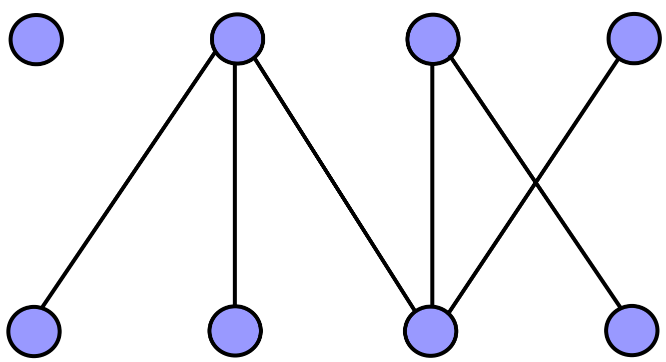
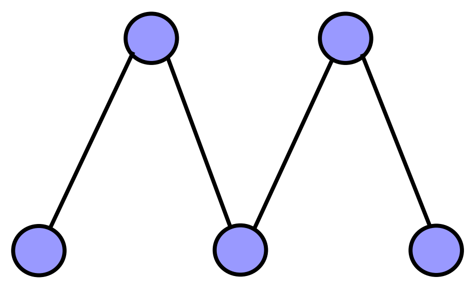
GRAPH THEORY and APPLICATIONS

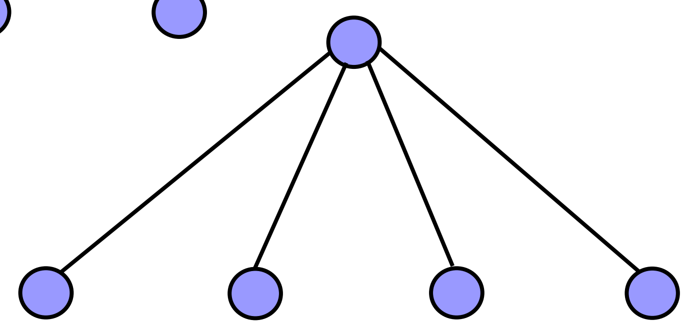
Trees

Properties

- Tree: a connected graph with no cycle (*acyclic*)
- Forest: a graph with no cycle
- Paths are trees.
- Star: A tree consisting of one vertex adjacent to all the others.







Trees as Models

- Trees are used in many applications: analysis of algorithms, compilation of algebraic expressions, theoretical models of computation, etc.
  - Search trees
  - Abstract models: sort techniques like heapsort.

Number of Edges

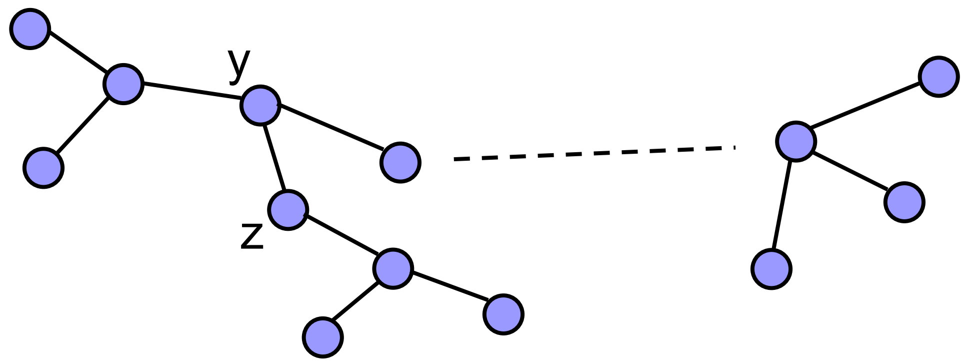
Theorem: In every tree T = (V, E),

$$|V| = |E| + 1$$

- Proof: by induction on number of edges.
  - If |E| = 0 then the tree consists of a single isolated vertex.
    $|V| = 1 = |E| + 1$
  - Assume that the theorem is true for trees of at most k edges.

Number of Edges

Consider a tree T, where $|E| = k + 1$



The edge (y,z) is removed: Two subtrees T_1 and T_2.

$|V| = |V_1| + |V_2|$ and $|E| = |E_1| + |E_2| + 1$.

Since $0 \leq |E_1| \leq k$ and $0 \leq |E_2| \leq k$,

$$|V_1| = |E_1| + 1 \text{ and } |V_2| = |E_2| + 1.$$

Consequently,

$$|V| = |V_1| + |V_2| = |E_1| + 1 + |E_2| + 1 = |E_1| + |E_2| + 1 + 1 = |E| + 1$$

Definition of Tree

Theorem: The following statements are equivalent for a loop-free undirected graph $G = (V, E)$:

A. G is a tree.
B. G is connected, but the removal of any edge from G disconnects G into two subgraphs that are trees.
C. G contains no cycle, and $|V| = |E| + 1$.
D. G is connected, and $|V| = |E| + 1$.
E. G contains no cycle, and if $a, b \in V$ with $(a, b) \notin E$, then the graph obtained by adding (a, b) to G has precisely one cycle.

Proof

- $A \Rightarrow B$
  - If G is a tree, then G is connected.
  - Let e = (a,b) be any edge of G. Then, *if* G-e is connected, there are at least two paths in G from a to b.
  - From two such paths we can form a cycle.
  - But G has no cycle.
  - Hence, G-e is disconnected and the vertices in G-e can be partitioned into two subsets:
    - Vertex a and those vertices that can be reached from a.
    - Vertex b and those vertices that can be reached from b.
  - These two connected components are trees, because a loop or cycle in either component would also be in G.

Proof (cont.)

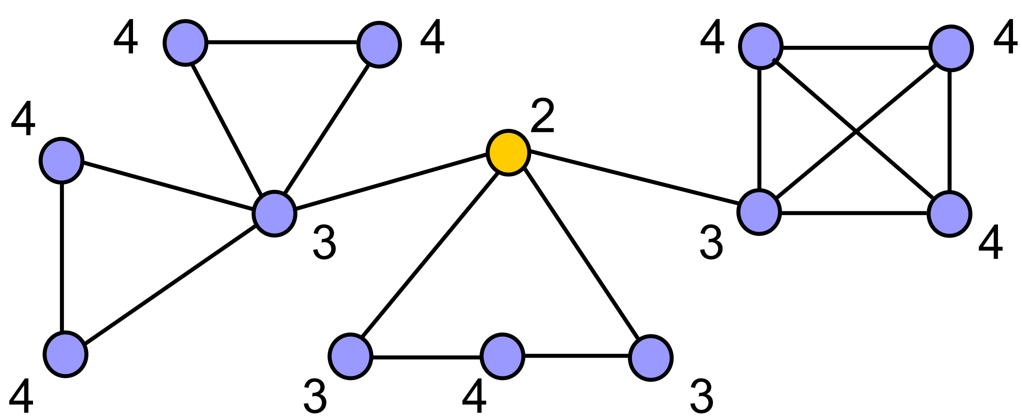
- $B \Rightarrow C$
  - If G contains a cycle then let e = (a,b) be an edge of the cycle.
  - But then, G-e is connected, contradicting the hypothesis in part B.
  - So, G contains no cycle.
  - Since G is a loop-free connected graph, we know that G is a tree.
  - Then, $|V| = |E| + 1$.

Proof (cont.)

- $C \Rightarrow D$
  - Assume G is not connected.
  - Let G_1, G_2, …, G_r be components of G.
  - For $1 \leq i \leq r$, select a vertex $v_i \in G_i$ and add the r-1 edges (v_1,v_2), (v_2,v_3), …, (v_{r-1},v_r) to G to form G'.
  - G' is a tree.
  - $|V| = |E'| + 1$
  - From C, $|V| = |E| + 1$, so $|E| = |E'|$ and $r-1 = 0$
  - With $r = 1$, it follows that G is connected.
- To complete the proof:
  - $D \Rightarrow E \wedge E \Rightarrow A$

More Definitions on Graphs

- Distance: If G has a (u,v) path, then the distance from u to v, d(u,v) is the least length of a (u,v) path.
- Eccentricity $\varepsilon(u)$ of a vertex u is $\max_{v \in V} d(u,v)$.
- The radius of a graph G is $\min_{u \in V} \varepsilon(u)$



Each vertex is labeled with its eccentricity.

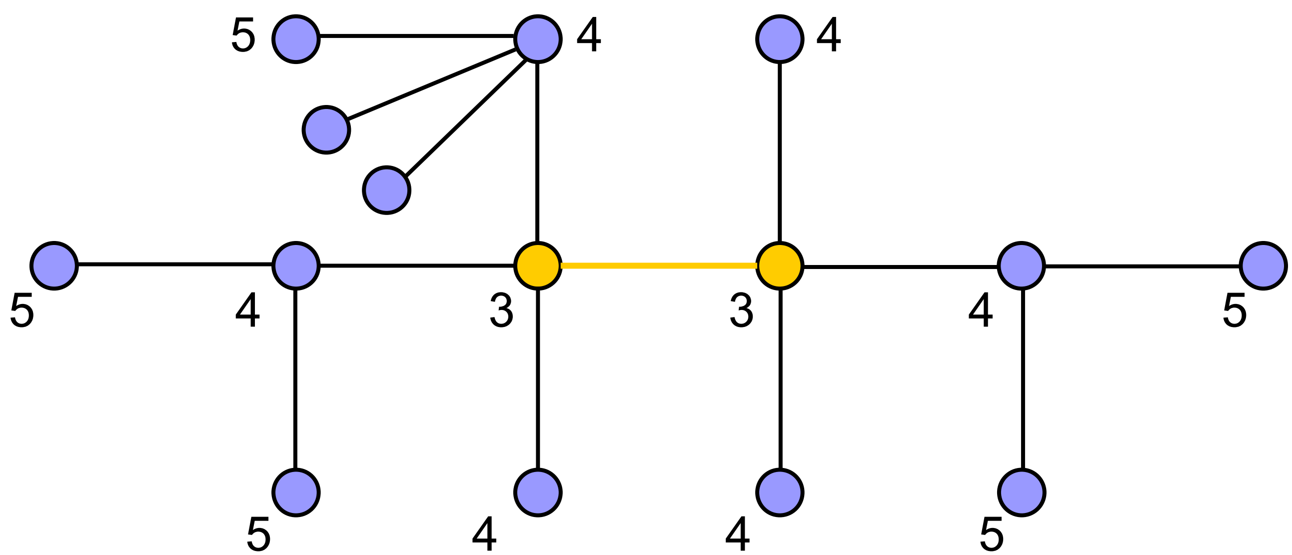
Radius: 2

Diameter: 4 (maximum eccentricity)

Center of a Tree

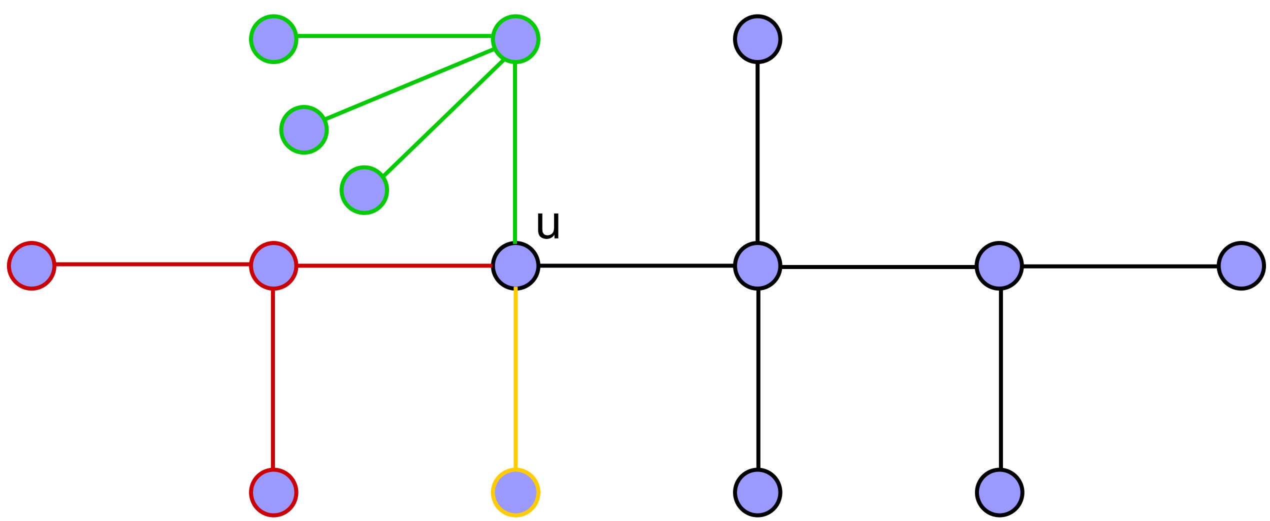
- Center: The subgraph induced by the vertices of minimum eccentricity.

Theorem: The center of a tree is a vertex or an edge (two adjacent vertices).



Branch

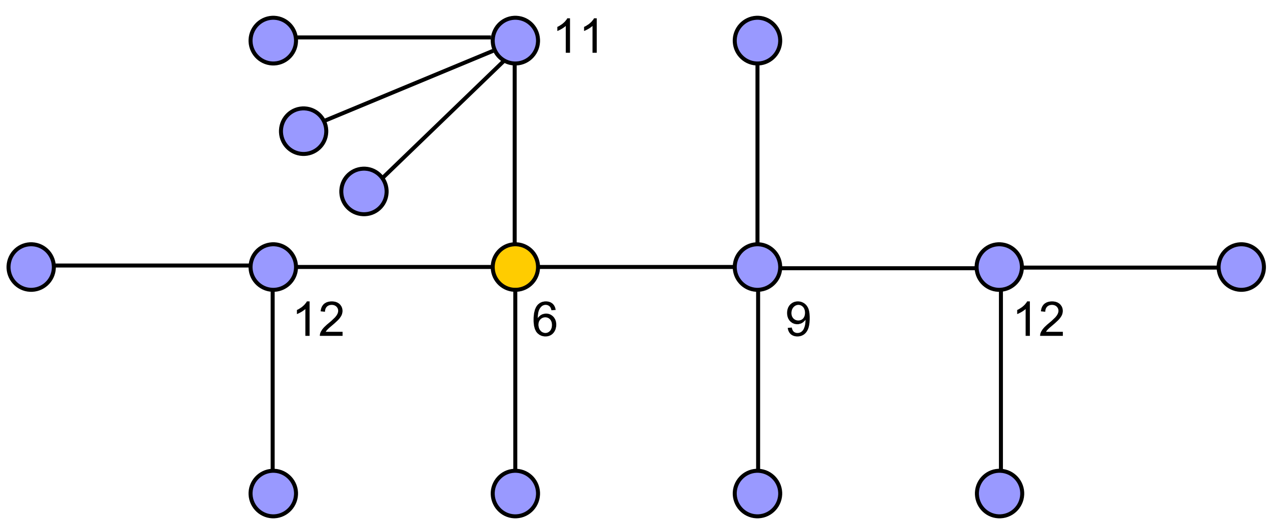
- A branch at a node u of a tree is a maximal subtree containing u as an endnode.
  - The number of branches at u is the degree of u.
  - The weight at a node u is the maximum number of edges in any branch at u.



Centroid of a Tree

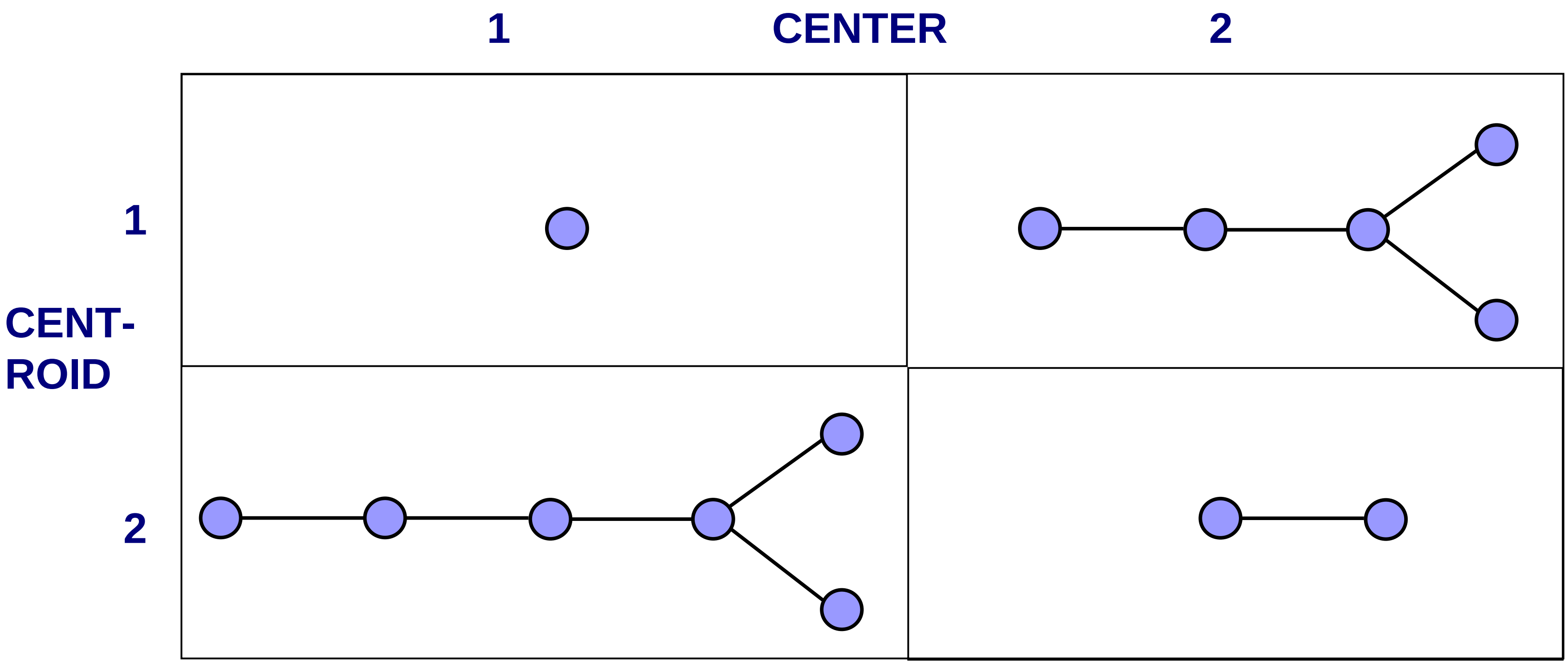
- A node v is a centroid, if v has minimum weight.
- The centroid of a tree consist of all centroid nodes.

Theorem: The centroid of a tree is a vertex or an edge (two adjacent vertices).



Center ≠ Centroid

- Smallest trees with one and two central and centroid nodes:



Wiener Index

- In a communication network, large diameter is acceptable if most pairs can communicate via shortest paths.
  - We study the average distance.
  - Average: Sum divided by n(n-1)/2. (all pairs)
  - It is equivalent to study:

$$D(G) = \sum_{u,v \in V} d_G(u,v)$$

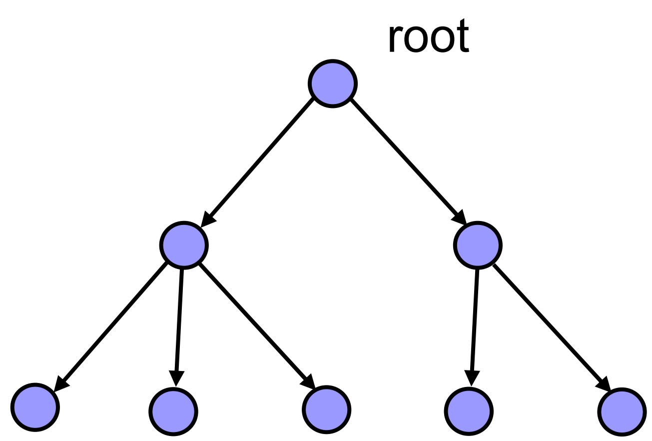
  - This sum is called the Wiener Index of G.

Directed Tree

- Edges of a tree may be directed.
- If <u,v> is a directed edge, then:
  - u is the parent of v,
  - v is the child of u.
- A vertex v is the root of a directed tree, if there are paths from v to every other vertex in the tree.

Rooted Tree

- Definition: A rooted tree is a tree in which we identify a vertex v as root (indegree: 0).
- Level of a vertex: Vertices at distance i from the root lie at level i+1.
- The height of the rooted tree is its maximum level.



Ordered Trees

- Ordered tree: A directed tree in which the set of children of each vertex is ordered.
- Binary Tree: An ordered tree in which no vertex has more than two children:
  - Left child and right child.
- Complete binary tree: Every vertex has either two children or none.
- Balanced complete binary tree: Every endpoint (leaf) has the same level.

Complete Trees

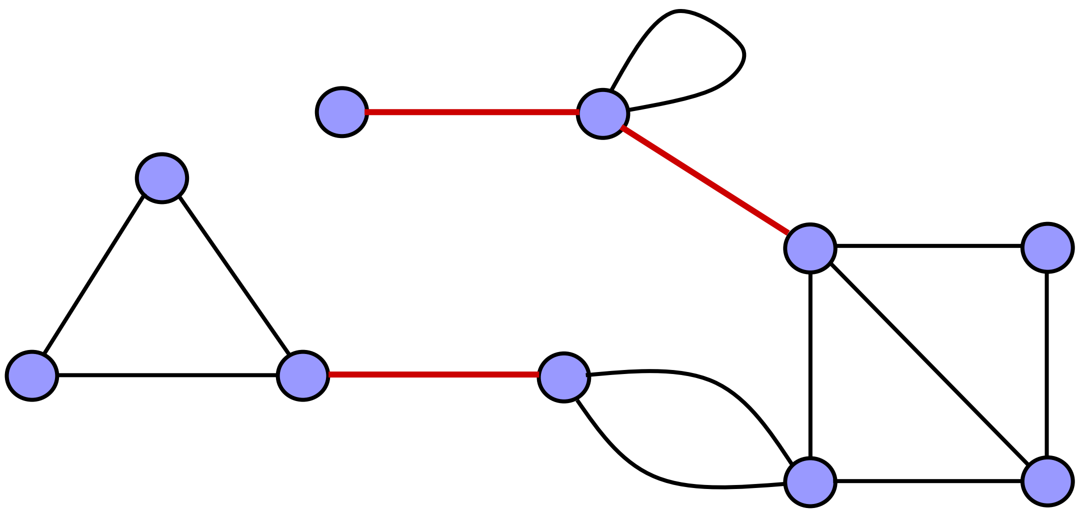
Theorem:

- A complete balanced binary tree of height h has $2^h - 1$ vertices.
- A complete balanced N-ary tree of height h has

$$\frac{N^h - 1}{N - 1}$$ vertices.

Cut Edge

- A cut edge of G is an edge e such that G – e is disconnected.

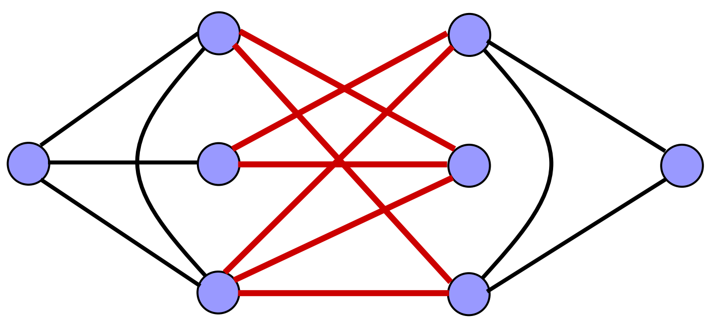


This graph has 3 cut edges.

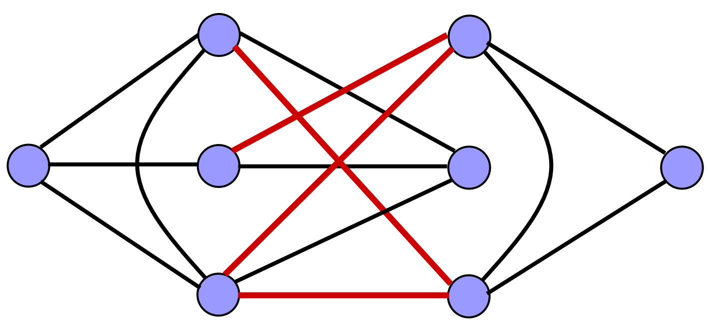
Theorem: A connected graph is a tree if and only if every edge is a cut edge.

Edge Cut

- For subsets S and S' of V,
  - [S,S'] is the set of edges with one end in S, the other in S'.
- Edge cut: A subset of E of the form [S,S'], where
  - S is a nonempty proper subset of V,
  - S' = V – S.
- Bond: A minimal nonempty edge cut of G.
- If G is connected, then a bond B is a minimal subset of E such that G – B is disconnected.



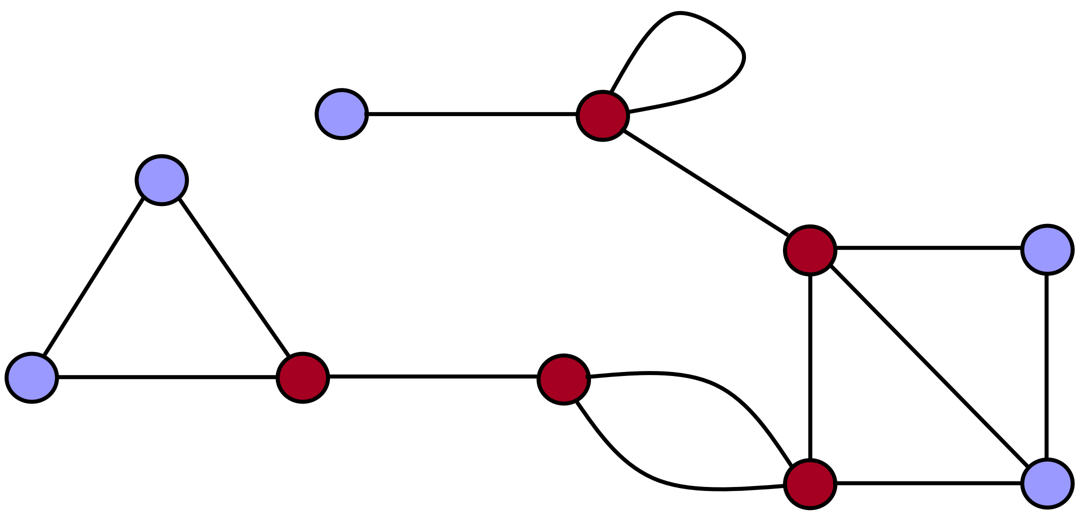
an edge cut



a bond

Cut Vertex

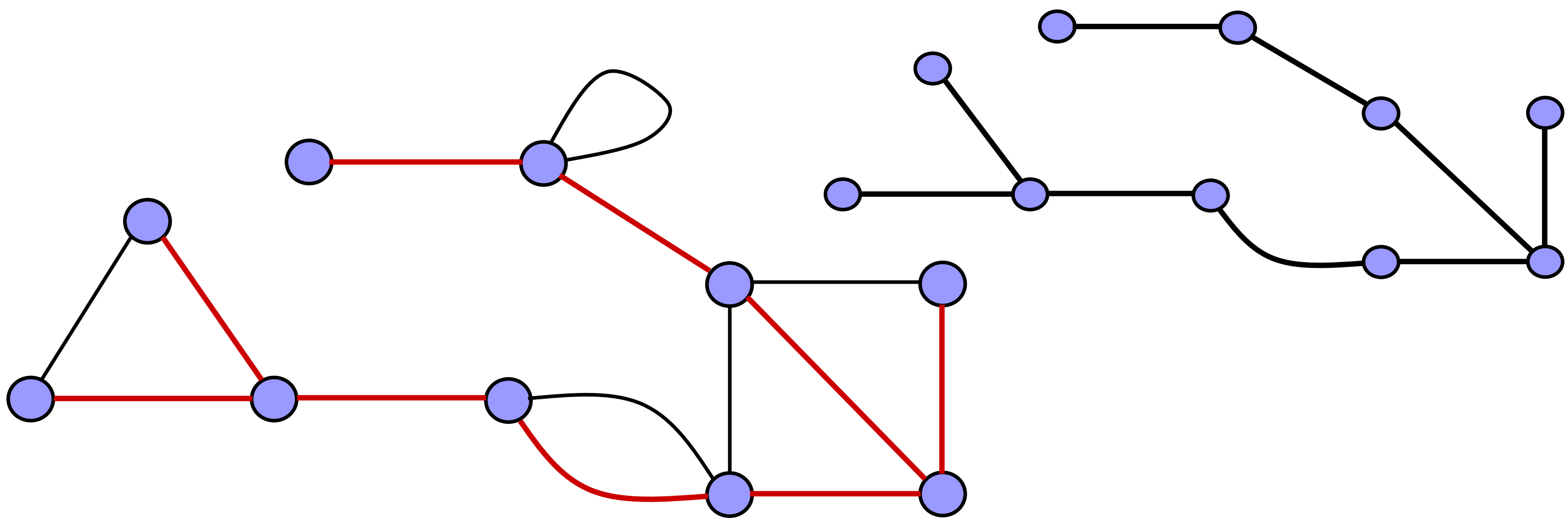
- A vertex v is a cut vertex if:
  - E can be partitioned into two nonempty subsets E_1 and E_2,
  - $G[E_1]$ and $G[E_2]$ have just the vertex v in common.



cut vertices

Spanning Tree

- A spanning tree of a connected undirected graph G is a subgraph which is a tree and which contains all the vertices of G.
  - The construction of a communication network
  - A road map or railway system



Minimum-weighted Spanning Tree

Problem: Given the cost of directly connecting any two nodes, problem is to find a network:

- at minimum cost
- and providing route between every two nodes

Solution: The solution is the minimum-weighted spanning tree of the associated weighted graph.

- Minimum-weighted spanning tree can be found by an efficient algorithm.

Steiner Tree

- A generalization of the minimum-weighted spanning tree problem:
  - Given a proper subset V' of the vertices of a graph
  - find a minimum-weighted tree which spans the vertices of V'.
- Such a tree is called Steiner Tree.
- No efficient algorithm is known for Steiner tree problem.

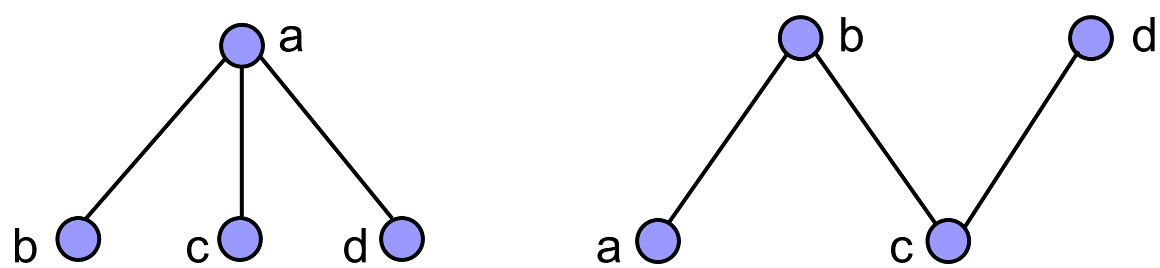
Enumeration of Trees

- With one or two vertices, only one tree can be formed.
- With three vertices there is one isomorphism class. The adjacency matrix is determined by which vertex is the center.

  - So, there are 3 trees with 3 vertices.

Enumeration of Trees

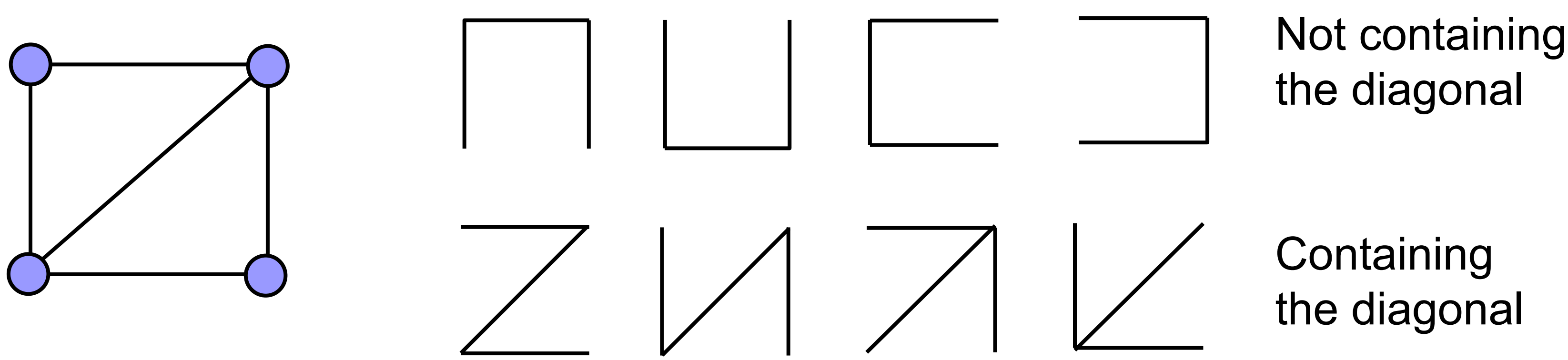
- With 4 vertices:
  - There are 4 stars and 12 paths
  - This yields to 16 trees.



- With 5 vertices, a careful study yields 125 trees.
- With n vertices, there are n^{n-2} trees: this is **Cayley's Formula**.

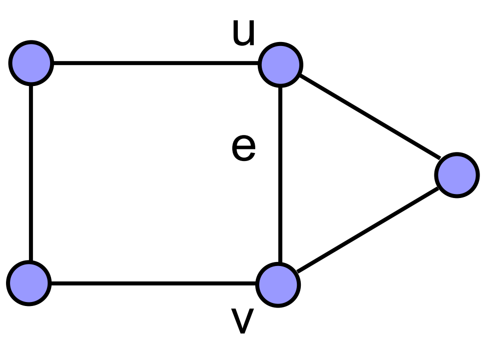
Spanning Trees in a Graph

- The complete graph with n vertices has all the edges that can be used in forming trees with n vertices.
- The number of spanning trees in a complete graph with n vertices is n^{n-2}.
- Can we find a method to compute the number of spanning trees in any graph?

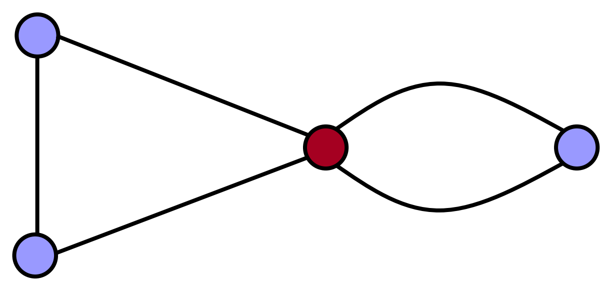


Contraction

- Definition: In a graph G, contraction of edge e with end points u and v is
  - the replacement of u and v with a single vertex
  - the incident edges of this vertex are the edges other than e that were incident to u and v.
- The resulting graph G·e has one less edge than G.



G

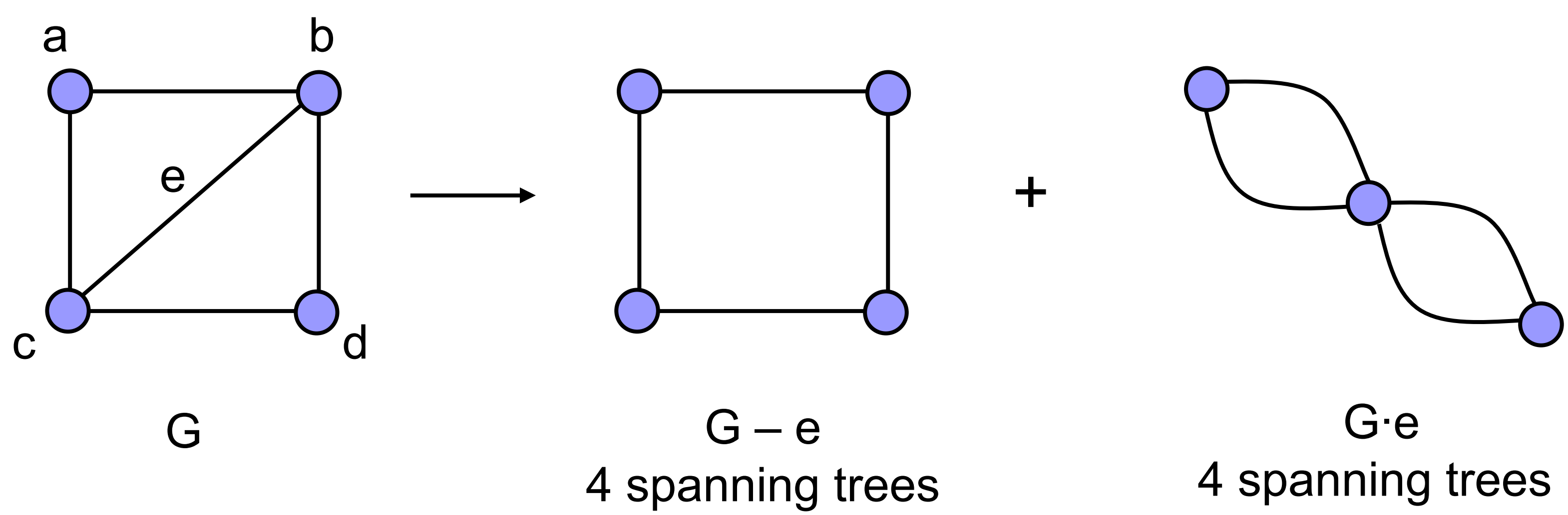

G·e

Recursive Solution

- Proposition: Let $\tau(G)$ denote the number of spanning trees of a graph G. If $e \in E$ is not a loop, then:

$$\tau(G) = \tau(G\text{-}e) + \tau(G \cdot e)$$

Example:



Recursive Solution

- This may lead to a recursive algorithm.
- We cannot apply the recurrence when e is a loop.
  - The loops do not affect the number of spanning trees.
  - Hence, we can delete loops as they arise.
- If we try to compute by deleting and contracting every edge, the amount of computation grows exponentially with the size of the graph.

Matrix Tree Computation

- Form a matrix:
  - Put the vertex degrees on the diagonal
  - The remaining elements are 0.
- Substract the adjacency matrix from it.

Example:

	a	b	c	d
a	2	0	0	0
b	0	3	0	0
c	0	0	3	0
d	0	0	0	2

−

	a	b	c	d
a	0	1	1	0
b	1	0	1	1
c	1	1	0	1
d	0	1	1	0

=

	a	b	c	d
a	2	-1	-1	0
b	-1	3	-1	-1
c	-1	-1	3	-1
d	0	-1	-1	2

Matrix for the Kite

Matrix Tree Computation

- Delete a row and a column of the resulting matrix.
- Take the determinant.

	a	b	c	d
a	2	-1	-1	0
b	-1	3	-1	-1
c	-1	-1	3	-1
d	0	-1	-1	2

→

	a	c	d
a	2	-1	0
b	-1	-1	-1
d	0	-1	2

- det: - 4 + 0 + 0 – 0 – 2 – 2 = - 8

Matrix Tree Theorem

- Given a loopless graph G:
  - Vertex set: $v_1, v_2, \ldots, v_n$
  - Let a_{ij} be the number of edges with endpoints v_i and v_j.
  - Let Q be the matrix in which entry (i,j) is:
    - $-a_{ij}$ when $i \neq j$
    - $d(v_i)$ when $i=j$.
  - If Q* is obtained by deleting rows s and column t of Q, then:

$$\tau(G) = (-1)^{s+t}\det(Q^*)$$

The Connector Problem

- A railway network connecting a number of towns is to be set up.
- Given:
  - the cost c_{ij} of constructing a direct line between towns i and j
- Design:
  - a network minimizing the total cost of construction.

Representing the connector problem

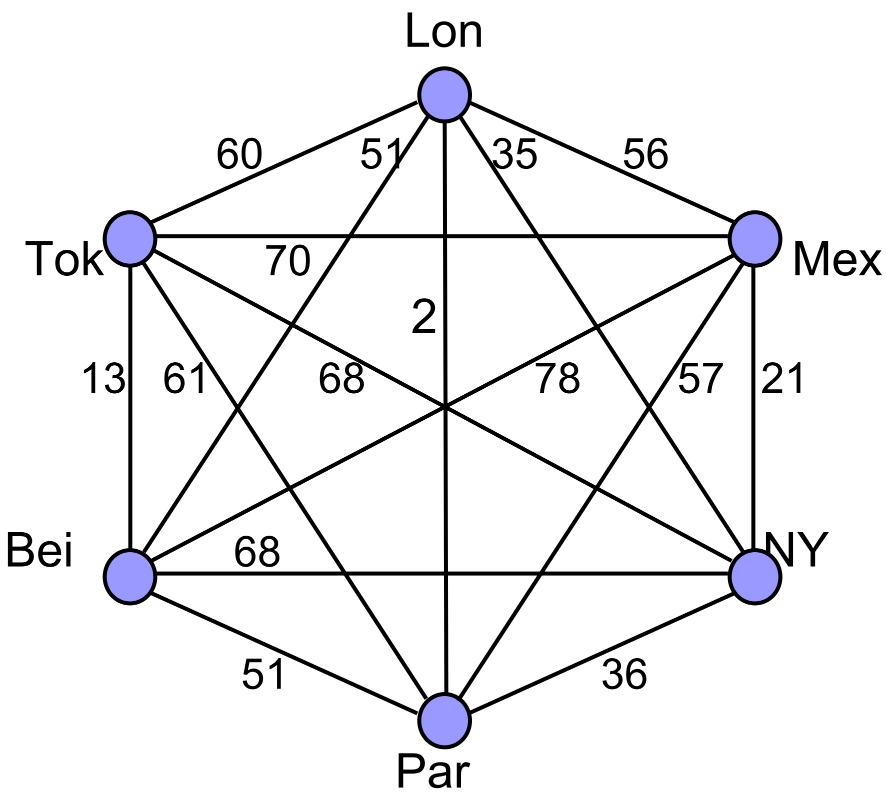
- Town = vertex
- direct line = edge
- Represent the map of possible lines as a graph.
- The problem becomes:
  - In a weighted (c_{ij}) graph, find a spanning subgraph of minimum weight.
  - As costs are positive numbers, this is equivalent to finding a minimum spanning tree.

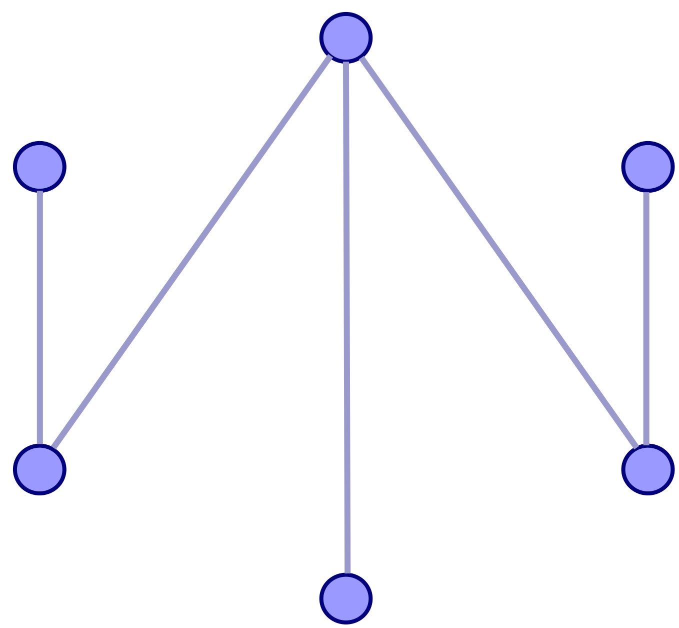
Minimum Spanning Tree: Kruskal's Algorithm

- Edges: $e_1, e_2, \ldots, e_n$
- Weights: $w(e_i)$

```
Choose a link e1 such that w(e1) is minimum.
count = 0
repeat
  if edges e1, e2, …, ei have been chosen then
    choose an edge ei+1 from E – {e1,e2,…,ei} so that:
       G[{e1, e2, …, ei+1}] contains no cycle
       w(ei+1) is minimum.
  endif
  count = count + 1
until the tree has n-1 edges (count == n-1).
```

Example





Kruskal’s Algorithm

- Theorem: Any spanning tree constructed by Kruskal’s algorithm is an optimal (minimum) tree.

  What about the complexity?
- The edges can be sorted in increasing order of weights. This takes O(e loge) time.
- At each step one edge is added to the tree, the algorithm ends when no more edges can be added.
  - Although the tree will contain n – 1 edges for an n node graph, we may need to examine e edges.
  - Hence, the number of steps necessary to construct the tree is e.

Complexity of Kruskal's Algorithm

- At each step we check that the new edge doesn't create a cycle.
  - The vertices are labeled so that at any stage, two vertices belong to the same component if they have the same label.
  - Initially, v_1 belongs to component 1, and so on.
  - Once e_i is added to the tree, the vertices at the ends are relabeled with the smaller of their two labels.
  - So, we can check whether a new edge creates a cycle, by checking the labels of its endpoints.
  - Relabeling may take O(n) comparisons.
- Therefore, the algorithm is O(e.n + eloge) = O(e.n)

The Directed Minimum Spanning Tree Problem

Problem Statement

- Consider a directed graph, G(V,A).
- Associated with each arc (i,j) is a cost c(i,j).
- Let |V|=n and |A|=m.

The problem is to find:

- A rooted directed spanning tree, G(V,S) where:
  - S is a subset of A such that the sum of c(i,j) for all (i,j) in S is minimized.
  - **The rooted directed spanning tree:** A graph which connects, without any cycle, all nodes with n-1 arcs.
  - Each node, except the root, has one and only one incoming arc.

Chu-Liu/Edmonds Algorithm

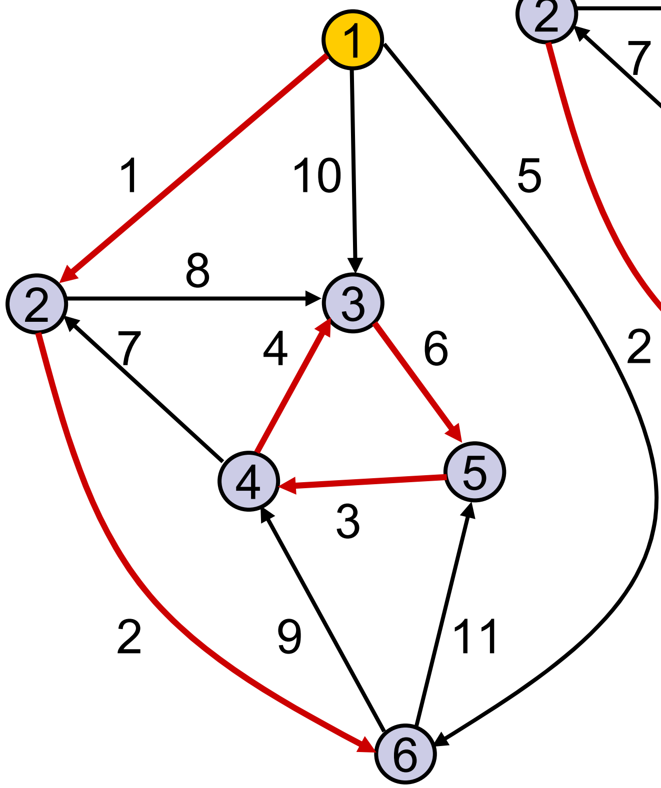
- **Discard the arcs entering the root if any.**
- **For each node other than the root**
  - **select the entering arc with the smallest cost**
- **If no cycle formed, G(V,S) is a MST. Otherwise, continue.**
- **For each cycle formed:**
  - **contract the nodes in the cycle into a pseudo-node k**
  - **modify the cost of each arc which enters a node j in the cycle from some node i outside the cycle according to the following equation:**

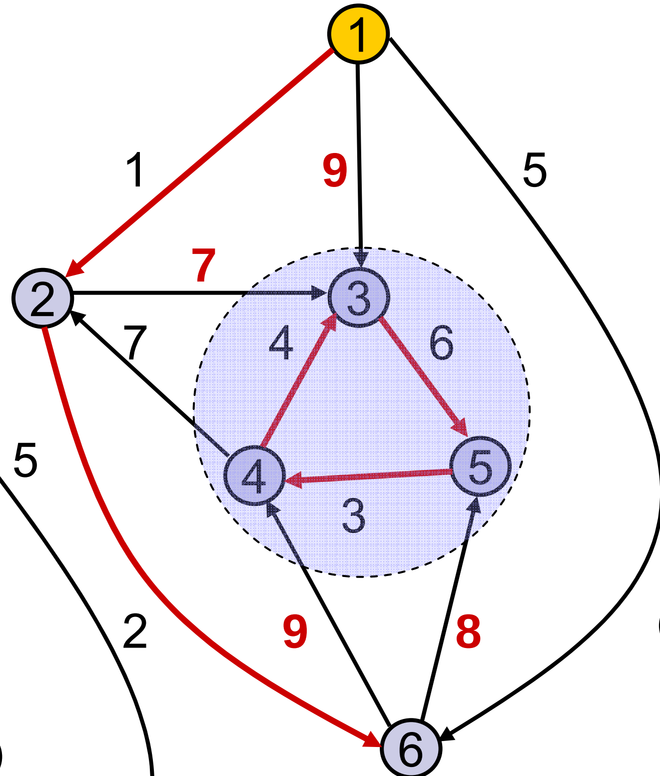
    c(i,k) = c(i,j)-(c(x(j),j) - min(c(in-cycle edges))

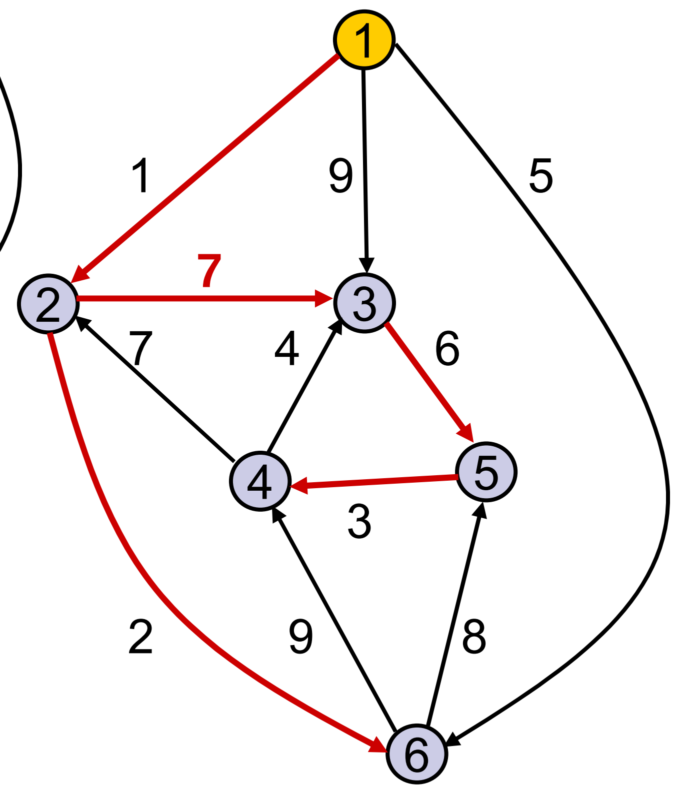
    where c(x(j),j) is the cost of the arc in the cycle which enters j.
- **For each pseudo-node:**
  - **select the entering arc with the smallest modified cost**
  - **Replace the arc in S (to same real node) by the new selected arc.**
- **Go to step 3.**

Directed MST Example

- Root: 1







Tree Application: Branch-and-Bound Method

Knapsack problem:

- A container
- Several items, each associated with:
  - a size, and
  - a value.
- Which items should we choose to pack in the container, so that:
  - The total value is maximized
  - The total size do not exceed container's size.

Tree Application: Branch-and-Bound Method

- To find the optimal solution, we need to examine all possible combinations.
- Difficult problem
- Suppose we have 5 items:

Item	A	B	C	D	E
Weight	3	8	6	4	2
Value	2	12	9	3	5

- Container size = 9

Tree Application

- Find a packing of:
  - Largest possible total value.
  - Total weight should not exceed 9.
- List all possible packings: 32 possibility
  - Choose the one with maximum value.
  - Not practical for large problem size.
- A more efficient procedure: **Branch-and-bound method**:
  - Search through a tree of possible solutions.

Solution: First Step

- List the items in decreasing order of value per unit weight:

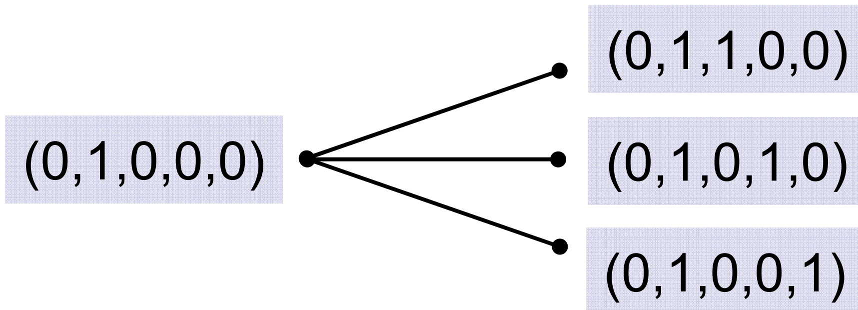
Order	**1**	**2**	**3**	**4**	**5**
Item	E	B	C	D	A
Weight	2	8	6	4	3
Value	5	12	9	3	2
Value per unit	2.5	1.5	1.5	0.75	0.67

Solution

- Denote each possible packing by a solution vector $(x_1, x_2, x_3, x_4, x_5)$
- $x_i = 1$, if item i is packed
- $x_i = 0$, otherwise
- (0,0,1,1,0) includes items 3 and 4 (C and D).
- Feasible solution: A solution which satisfies the weight constraint.
  - (0,0,1,1,0) is infeasible. Weight = 10

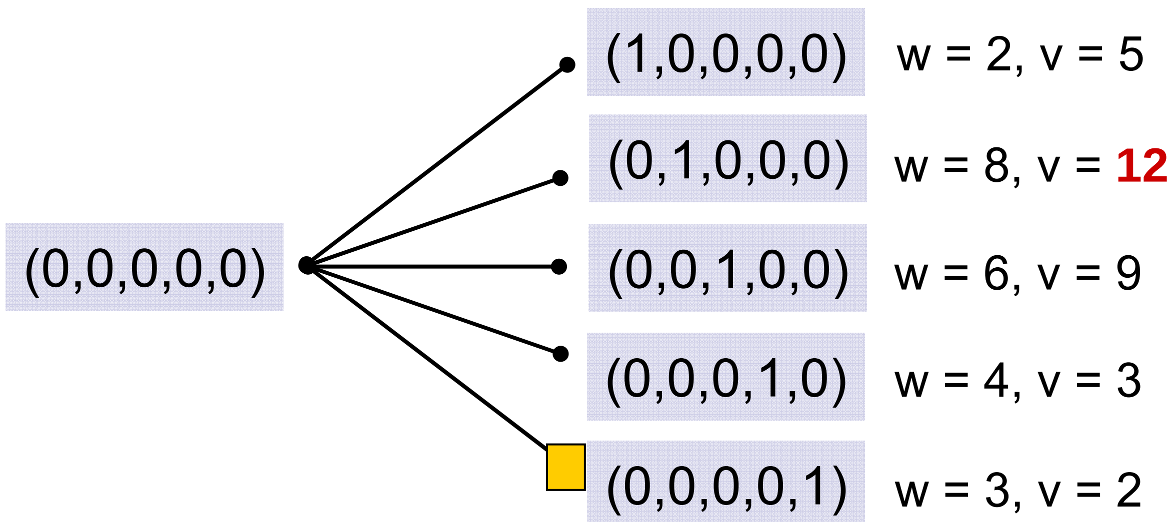
Branching out

- From vector (0,1,0,0,0) we can branch out:



- New solutions have 1 more item.
- We may change only the positions to the right of the last 1.
- The branch-and-bound starts with a null solution

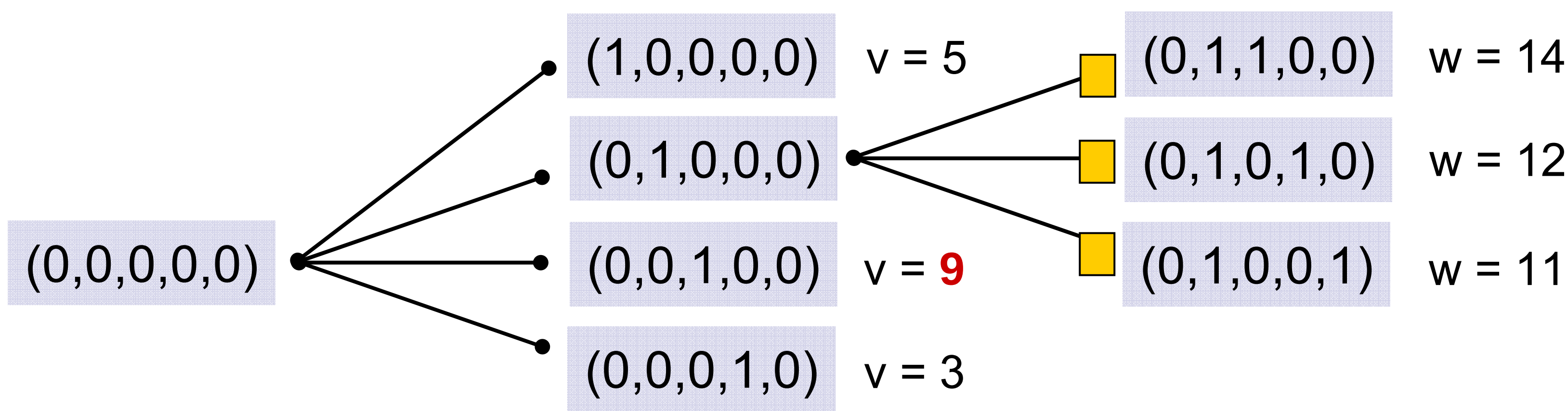
Example



- Store: v = 12, solution = (0,1,0,0,0)
- (0,0,0,0,1) is marked with a square:
  We cannot continue the branching from this vertex.

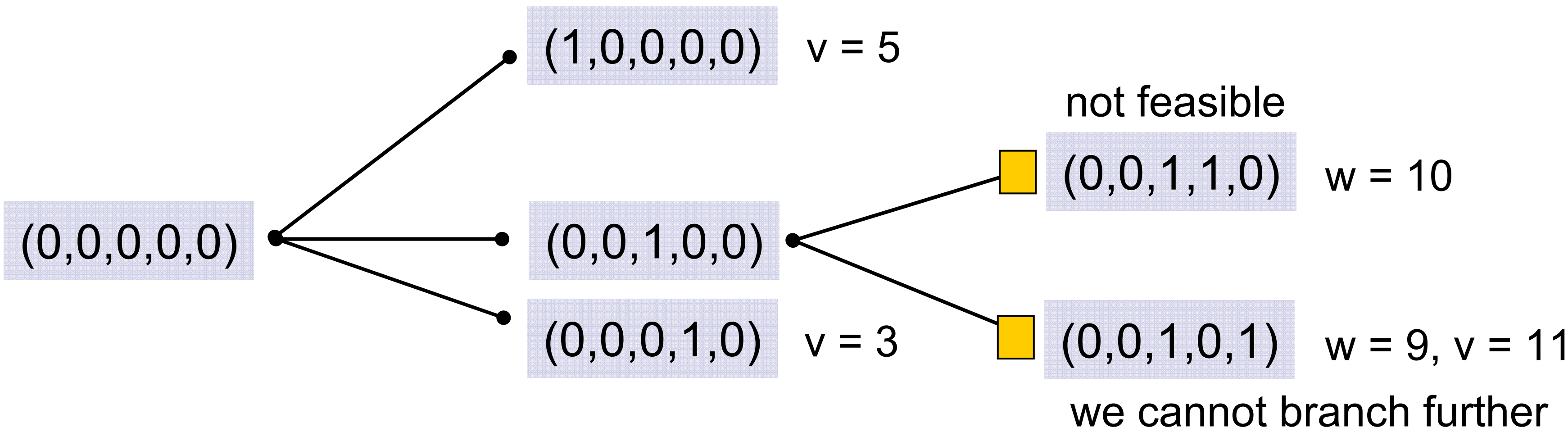
Example

- Delete the marked vertex.
- Continue the branching from the solution with the highest value.



- All three new solutions are infeasible.
- Continue from (0,0,1,0,0)

Example



- Cut the marked branches.
- Continue from vertex: (1,0,0,0,0)

Home study:

- Finish the branch-and-bound example.
- Research
  - Prim's Algorithm