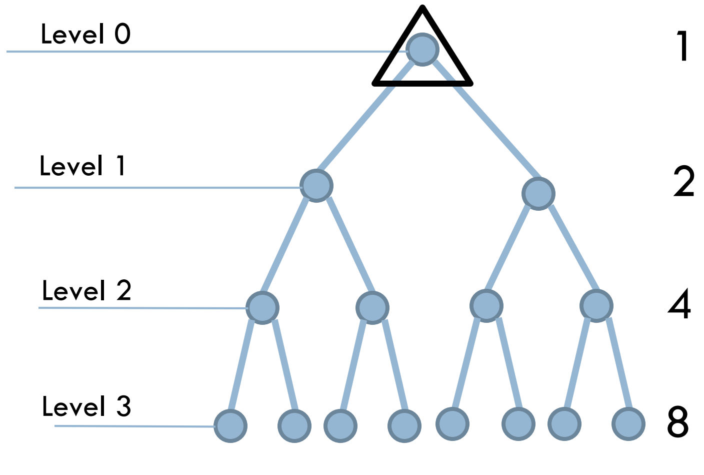


Levels of a binary tree

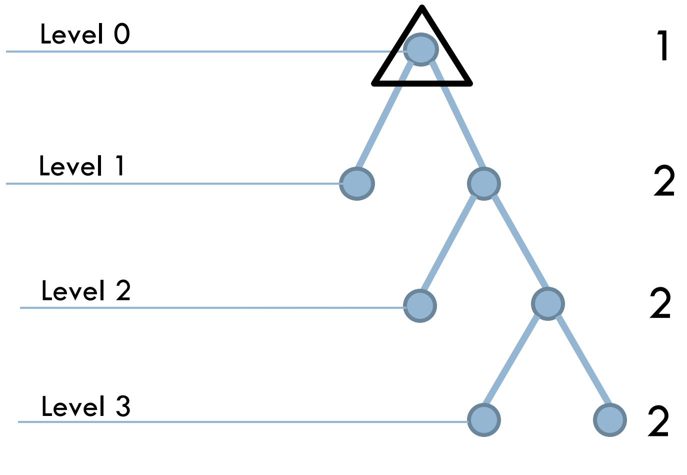
- In a binary tree, a vertex v_i is said to be at level L_i if v_i is at a distance of L_i from the root
- The root is assumed to be at level 0
- Hence at level 0, there could be only 1 vertex
  - at level 1, at most $2^1 = 2$ vertices (either 0 or 2)
  - at level 2, at most $2^2 = 4$ vertices & so on
  - at level k, at most 2^k vertices
- Hence maximum no. of vertices possible in a k-level binary tree is therefore
- $V_{max} = 2^0 + 2^1 + 2^2 + \ldots\ldots + 2^k$
- $V_{min} = 1 + 2 + 2 + \ldots\ldots + 2$ (k times)$= 2k+1$

Examples

- Here k=3
- $V_{max} = 2^0 + 2^1 + 2^2 + 2^3 = 1+2+4+8 = 15$ vertices
- $V_{min} = 2*3+1 = 7$ vertices



Total 15 vertices



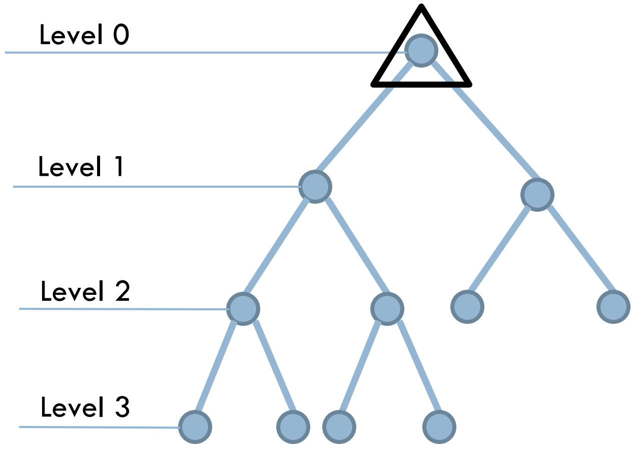
Total 7 vertices

Height of a binary tree

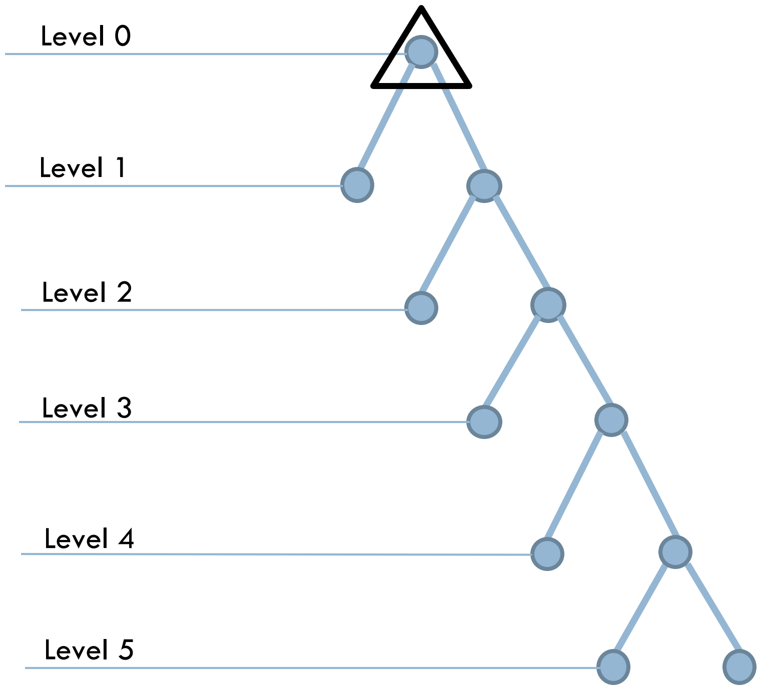
- No. of levels possible in an ‘n vertex’ binary tree
- $L_{min} = \lceil \log_2(n+1)-1 \rceil$
- This is obtained when all levels are filled with maximum possible vertices (tightly packed)
- $L_{max} = \frac{(n-1)}{2}$
- This is obtained when the farthest vertex is made as far as possible from the root; all levels contain just 2 vertices (loosely packed)

Example

- Consider a binary tree with n=11
  - $L_{min} = \lceil \log_2(n+1)-1 \rceil = \lceil \log_2(12)-1 \rceil = 3$
  - $L_{max} = \frac{(n-1)}{2} = 5$



Min level binary tree for n=11



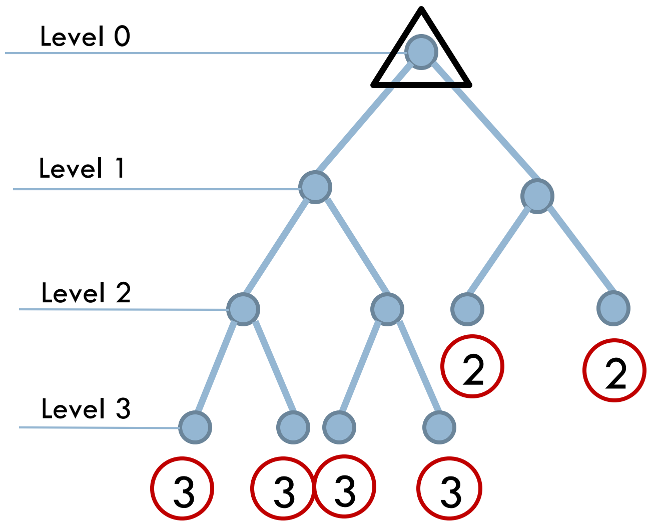
Max level binary tree for n=11

Application of binary trees

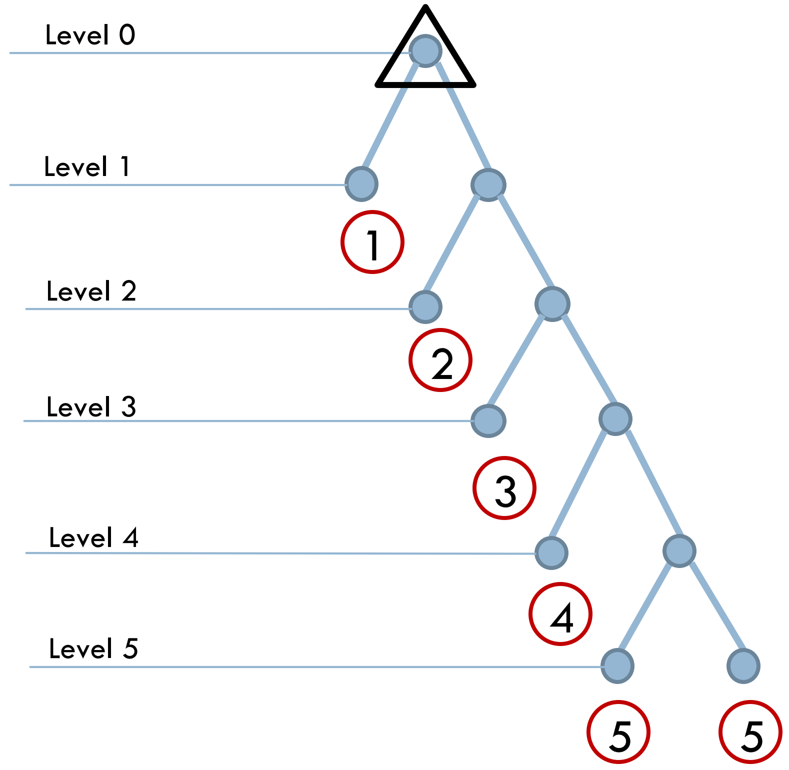
- Search procedures:
  - Each vertex is a test (yes/no)
  - Outcome of the test decides to choose one of the two vertices at the next level
  - The process continues until we reach a pendant vertex, where we get the final output
  - If the height of the tree is made as small as possible, we could reduce the number of tests
  - Eg: binary search

Path length of a binary tree

- Path length of a tree is defined as the sum of the path lengths of the pendant vertices from the root OR
- Sum of the levels of all pendant vertices
- It is an important factor as it is related to the execution time of an algorithm
- $P = \Sigma L_i \;\; \forall$ pendant vertex V_i
- $P = 3+3+3+3+2+2 = 16$



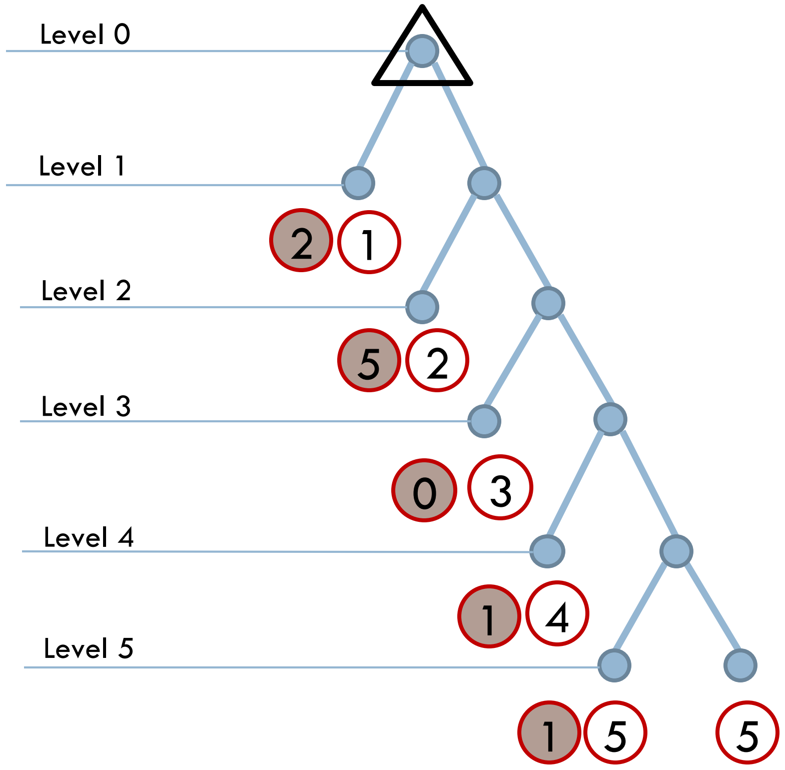
- Obtain the path length of the binary tree given

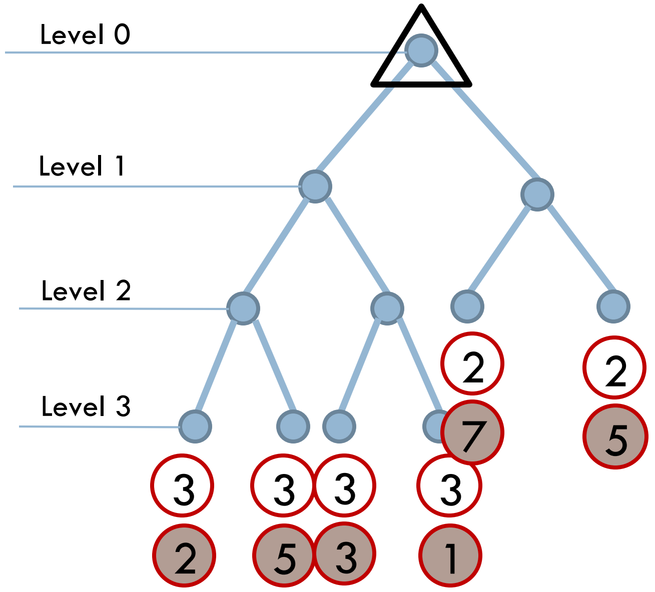


Weighted Path Length

- All pendant vertices are associated with a positive real number, called weight w
- Path length of each pendant vertex is multiplied by the corresponding weight to obtain weighted path length of the pendant vertex
- Hence weighted path length of the binary tree is the sum of the weighted path lengths of the pendant vertices
- $P_w = \Sigma\ w_i\ L_i\ \forall$ pendant vertex V_i
- In applications, we would be required to build binary trees with minimum weighted path length

- $P_w = \Sigma\, w_i L_i = 3x2+3x5+3x3+3x1+2x7+2x5 = 57$

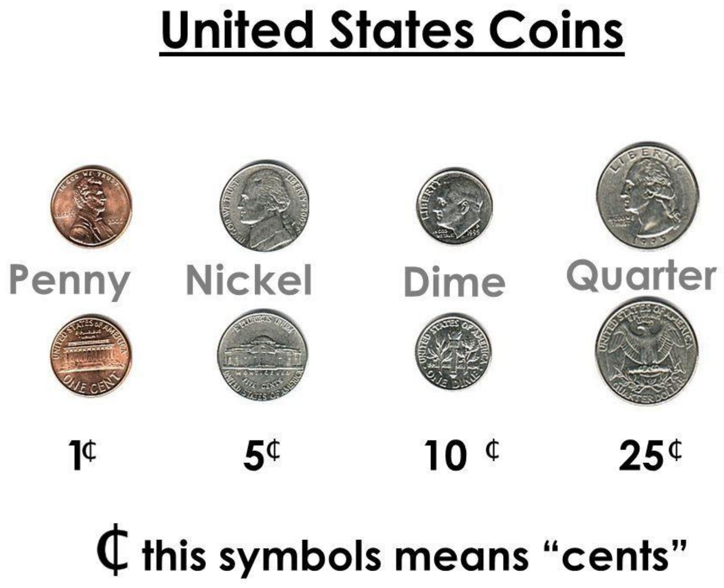




- Problem: Consider a coke machine that accepts coins. Suppose the machine lets only 4 types of coins to go through the slot, say pennies, nickels, dimes & quarters. The machine performs a sequence of tests to identify the coin. Probability of a coin being a penny, nickel, dime & a quarter are 0.05, 0.15, 0.5 & 0.30 respectively. For each type of coin, 1 test is done. Suppose all 4 tests take the same amount of time. The result of each test is yes or no. Then in what order must the tests be done, so as to minimize the overall coin determination time.

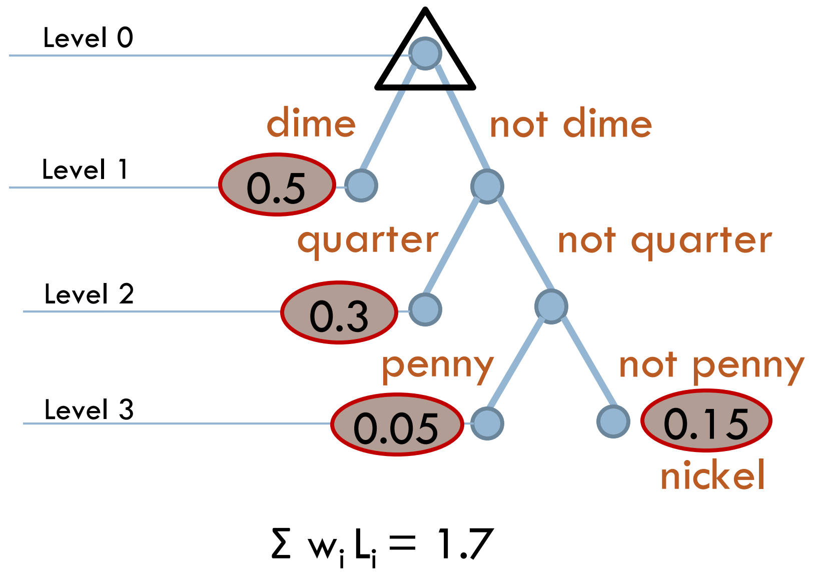
Solution

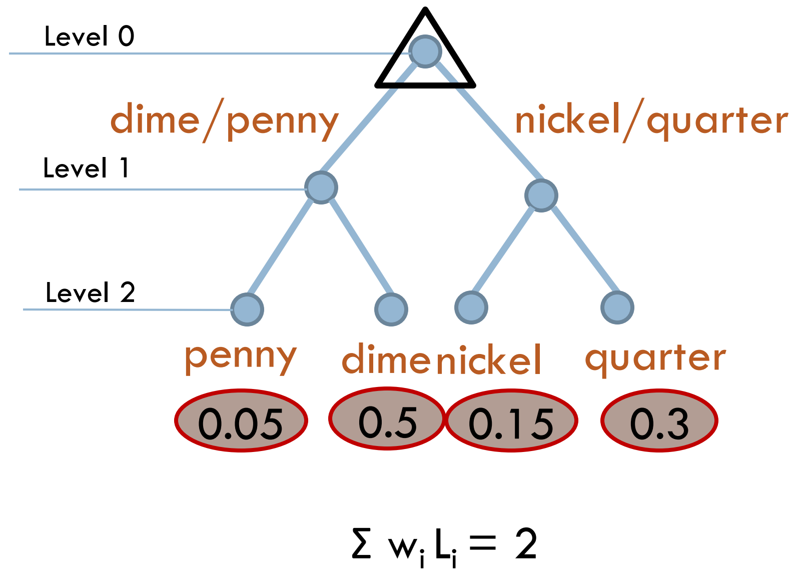
- Construct a binary tree with 4 pendant vertices, with their corresponding weight, as w1=0.05, w2= 0.15, w3= 0.5, w4 =0.3, such that the weighted path length is the minimum



Possible binary trees

- If ‘t’ is the time taken for each test, then weighted path lengths are
  - $\Sigma\, w_i L_i = 1x0.5+2x0.3+3x0.05+3x0.15=1.7t$
  - $\Sigma\, w_i L_i = 2x0.5+2x0.3+2x0.05+2x0.15= 2t$

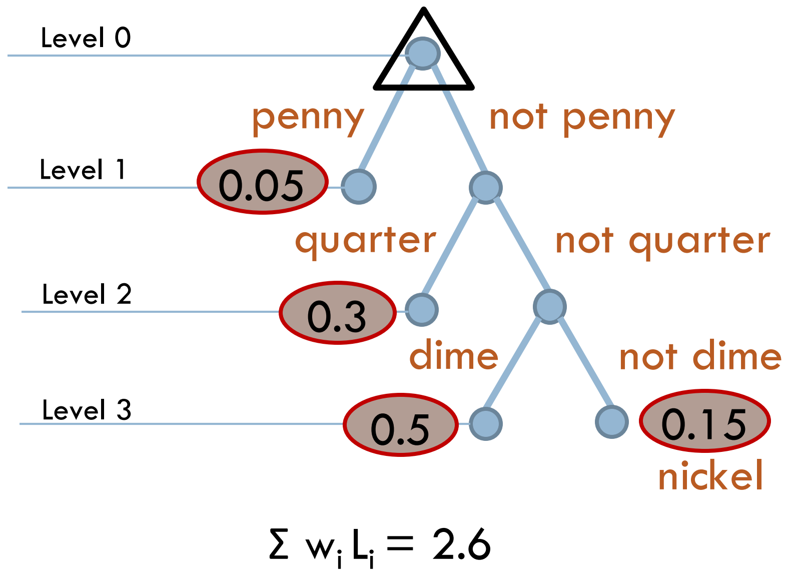


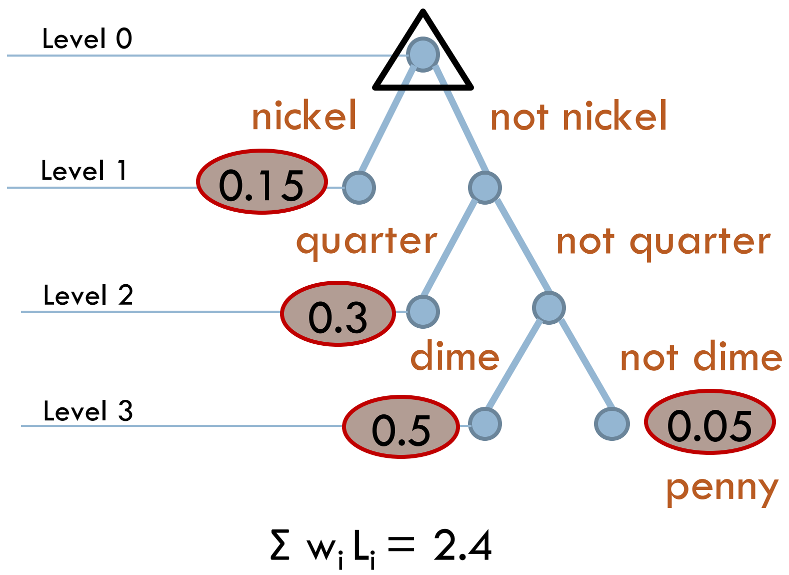


Possible binary trees

- $\Sigma\, w_i L_i = 1x0.05+2x0.3+3x0.5+3x0.15=2.6t$
- $\Sigma\, w_i L_i = 1x0.15+2x0.3+3x0.5+3x0.05=2.4t$

- Hence the first binary tree would be our solution





- Theorem 23: The number of vertices 'n' in a binary tree is always odd
- Proof: In a binary tree, only the root node has even degree
- In an 'n' vertex binary tree, remaining (n-1) vertices are of odd degree
- The no. of odd degree vertices in any graph is even
- Therefore the quantity (n-1) must always be even
- Then n is odd

- Theorem 24: In a binary tree with n vertices, the number of pendant vertices, p = $\frac{(n+1)}{2}$
- Proof:
- In a binary tree, no. of root vertex = 1 & the degree of root vertex is 2
- Let p denote the no. of pendant vertices degree of each pendant vertex is 1
- No. of internal vertices = n-p-1 & the degree of each internal vertex is 3
- We know that the sum of the degrees of all vertices in a graph is twice the no. of edges
- In a binary tree with n vertices, the no. of edges = n-1; hence we may get
- 2x1 + 1xp + 3x(n-p-1) = 2(n-1)
- 2+p+3n-3p-3 = 2n-2
- 2-3+2 +3n-2n = 3p-p
- 1+n = 2p → p = $\frac{(n+1)}{2}$

- Theorem 25: The maximum possible level or height of a binary tree with n vertices is given by
  - $L_{min} = \lceil \log_2(n+1)-1 \rceil$
  - $L_{max} = \frac{(n-1)}{2}$
- Proof:
- At level 0, we have 1 vertex
- At level 1, we have 2 vertices
- At level 2, we have 3 vertices
- Maximum no. of vertices in a k-level binary tree is
- $V_{max} = 2^0 + 2^1 + 2^2 + \ldots\ldots + 2^k$
- That means, $n <= 2^0 + 2^1 + 2^2 + \ldots\ldots + 2^k$

- Sum of a GP = $\frac{ar^{n+1}-1}{r-1}$
- Hence r=2, a=1
- then n <= $\frac{2^{k+1}-1}{2-1}$
- n <= 2^{k+1} -1
- n+1 <= 2^{k+1}
- $\log_2(n+1) <= (k+1)\log_2 2$
- $\log_2(n+1) <= k+1$
- $\log_2(n+1) - 1 <= k$
- $K >= \log_2(n+1) - 1$
- Hence $L_{min} = \lceil \log_2(n+1)-1 \rceil$

- Now, to construct a binary tree with n vertices such that the farthest vertex is as far as possible from the root, we must have exactly 2 vertices at each level except at level 0
- If k is the max level, then
  - At level 0, no. of vertex =1 (only the root vertex)
  - At level 1, no. of vertices = 2
  - At level 2, no. of vertices = 2
  - At level k, no. of vertices = 2
- Then total no. of vertices = 1+2+2+2+.....(k times) = n

$$1+2k = n$$

$$k = \frac{(n-1)}{2}$$

- Hence $L_{max} = \frac{(n-1)}{2}$

Sum up

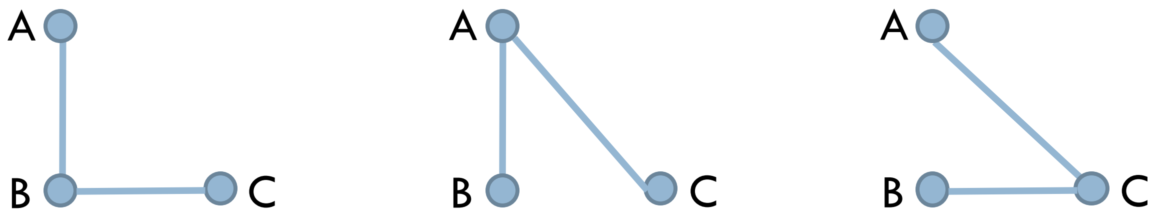
- Binary trees are rooted trees
- Root vertex has degree 2, all internal vertices are of degree 3 & all pendant vertices are of degree 1
- The total no. of vertices, n in a binary tree is always odd.
- The no. of internal vertices in a binary tree is (n-p-1)
- The no. of pendant vertices, p = $\frac{(n+1)}{2}$
- Max no. of vertices possible for a k-level binary tree is
- $V_{max} = 2^0 + 2^1 + 2^2 + \ldots\ldots + 2^k$
- The max possible level for a n vertex binary tree is
- min $L_{max} = \lceil \log_2(n+1)-1 \rceil$
- max $L_{max} = \frac{(n-1)}{2}$
- Path length of a binary tree $P = \Sigma L_i \ \forall$ pendant vertex V_i
- Weighted path length, $P_w = \Sigma w_i L_i \ \forall$ pendant vertex V_i

Counting trees

- The total number of trees that one can construct with a given number of vertices
- The count is different for labeled and unlabeled trees

Labeled trees

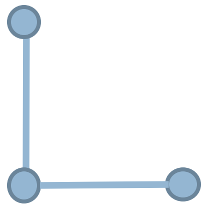
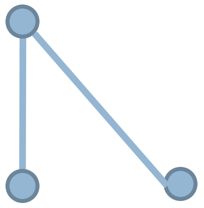
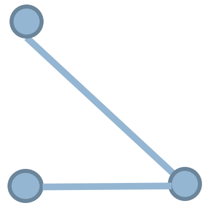
- A tree in which each vertex is assigned a unique label
- According to Cayley's theorem, the no. of different labeled trees possible with n vertices is n^{n-2}
- Eg: no. of labeled trees possible for n=3 is $3^{3-2} = 3$



Problem

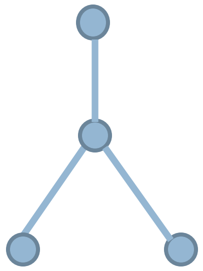
- Draw all labelled trees for n=4

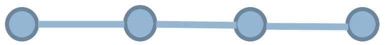
Unlabeled trees

- A tree in which there is no distinction between the vertices by their name. However vertices differ in their degree
- Eg: Count all unlabeled trees for n=3
- There is only 1 tree possible with n=3
- 
- Through we can draw  and 
- It is all the same, since unlabeled

Problem

- Count all unlabeled trees for n=4
- Solution:





Problem

- Count all unlabeled trees for n=5

Application of counting trees

- In 1857, Arthur Cayley discovered trees while he was trying to count the structural isomers of saturated hydrocarbons
- He used a connected graph to represent hydrocarbon molecules

Saturated hydrocarbons

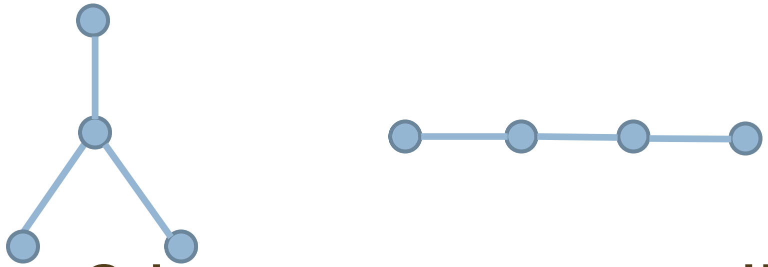
- They contain carbon & hydrocarbon atoms
- General formula is C_kH_{2k+2}
- Carbon atoms have valency 4 & hydrogen atoms have valency 1
- Since their chemical structure assembles that of a tree, properties of trees are used to study them further
- These hydrocarbons can be represented as labeled trees, where carbon & hydrogen atoms are denoted as vertices & the bonding between them is shown as edges

- Since in C_kH_{2k+2}
  - No. of C atoms= k &
  - No. of H atoms= 2k+2
  - Total no. of vertices in the tree = k+2k+2 = 3k+2
- Since in C_kH_{2k+2}
  - degree of each C atom = 4
  - degree of each H atom = 1
  - Total degree of the graph = 4k+2k+2=6k+2
  - Hence, no. of edges = $\frac{(6k+2)}{2}$ = 3k+1

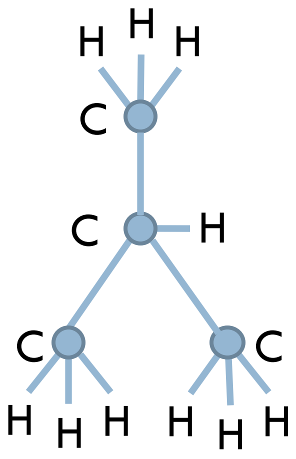
Problem

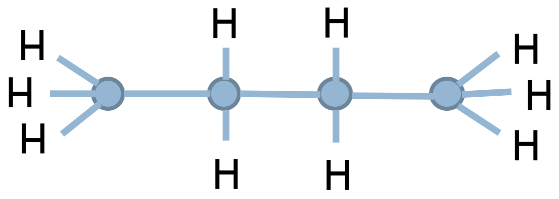
- How many structural isomers are there for Butane (C_4H_{10})?
- Graph theoretic problem: How many different unlabeled trees are possible with n = 4 ?
- Solution: we have omitted the hydrogen atom vertices, because hydrogen atom vertices are pendant vertices, they go with carbon atoms only one way. Hence they make no contribution to isomerism.
- As a result the tree contains only C atoms. So we can use unlabeled trees, as there is no distinction between C atom vertices

- Possible unlabeled trees for n=4 are:



- Hence 2 isomers are possible for butane:


iso Butane


n Butane

Problem

- Find out the structural isomers of pentane

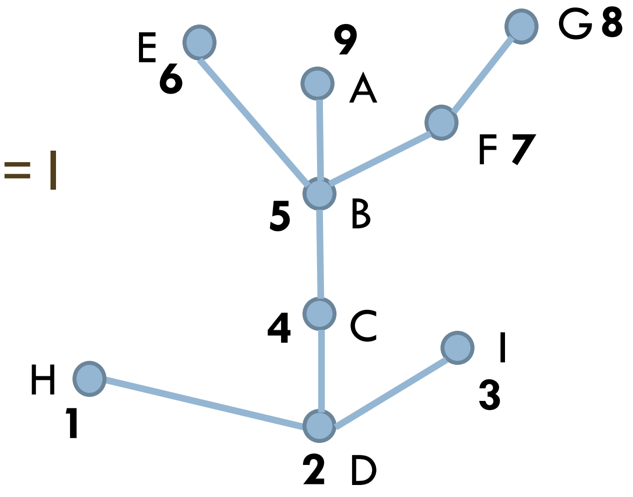
Cayley’s theorem

- Theorem 26: The number of labelled trees with 'n' vertices (n>=2) is n^{n-2}
- Proof:
- Suppose we have a tree with n uniquely labeled vertices
- Number all vertices with numbers from 1 to n
- Find the pendant vertex with the least number. Let it be P_1. Let a_1 be the vertex adjacent to p_1
- Delete vertex p_1 along with the incident edge from the tree
- Start a prufer sequence; add a_1 to the prufer sequence (a_1)

- From the remaining tree with (n-1) vertices, find the pendant vertex with the least number. Let it be p_2. Let a_2 be the vertex adjacent to p_2
- Delete vertex p_2 along with the incident edge from the tree
- Add a_2 to the prufer sequence (a_1, a_2,)
- Continue the process until only 2 vertices are left in the tree
- Now the prufer sequence (a_1, a_2, …, a_{n-2})uniquely defines the tree
- Thus with n vertices, we can get n^{n-2} different labeled trees

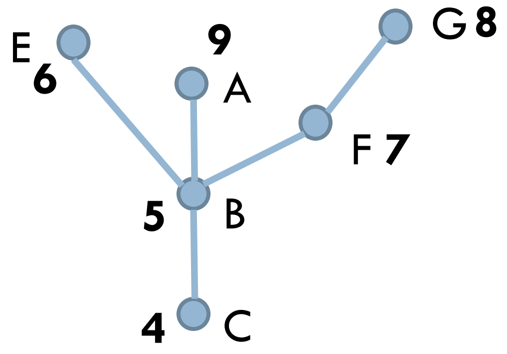
Example

- Consider the uniquely labelled tree given
- Number the vertices
- The pendant vertex with least number is H; hence p_1=vertex H
  - a_1=vertex D(2) (adjacent to p_1)
  - Prufer seq = (2)
  - Delete H
- Next pendant vertex with least number p_2= I
  - a_2=vertex D(2) (adjacent to I)
  - Prufer seq = (2, 2)
  - Delete I
- Next pendant vertex is p_3= D
  - a_1=vertex C(4) (adjacent to D)
  - Prufer seq = (2, 2, 4)
  - Delete C



Example

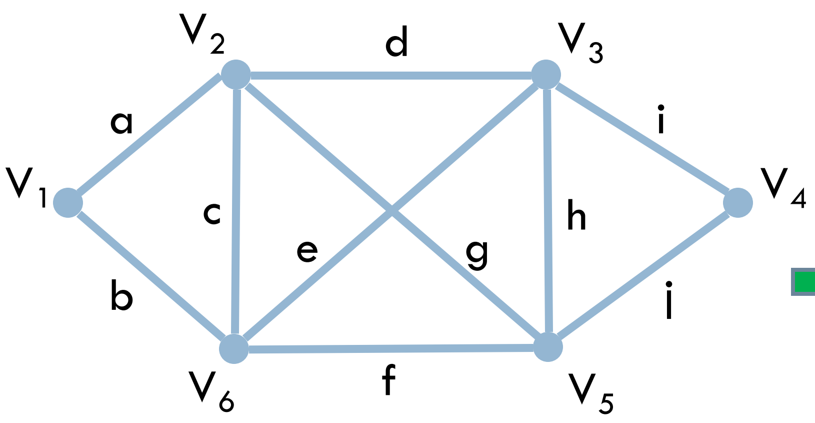
- Next is p_4= C
  - a_4=vertex B(5) (adjacent to C)
  - Prufer seq = (2, 2, 4, 5)
  - Delete C
- Next is p_5= E
  - a_5=vertex B(5) (adjacent to E)
  - Prufer seq = (2, 2, 4, 5, 5)
  - Delete E
- Next is p_6= G
  - a_6=vertex F(7) (adjacent to G)
  - Prufer seq = (2, 2, 4, 5, 5, 7)
  - Delete G
- Next is p_7= F
  - a_7=vertex B(5) (adjacent to F)
  - Prufer seq = (2, 2, 4, 5, 5, 7, 5)
  - Delete F



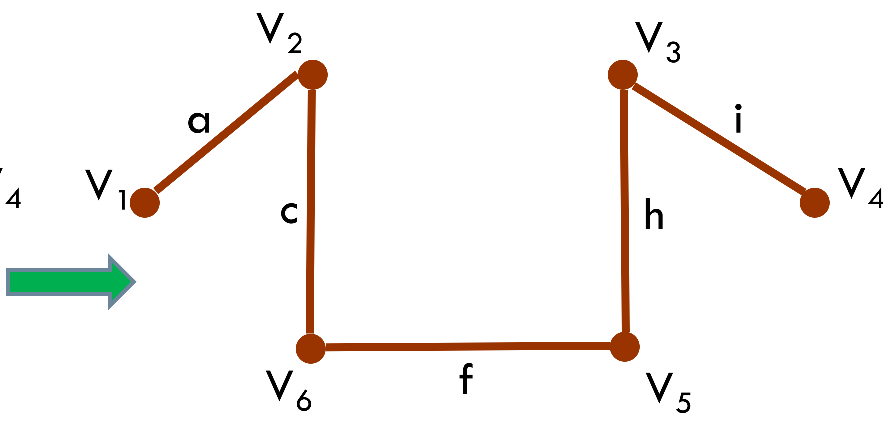
- Now the sequence is unique for the labelled tree
- Since the sequence contains 7 elements, and n=9, we can get $n^{n-2} = 9^7$ such unique sequences

Spanning trees

- A tree T is said to be a spanning a tree of a connected graph G if
  - T is a subgraph of G and
  - T contains all vertices of G



: **G**

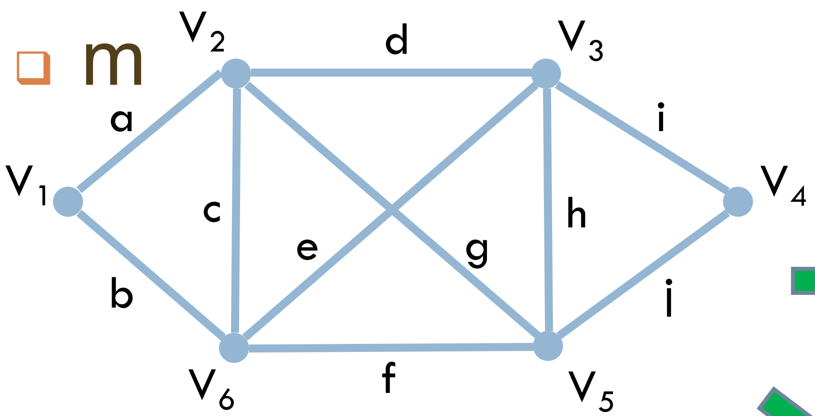


: **T**

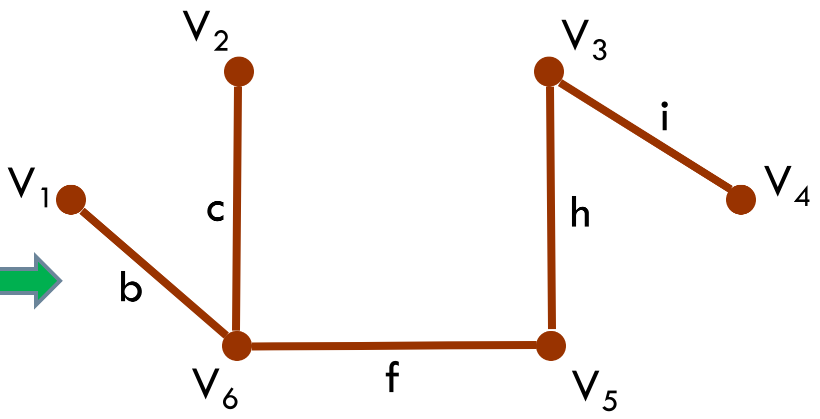
- Since the spanning tree depicts the skeleton of the graph G, T is also known as the skeleton/scaffolding of G
- Since spanning trees are the largest trees among all possible trees of G, it is also known as maximal tree subgraph or maximal tree of G
- Note:
- Spanning are always referred with respect to some connected graph
- For a given connected graph there can be many spanning trees

Different SPTs of a graph

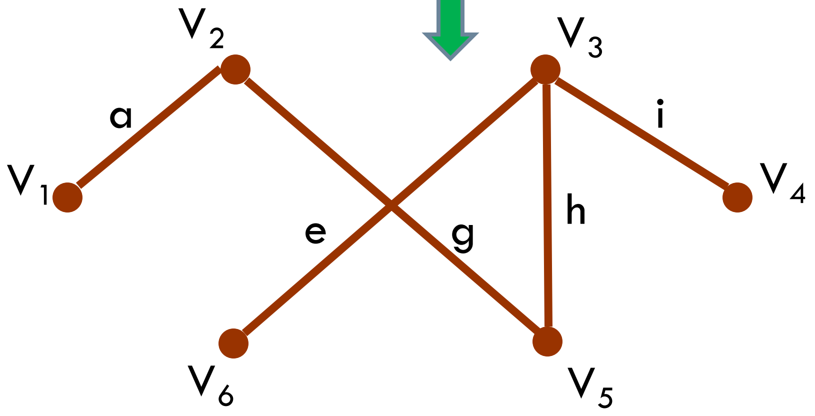
- m



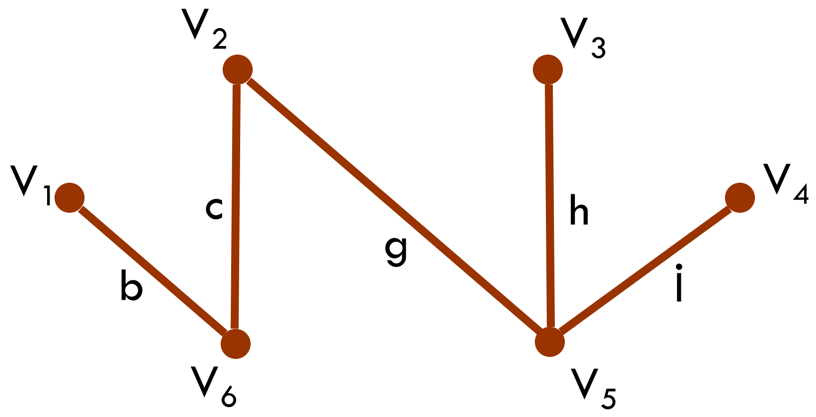
: G



: SPT 1



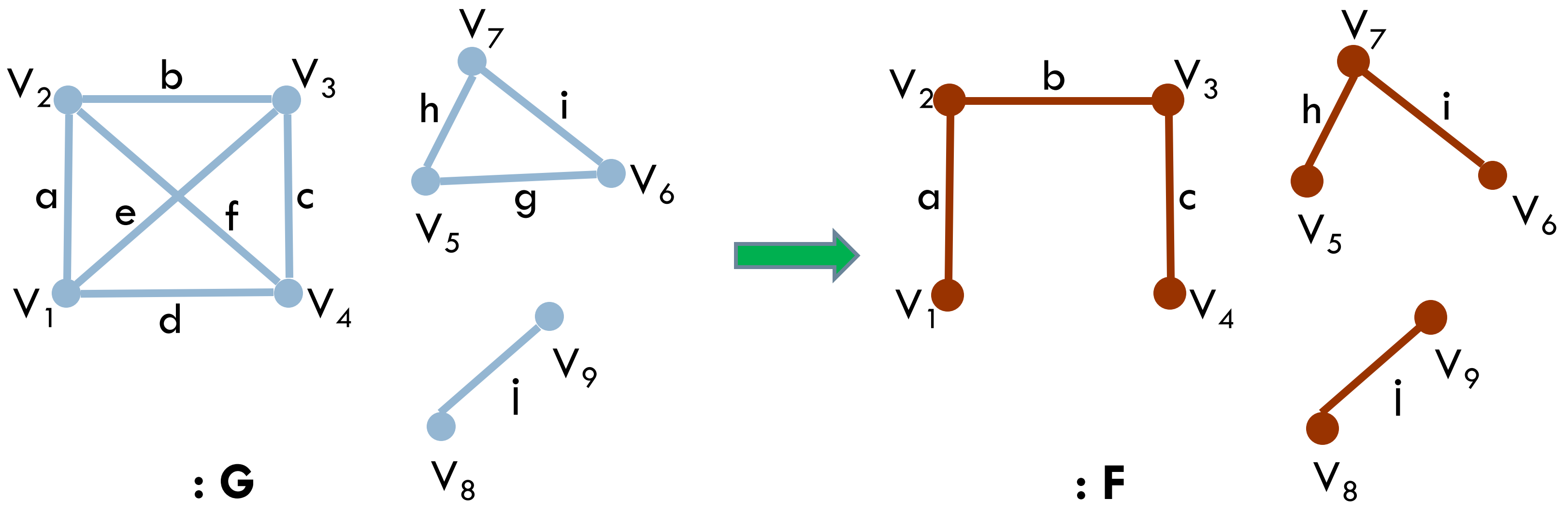
: SPT 2



: SPT 3

Spanning forest

- In a disconnected graph, a spanning tree can be found for each connected component, together which is known as a spanning forest
- Hence a disconnect graph with k components will have a spanning forest with k spanning trees



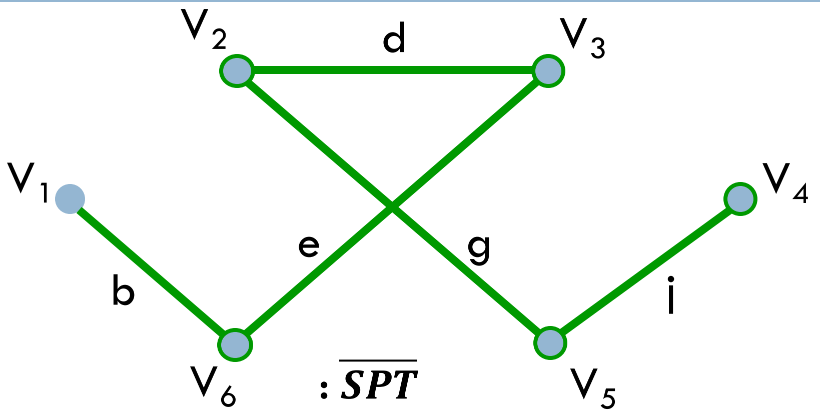
Finding the spanning tree

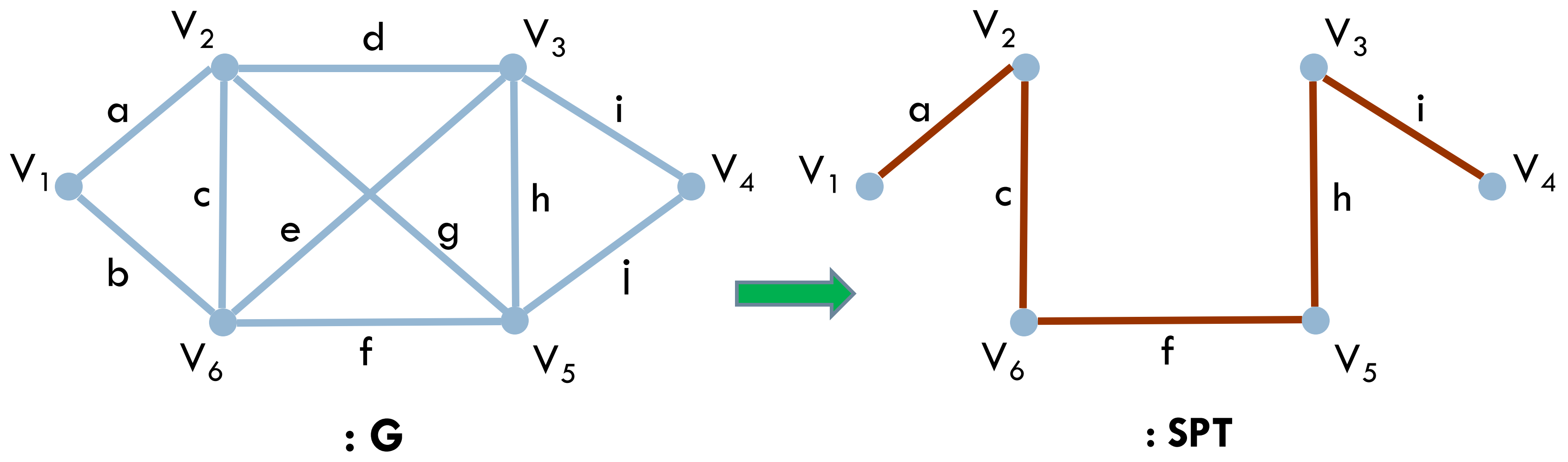
- For a given connected graph G,
  1) If G has no circuits, G itself is the SPT
  2) If G has a circuit, remove one edge from the circuit such that G still remains connected
  3) Repeat step 2 until G is circuit-less
  4) The remaining graph is a SPT

Branch & Chord

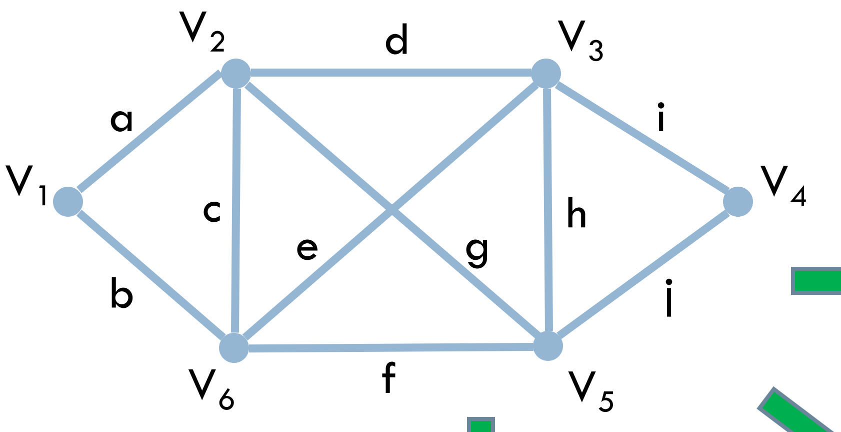
- W.r.t a given SPT, an edge of the graph G that is a part of the SPT is called a branch of the tree
- An edge of G that is not included in the SPT is called a chord of the graph G
- Hence, branches + chords = Edge set of G
- If T is the spanning tree of H and $\overline{T}$ be the complement of T in G then T U $\overline{T}$ = G
- Note:
  - A graph may have many spanning trees
  - Branches and Chords are defined wrt one SPT
  - An edge that is a branch of one SPT may be a chord for another SPT of the same graph

- For the SPT chosen,
  - Branches={a,c,f,h,i}
  - Chords={b,d,e,g,j}

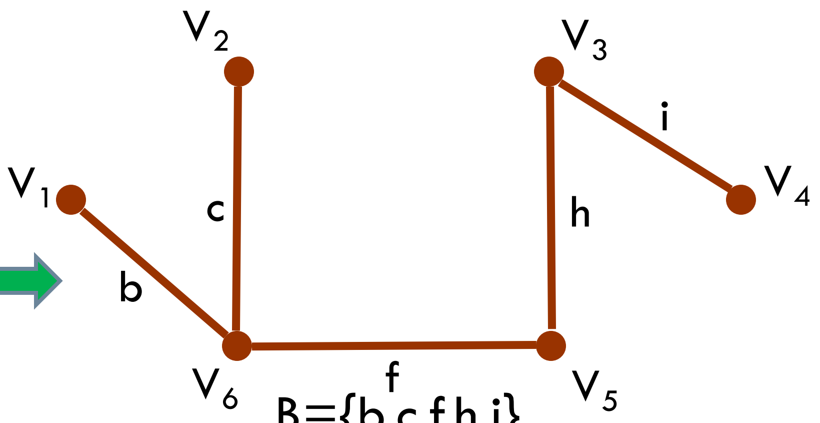




Different SPTs of a graph

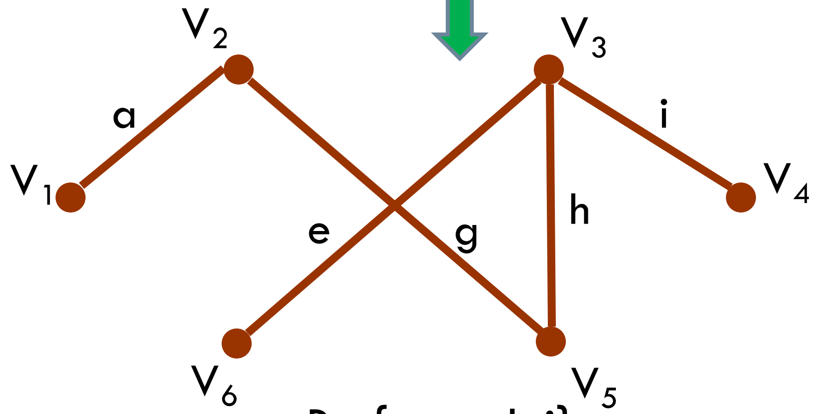


: **G**



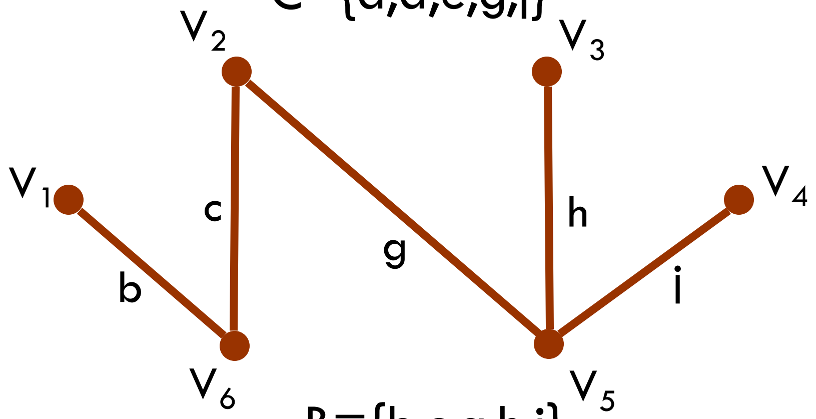
: **SPT 1**

B={b,c,f,h,i}
C={a,d,e,g,j}



: **SPT 2**

B={a,e,g,h,i}
C={b,c,d,f,j}



: **SPT 3**

B={b,c,g,h,j}
C={a,d,e,f,i}

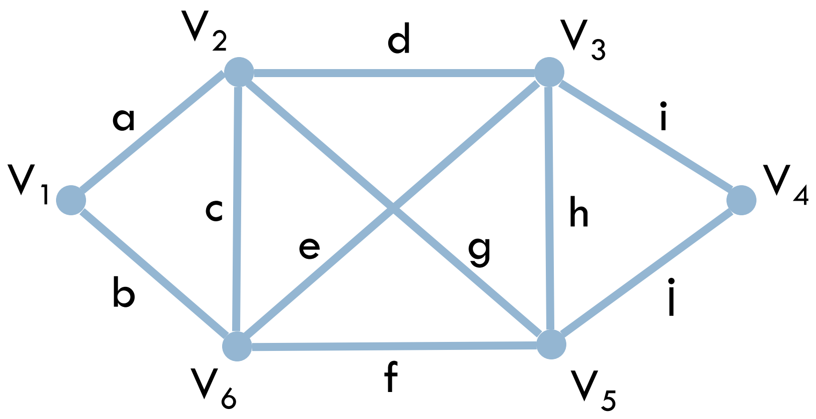
- Theorem 27: Every connected graph G has at least one spanning tree
- Proof:
- If the connected graph G has no circuits, then it is its own spanning tree
- If it has a cycle/circuit, delete one edge from the cycle such that the graph still remains connected
- Repeat the above step until there are no more cycles in the graph
- The final graph will hence contain all vertices of the graph and no cycles
- Hence it is a SPT

- Theorem 28: Any SPT of a connected graph with n vertices and e edges has (n-1) tree branches and (e-n+1) chords
- Proof:
- The connected graph has n vertices and e edges
- Any SPT of the graph will contain n vertices and (n-1) edges
- The remaining edges of the graph are chords, i.e. e-(n-1)= e-n+1

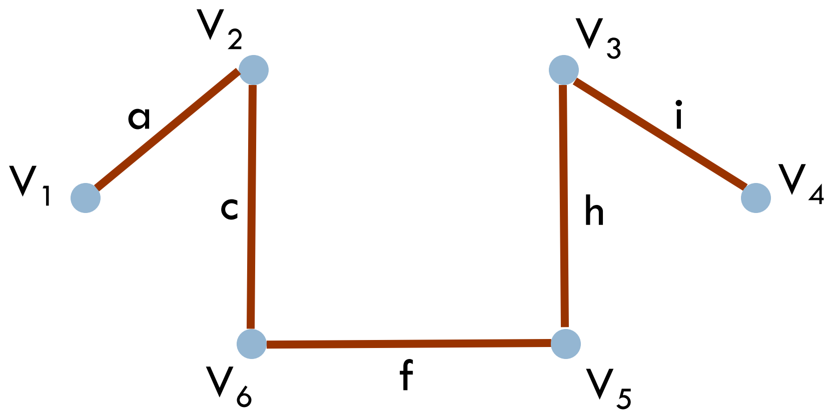
Rank & Nullity

- For a graph with n vertices, e edges and k components
  - Rank, $r = n-k$
  - Nullity, $\mu = e-n+k$
- If the graph is connected, then $k=1$; hence
  - Rank, $r = n-1$
  - Nullity, $\mu = e-n+1$
- <u>Note:</u>
  - r = no. of branches in a SPT of the graph
  - μ = no. of chords in the graph wrt the SPT

Examples



: **G**



: **SPT**

B={a,c,f,h,i}

C={b,d,e,g,j}

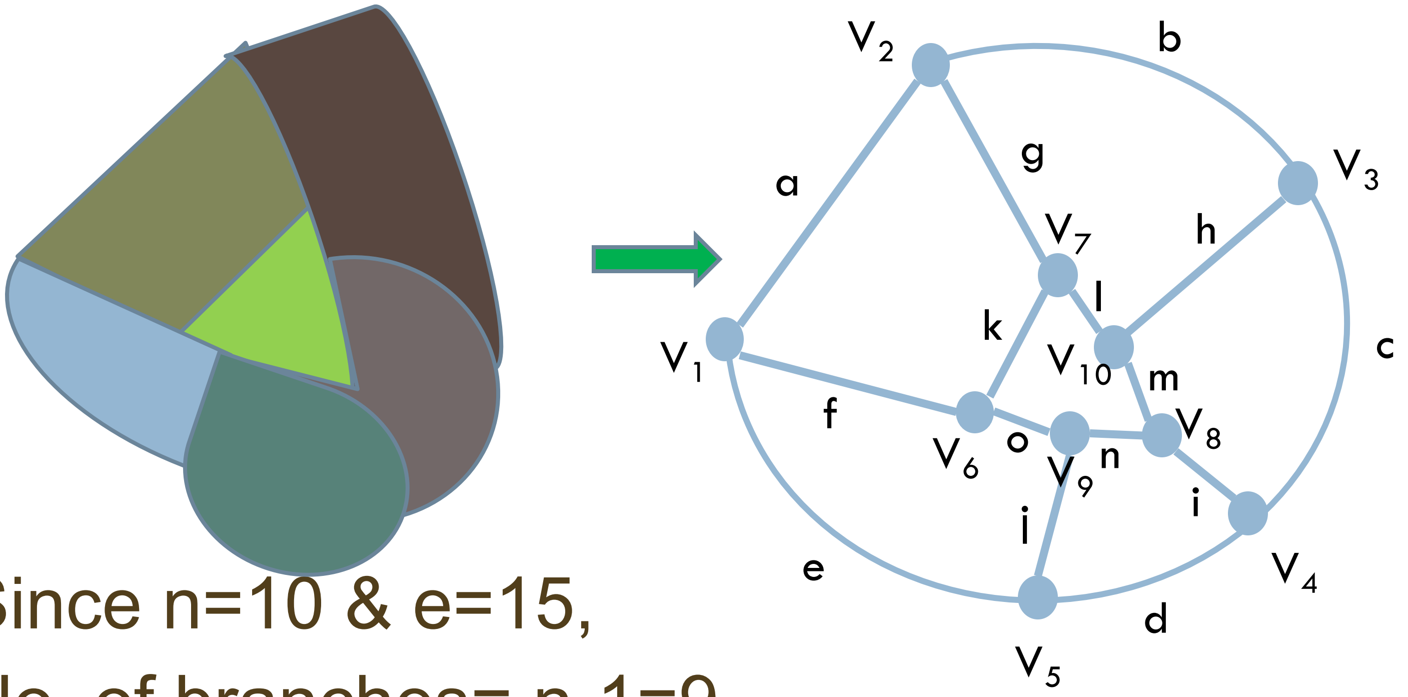
- Rank r = n-1 = 6-1 = 5
- Nullity, μ = e-n+1 = 10-6+1 = 5

Applications

- Real life problem: An electric network contains e elements and n nodes. What is the minimum no. of elements that must be removed so as to make the network circuit-less?
- Graph theoretic problem: Represent the network as a connected graph. Let the edges represent the elements and vertices the nodes. How many edges need to be removed so as make the graph circuit-less?

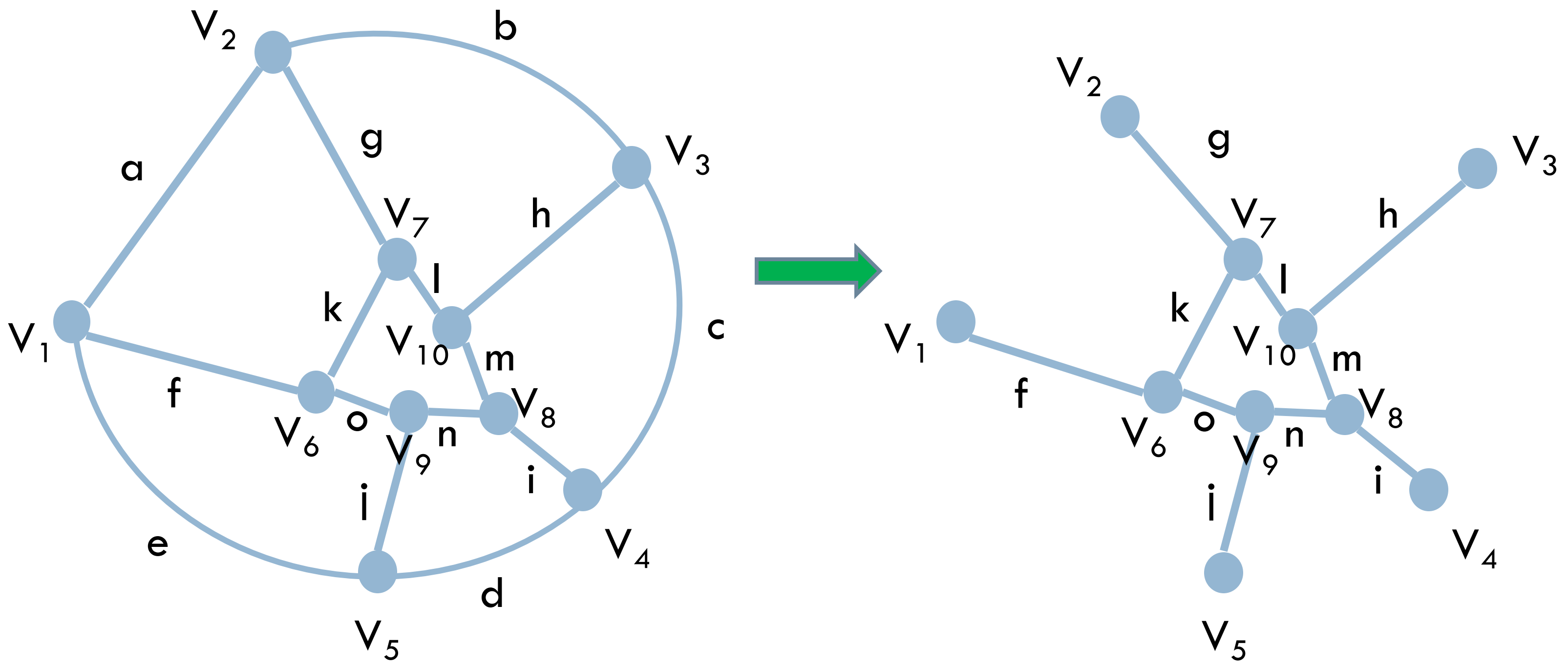
- Solution:
  - No. of branches in the SPT = n-1
  - No. of chords = e-n+1
  - Hence removal of e-n+1 elements can make the network circuit-less

- Real life problem: Consider a farm consisting of 6 walled plots of land and these plots are filled with water, then how many walls need to be broken so as to drain out the water?
- Graph theoretic problem: Represent the farm as connected graph with the walls as edges and corners as vertices. How many edges need to be removed so as make the graph circuit-less?



- Since n=10 & e=15,
- No. of branches= n-1=9
- No. of chords = e-n+1 =6

- Hence, removing 6 chords can make the graph a tree; then water can flow out easily



Properties of spanning trees

- A connected graph G may have any number of spanning trees
- The no. of vertices in any spanning tree is n; these are the vertices of the graph G
- The no. of edges in any spanning tree is n-1; these are some of the edges of the graph G
- The edges of a SPT are known as its branches
- The edges of the graph not included in a SPT are known as chords of the graph wrt the SPT
- Every SPT will contain (n-1) branches; but the set of (n-1) branches will be different for different SPTs
- Every SPT will leave (e-n+1) chords in the graph; the set of chords will be different for different SPTs

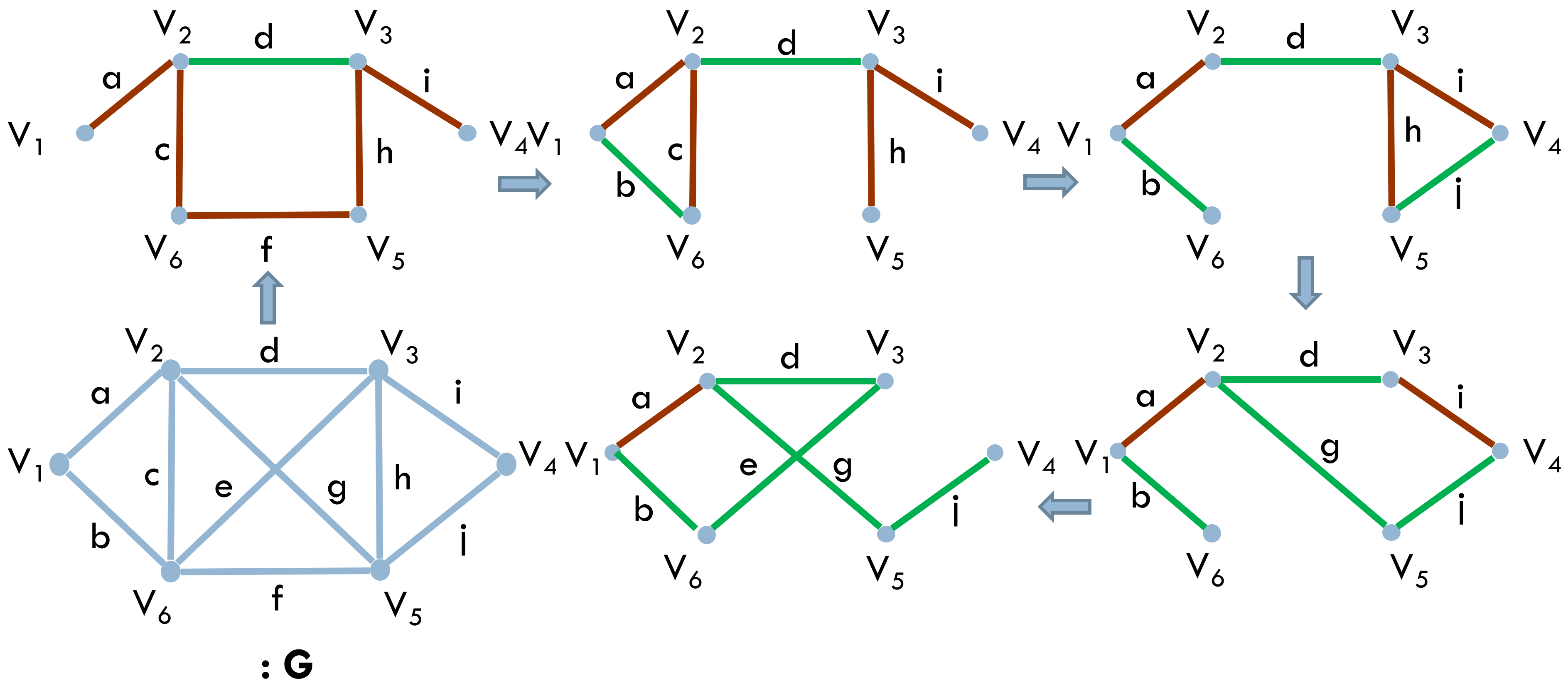
- Theorem 29: A connected graph is a tree iff adding an edge between any two vertices in G creates exactly one circuit
- Proof: suppose that the connected graph G is a tree
- Add an edge between any two vertices of the tree say v_i & v_j
- Since v_i & v_j are vertices of the tree, an edge between them creates a circuit, as there was already a path between v_i & v_j in the tree
- Since there could be only one path between every pair of vertices in a tree, adding an edge can create only one circuit

- Conversely: suppose that adding an edge between any two vertices of G creates exactly one circuit
- That means there was only path between every pair of vertices in G
- Which in turn implies that G was a tree

Finding all SPTs

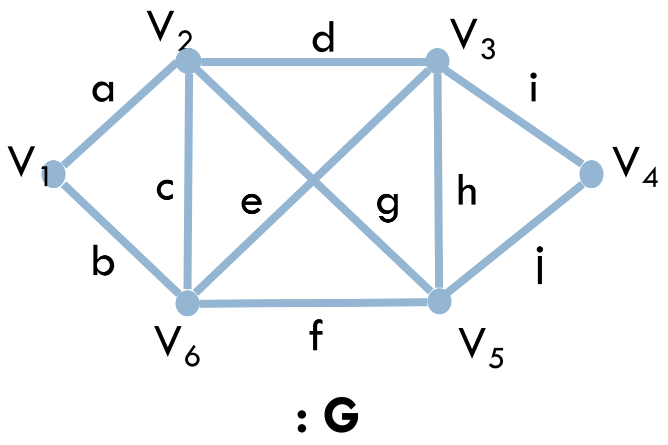
- From a given SPT of a graph, we can find the other SPTs of the same graph by the procedure of elementary tree transformation or cyclic interchange
  1) Start with a given SPT
  2) Add a chord to the SPT so that a fundamental circuit is formed
  3) Remove 1 branch from the circuit so formed. This will break the circuit and generate a new SPT
  4) Repeat steps 2 & 3 until all SPTs are obtained

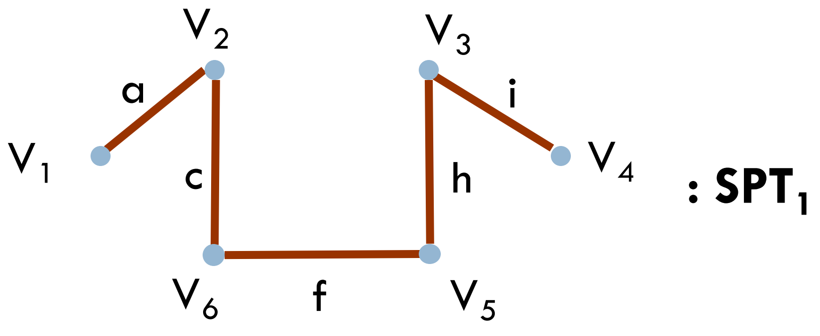
Example

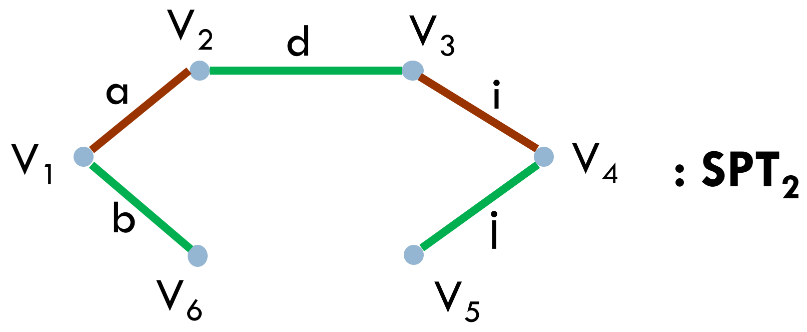


Distance b/w two SPTs

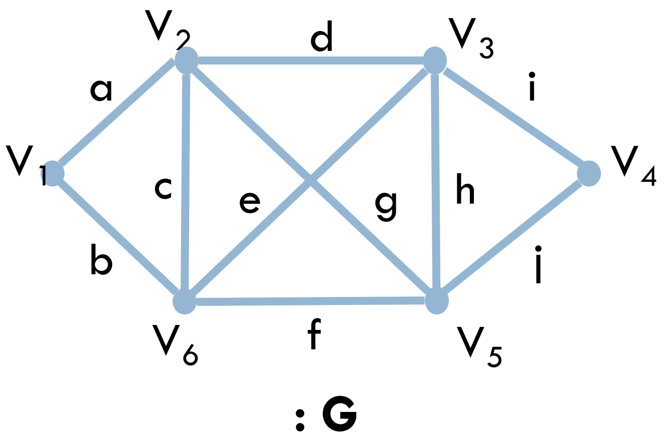
- Distance b/w two SPTs of a graph is the no. of branches in which they differ
- Distance b/w SPT_1 & SPT_2 = 3

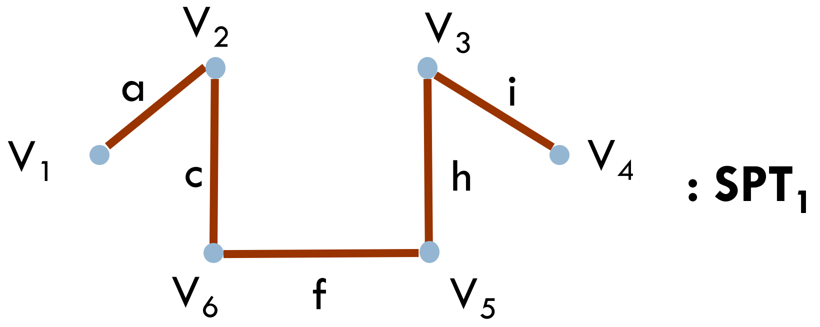


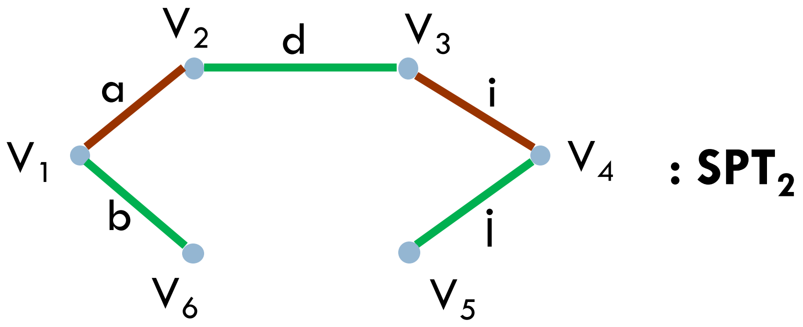




- Hence distance b/w two SPT_i & SPT_j is given by
  - $d(SPT_i , SPT_j)= ½ N(SPT_i \oplus SPT_j)$
  - Where N gives the no. of edges
  - Here, $N(SPT_1 \oplus SPT_2) = 6$
  - $d(SPT_1,SPT_2)=6/2=3$







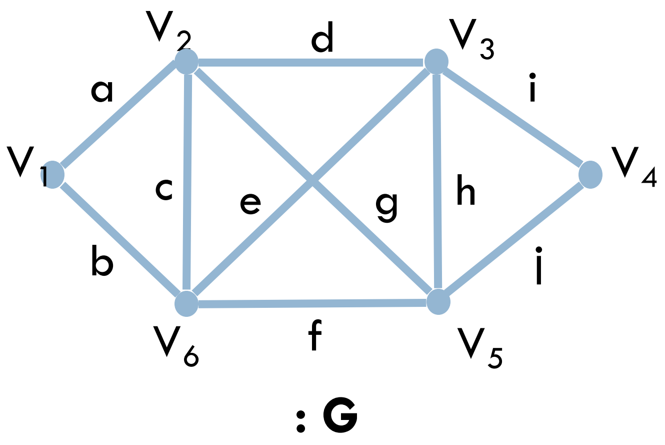
Maximum distance b/w two SPTs

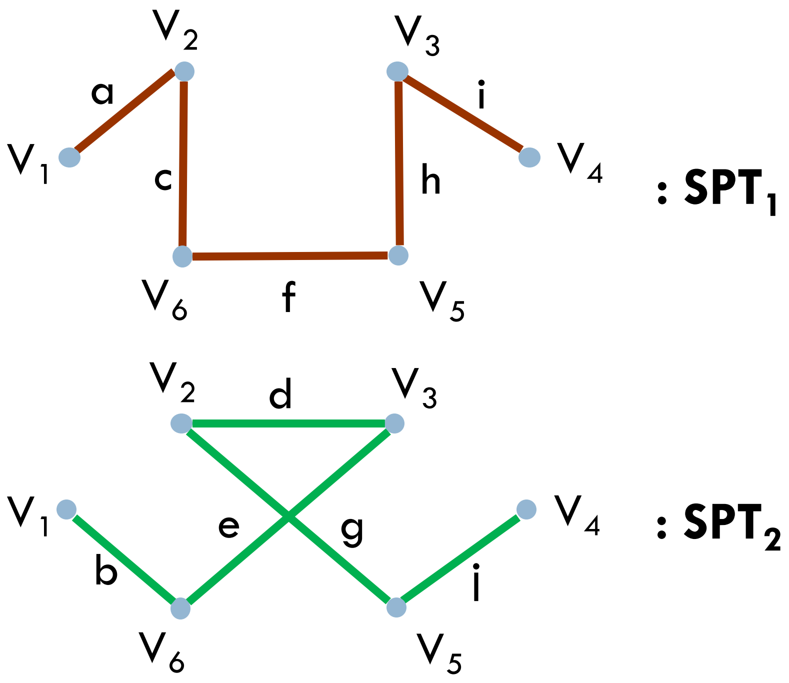
- Two SPTs are at maximum distance, when they are edge disjoint; i.e they have no edge in common
- Since maximum no. of edges in a SPT is (n-1), the maximum distance possible b/w any 2 SPTs trees of a graph = n-1
- But if there aren't enough chords left, the maximum distance possible b/w any 2 SPTs get limited to the no. of chords available

- For n vertices and e edges in G,
  - If e=2(n-1) (exactly n-1 chords available)
    - $d(SPT_i , SPT_j) = n-1 = r$
  - If e> 2(n-1) (more chords left)
    - $d(SPT_i, SPT_j) = n-1 = r$
  - If e<2(n-1) (not enough chords)
    - $d(SPT_i , SPT_j) = e-n+1 = \mu$
- Hence we can conclude
  - if $e >= 2(n-1) \rightarrow d(SPT_i , SPT_j) = r$
  - if $e < 2(n-1) \rightarrow d(SPT_i , SPT_j) = \mu$
- More precisely,
  - $d(SPT_i , SPT_j) = \min(r, \mu)$

Example : when r = μ

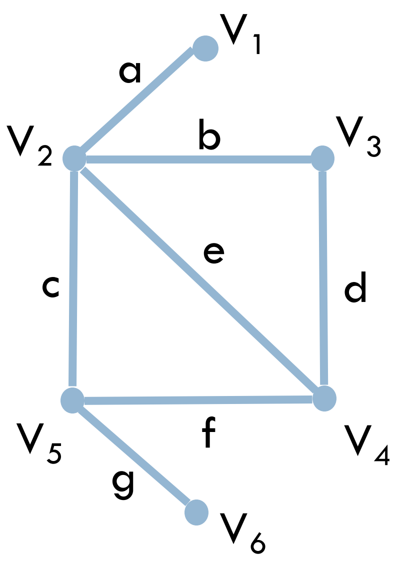
- Here n=6 & e=10; r = 5 & **μ** = 5
- Hence d(SPT_1,SPT_2)= 5



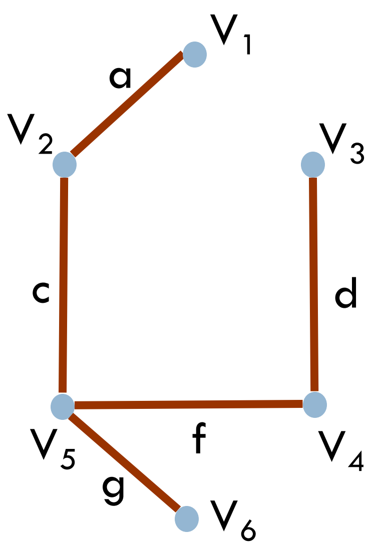


when r > µ

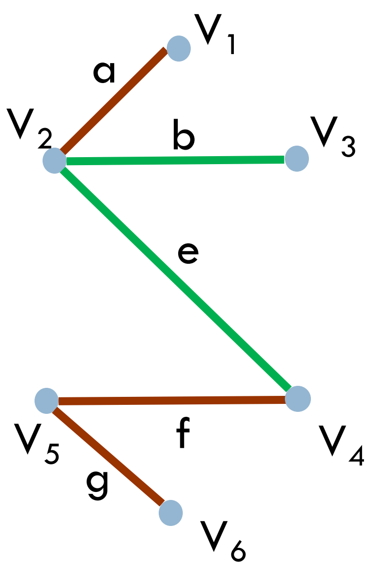
- Here n=6 & e=7; r = 5 & **µ** = 2
- d(SPT_1,SPT_2)= 2



: G



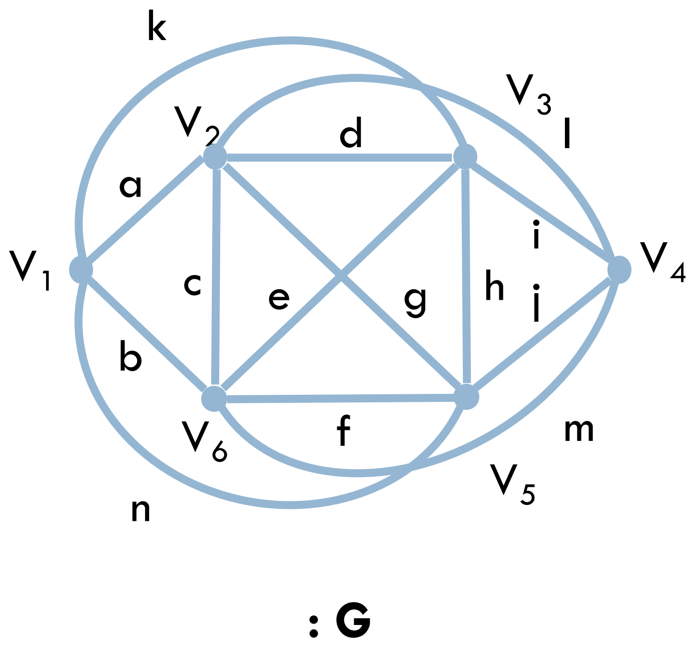
: SPT 1

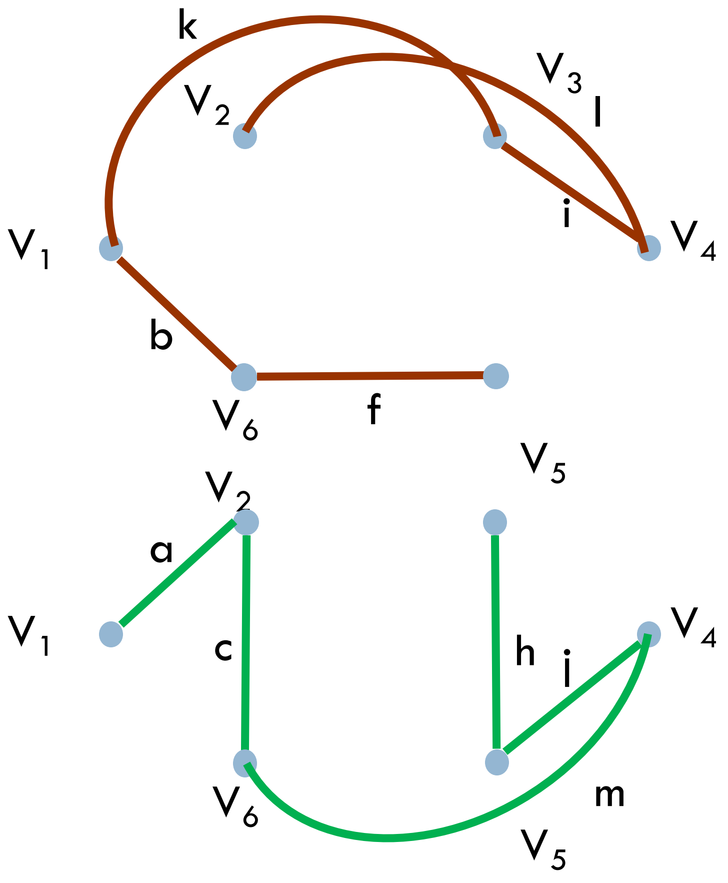


: SPT 2

when r < μ

- Here n=6 & e=14; r = 5 & μ = 9
- d(SPT_1,SPT_2)= 5



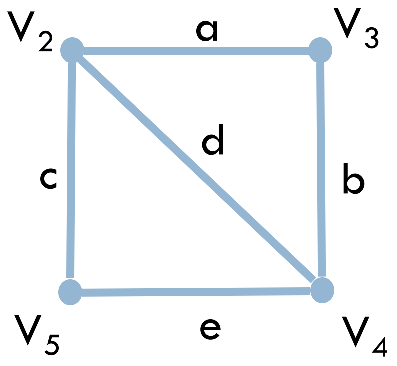


- Hence maximum distance b/w any two SPTs of a graph is either
  - n-1 (rank,r) or
  - e-n+1 (nullity, μ)

  Whichever is smaller
- max d(SPT_i , SPT_j)= min(r, μ)

Problem

- Find the max distance b/w the SPTs of the graph G

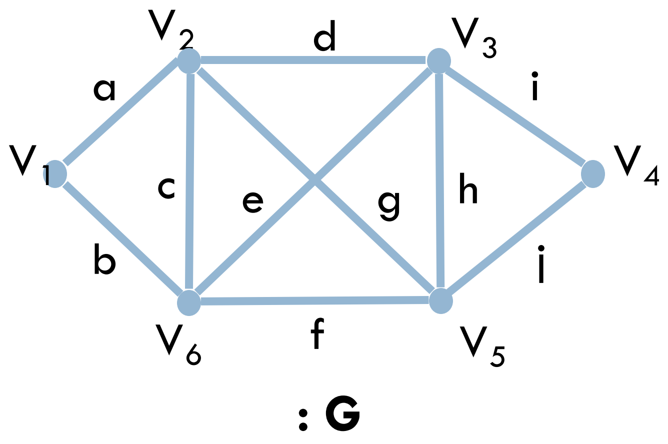


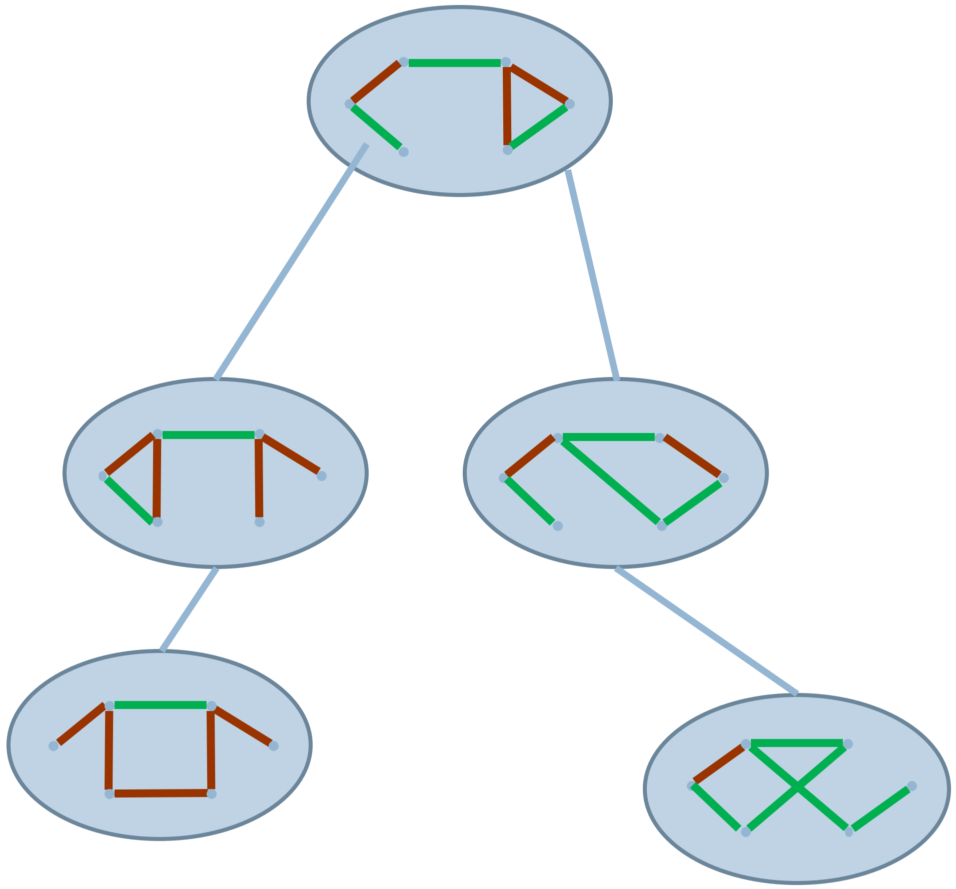
: G

Tree graph

- All SPTs of a graph are represented as vertices of a tree
- The cyclic interchange from one SPT to another is represented as an edge between them

Example





Central spanning tree

- The SPT of a graph which has minimum distance with all other spanning trees of the same graph
- Same concept as center of a tree

- Theorem 30: The distance between the spanning trees of a graph is a metric
- Proof:
- Distance b/w any two spanning trees is always positive or zero
  - $d(SPT_i , SPT_j) >= 0$
  - $d(SPT_i , SPT_j) = 0$ if $i=j$
- The number of branches by which SPT_i differs from SPT_j is the same as that SPT_j differs from SPT_i
  - $d(SPT_i , SPT_j) = d(SPT_j , SPT_i)$
- Triangular inequality
  - $d(SPT_i , SPT_j) <= d(SPT_i , SPT_k) + d(SPT_k , SPT_j)$

- Theorem 31: Starting from any one SPT, we can obtain every other SPT of G by successive cyclic interchanges
- Proof:

Shortest Spanning trees

- In the context of weighted graphs, where a numerical value (weight) is associated with each edge of the graph
- The shortest SPT of the graph is the one with minimum sum of weights
- Similar to lightest Ham circuit (travelling salesman problem)
- Also known as
  - Shortest distance SPT
  - Minimal SPT

Application

- Real life problem: Suppose if we need to construct roads to connect ‘n’ cities. Which are the cities that need to directly connected by roads so that construction cost can be minimized?
- Graph theoretic problem: Draw a complete graph with ‘n’ vertices. On each edge note down the construction cost. Find the minimum SPT of the graph. The branches of which represents the roads that are to be constructed

Finding the min SPT

- Many algorithms are available to find the minimum SPT of weighted graphs
  - Kruskal’s algorithm
  - Prim’s algorithm

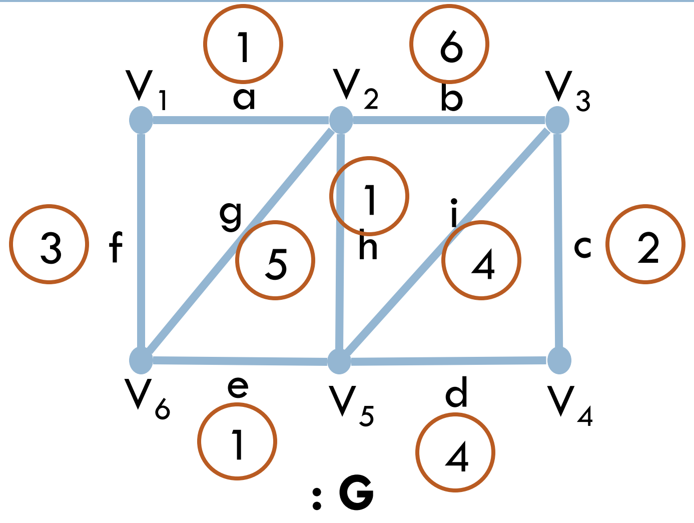
Kruskal’s algorithm

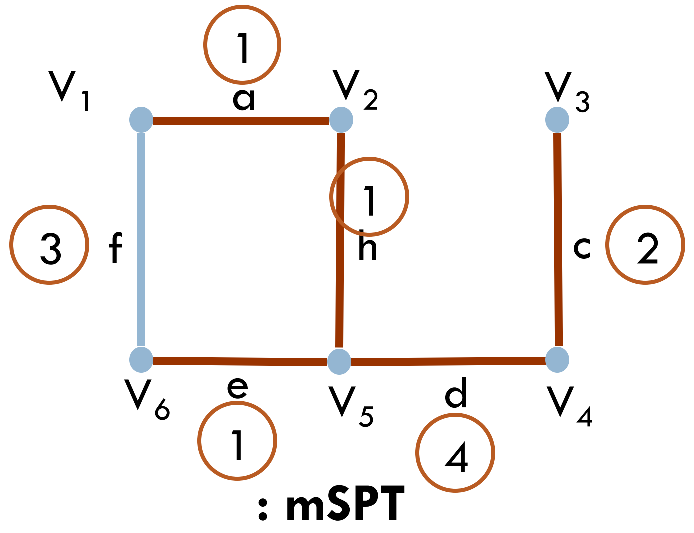
1) List all edges of the graph in the increasing order of their weights
2) Choose the edge with the smallest weight to be the first branch of the SPT
3) Choose the next smallest-weight edge such that it doesn’t make a circuit with the preciously selected edges
4) Continue the process until (n-1) edges have been chosen

Example

- Edge listing:

wt	edge	
1	a	✓
1	h	✓
1	e	✓
2	c	✓
3	f	✗
4	d	✓
4	i	
5	g	
6	b	





Weight of the mSPT=9

Prim’s algorithm

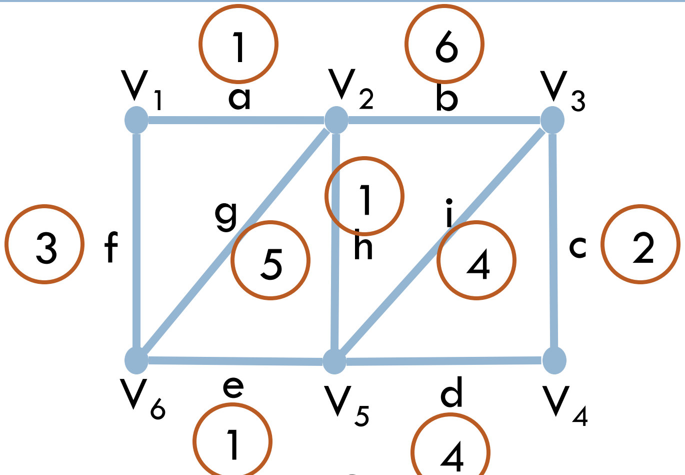
1) Draw ‘n’ isolated vertices and label them as v_1, v_2, v_3, …v_n
2) Tabulate the weights of the edges in an nxn matrix
3) Set the weights of non existent edges as ∞
4) Start from vertex v_1 and find its nearest neighbor (edge with minimum distance). Nearest neighbor is found by choosing the one which has least value in row_1 , say v_i. Draw the edge b/w v_1 and v_i in the null graph. Consider the edge as a subgraph

5) Now find the nearest neighbor of the subgraph. It is found by choosing the one with least value in rows 1 & i, say v_j. Add this edge to the subgraph.
6) Repeat step 3 until (n-1) edges have been chosen

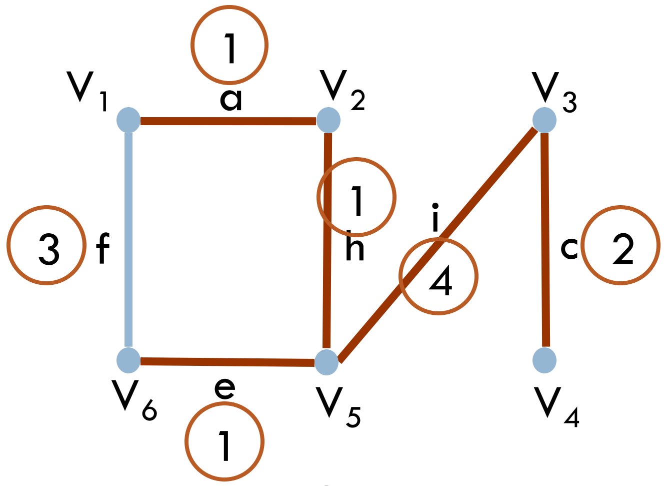
Example

	V_1	V_2	V_3	V_4	V_5	V_6
V_1	-	1 ✓	∞	∞	∞	3 ✗
V_2	1 ✓	-	6	∞	1 ✓	5
V_3	∞	6	-	2 ✓	4 ✓	∞
V_4	∞	∞	2 ✓	-	4	∞
V_5	∞	1 ✓	4 ✓	4	-	1 ✓
V_6	3 ✗	5	∞	∞	1 ✓	-

Weight of the mSPT= 1+1+1+4+2=9



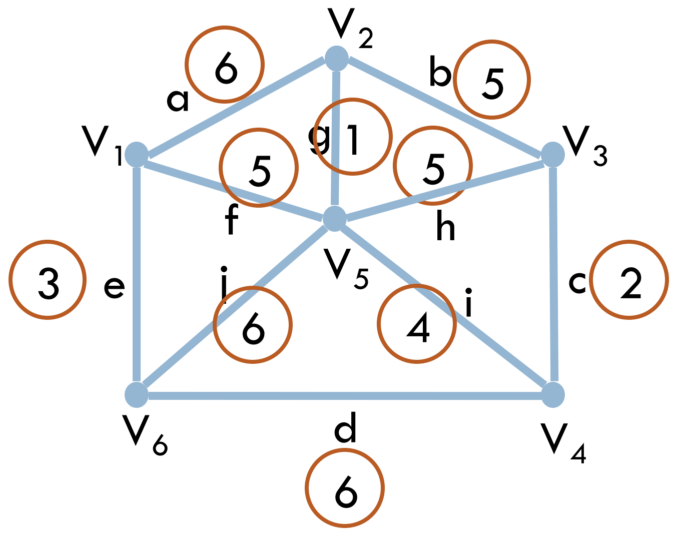
: G



: mSPT

Problem

- Find the mSPT for the given graph using
  - Kruskal's
  - Prim's



: G

Fundamental circuit

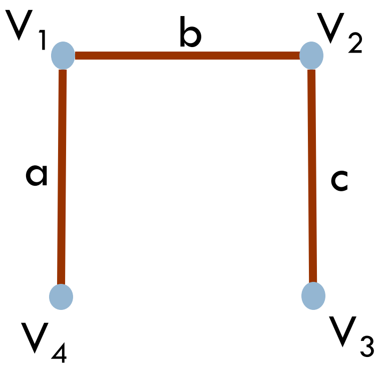
- W.r.t a given SPT, the circuit formed in the SPT by adding a chord is referred to as a Fundamental circuit
- Since each chord can generate 1 circuit, the no. of fundamental circuits possible for a graph is given by the no. of chords in the graph

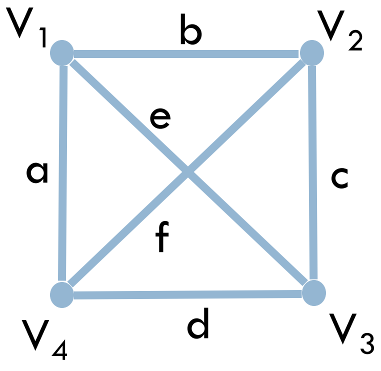
Example

: spt 1

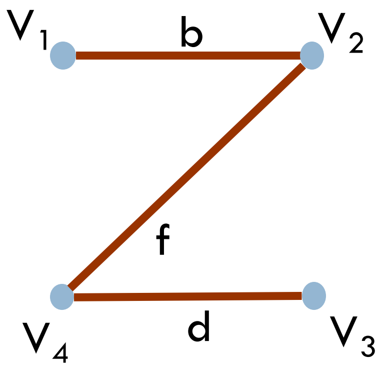
B={a,b,c}

C={d,e,f}





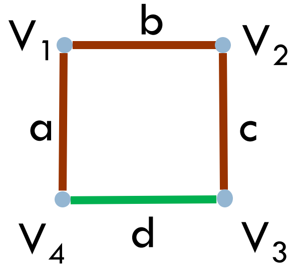
: G



: spt 2

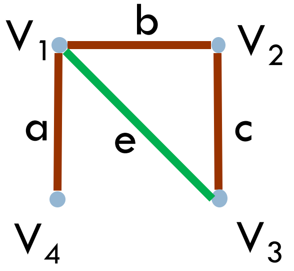
B={b,d,f}

C={a,c,e}



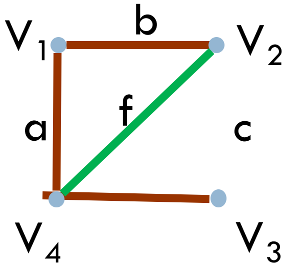
: FC 1

(adding chord d)



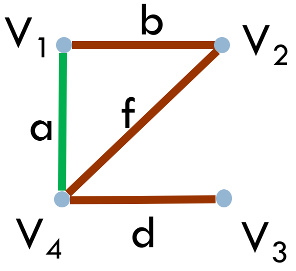
: FC 2

(adding chord e)



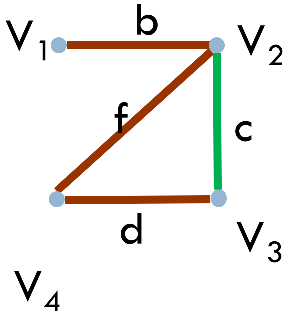
: FC 3

(adding chord f)



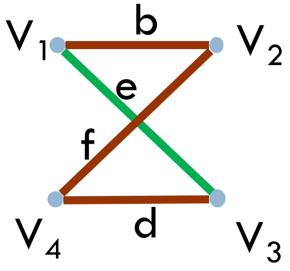
: FC 1

(adding chord a)



: FC 2

(adding chord c)

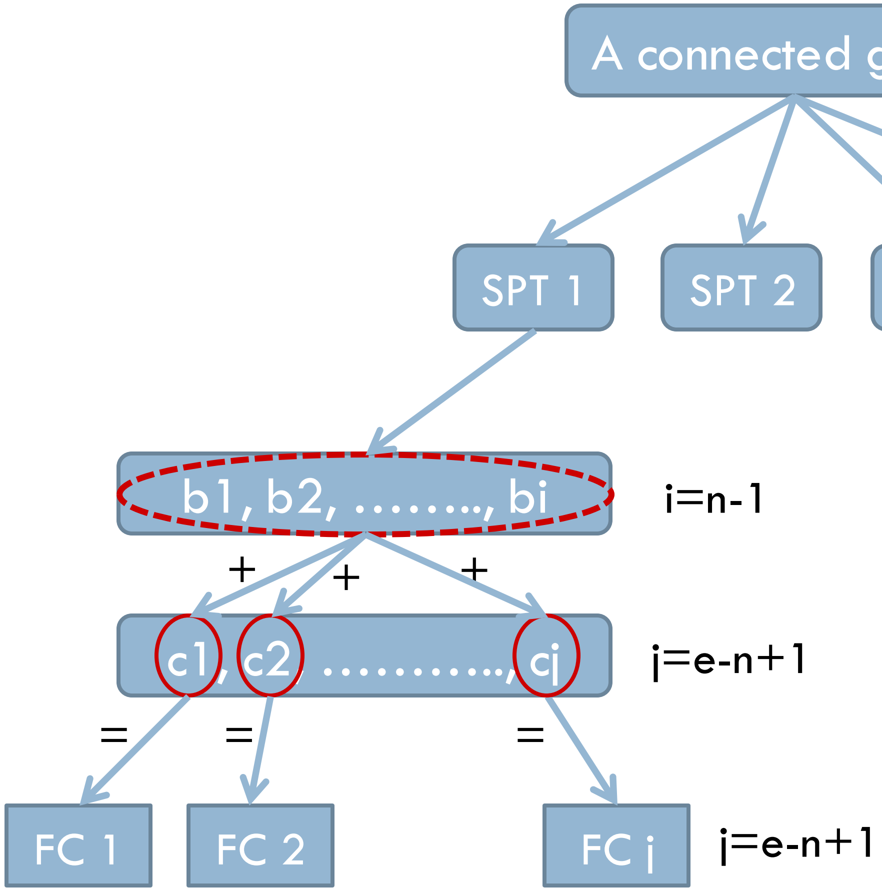


: FC 3

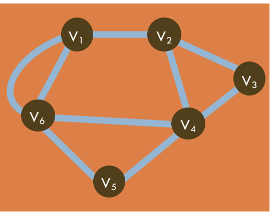
(adding chord e)

Sum up

- A connected graph will have many SPTs
- For each SPT, some of the (n-1) edges form the branches and remaining e-n+1 edges form the chords
- A fundamental circuit is always mentioned wrt a SPT
- A fundamental circuit contains 1 chord along with all/some branches of the SPT
- Each chord produces one fundamental circuit
- Hence no. of fundamental circuits possible for a SPT is given by the no. of chords
- A circuit that contains 2 chords is not a fundamental circuit


A connected graph
SPT 1
SPT 2
...
SPT k
b1, b2, ..., bi
i=n-1
+
+
+
c1, c2, ..., cj
j=e-n+1
=
=
=
FC 1
FC 2
FC j
j=e-n+1

End of module 3



Module 4

Graph connectivity & Planar graphs

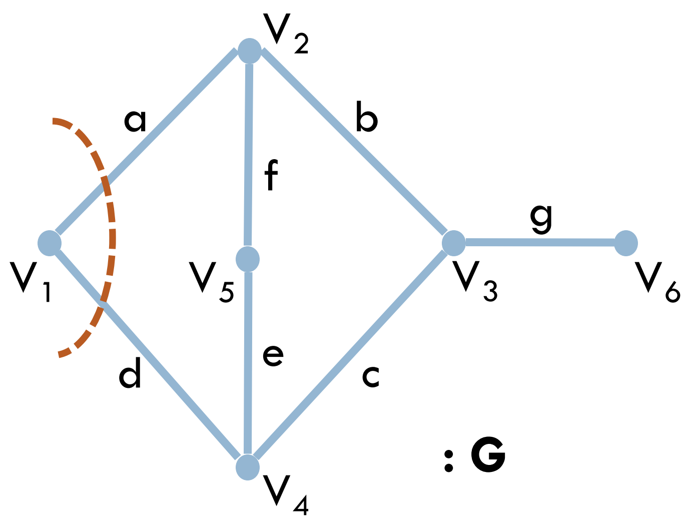
Contents

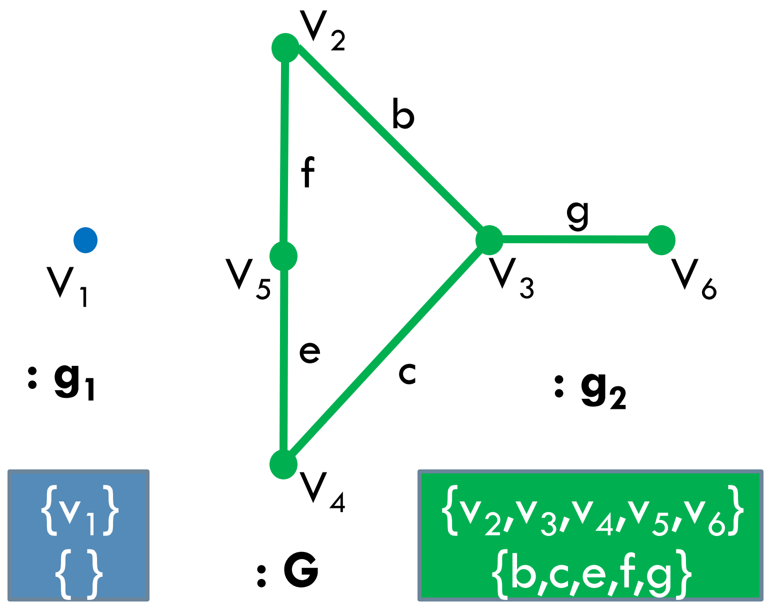
Cut-sets

- In a connected graph G, a cut-set is a set of edges removal of which leaves the graph disconnected, provided removal of no proper subset of the set disconnects G
- A cut-set cuts the graph into two components such that no path exists between the two
- It is the minimal set of edges removal of which reduces the rank of the graph by one
- A cut-set is also known as
  - Minimal cut-set
  - Proper cut-set
  - Co-cycle

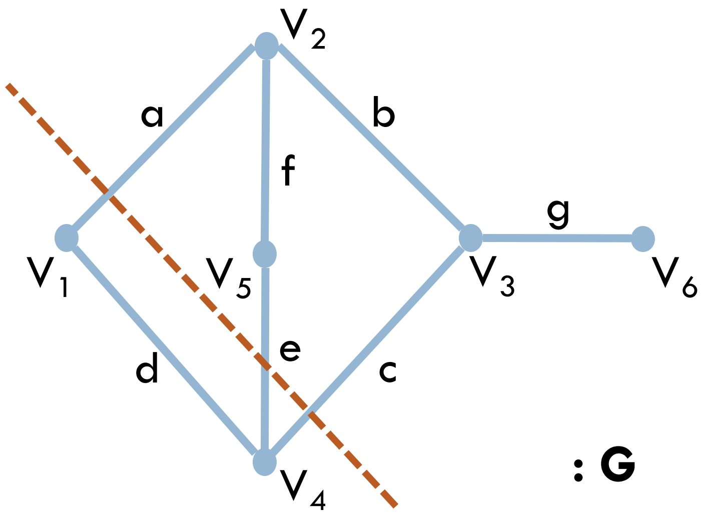
Example

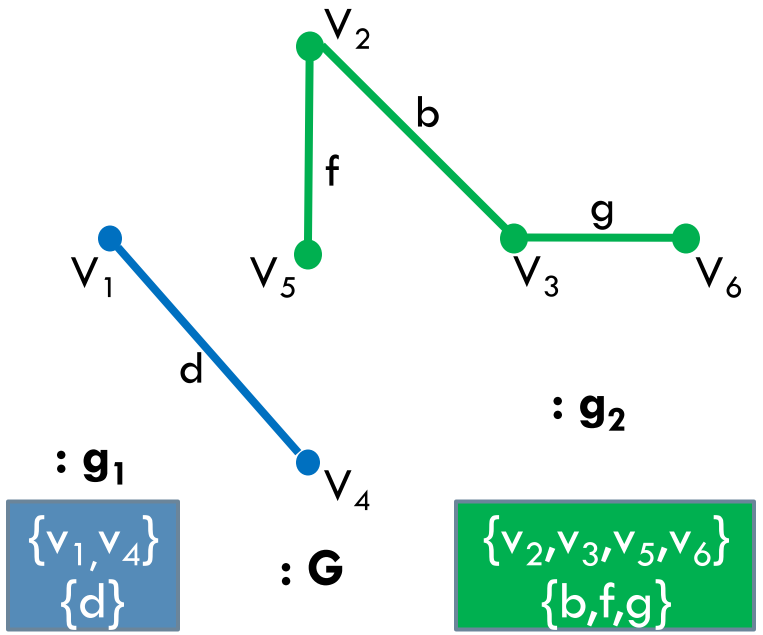
- In graph G, {a,d} is a cut-set; cs_1= {a,d}
- Hence G-cs_1 = 2 sub-graphs of G, g_1 & g_2
- In graph G, n=6, k=1→rank = n – k = 5
- After removing the cut-set, n=6,k=2 → rank = n – k = 4
- Hence removal of a cut-set reduces the rank of the graph by one





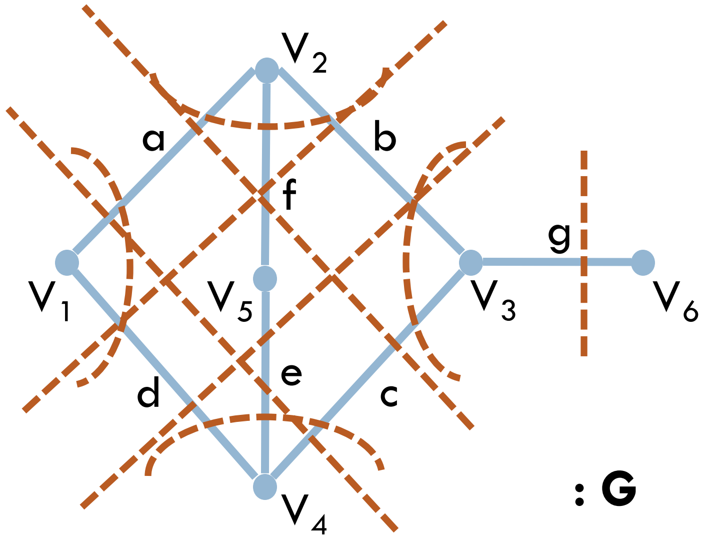
- In graph G, {a,e,c} is another cut-set
- cs_2= {a,e,c}
- Hence G-cs_2 = 2 sub-graphs of G, g_1 & g_2





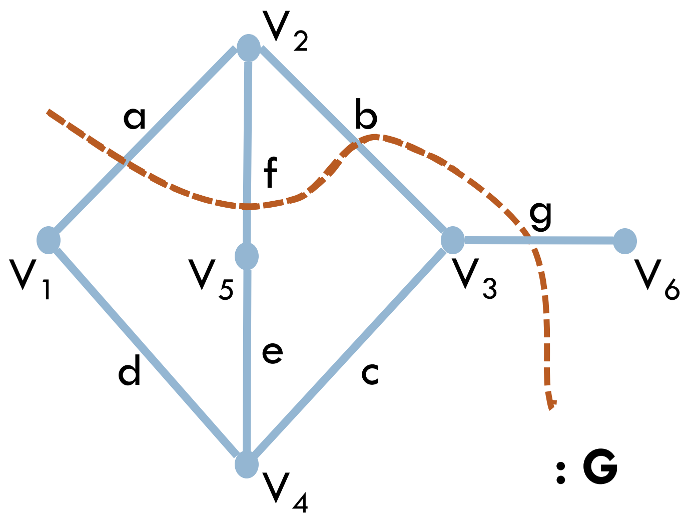
List all cut-sets

- $cs_1 = \{a,d\}$
- $cs_2 = \{a,b,f\}$
- $cs_3 = \{b,c\}$
- $cs_4 = \{d,e,c\}$
- $cs_5 = \{a,e,c\}$
- $cs_6 = \{a,f,c\}$
- $cs_7 = \{b,f,d\}$
- $cs_8 = \{b,e,d\}$
- $cs_9 = \{g\}$



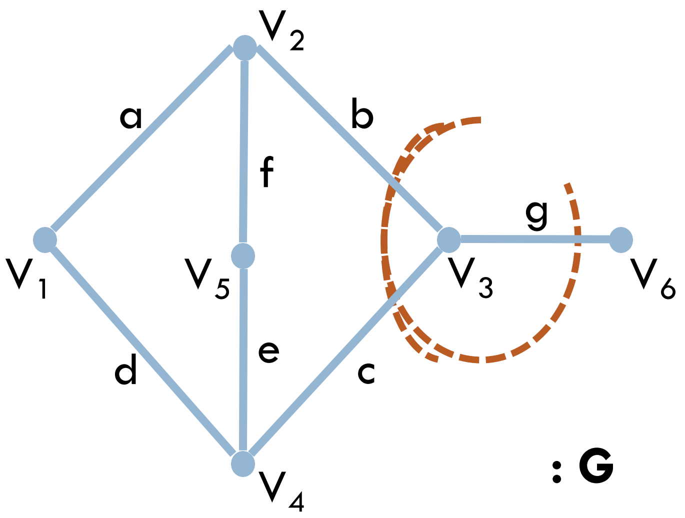
Wrong cut-sets

- Is {a,f,b,g} a cut-set of G ?
- No, coz removal of {a,f,b,g} cuts the graph into three
- Also the proper subset (a,f,b} of {a,f,b,g} is itself a cut-set
- Subset of a cut-set cannot be a cut-set



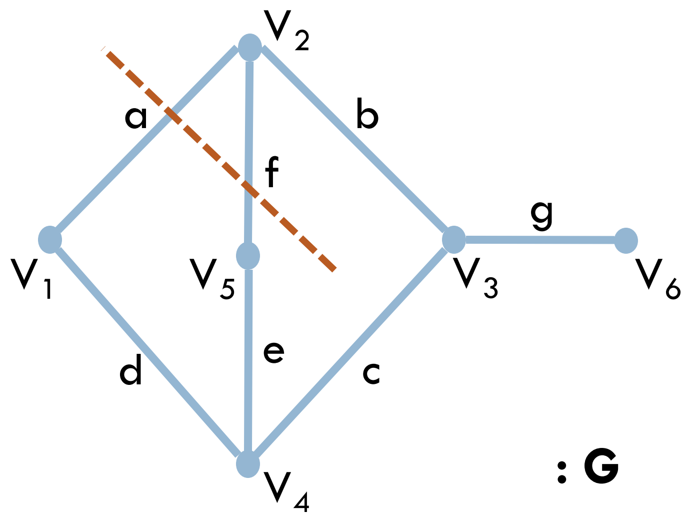
Wrong cut-sets

- Is {b,c,g} a cut-set of G ?
- No, coz it cuts the graph into three
- Moreover, subset {b,c} itself is a cut-set



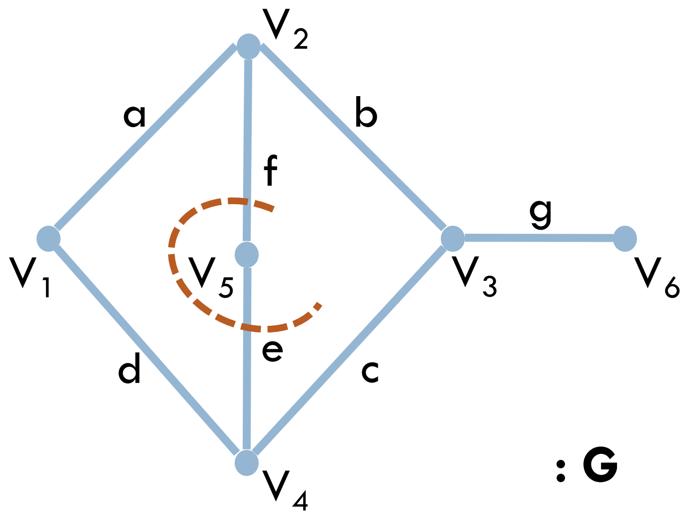
Wrong cut-sets

- Is {a,f} a cut-set of G ?
- No, coz removal of {a,f} does not cut the graph into two



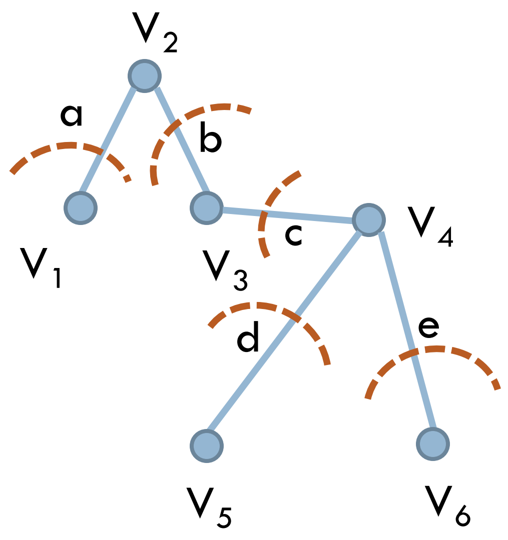
Right cut-sets

- Is {e,f} a cut-set of G?
- Yes, coz {e,f} cuts the graph into two & none of the proper subsets of {e,f} is a cut-set



Cut-set in a tree

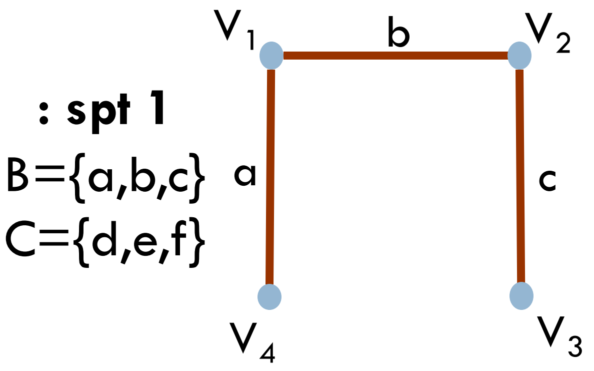
- Since removal of any edge in a tree breaks the tree into two, every edge of a tree is a cut-set
- Cut-sets→ {a} {b} {c} {d} {e}



Fundamental cut-set

- W.r.t a given SPT, a cut-set of the graph is said to be fundamental, if it contains exactly one branch of the SPT along with some/all of the chords
- Since each branch can generate 1 cut-set, the no. of fundamental cut-sets possible for a graph is given by the no. of branches in the spanning tree

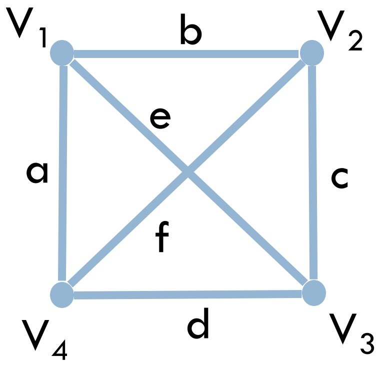
Example



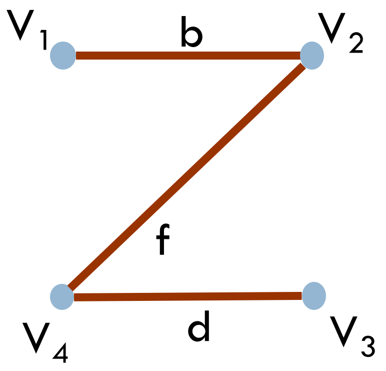
: spt 1

$B=\{a,b,c\}$

$C=\{d,e,f\}$



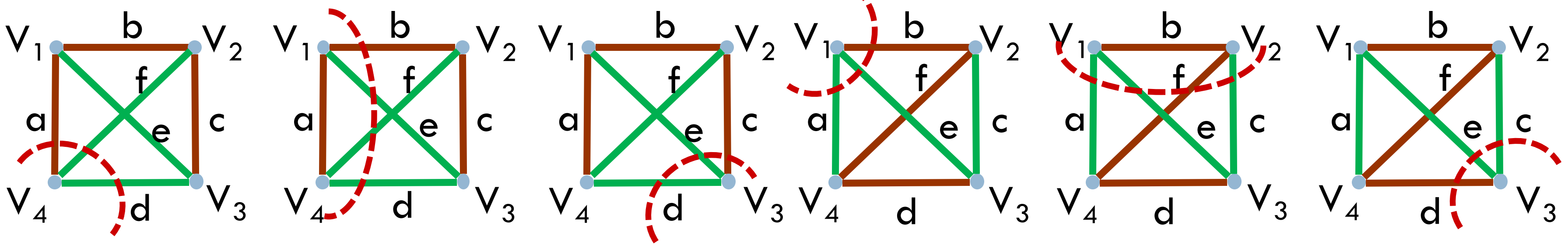
: G



: spt 2

$B=\{b,d,f\}$

$C=\{a,c,e\}$



:FCS1{a,e,d} (using branch a) :FCS2{b,f,e,d} (using branch b) :FCS3{c,e,d} (using branch c) :FCS1{b,e,a} (using branch b) :FCS2{a,f,e,c} (using branch f) :FCS3{c,e,d} (using branch d)

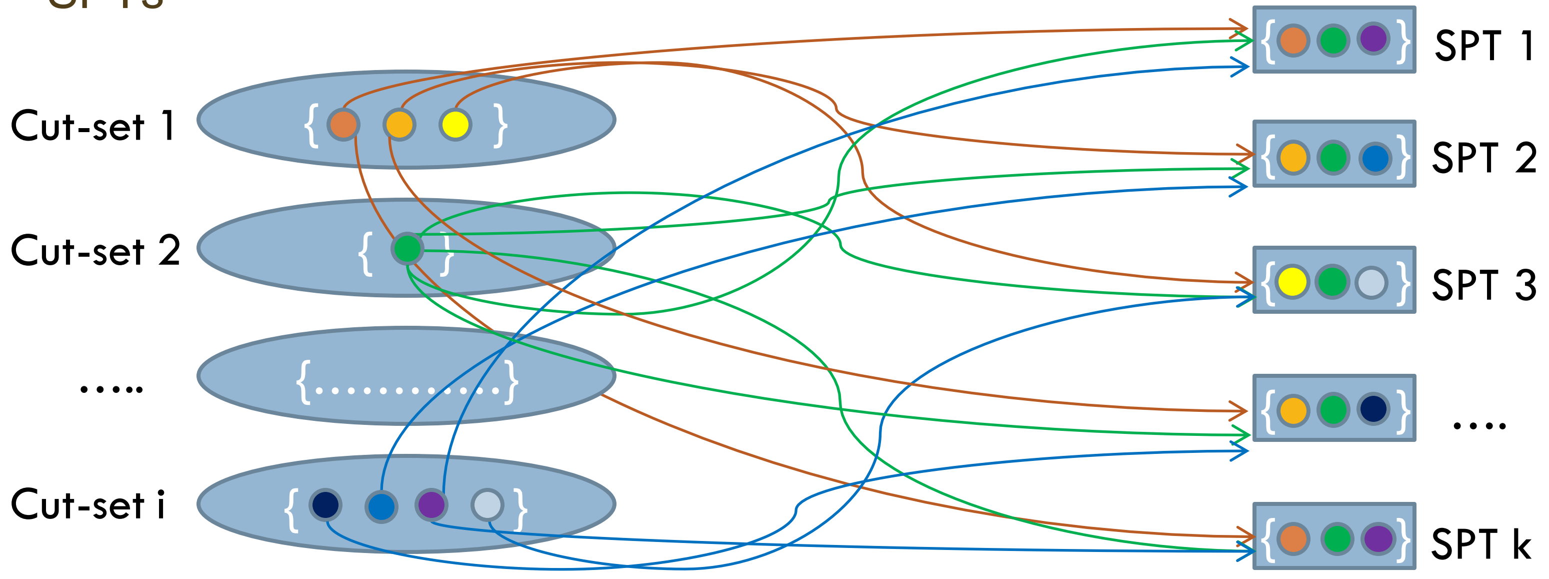
Sum up

- A connected graph will have many SPTs
- For each SPT, some of the (n-1) edges form the branches and remaining e-n+1 edges form the chords
- A fundamental cut-set is always mentioned w.r.t a SPT
- A fundamental cut-set contains 1 branch along with all/some chords of the SPT
- Each branch produces one fundamental cut-set
- Hence no. of fundamental cut-sets possible for a graph is given by the no. of branches in its SPT
- A cut-set that contains 2 branches is not a fundamental cut-set

- Theorem 32: Every cut-set in a connected graph G must contain at least one branch of every spanning tree of G
- Proof: Let G be a connected graph and S be a cut-set of G
- Assume that we have a SPT T that does not have any of its branches in S
- Then removing S from G does not remove any of the branches from G
- Since the spanning tree remains completely in the graph and any SPT shall contain all the vertices of the graph, removal of S still leaves the graph connected
- But it is not possible. Removal of any cut-set must leave the graph disconnected
- Hence our assumption cannot be true
- There can be no SPT without any of its branches in any cut-set of G
- Hence the theorem

Note

- Every cut-set must contain at least one branch of every SPT
- However if there are k SPTs, it is not necessary that every cut-set must contain k elements; an edge may be common to many SPTs
- If a cut-set has a single edge then that edge has to be a branch in all the SPTs



- Theorem 33: In a connected graph G, any minimal set of edges containing at least one branch of every SPT of G is a cut-set
- Proof: In a connected graph G, let Q be a minimal set of edges containing at least one branch of every SPT of G
- Remove Q from G. The remaining graph will not contain any of the SPTs. That means now the graph is disconnected
- Also, since Q is the minimal set of edges containing branches from all SPTs, returning any one edge to G-Q will create at least one SPT thereby making the graph connected as well
- Then we can say that Q is the minimal set of edges removal of which disconnects G, which is indeed the definition of a cut-set
- Hence Q is a cut-set

- Theorem 34: Every circuit has an even no. of edges in common with any cut-set
- Proof: Consider a cut-set S in graph G. let the removal of S partition the vertices of G into two disjoint subsets V_1 and V_2.
- Consider a circuit ρ in G(before the removal of S). If all the vertices ofρ lies entirely within V_1 or entirely within V_2, then S will have no edge in common with ρ i.e, zero no. of edges in common (even)
- Whereas if some of the vertices of ρ lies in V_1 and some in V_2, then in order to traverse the circuit we need to go back and forth between V_1 and V_2 and finally need to reach back at the starting point
- Hence the no. of edges we traverse be tween V_1 and V_2 must be even. And these edges could be only from S. Therefore no. of edges common to S and ρ is even

- Theorem 35: The ringsum of any two cut-sets in a graph is either a third cut-set or an edge disjoint union of cut-sets
- Proof: Let S_1 be a cut-set of the graph that partitions the vertex set V into V_1 and V_2
- Let S_2 be another cut-set of the graph that partitions the V into V_3 and V_4
- Clearly, $V_1 \cup V_2 = V$ and $V_1 \cap V_2 = \emptyset$

  $V_3 \cup V_4 = V$ and $V_3 \cap V_4 = \emptyset$
- Now consider the subset$(V_1 \cap V_4) \cup (V_2 \cap V_3)$ as V_5 which is in fact $V_2 \oplus V_3$; similarly consider subset$(V_1 \cap V_3) \cup (V_2 \cap V_4)$ as V_6 which is same as $V_2 \oplus V_3$
- Now $S_1 \oplus S_2$ seem to contain only those edges between V_5 and V_6. Also there are no other edges between V_5 & V_6 which implies $V_5 \cup V_6 = V$ and $V_5 \cap V_6 = \emptyset$
- Then $S_1 \oplus S_2$ is a cut-set of G if V_5 and V_6 each remain connected after the removal of $S_1 \oplus S_2$; otherwise $S_1 \oplus S_2$ is the union of cut-sets

Example

- Eg 1: cut-sets S_1= {d,e,f} & S_2={f,i,h}

  $S_1 \oplus S_2 = (S_1 \cup S_2) - (S_1 \cap S_2)$

  = {d,e,f,i,h} – { f }

  = {d,e,i,h}

  → again a cut-set

- Eg 2: cut-sets S_1= {a,b} & S_2={b,c,e,f}

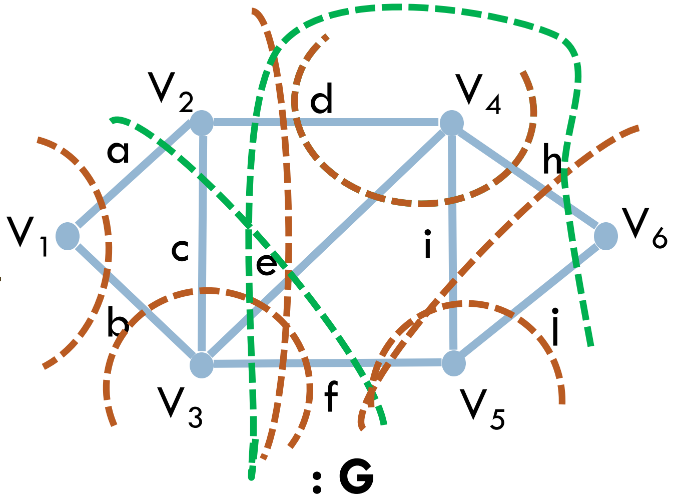
  $S_1 \oplus S_2$ = {a,c,e,f}

  → again another cut-set

- Eg 3: cut-sets S_1= {d,e,i,h} & S_2={f,i,j,}

  $S_1 \oplus S_2$ = {d,e,f,h,j}

  → union of two cut-sets {d,e,f} and {h,j}



- Theorem 36: W.r.t a given SPT T, a chord c_i that determines a fundamental circuit ρ occurs in every fundamental cut-set associated with the branches in ρ and in no other
- Proof: T is the given SPT
- Let ρ be the fundamental circuit determined by the chord c_i
- ρ in = $\{c_i , b_1, b_2, \ldots, b_k \}$
- Let S_1 be the fundamental cut-set associated with branch b_1
- $S_1=\{b_1,c_1, c_2, \ldots.c_q\}$
- Since the number of edges common to ρ and S_1 must be even, c_i must be in S_1
- The same is true for fundamental cut-sets made by branches $b_2,b_3,\ldots b_k$
- On the other hand suppose that c_i occurs in some fundamental cut-set S_{k+1} made by a branch other than $b_1,b_2,\ldots.b_k$. Since none of the branches is in S_{k+1}, there is only 1 edge - c_i - common to S_{k+1} and the fundamental circuit ρ which is not possible
- Hence the theorem

- **Theorem 37:** With respect to a given spanning tree T, a branch bi that determined a fundamental cut-set S is contained in every fundamental circuit associated with the chord in S, and in no others
- **Proof:** T is the given spanning tree
- Let S be the fundamental cut-set determined by the branch b_i

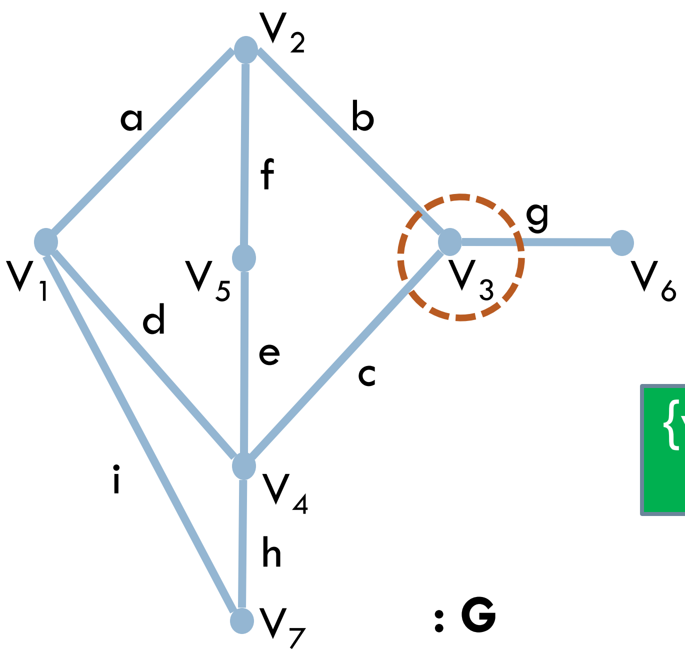
  $S = \{b_i, C_1, C_2, \ldots\ldots.C_q\}$
- Let $\boldsymbol{\rho}_1$ be the fundamental circuit determined by the chord C_1
- $\boldsymbol{\rho}_1 = \{C_1, b_1, b_2 \ldots..b_K\}$
- Since the no. of edges must be S and $\boldsymbol{\rho}_1$ must be even, b_i must be in $\boldsymbol{\rho}_1$.
- The same is true for the fundamental circuits made by chords C_2, $C_3 \ldots C_q$
- On the other hand, suppose that b_i occurs in some fundamental circuit $\boldsymbol{\rho}_{q+1}$ made by a chord other than $C_1, C_2, \ldots.. C_q$. Since none of the chords C_1, C_2, ….. C_q is in $\boldsymbol{\rho}_{q+1}$, there is only 1 edge b_i common to a circuit $\boldsymbol{\rho}_{q+1}$ & cut-set S, which is not possible
- Hence the theorem

Cut-vertices

- In a connected graph G, a cut-vertex is a set of vertices removal of which leaves the graph disconnected, provided removal of no proper subset of the set disconnects G
- A cut-vertex cuts the graph into two or more components, such that no path exists between the components

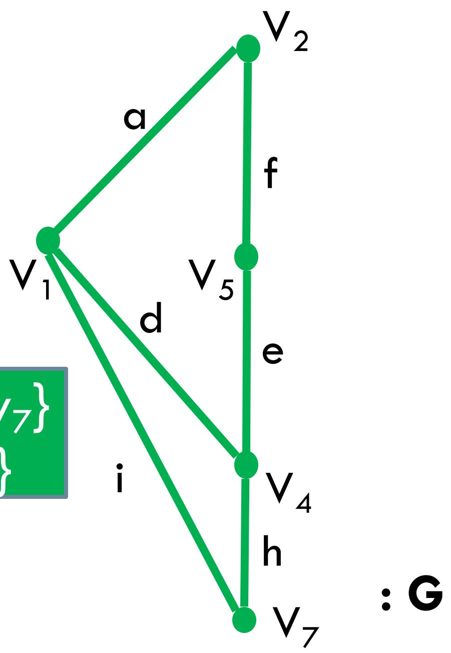
Example

- In graph G, $\{V_3\}$ is a cut-vertex; $cv_1 = \{V_3\}$
- Hence $G\text{-}cv_1$ = 2 sub-graphs of G, g_1 & g_2



: g_1

$\{v_1, v_2, v_4, v_5, v_7\}$
$\{a, d, e, f, h, i\}$

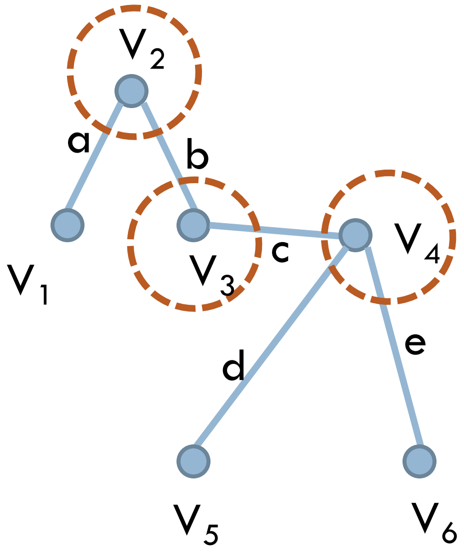


: g_2

$\{v_6\}$
$\{\ \}$

Cut-vertex in a tree

- Since removal of any vertex other than the pendant vertices breaks the tree, every vertex of a tree is a cut-vertex
- Cut-vertices$\rightarrow$ $\{V_2\}$ $\{V_3\}$ $\{V_4\}$

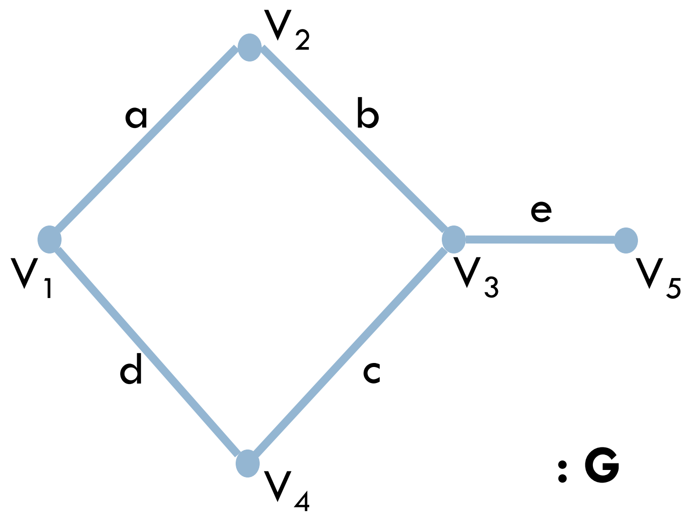


Edge connectivity (E_c)

- Minimum no. of edges removal of which disconnects the graph or reduces the rank of the graph by one
- It is given by the size of the smallest cut-set

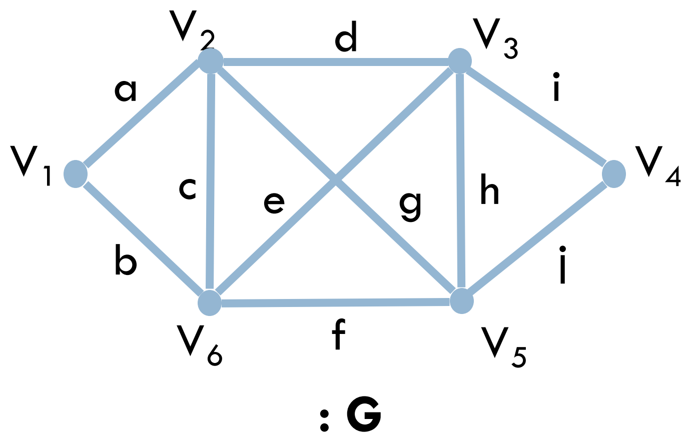
Example

- Cut-sets are
  - {a,d}
  - {a,b}
  - {a,c}
  - {b,c}
  - {b,d}
  - {c,d}
  - {e}
- Smallest cut-set is {e} → contains one element
- Hence edge connectivity E_c of graph G is 1



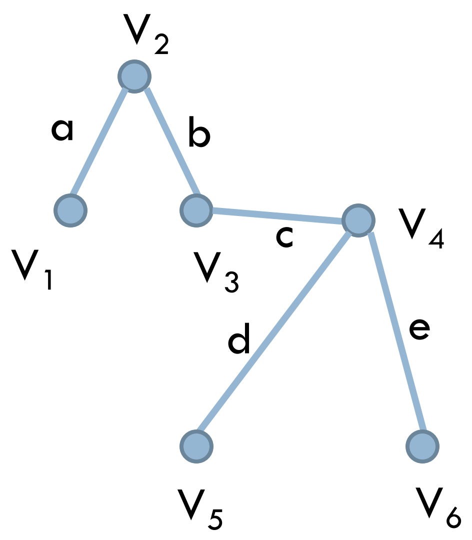
Problem

- Find the edge connectivity E_c of the graph given



Edge connectivity of a tree

- Since a tree can be broken by the removal of a single edge, edge connectivity of a tree is always 1
- Cut-sets of the tree are
- {a} , {b} , {c} , {d} , {e}
- Hence E_c is 1

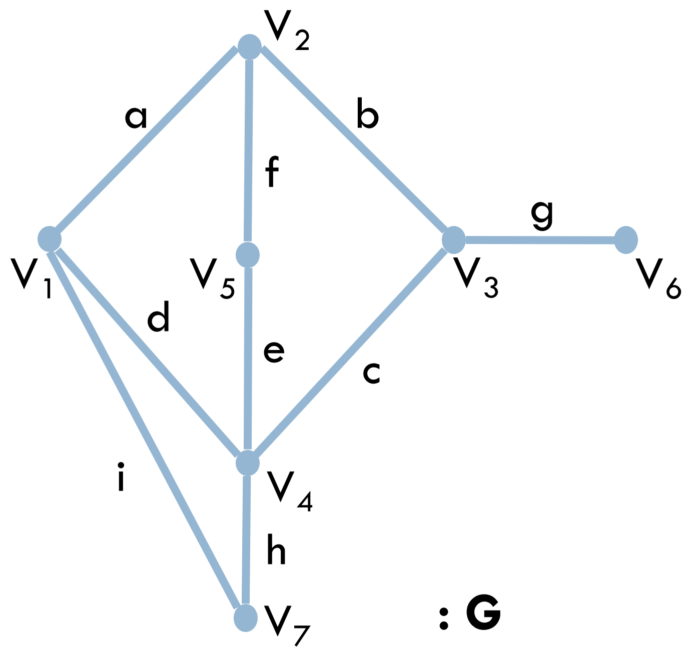


Vertex connectivity (V_c)

- Minimum no. of vertices removal of which disconnects the graph
- It is given by the size of the smallest cut-vertex

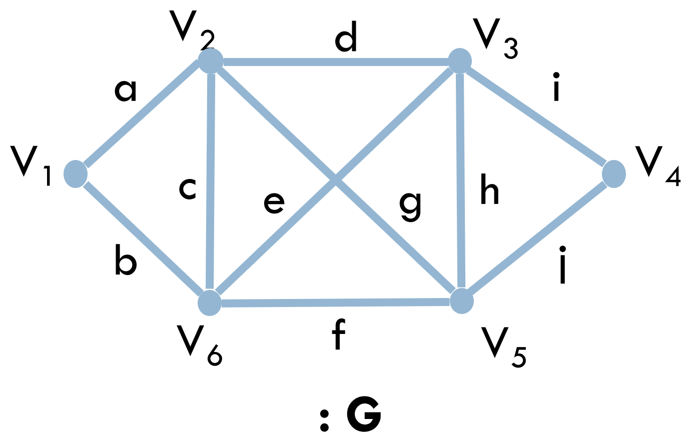
Example

- Cut-vertices are
  - $\{V_3\}$
  - $\{V_1, V_4\}$
  - $\{V_2, V_4\}$
- Smallest cut-vertex is $\{V_3\}$ → contains one element
- Hence vertex connectivity V_c of graph G is 1



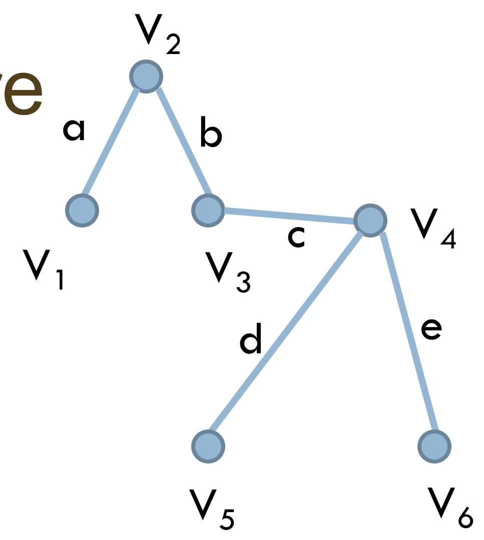
Problem

- Find the vertex connectivity V_c of the graph given



Vertex connectivity of a tree

- Since a tree can be broken by the removal of a single non-pendant vertex, vertex connectivity of a tree is always 1
- Cut-vertices of the tree are
- $\{V_2\}$, $\{V_3\}$, $\{V_4\}$
- Hence V_c is 1

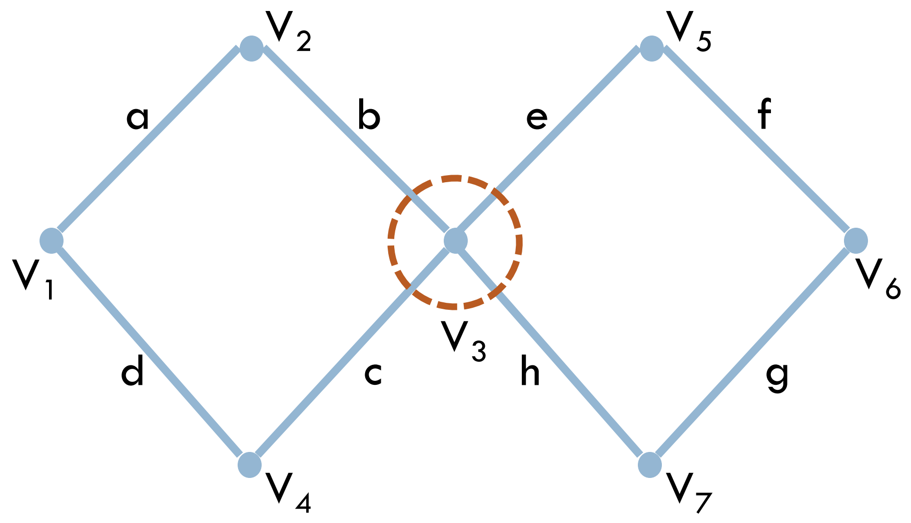


Separable graph

- A connected graph is said to be separable if its vertex connectivity is one
- If removal of a single vertex disconnects the graph, then it is separable
- The vertex, removal of which disconnects the graph is called an articulation point or cut-vertex or cut-node
- In such a graph, there would be a subgraph g such that g & $\overline{g}$ have only 1 vertex in common

Example

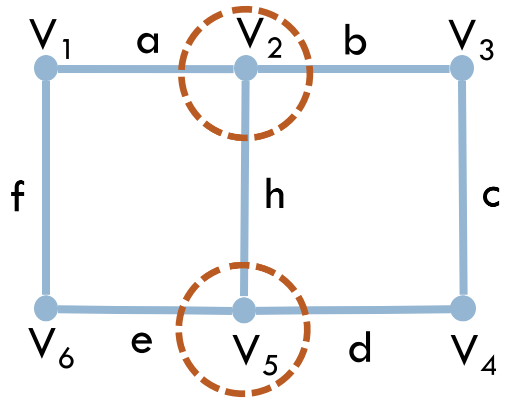
- Smallest cut-vertex is $\{V_3\}$; contains 1 element
- Hence G is a separable graph
- V_3 is the articulation point
- Removal of V_3 disconnects the graph



: G

Example

- Smallest cut-vertex is $\{V_2, V_5\}$; contains 2 elements
- Hence G is a non-separable graph



: G

- **Theorem 38:** A vertex V in a connected graph G is a cut–vertex iff these exists two vertices x & y in G such that every path between x & y passes through V
- **Proof**: Let V be a cut-vertex of graph G
- Then removal of V from G must disconnect the graph into two components, such that the components are not empty. Each component must contain at least an isolated vertex
- Let x be a vertex from first component & y from the other component
- If there exists a path between x & y, other than through vertex V, then removal of V will not disconnect the graph. But since V is a cut-vertex, removal of V must disconnect the graph
- So there can be no path between x & y other than through V

- **<u>Conversely:</u>** If x & y are two vertices of G such that all paths between x & y are through vertex V
- Then removal of V from G makes x & y not reachable from each other as all paths between x & y have been broken
- Since no path exists between x & y, then x & y must be lying in different components, which implies that the graph has been disconnected by the removal of V
- Any vertex V, removal of which disconnects a graph is a cut vertex. Hence here, V is a cut-vertex
- Hence the theorem

- **Theorem 39:** The edge connectivity of a graph cannot exceed the degree of the vertex with the smallest degree in G
- **Proof:** Let V_i be the vertex with the smallest degree.
- Let $d(V_i)$ represent the degree of V_i
- Vertex V_i can be separated from the graph by removing all the $d(V_i)$ edges incident on it
- Hence $d(V_i)$ is the edge connectivity of the graph
- Hence the theorem

- **Theorem 40:** The vertex connectivity of any graph G can never exceed the edge connectivity of G
- **Proof:** Let α denote the edge connectivity of G
- Then, there must exist a cut–set with α edges. Let it be S.
- S partitions the vertex set of the graph into two. Let they be V_1 & V_2
- By removing at most α vertices from V_1 (or V_2) on which the α edges were incident, we can bring the same effect on the graph i.e, we can disconnect the graph in the same way how S disconnected the graph. However if any other edges were incident on these vertices, they too would get deleted
- However the vertex connectivity would be α itself

K-connected graph

- A graph whose vertex connectivity is K
- Every pair of vertices in a k-connected graph is joined by at least k non-intersecting paths

Properties of cut-sets

- Cut-set → set of edges removal of which disconnects the graph
- Edge connectivity → no. of edges in the smallest cut-set
- Cut-set in tree → every edge of a tree is a cut-set
- Edge connectivity of any tree → is always 1
- Fundamental cut-set → a cut-set that contains exactly one branch of the spt
- Number of fundamental cut-sets → no. of branches in the spt
- Fundamental circuit → a circuit that contains exactly one chord
- Number of fundamental circuits → no. of chords in the graph
- Every cut-set will contain at least one branch of every spt

Properties of cut-vertices

- Cut-vertex → set of vertices removal of which disconnects the graph
- Vertex connectivity → no. of vertices in the smallest cut-vertex
- Cut-vertex in tree → Every vertex (other than pendant vertex) in a tree is a cut-vertex
- Vertex connectivity of any tree → always 1
- Separable graph→ graph whose vertex connectivity is 1
- Edge connectivity → cannot exceed the smallest degree
- The vertex connectivity → cannot exceed edge connectivity
- A graph is K-connected → vertex connectivity is K

Combinatorial representation of Graphs

- Any graph exists as an abstract object irrespective of its size and shape in drawing
- Such a combinatorial/abstract representation of a graph is given by
- $G = (V, E, \varphi)$
- Where $V = \{V_1, V_2, \ldots . V_n\}$ is the set of n vertices

  $E = \{e_1, e_2, \ldots . e_e\}$ is the set of e edges

  $\varphi = E \rightarrow V$
- φ is the mapping from set E to set V
- φ defines the relationship between the sets V & E

Example

- $V = \{V_1, V_2, V_3, V_4, V_5\}$
- $E = \{a,b,c,d,e,f,g\}$
- $\varphi = a \rightarrow (V_1,V_5)$

  $b \rightarrow (V_1, V_2)$

  $c \rightarrow (V_1, V_4)$

  $d \rightarrow (V_5, V_4)$

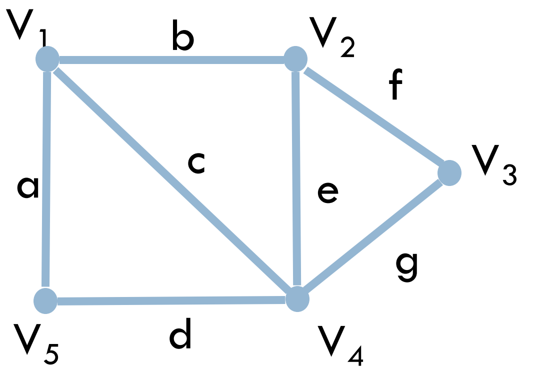
  $e \rightarrow (V_2, V_4)$

  $f \rightarrow (V_2, V_3)$

  $g \rightarrow (V_4, V_3)$
- Here $a \rightarrow (V_1, V_5)$ implies that object a from set E is mapped onto the unordered pair (V_1,V_5) of objects from set V

Geometric representation of Graphs

- An abstract/combinatorial graph can be geometrically represented in many ways without altering the definition of the graph
- Geometric representation of the abstract graph on the previous slide

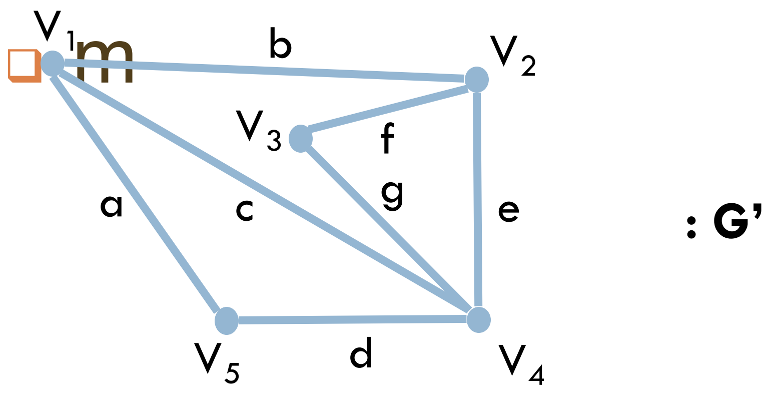


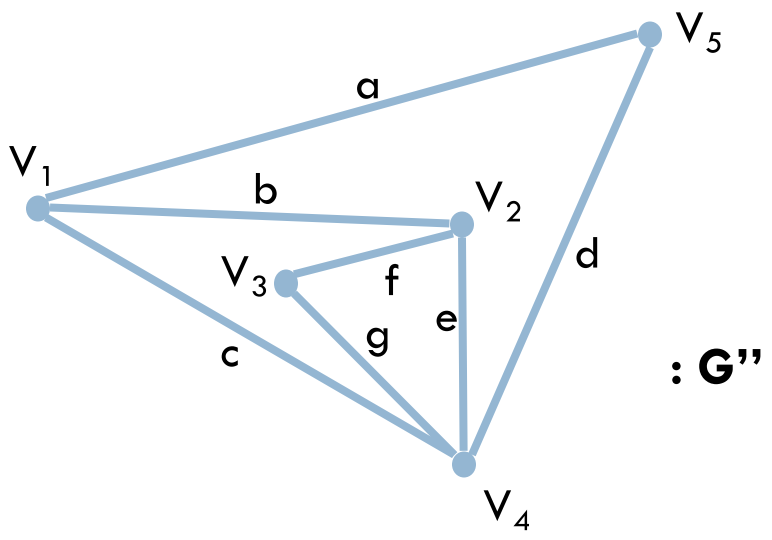
: G

Examples

- G' and G'' are also geometric representations of the same abstract definition
- It can be seen that G, G' & G'' are all isomorphic to each other

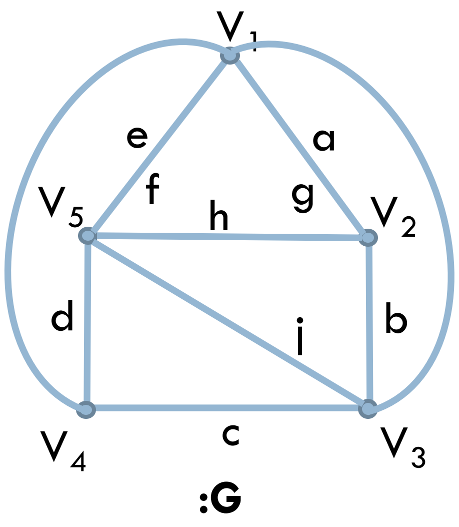
- m

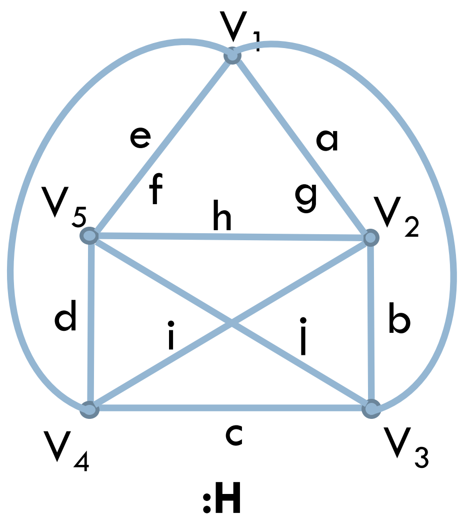




Planar & non-planar graphs

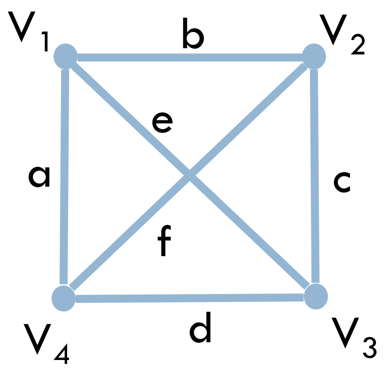
- A graph G is said to be planar if there exists some geometric representation of G which can be drawn on a plane without none of its edges intersecting
- A graph that cannot be drawn on a plane without edges crossing over is called a non planar graph
- G is planar and H is non-planar



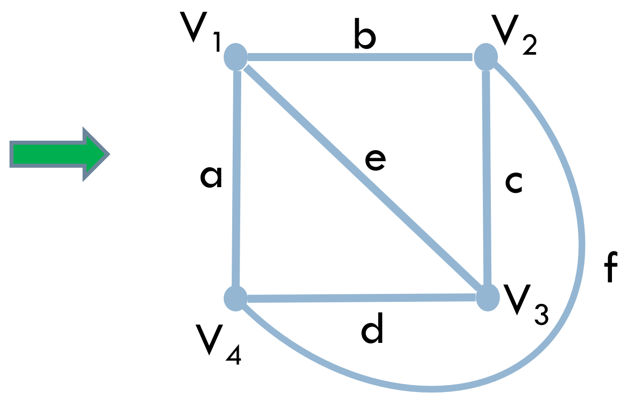


Problem

- Is the given graph G a planar graph?
- Yes, coz it can be re-drawn without none of its edges intersecting



: G



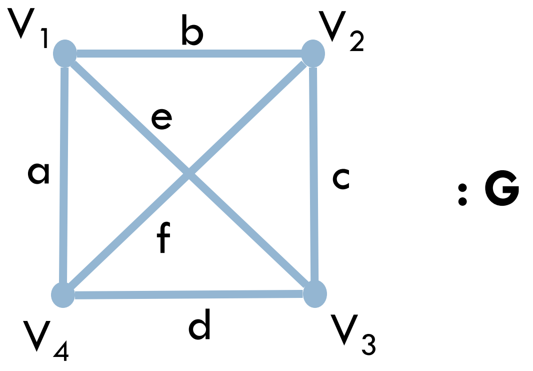
: G'

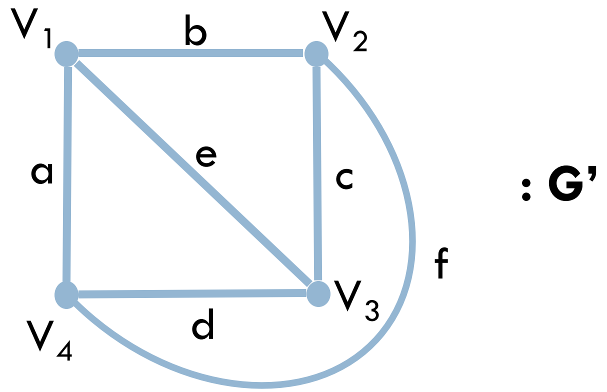
Embedding

- A drawing of a geometric representation of a graph on any surface without edges intersecting is called an embedding
- Only planar graphs can have embeddings
- Such an embedding of a planar graph G on a plane is called plane representation of G

Example

- G is a planar graph, but it is not an embedding as some of its edges crosses over others
- Where as G' is an embedding

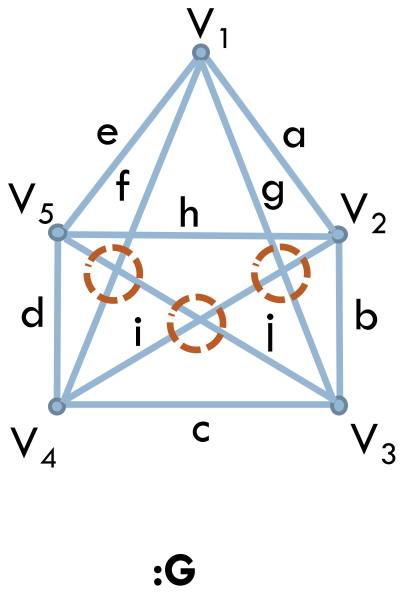


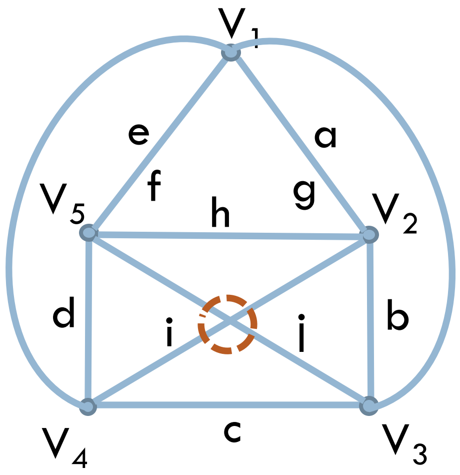


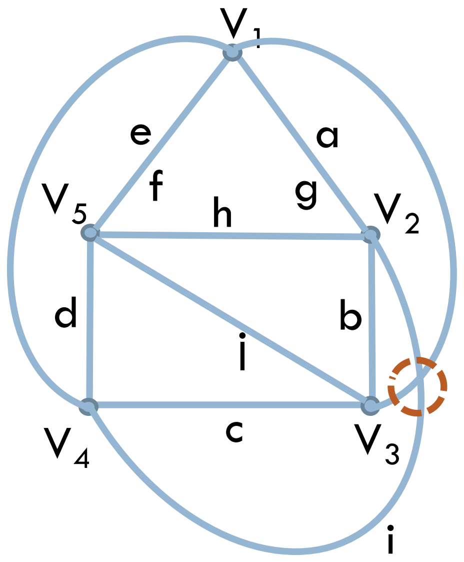
- <u>Note:</u> for a graph to be planar, there must at least one geometric representation that is an embedding i.e, we must be able to draw the graph without its edges intersecting

Example

- The graph G is non-planar as we are not able draw an embedding







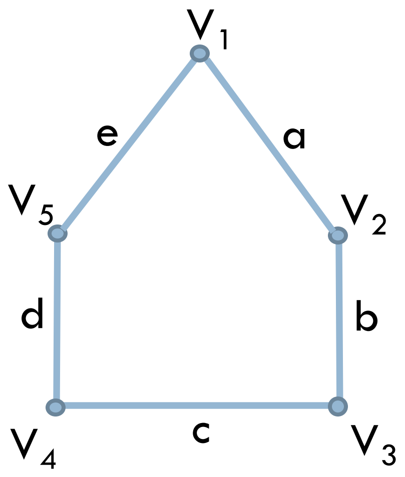
Kuratowski's 2 non-planar graphs

- The Polish mathematician Kasimir Kuratowski
- Non-planar property of the 2 graphs
  - Complete graph with 5 vertices (K_5)
  - Bi-partite graph with 6 vertices ($K_{3,3}$)

Complete graph with 5 vertices (K_5)

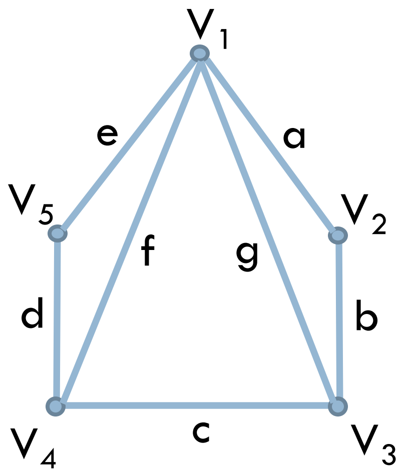
- Theorem 43: The complete graph with 5 vertices (K_5) is non planar
- Proof: Let the 5 vertices of the graph be V_1, V_2, V_3, V_4 and V_5
- Since it is a complete graph, every vertex needs to be connected to every other vertex by an edge
- There must be a circuit going from V_1 to V_2 to V_3 to V_4 to V_5 and back to V_1; that is a pentagon that divides the region into 2 - inside & outside of the pentagon

:G



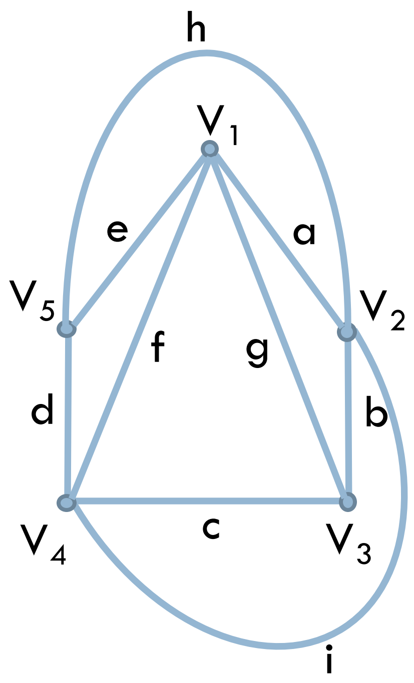
- Now we need V_1 to be connected to V_3 & V_4. V_1 can be connected to V_3 along an edge inside the pentagon. Similarly V_1 can be connected to V_4 also, inside the pentagon

:G



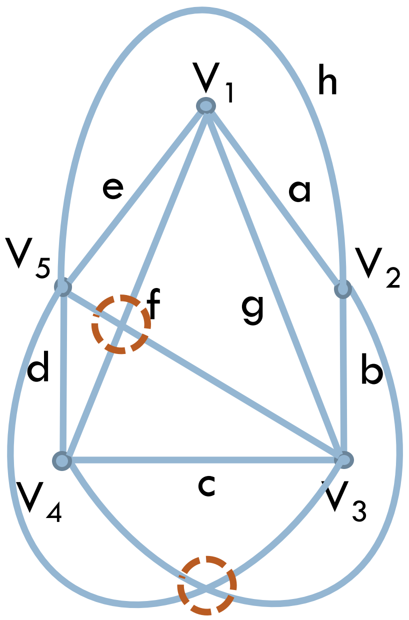
- Next we need V_2 to be connected to V_4 & V_5
- Drawing an edge inside the pentagon is not possible as it will intersect the previously drawn edges. So let as draw these 2 edges along the outside region

:G



- Now V_1, V_2, & V_4 have degrees 4 each. V_3 & V_5 have degrees 3 each. So the remaining edge to be drawn is between V_3 & V_5. We cannot draw this edge inside or outside, without intersecting previous edges
- Hence this graph cannot be embedded in a plane
- So it is non-planar

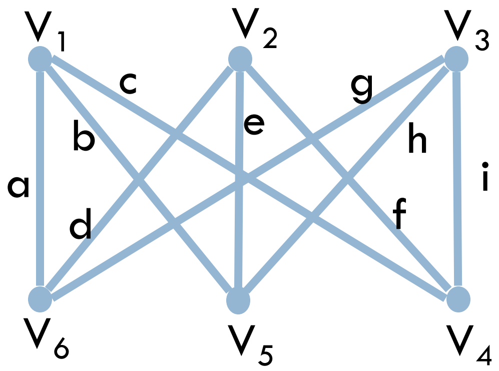
:G



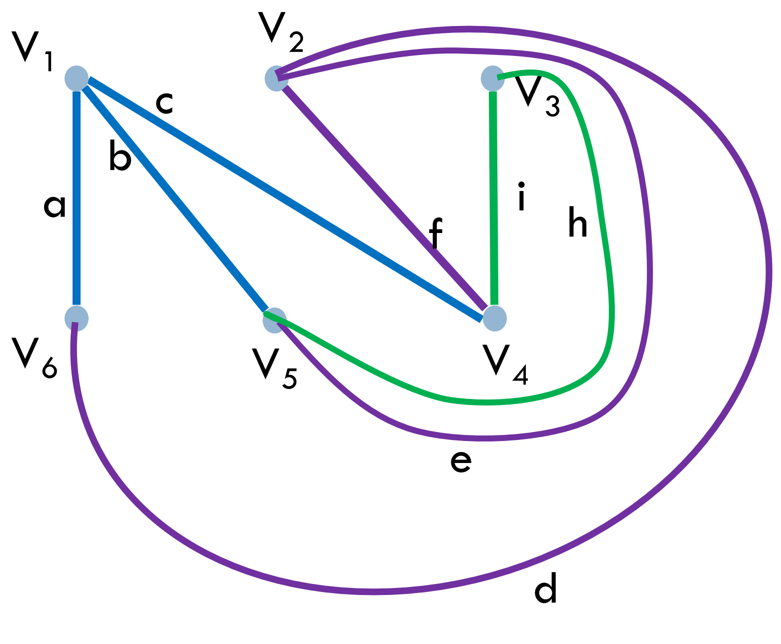
Bi-partite graph with 6 vertices ($K_{3,3}$)

- Theorem 44: Kuratowski's 2nd graph is non-planar
- Proof: The graph is bi-partite graph with 6 vertices
- Here the vertex set V is divided into two V' & V''
- Every vertex in V' is connected to every vertex in V'' by an edge
- $V'=\{V_1,V_2,V_3\}$ & $V''=\{V_4,V_5,V_6\}$
- $E = \{a,b,c,d,e,f,g,h,i\}$

: **G**

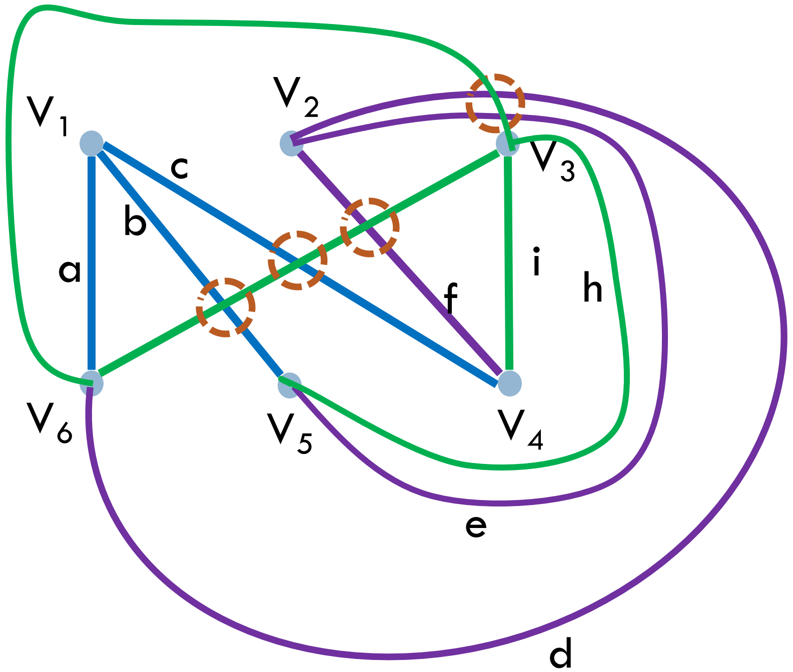


- From vertex V_1, draw edges towards V_4, V_5 & V_6. Now V_1 has degree 3
- From vertex V_2, draw an edge toward V_4
- From V_3 draw an edge towards V_4
- Again at V_2, draw edge towards V_5 & V_6 using curved lines so as to avoid intersecting
- Also from V_3 to V_5



: G

- Now 1 more edge is required between V_3 & V_6. We cannot draw the edge without intersecting previous edges
- Hence the proof



: G

Properties common to Kuratowski's 2 graphs

1) Both are regular graphs
   - K_5 is a regular graph with degree 4 each
   - $K_{3,3}$ is a regular graph with degree 3 each
2) Both are non planar
3) Removal of one edge or one vertex makes both of them planar
4) Both are the smallest non planar graphs
   - K_5 is the non planar graph with smallest no. of vertices (5 vertices)
   - $K_{3,3}$ is the non planar graph with smallest number of edges (9 edges)

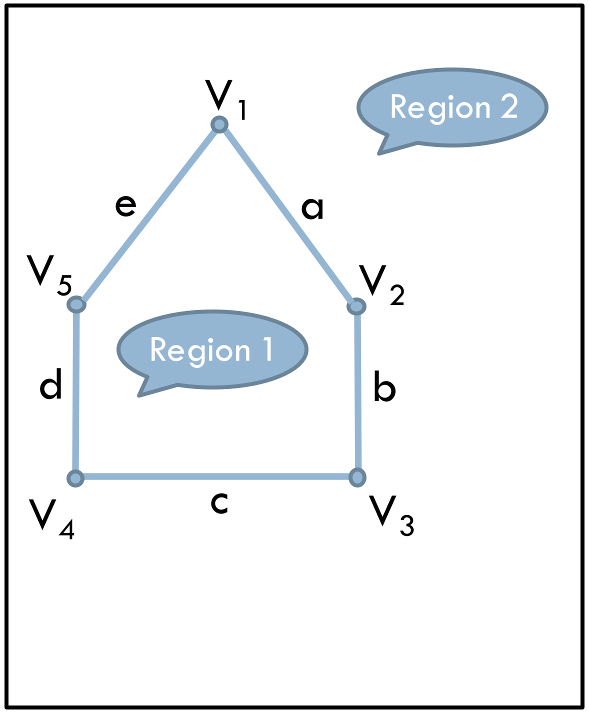
Embedding of a planar graph without curved lines

- It may appear that in order to draw a planar graph without its edges intersecting, we need to use curved lines as some edges; but this is not true
- Fary proved that every planar graph can be drawn using straight lines and of course, without edges intersecting

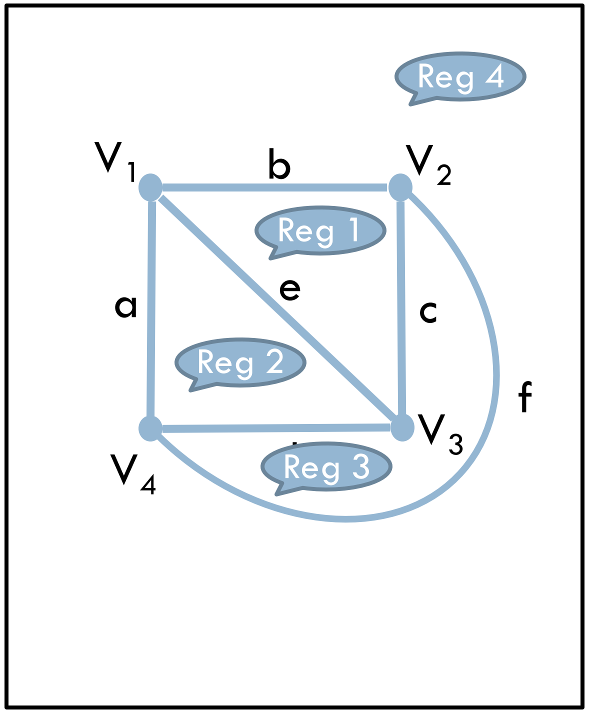
Region or face

- The plane representation of a graph (embedding) divides the plane into regions. A region is characterized by the set of edges forming its boundary
- Hence non-planar graphs cannot have regions defined, as they have their edges intersecting
- And planar graphs which are not an embedding too cannot have regions defined
- Thus region is a property, specific to the geometric representation of a graph and not to the abstract representation
- Regions are also known as faces, windows or meshes

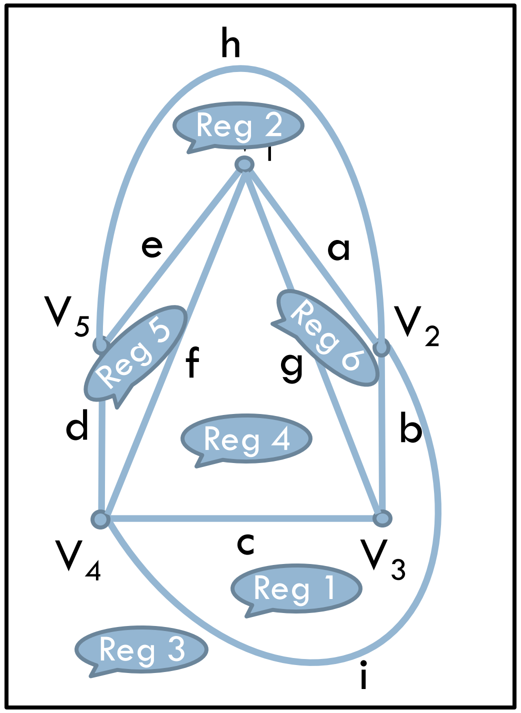
Examples: planar graphs



: G



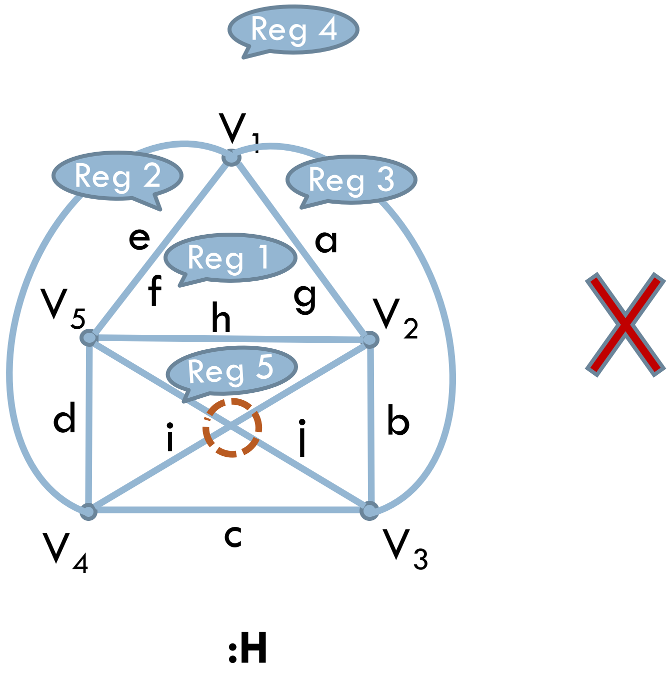
: H



: J

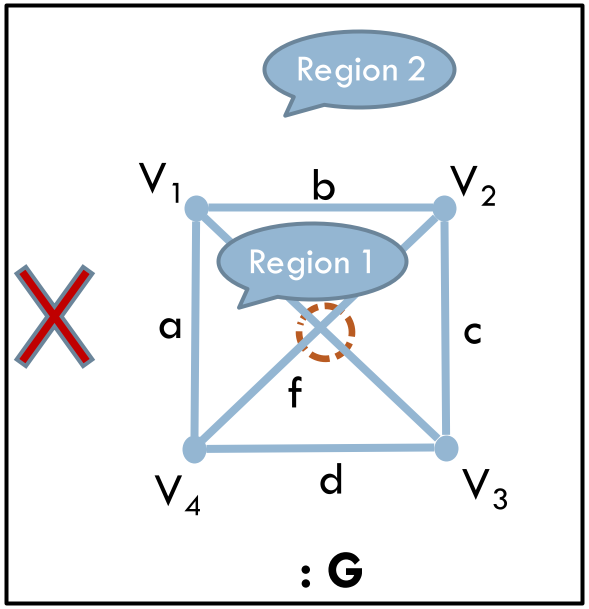
Example: non-planar

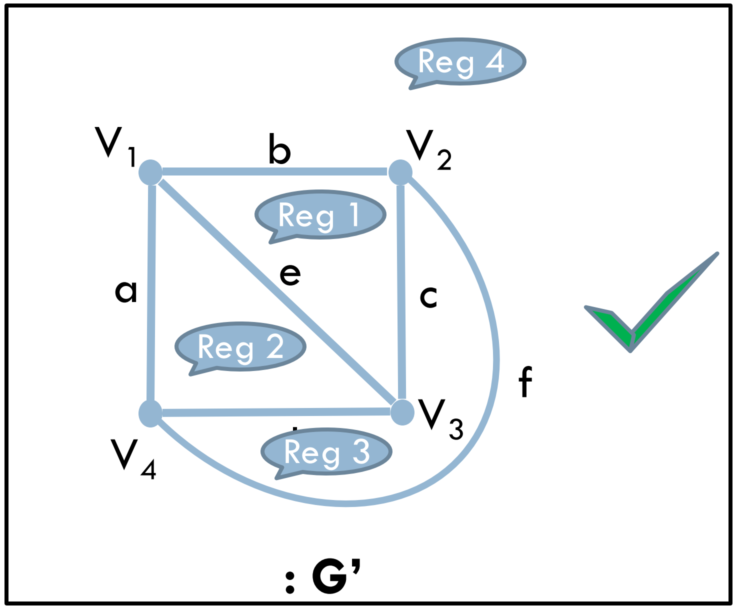
- Since the faces do not have proper boundaries, regions cannot be defined for non-planar graphs



Example: planar but not an embedding

- Though planar, G is not an embedding; hence cannot have regions defined
- G' is an embedding of the same planar graph G; hence 4 regions can be identified





Infinite Region

- The portion of the plane lying outside a graph embedded in a plane is called the infinite region
- Also known as unbounded, outer or exterior region
- Like other regions, infinite region is also characterized by a set of edges
- A planar graph can be embedded in a plane in different ways. By changing the embedding of a planar graph, we can change the infinite region

- G & G' are different planar embeddings of the same graph. The infinite region boundary is different for both

Regions of graph G are

Region 1 = {a,d,e}

Region 2 = {e,b,c}

Region 3 = {a,b,f}

Infinite region = {c,d,f}

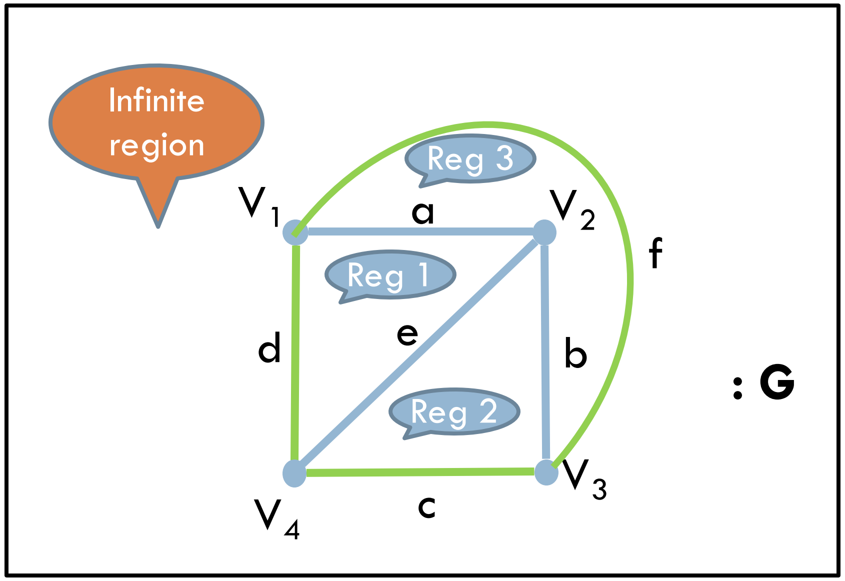
Regions of graph G' are

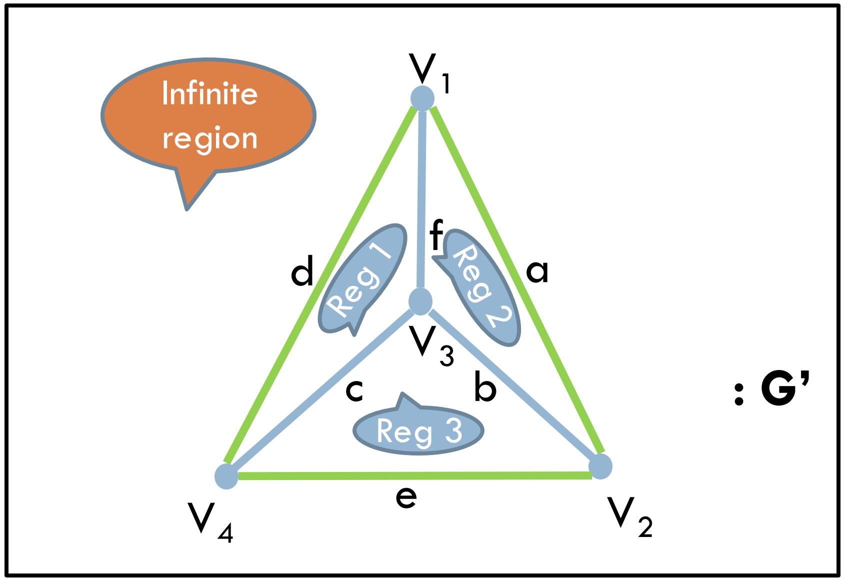
Region 1 = {c,d,f}

Region 2 = {a,b,f}

Region 3 = {e,c,b}

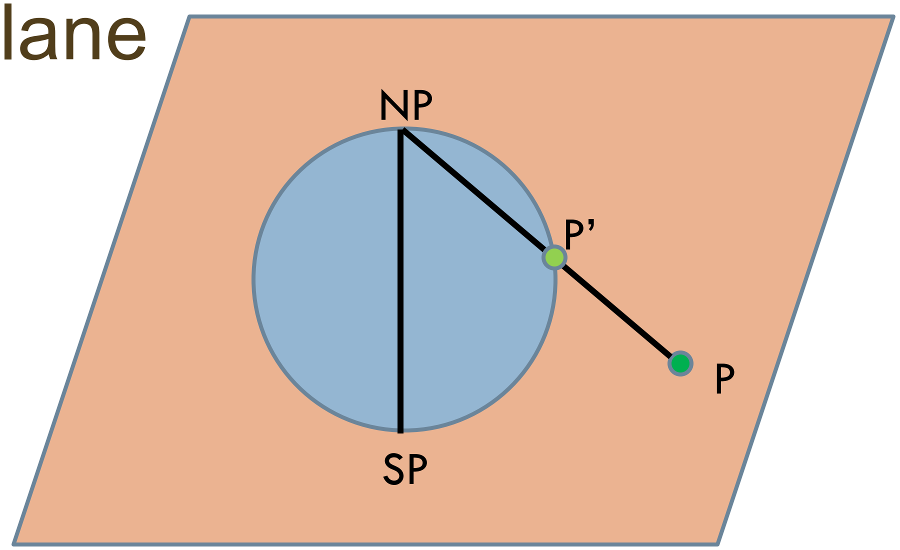
Infinite region = {a,e,d}





Embedding on a Sphere

- To eliminate the distribution between finite & infinite regions, a planar graph can be embedded on the surface of a sphere
- It is done by stereographic projection of a sphere on a plane



- **Theorem 45:** A graph can be embedded on the surface of a sphere if and only if it can be embedded on a plane
- **Proof:** Place the sphere on the plane and note the point of contact as SP (south pole)
- From the point SP, draw a straight line perpendicular to the plane. The point where this line meets the circumference of the sphere is noted as NP
- For any point P on the plane, there is a corresponding point p’ on the sphere and vice versa
- To obtain P’, draw a straight line from P to meet NP. Point where this line intersects the circumference of the sphere is the point p’
- Thus we can say that there is a one-to-one correspondence between the points on the sphere and the finite points on the plane
- Points at infinity corresponds to NP
- Hence the theorem

- **Theorem 46:** A planar graph may be embedded in a plane such that any specific region can be made the infinite region
- **Proof:** A planar graph embedded in the surface of a sphere divides the surface into different regions.
- Each region on the sphere is finite, the infinite region has been mapped on to the point NP
- Now, it is clear that by suitably rotating the sphere, we can make any specific region to be the infinite region on the plane
- Hence the theorem

Number of Regions in a planar graph

- In all the possible embeddings of a planar graph, the no. of regions in the graph would be the same
- It is known as Euler's formula
- It is given by f = e-n+2
  - f→ no. of regions/faces
  - e→ no. of edges
  - n→ no. of vertices

Euler’s formula

- **Theorem 47:** A connected planar graph with n vertices and e edges has e-n+2 regions
- **Proof:** Assume a simple graph. Though a self loop or a parallel edge does not affect the formula, as increasing an edge equally increases the no. of regions
- Similarly let us not consider those edges that do not form the boundary of any region. Such edges also do not affect the formula, as each such edge increases n by 1, ‘e-n’ remains unaltered
- Assume the planar embedding of the graph to be containing all straight lines as edges
- Now, the graph looks like a net of polygons

- **<u>Calculating total no. of edges</u>**
- Let k_3= no. of triangles

  k_4= no. of quadrilaterals

  k_5= no. of pentagons

  ..

  k_r= no. of r-sided polygons
- Hence total no. of edges in the entire net of polygon would be
- $3k_3 + 4k_4 + 5k_5 \ldots\ldots..+ rk_r + p = 2^*e$ Equation 1
- Where e is the total no. of edges in the polyhedron
- As each edge would be counted twice, we take 2e

- **Calculating total no. of faces**
- Total no. of faces/regions would be
- $k_3 + k_4 + k_5 \ldots\ldots\ldots + k_r + 1 = f$ Equation 2
- The outer region/infinite region is also to be counted along with all interior regions; hence the ‘1’
- f is the total no. of regions/faces

- **Sum of all interior angles of the polyhedron:**
- Sum of all interior angles of a p–sided polygon is $(p-2)\pi$
- Taking the sum of interior angles for each polygon inside the polyhedron,
- $k_3(3-2)\pi + k_4(4-2)\pi + k_5(5-2)\pi + \ldots\ldots + k_r(r-2)\pi$ Equation 3
- **Sum of all exterior angles of the polyhedron:**
- Sum of all exterior angles of a p–sided polygon is $(p+2)\pi$ Equation 4
- **Total angle sum of the polyhedron:**
- At each vertex we have an angle of 360^o
- Since we have 'n' vertices,
- Total angle sum $= n * 360^o = n * 2\pi = 2\pi n$ Equation 5

Equation 3 + Equation 4 = Equation 5

- Sum of interior angles + sum of exterior angles = total angle sum of polyhedron
- $k_3(3-2)\pi + k_4(4-2)\pi + \ldots.. + k_r(r-2)\pi + (p+2)\pi = 2\pi n$
- $\pi(3k_3 + 4k_4 + \ldots + rk_r) - 2k_3\pi - 2k_4\pi - \ldots. - k_r\pi + (p+2)\pi = 2\pi n$
- From eq 1, we have $3k_3 + 4k_4 + \ldots..+ rk_r = 2e - p$
- $\pi(2e - p) - 2\pi(k_3 + k_4 + \ldots\ldots\ldots. + k_r) + (p+2)\pi = 2\pi n$
- From eq 2, we have $k_3 + k_4 + k_5 \ldots\ldots..+ k_r + 1 = f$

- $\pi(2e - p) - 2\pi(f - 1) + (p+2)\pi = 2\pi n$
- $2e - p - 2f + 2 + p + 2 = 2n$
- $2e - 2f + 4 = 2n$
- $e - f + 2 = n$
- $f = e - n + 2$

<u>Note:</u>

In any simple connected planar graph,

$e \geq 3/2\, f$ and $e \leq 3n-6$

Recalling Kuratowski's two graphs

- 1) Complete graph with 5 vertices (K_5)
- n = 5; e = 10; 3n – 6 = 9
- According to the above result for a planar graph, $e \leq 3n-6$
- Here $10 \leq 9$ ✗
- Therefore K_5 is non-planar
- f = e-n+2= 10-5+2 = 7; 3/2 *f = 21/2 = 10.5
- According to the above result for a planar graph, $e \geq 3/2\ f$
- Here $10 \geq 10.5$ ✗
- Therefore K_5 is non-planar

- 2) Bipartite graph with 6 vertices ($K_{3,3}$)
- $n = 6$; $e = 9$; $3n – 6 = 12$
- According to the above result for a planar graph, $e \leq 3n-6$
- Here $9 \leq 12$ ✓
- But $K_{3,3}$ is non-planar
- $f = e-n+2 = 9 – 6 +2 = 5$; $3/2 f = 3/2 * 5 = 7.5$
- According to the above result for a planar graph, $e \geq 3/2 f$
- Again $9 \geq 7.5$ ✓
- But $K_{3,3}$ is non-planar
- Hence we conclude that $e \leq 3n-6$ is only a sufficient condition, not necessary for a graph to be planar

Detection of Planarity

- In order to check whether a given graph is planar or not the following steps of Elementary reduction can be used

Elementary Reduction

- **Step 1:** Of the graph is disconnected, we need to check whether each component is planar. If all components are planar, then the disconnected graph is said to be planar
- If the graph is a separable graph, we need to check whether each block is planar. It all blocks are planar, then the separable graph is planar
- Now, let our graph G = $\{G_1, G_2, \ldots\ldots\ldots\ldots G_k\}$
- Where each G_i is a non-separable block of G
- Test each G_i for planarity

- **Step 2 :** Remove all self loops as self loops does not affect planarity
- **Step 3 :** Similarly remove all parallel edges as they too do not have anything to do with planarity
- **Step 4 :** Merge edges in series, by ignoring their common vertex
- **Step 5 :** Repeat step 3 and 4 repeatedly until no more edges can be deleted.
- **Step 6 :** Now the resulting graph may contain
  1. A single edge
  2. A complete graph with 4 vertices
  3. A non separate graph with $n \geq 5$ & $e \geq 7$
- If it is (i) or (ii) then our graph is planar; no need of further clarification
- But if it is (iii) continue to step 7

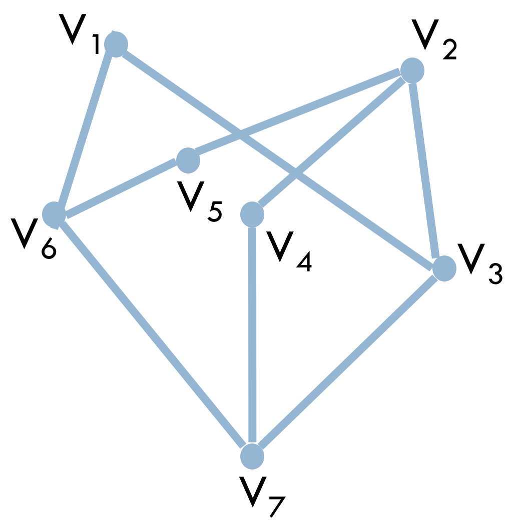
- **Step 7**: Check whether $e \leq 3n - 6$ for the resultant graph
- If the condition is not satisfied, then our graph is planar. But if not, may or may not be planar. So we need to check further
- **Step 8:** Check whether the resultant graph contain either of Kuratowski's graphs or their homeomorphic graphs
- If our graph contains K_5, $K_{3,3}$ or graphs homeomorphic to K_5 and $K_{3,3}$, then our graph is certainly non-planar

Homeomorphic graphs

- Two graphs are said to be homeomorphic if one can be obtained from the other by creating edges in series or by merging edges in series

Problem

- Check whether the given graph is planar by the method of elementary reduction



: G

Kuratowski's theorem

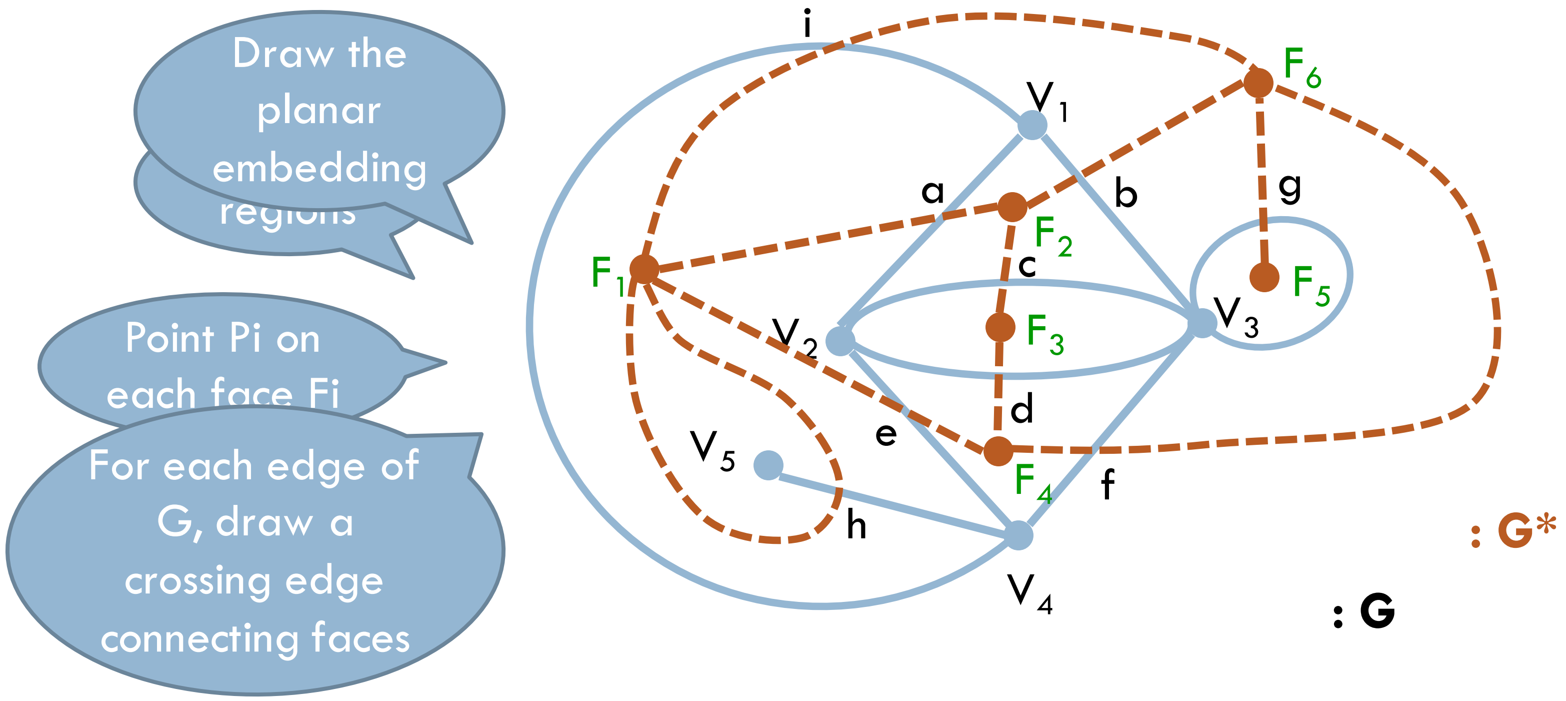
- Theorem 48: A necessary & sufficient condition for a graph G to be planar is that G does not contain either of Kuratowski's two graphs or any graph homeomorphic to them
- Proof: We know that Kuratowski's 2 graphs are no-planar and they cannot be embedded in a plane
- So if any graph contains any of the above graphs as subgraphs, then surely the main graph too could not be embedded in a plane. So the main graph is also non-planar
- If the given graph contains subgraphs that are homeomorphic to any of the Kuratowski's graph, then the given graph is also non-planar coz any graph homeomorphic to K_5 and $K_{3,3}$ is also non-planar

Geometric dual of a planar graph

- In order to obtain the geometric dual of a planar graph
  1) Start with a plane representation of the planar graph (planar embedding)
  2) Name the regions or faces are F_1, F_2, F_3…… F_{e-n+2}
  3) Place a point P_i in each face F_i
  4) For each edge of G, draw a line crossing the edge connecting the two faces on either sides; For an edge lying entirely in a region, draw a self loop at the point that passes through the edge
  5) Name the new graph as G* which forms the dual of G

Example

- Let G be the plane representation of a graph



Properties of duals G & G*

- A self loop in G yields a pendant edge in G*
- A pendent edge in G yields a self loop in G*
- Edges in series in G becomes parallel edges in G*
- Parallel edges in G becomes edges in series in G*
- Number of edges forming the boundary of a face F_i in G becomes the degree of the vertex P_i in G*
- Degree of a vertex V_i in G becomes the number of edges forming the boundary of the face F_i in G*

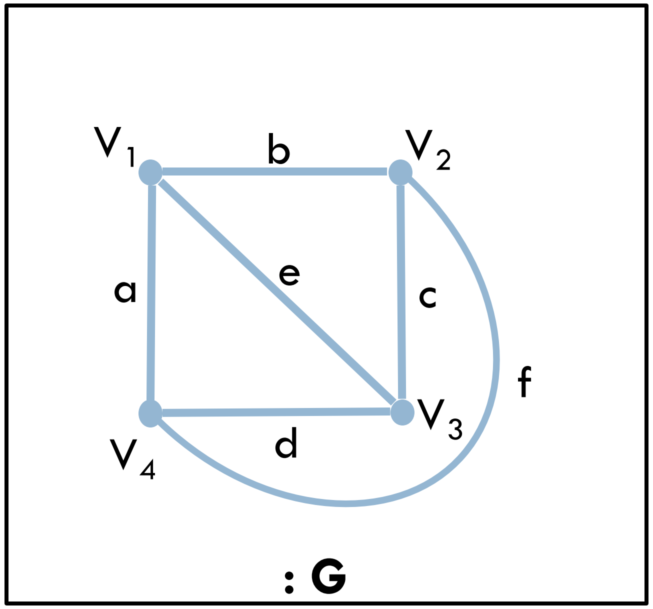
- Since G is planar, G* is also planar
- If n, e, f, r & μ denotes the no. of vertices, no. of edges, no. of faces, rank & nullity of G & n*, e*, f*, r*& μ * denotes the corresponding quantities in G*
- There is a one-to-one correspondence between the edges of G & G*. Every edge of G intersects the corresponding edge of G*. However, number of vertices may change

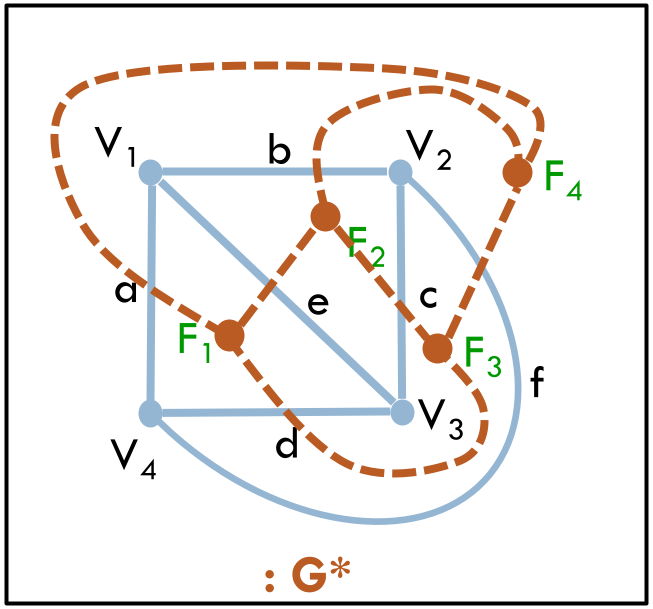
All duals of G

- A planar graph G may have different planar embeddings. For each planar embedding, we can obtain a corresponding geometric dual
- A planar graph G will have a unique dual if & only if it has a unique planar embedding
- If G & G’ are isomorphic, then their corresponding duals G* & G’* may not be isomorphic

Self dual Graphs

- If a planar graph G is isomorphic to its own dual, it is called a self dual graph
- Example



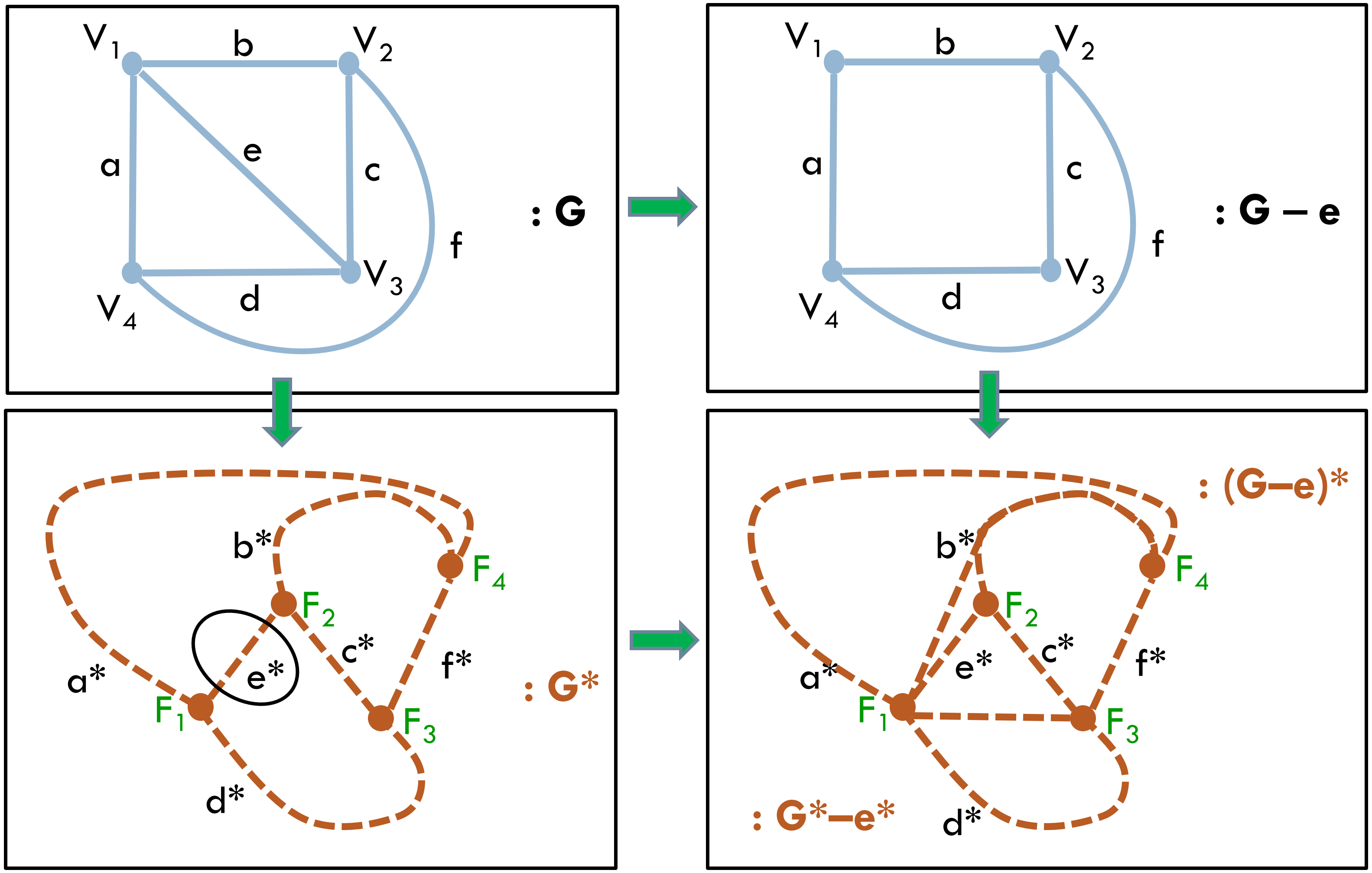


Dual of a Subgraph

- Let G* be the dual of G
- Let ‘a’ be an edge in G & a*, the corresponding edge in G*
- To find the dual of G-a; that is the dual of the graph G after deleting the edge ‘a’ i.e, (G-a)*
- This can be directly obtained from G*
- If ‘a’ was a boundary of 2 regions in G, then by deleting a* from G*, we can obtain (G-a)* ; deleting the edge will require to fuse the end vertices

- Else if ‘a’ was a not any boundary in G, then a* would be a self loop in G*. Then deleting the self loop yields (G-a)*
- G – q

Example



V_1
b
V_2
a
e
c
f
V_4
d
V_3
: G
V_1
b
V_2
a
c
f
V_4
d
V_3
: G – e
b*
F_4
F_2
c*
f*
a*
e*
F_1
F_3
d*
: G*
: (G–e)*
b*
F_4
F_2
c*
e*
f*
a*
F_1
F_3
: G*–e*
d*

Dual of a homeomorphic graph

- Let G* be the dual of G
- Let ‘a’ be an edge in G & a* be its corresponding edge in G*
- Suppose we create a new vertex in G by introducing a vertex of degree 2 on edge a. This will create a new edge as well. Let it be b. Now the dual of G+b will contain a new edge b* which appears as an edge parallel to a*
- Similarly merging 2 edges in series will simply eliminate one of the corresponding parallel edges in G*
- Thus dual of a homeomorphic graph of G can be obtained from G*

Combinatorial Dual

- G* is said to be combinatorial dual of G if there is a one to one correspondence between the edges of G & G* such that if g is any subgraph of G & h is the corresponding subgraph of G* then
- Rank (G*-h) = rank (G*) – nullity (g)

- **Theorem 49:** A necessary and sufficient condition for two planar graphs G_1 & G_2 to be duals of each other is that, there is a one-to-one correspondence between the edges of G_1 & G_2 such that a set of edges in G_1 forms a circuit if & only if the corresponding set in G_2 forms a cut-set
- **Proof:** Since every edge of G will be intersected by exactly one edge of G*, there must ne a one to one correspondence between the edges of G_1 & G_2
- Now, consider a planar representation of G & its dual G*. Let ρ be an arbitrary circuit in G. ρ will form will form some simple closed curve in G, dividing the plane into 2 areas, one inside ρ & the other outside ρ

- Now the vertices of G* can be viewed as two non empty disjoint subsets, those vertices that represent regions inside ρ & those that represents regions outside ρ and this partition is brought by the set of edges in ρ^*. Hence ρ^* is a cut-set in G*
- Similarly every cut-set S* in G* will have a unique circuit S in G

- **Conversely:** Suppose there are two planar graphs G & G' such that there is one to one correspondence between their edges and also one to one correspondence between the cut-sets of G & the circuits of G' and vice versa
- Let G* be a dual of G
- Then there is a one to one correspondence between the cut-sets of G & the circuits of G' & also between the cut-sets of G & the circuits of G*
- Therefore there is a one to one correspondence between the circuits of G' & G* implying that G' & G* are 2-isomorphic. Then G' must be a dual of G
- (Based on the theorem: Two graphs are 2- isomorphic if & only if they have circuit correspondence)

- **Theorem 50:** A graph has a dual if and only if it is planar
- **Proof:** Let us prove that a non planar graph does not have a dual
- Let G be a non-planar graph. Then according to Kuratowski's theorem, G contains either K_5 or $K_{3,3}$ or a graph homeomorphic to them
- Any graph can have a dual only if every subgraph of that graph & every graph homeomorphic to that graph has a dual
- From the above 2 statements, we can say that if K_5 and $K_{3,3}$ cannot have a dual then none of the non-planar graphs can have a dual

- To prove that $K_{3,3}$ do not have a dual, assume the contradiction that $K_{3,3}$ has a dual D
- Since $K_{3,3}$ has 9 edges, so must be D
- All cut-sets in $K_{3,3}$ must have corresponding circuits in D & vice versa
- Since $K_{3,3}$ do not have any cut-set containing 2 edges, D cannot have any circuit containing 2 edges. That means D cannot contain any parallel edges

- Since every circuit in $K_{3,3}$ is of length 4 or 6, D cannot have any cut set with less than 4 edges, which implies every vertex in D has degree of at least 4
- Since D has no parallel edges & every vertex has degree of minimum 4, D must contain at least 5 vertices, each of degree 4 or D may contain more than 5 vertices with larger degrees
- D must then at least contain $\frac{5*4}{2}$ = 10 edges, contradicting to the fact that D has only 9 edges
- So there can be no such D. Hence $K_{3,3}$ cannot have a dual

- Similarly we can prove that K_5 do not have a dual
- Assume the contradiction that K_5 has a dual, H
- Since K_5 has 10 edges, H must also have 10 edges
- All cut-sets in K_5 must have corresponding circuits in H vice versa
- Since K_5 do not have any cut-set with 2 edges, it cannot have a circuit with 2 edges. That means H has no parallel edges
- Since every cut-set in K_5 contains 4 or 6 edges, H can have circuits of length 4 or 6 only

- Consider a circuit of length 6 (hexagon) in H. Now, we cannot add the remaining 4 edges, without creating parallel edges or circuits of length three
- So in order to add the remaining 4 edges without violating the rules (parallel edge & circuits of length 3) we assume H to have 7 vertices, with degree at least 3
- Then H must have $\frac{7*3}{2}$ =11 edges, contradicting that H has 10 edges
- So there can be no such dual for K5

End of module 4